RM0522
Reference manual

STM32C5 Arm®-based 32-bit MCUs

Introduction

This document is addressed to application developers. It provides complete information on how to use the STM32C5 microcontrollers memory and peripherals.

For ordering information, mechanical, and electrical device characteristics, refer to the corresponding datasheets.

For information on the Arm® Cortex®-M33 core, refer to the corresponding Arm® Technical Reference Manual available on http://infocenter.arm.com.

The STM32C5 microcontrollers include ST state-of-the-art patented technology.

Related documents

- STM32C53xxx datasheet (DS15125)
- STM32C542xx datasheet (DS15135)
- STM32C55xxx datasheet (DS14928)
- STM32C562xx datasheet (DS14927)
- STM32C59xxx datasheet (DS15136)
- STM32C5A3xx datasheet (DS15137)
- STM32C53x/C542 erratasheet (ES0676)
- STM32C55x/C562 erratasheet (ES0661)
- STM32C59x/C5A3 erratasheet (ES0677)
- STM32 microcontroller system memory boot mode (AN2606)
- STM32 Cortex®-M33 MCUs programming manual (PM0264)

Contents

1 Documentation conventions 85

1.1 General information 85

1.2 List of abbreviations for registers 85

1.3 Register reset value 86

1.4 Glossary 86

2 Memory and bus architecture 87

2.1 System architecture 87

2.1.1 Fast C-bus 88

2.1.2 Slow C-bus 89

2.1.3 S-bus 89

2.1.4 LPDMA1 and LPDMA2 buses 89

2.1.5 Bus matrix 89

2.1.6 AHB/APB bridges 89

2.2 Memory organization 90

2.2.1 Introduction 90

2.2.2 Memory map and register boundary addresses 91

2.2.3 Embedded SRAM 98

2.2.4 Flash memory overview 98

3 System security 99

3.1 Key security features 99

3.2 Resource isolation 100

3.2.1 Temporal isolation using secure hide protection (HDP) 100

3.2.2 Resource isolation using Cortex privileged mode 100

3.3 Secure execution 101

3.3.1 Memory protection unit (MPU) 101

3.3.2 Embedded flash memory write protection 101

3.3.3 Tamper detection and response 101

3.4 Secure storage 103

3.4.1 Hardware secret key management 104

3.5 Unique ID 105

3.6 Crypto engines 105

3.6.1 Cryptographic engines features . . . 105
3.6.2 Secure AES co-processor (SAES) . . . 106
3.7 Product life cycle . . . 107
3.7.1 Life cycle management with readout protection (RDP) . . . 108
3.8 Summary of protection mechanism . . . 110

4 Boot modes . . . 112
4.1 STM32C5 boot modes . . . 112

5 RAMs configuration controller (RAMCFG) . . . 114
5.1 RAMCFG introduction . . . 114
5.2 RAMCFG main features . . . 114
5.3 RAMCFG functional description . . . 114
5.3.1 Internal SRAMs features . . . 114
5.3.2 Error code correction (SRAM2) . . . 115
5.3.3 Write protection (SRAM2) . . . 116
5.3.4 Software erase . . . 116
5.4 RAMCFG low-power modes . . . 117
5.5 RAMCFG interrupts . . . 117
5.6 RAMCFG registers . . . 118
5.6.1 RAMCFG memory x control register (RAMCFG_MxCR) . . . 118
5.6.2 RAMCFG memory x interrupt status register (RAMCFG_MxISR) . . . 119
5.6.3 RAMCFG memory x erase key register (RAMCFG_MxERKEYR) . . . 119
5.6.4 RAMCFG memory 2 interrupt enable register (RAMCFG_M2IER) . . . 120
5.6.5 RAMCFG memory 2 ECC single error address register (RAMCFG_M2SEAR) . . . 120
5.6.6 RAMCFG memory 2 ECC double error address register (RAMCFG_M2DEAR) . . . 121
5.6.7 RAMCFG memory 2 interrupt clear register (RAMCFG_M2ICR) . . . 121
5.6.8 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1) . . . 122
5.6.9 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2) . . . 122
5.6.10 RAMCFG memory 2ECC key register (RAMCFG_M2ECCKEYR) . . . 122
5.6.11 RAMCFG memory 2 write protection register 3 (RAMCFG_M2WPR3) . . . 123

5.6.12 RAMCFG memory 2 write protection register 4 (RAMCFG_M2WPR4) . . . 123
5.6.13 RAMCFG register map . . . 124

6 Embedded flash memory (FLASH) . . . 126
6.1 FLASH introduction . . . 126
6.2 FLASH main features . . . 126
6.3 FLASH functional description . . . 127
6.3.1 FLASH block diagram . . . 127
6.3.2 FLASH signals . . . 127
6.3.3 Flash memory architecture and usage . . . 129
6.3.4 FLASH read operations . . . 131
6.3.5 FLASH program operations . . . 133
6.3.6 FLASH erase operations . . . 136
6.3.7 FLASH parallel operations . . . 138
6.3.8 Flash memory error protections . . . 138
6.3.9 OTP and RO memory access . . . 138
6.3.10 FLASH data area . . . 141
6.3.11 FLASH bank swapping . . . 142
6.3.12 FLASH reset and clocks . . . 147
6.4 FLASH option bytes . . . 149
6.4.1 About option bytes . . . 149
6.4.2 Option-byte loading . . . 149
6.4.3 Option-byte modification . . . 150
6.4.4 Option-byte overview . . . 151
6.4.5 Description of user and system option bytes . . . 152
6.4.6 Description of data protection option bytes . . . 153
6.4.7 Description of data FLASH option bytes . . . 153
6.4.8 Description of boot-address option byte . . . 154
6.4.9 Specific rules for modifying option bytes . . . 154
6.5 FLASH security and protections . . . 155
6.5.1 Hide protection (HDP) . . . 155
6.5.2 Flash memory register privileged and unprivileged modes . . . 157
6.5.3 Flash memory banks attributes in case of bank swap . . . 157
6.5.4 Flash memory configuration protection . . . 158
6.5.5 Write protection . . . 159
6.5.6 FLASH data area protections . . . 159

6.5.7 RDP protection and life cycle management . . . 159
6.5.8 RDP level transitions . . . 160
6.5.9 OEM key management . . . 161
6.5.10 BS key management . . . 162
6.5.11 One-time-programmable and read-only memory protections . . . 163
6.6 System memory . . . 163
6.6.1 System memory introduction . . . 163
6.6.2 Bootloader user functions . . . 163
6.7 FLASH low-power modes . . . 165
6.8 FLASH error management . . . 166
6.8.1 Write protection error (WRPERR) . . . 166
6.8.2 Programming sequence error (PGSERR) . . . 167
6.8.3 Strobe error (STRBERR) . . . 168
6.8.4 Inconsistency error (INCERR) . . . 168
6.8.5 Error correction code error (ECCC/ECCD) . . . 169
6.8.6 Option-byte change error (OPTCHANGEERR) . . . 171
6.8.7 Miscellaneous HardFault errors . . . 172
6.9 FLASH interrupts . . . 172
6.10 FLASH registers . . . 173
6.10.1 FLASH access control register (FLASH_ACR) . . . 174
6.10.2 FLASH key register (FLASH_KEYR) . . . 175
6.10.3 FLASH option key register (FLASH_OPTKEYR) . . . 175
6.10.4 FLASH operation status register (FLASH_OPSR) . . . 176
6.10.5 FLASH option control register (FLASH_OPTCR) . . . 177
6.10.6 FLASH status register (FLASH_SR) . . . 178
6.10.7 FLASH control register (FLASH_CR) . . . 180
6.10.8 FLASH clear control register (FLASH_CCR) . . . 183
6.10.9 FLASH privilege configuration register (FLASH_PRIVCFGR) . . . 183
6.10.10 FLASH HDP extension register (FLASH_HDPEXTR) . . . 184
6.10.11 FLASH option status register (FLASH_OPTSR_CUR) . . . 185
6.10.12 FLASH option status register (FLASH_OPTSR_PRG) . . . 186
6.10.13 FLASH option status register 2 (FLASH_OPTSR2_CUR) . . . 188
6.10.14 FLASH option status register 2 (FLASH_OPTSR2_PRG) . . . 189
6.10.15 FLASH unique boot entry register (FLASH_BOOTR_CUR) . . . 190
6.10.16 FLASH unique boot entry address (FLASH_BOOTR_PRG) . . . 190
6.10.17 FLASH OTP block lock (FLASH_OTPBLR_CUR) . . . 191

6.10.18 FLASH OTP block lock (FLASH_OTPBLR_PRG) 192
6.10.19 FLASH bootloader interface selection (FLASH_BL_COM_CFG_CUR) 192
6.10.20 FLASH bootloader interface selection (FLASH_BL_COM_CFG_PRG) 193
6.10.21 FLASH OEM key register 1 (FLASH_OEMKEYR1_PRG) 193
6.10.22 FLASH OEM key register 2 (FLASH_OEMKEYR2_PRG) 194
6.10.23 FLASH OEM key register 3 (FLASH_OEMKEYR3_PRG) 194
6.10.24 FLASH OEM key register 4 (FLASH_OEMKEYR4_PRG) 194
6.10.25 FLASH boundary scan key register (FLASH_BSKEYR_PRG) 195
6.10.26 FLASH write page protection for bank1 (FLASH_WRP1R_CUR) 195
6.10.27 FLASH write-page protection for bank1 (FLASH_WRP1R_PRG) 196
6.10.28 FLASH HDP bank1 register (FLASH_HDP1R_CUR) 196
6.10.29 FLASH HDP bank1 register (FLASH_HDP1R_PRG) 197
6.10.30 FLASH ECC correction register (FLASH_ECCCORR) 197
6.10.31 FLASH ECC detection register (FLASH_ECCDETR) 198
6.10.32 FLASH ECC data (FLASH_ECCDR) 199
6.10.33 FLASH write page protection for bank2 (FLASH_WRP2R_CUR) 200
6.10.34 FLASH write page protection for bank2 (FLASH_WRP2R_PRG) 200
6.10.35 FLASH HDP bank2 register (FLASH_HDP2R_CUR) 201
6.10.36 FLASH HDP bank2 register (FLASH_HDP2R_PRG) 201
6.10.37 FLASH register map 202

7 Instruction cache (ICACHE) 207

7.1 ICACHE introduction 207
7.2 ICACHE main features 207
7.3 ICACHE implementation 208
7.4 ICACHE functional description 208
7.4.1 ICACHE block diagram 209
7.4.2 ICACHE reset and clocks 209
7.4.3 ICACHE TAG memory 210
7.4.4 Direct-mapped ICACHE (1-way cache) 211
7.4.5 ICACHE enable 212
7.4.6 Cacheable and noncacheable traffic 212
7.4.7 Address remapping 213
7.4.8 Cacheable accesses 215
7.4.9 Dual-master cache 216

7.4.10 ICACHE maintenance . . . 216
7.4.11 ICACHE performance monitoring . . . 217
7.4.12 ICACHE boot . . . 217
7.5 ICACHE low-power modes . . . 217
7.6 ICACHE error management and interrupts . . . 218
7.7 ICACHE registers . . . 218
7.7.1 ICACHE control register (ICACHE_CR) . . . 218
7.7.2 ICACHE status register (ICACHE_SR) . . . 219
7.7.3 ICACHE interrupt enable register (ICACHE_IER) . . . 220
7.7.4 ICACHE flag clear register (ICACHE_FCR) . . . 220
7.7.5 ICACHE hit monitor register (ICACHE_HMONR) . . . 221
7.7.6 ICACHE miss monitor register (ICACHE_MMONR) . . . 221
7.7.7 ICACHE region x configuration register (ICACHE_CRRx) . . . 221
7.7.8 ICACHE register map . . . 222

8 Power control (PWR) . . . 224
8.1 PWR introduction . . . 224
8.2 PWR main features . . . 224
8.3 PWR pins and internal signals . . . 224
8.4 PWR power supplies and supply domains . . . 225
8.4.1 External power supplies . . . 225
8.4.2 Internal regulator . . . 226
8.4.3 Independent analog peripherals supply . . . 226
8.4.4 RTC domain . . . 226
8.5 PWR system supply voltage regulation . . . 227
8.5.1 Embedded voltage regulator operating modes . . . 227
8.6 PWR power supply and temperature supervision . . . 227
8.6.1 Power-on reset (POR)/power-down reset (PDR) . . . 227
8.6.2 Programmable voltage detector (PVD) . . . 228
8.7 Power modes . . . 229
8.7.1 Slowing down system clocks . . . 233
8.7.2 Peripheral clock gating . . . 233
8.7.3 Low-power modes . . . 233
8.7.4 Sleep mode . . . 234
8.7.5 Stop 0 mode . . . 235
8.7.6 Stop 1 mode . . . 237

8.7.7 Standby mode . . . 239
8.7.8 Power mode output pins . . . 242
8.8 PWR privileged protection . . . 242
8.9 PWR interrupts . . . 243
8.10 PWR registers . . . 243
8.10.1 PWR power mode control register (PWR_PMCR) . . . 243
8.10.2 PWR status register (PWR_PMSR) . . . 244
8.10.3 PWR RTC domain control register (PWR_RTCCR) . . . 245
8.10.4 PWR voltage monitor control register (PWR_VMCR) . . . 245
8.10.5 PWR voltage monitor status register (PWR_VMSR) . . . 246
8.10.6 PWR wake-up status clear register (PWR_WUSCR) . . . 246
8.10.7 PWR wake-up status register (PWR_WUSR) . . . 247
8.10.8 PWR wake-up configuration register (PWR_WUCR) . . . 247
8.10.9 PWR I/O retention register (PWR_IORETR) . . . 248
8.10.10 PWR privilege configuration register (PWR_PRIVCFGR) . . . 249
8.10.11 PWR register map . . . 249

9 Reset and clock control (RCC) . . . 251
9.1 RCC introduction . . . 251
9.2 RCC pins and internal signals . . . 251
9.3 RCC reset functional description . . . 251
9.3.1 Power reset . . . 251
9.3.2 System reset . . . 252
9.3.3 RTC domain reset . . . 253
9.3.4 Reset source identification . . . 253
9.4 RCC clocks functional description . . . 254
9.4.1 HSE clock . . . 256
9.4.2 HSI oscillator and HSIS clock . . . 257
9.4.3 PSI oscillator and PSIS clock . . . 258
9.4.4 HSIDIV3 and PSIDIV3 clock . . . 260
9.4.5 HSIK and PSIK clock . . . 260
9.4.6 LSE clock . . . 261
9.4.7 LSI clock . . . 262
9.4.8 System clock (SYSCLK) selection . . . 262
9.4.9 Handling clock generators in Stop and Standby modes . . . 262
9.4.10 Clock security system (CSS) . . . 263

9.4.11 Clock output generation (MCO1/MCO2/LSCO) 264
9.4.12 Kernel clock selection 265
9.4.13 RTC and TAMP clock 271
9.4.14 Timer clock 271
9.4.15 Peripherals clock gating 271
9.5 RCC privilege functional description 272
9.6 RCC low-power modes 272
9.7 RCC interrupts 273
9.8 RCC registers 274
9.8.1 RCC clock control register (RCC_CR1) 274
9.8.2 RCC clock control register (RCC_CR2) 277
9.8.3 RCC clock configuration register1 (RCC_CFGR1) 279
9.8.4 RCC CPU domain clock configuration register 2 (RCC_CFGR2) 281
9.8.5 RCC clock source interrupt enable register (RCC_CIER) 283
9.8.6 RCC clock source interrupt flag register (RCC_CIFR) 285
9.8.7 RCC clock source interrupt clear register (RCC_CICR) 286
9.8.8 RCC AHB1 reset register (RCC_AHB1RSTR) 288
9.8.9 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) 289
9.8.10 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) 291
9.8.11 RCC APB1 peripheral low reset register (RCC_APB1LRSTR) 291
9.8.12 RCC APB1 peripheral high reset register (RCC_APB1HRSTR) 293
9.8.13 RCC APB2 peripheral reset register (RCC_APB2RSTR) 294
9.8.14 RCC APB3 peripheral reset register (RCC_APB3RSTR) 295
9.8.15 RCC AHB1 peripheral clock register (RCC_AHB1ENR) 296
9.8.16 RCC AHB2 peripheral clock register (RCC_AHB2ENR) 297
9.8.17 RCC AHB4 peripheral clock register (RCC_AHB4ENR) 299
9.8.18 RCC APB1 peripheral clock register (RCC_APB1LENR) 300
9.8.19 RCC APB1 peripheral clock register (RCC_APB1HENR) 302
9.8.20 RCC APB2 peripheral clock register (RCC_APB2ENR) 303
9.8.21 RCC APB3 peripheral clock register (RCC_APB3ENR) 304
9.8.22 RCC AHB1 sleep clock register (RCC_AHB1LPENR) 305
9.8.23 RCC AHB2 sleep clock register (RCC_AHB2LPENR) 306
9.8.24 RCC AHB4 peripheral sleep clock register (RCC_AHB4LPENR) 308
9.8.25 RCC APB1 sleep clock register (RCC_APB1LLPENR) 309
9.8.26 RCC APB1 sleep clock register (RCC_APB1HLPENR) 311
9.8.27 RCC APB2 sleep clock register (RCC_APB2LPENR) 312
9.8.28 RCC APB3 sleep clock register (RCC_APB3LPENR) 313

9.8.29 RCC kernel clock configuration register (RCC_CCIPR1) 314
9.8.30 RCC kernel clock configuration register (RCC_CCIPR2) 315
9.8.31 RCC kernel clock configuration register (RCC_CCIPR3) 317
9.8.32 RCC RTC domain control register (RCC_RTCCR) 318
9.8.33 RCC reset status register (RCC_RSR) 320
9.8.34 RCC privilege configuration register (RCC_PRIVCFGR) 322
9.8.35 RCC register map 322

10 Clock recovery system (CRS) 327
10.1 CRS introduction 327
10.2 CRS main features 327
10.3 CRS implementation 327
10.4 CRS functional description 328
10.4.1 CRS block diagram 328
10.4.2 CRS internal signals 328
10.4.3 Synchronization input 329
10.4.4 Frequency error measurement 329
10.4.5 Frequency error evaluation and automatic trimming 330
10.4.6 CRS initialization and configuration 331
10.5 CRS in low-power modes 332
10.6 CRS interrupts 332
10.7 CRS registers 332
10.7.1 CRS control register (CRS_CR) 332
10.7.2 CRS configuration register (CRS_CFGR) 334
10.7.3 CRS interrupt and status register (CRS_ISR) 335
10.7.4 CRS interrupt flag clear register (CRS_ICR) 337
10.7.5 CRS register map 337

11 General-purpose I/Os (GPIO) 339
11.1 GPIO introduction 339
11.2 GPIO main features 339
11.3 GPIO functional description 339
11.3.1 General-purpose I/O (GPIO) 341
11.3.2 I/O pin alternate function multiplexer and mapping 342
11.3.3 I/O port control registers 342
11.3.4 I/O port data registers 343

11.3.5 I/O data bitwise handling . . . 343
11.3.6 GPIO locking mechanism . . . 343
11.3.7 I/O alternate function input/output . . . 343
11.3.8 External interrupt/wake-up lines . . . 344
11.3.9 Input configuration . . . 344
11.3.10 Output configuration . . . 345
11.3.11 Alternate function configuration . . . 345
11.3.12 Analog configuration . . . 346
11.3.13 Using the HSE or LSE oscillator pins as GPIOs . . . 346
11.3.14 Using the GPIO pins in the RTC supply domain . . . 347
11.3.15 I/Os state retention during standby mode . . . 347
11.3.16 Privileged and unprivileged modes . . . 347
11.3.17 I/O compensation cell . . . 347
11.4 GPIO registers . . . 347
11.4.1 GPIO port mode register (GPIOx_MODER) (x = A to H) . . . 348
11.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to H) . . . 348
11.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to H) . . . 349
11.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to H) . . 349
11.4.5 GPIO port input data register (GPIOx_IDR) (x = A to H) . . . 350
11.4.6 GPIO port output data register (GPIOx_ODR) (x = A to H) . . . 350
11.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to H) . . . 351
11.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to H) . . . 351
11.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to H) . . . 352
11.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H) . . . 353
11.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to H) . . . 354
11.4.12 GPIO register map . . . 355

12 System configuration, boot, and security (SBS) . . . 357
12.1 SBS introduction . . . 357
12.2 SBS main features . . . 357
12.3 SBS functional description . . . 357
12.3.1 SBS block diagram . . . 357
12.3.2 SBS signals . . . 358
12.3.3 SBS reset and clocks . . . 358
12.3.4 SBS I/O compensation cell management . . . 358
12.3.5 SBS boot control . . . 359
12.3.6 SBS hardware secure storage control . . . 360

12.4 SBS interrupts . . . 361
12.5 SBS registers . . . 361
12.5.1 SBS temporal isolation control register (SBS_HDPLCR) . . . 361
12.5.2 SBS temporal isolation status register (SBS_HDPLSR) . . . 362
12.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR) . . . 362
12.5.4 SBS product mode and configuration register (SBS_PMCR) . . . 363
12.5.5 SBS FPU interrupt mask register (SBS_FPUIMR) . . . 364
12.5.6 SBS memory erase status register (SBS_MESR) . . . 365
12.5.7 SBS compensation cell for I/Os control and status register (SBS_CCCSR) . . . 365
12.5.8 SBS compensation cell for I/Os value register (SBS_CCVALR) . . . 366
12.5.9 SBS compensation cell for I/Os software code register (SBS_CCSWCR) . . . 367
12.5.10 SBS Class B register (SBS_CFGR2) . . . 367
12.5.11 SBS CPU lock register (SBS_CLCKR) . . . 368
12.5.12 SBS ECC NMI mask register (SBS_ECCNMIR) . . . 369
12.5.13 SBS register map . . . 369

13 Peripherals interconnect matrix . . . 371
13.1 Interconnect matrix introduction . . . 371
13.2 Connection summary . . . 371
13.3 Interconnection details . . . 374

14 Low-power direct memory access controller (LPDMA) . . . 382
14.1 LPDMA introduction . . . 382
14.2 LPDMA main features . . . 382
14.3 LPDMA implementation . . . 383
14.3.1 LPDMA instances . . . 383
14.3.2 LPDMA channels . . . 383
14.3.3 LPDMA in low-power modes . . . 384
14.3.4 LPDMA requests . . . 384
14.3.5 LPDMA block requests . . . 387
14.3.6 LPDMA channels with peripheral early termination . . . 387
14.3.7 LPDMA triggers . . . 388
14.4 LPDMA functional description . . . 390
14.4.1 LPDMA block diagram . . . 390

14.4.2 LPDMA channel state and direct programming without any linked-list . . . 390
14.4.3 LPDMA channel suspend and resume . . . 391
14.4.4 LPDMA channel abort and restart . . . 392
14.4.5 LPDMA linked-list data structure . . . 393
14.4.6 Linked-list item transfer execution . . . 395
14.4.7 LPDMA channel state and linked-list programming in run-to-completion mode . . . 395
14.4.8 LPDMA channel state and linked-list programming in link step mode . 400
14.4.9 LPDMA channel state and linked-list programming . . . 406
14.4.10 LPDMA direct transfers . . . 408
14.4.11 LPDMA transfer request and arbitration . . . 410
14.4.12 LPDMA triggered transfer . . . 414
14.4.13 LPDMA circular buffering with linked-list programming . . . 415
14.4.14 LPDMA transfer in peripheral flow-control mode . . . 417
14.4.15 LPDMA privileged/unprivileged channel . . . 418
14.4.16 LPDMA error management . . . 418
14.5 LPDMA in debug mode . . . 420
14.6 LPDMA in low-power modes . . . 420
14.7 LPDMA interrupts . . . 420
14.8 LPDMA registers . . . 422
14.8.1 LPDMA privileged configuration register (LPDMA_PRIVCFGR) . . . 422
14.8.2 LPDMA configuration lock register (LPDMA_RCFGLOCKR) . . . 423
14.8.3 LPDMA masked interrupt status register (LPDMA_MISR) . . . 423
14.8.4 LPDMA channel x linked-list base address register (LPDMA_CxLBAR) . . . 424
14.8.5 LPDMA channel x flag clear register (LPDMA_CxFCR) . . . 424
14.8.6 LPDMA channel x status register (LPDMA_CxSR) . . . 425
14.8.7 LPDMA channel x control register (LPDMA_CxCR) . . . 426
14.8.8 LPDMA channel x transfer register 1 (LPDMA_CxTR1) . . . 429
14.8.9 LPDMA channel x transfer register 2 (LPDMA_CxTR2) . . . 431
14.8.10 LPDMA channel x block register 1 (LPDMA_CxBR1) . . . 433
14.8.11 LPDMA channel x source address register (LPDMA_CxSAR) . . . 435
14.8.12 LPDMA channel x destination address register (LPDMA_CxDAR) . . . 436
14.8.13 LPDMA channel x linked-list address register (LPDMA_CxLLR) . . . 437
14.8.14 LPDMA register map . . . 438

15 Nested vectored interrupt controller (NVIC) . . . 440

15.1 NVIC main features 440
15.2 SysTick calibration value register 440
15.3 Interrupt and exception vectors 440

16 Extended interrupts and event controller (EXTI) 445
16.1 EXTI main features 445
16.2 EXTI block diagram 446
16.2.1 EXTI connections between peripherals and CPU 447
16.2.2 EXTI interrupt/event mapping 447
16.3 EXTI functional description 448
16.3.1 EXTI configurable event input wake-up 449
16.3.2 EXTI direct event input wake-up 449
16.3.3 EXTI mux selection 450
16.4 EXTI functional behavior 450
16.5 EXTI event protection 451
16.5.1 EXTI privilege protection 452
16.6 EXTI registers 452
16.6.1 EXTI rising trigger selection register (EXTI_RTSR1) 452
16.6.2 EXTI falling trigger selection register (EXTI_FTSR1) 453
16.6.3 EXTI software interrupt event register (EXTI_SWIER1) 454
16.6.4 EXTI rising edge pending register (EXTI_RPR1) 454
16.6.5 EXTI falling edge pending register (EXTI_FPR1) 455
16.6.6 EXTI privilege configuration register (EXTI_PRIVCFGR1) 455
16.6.7 EXTI rising trigger selection register 2 (EXTI_RTSR2) 456
16.6.8 EXTI falling trigger selection register 2 (EXTI_FTSR2) 457
16.6.9 EXTI software interrupt event register 2 (EXTI_SWIER2) 458
16.6.10 EXTI rising edge pending register 2 (EXTI_RPR2) 459
16.6.11 EXTI falling edge pending register 2 (EXTI_FPR2) 460
16.6.12 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2) 461
16.6.13 EXTI external interrupt selection register (EXTI_EXTICR1) 461
16.6.14 EXTI external interrupt selection register (EXTI_EXTICR2) 463
16.6.15 EXTI external interrupt selection register (EXTI_EXTICR3) 464
16.6.16 EXTI external interrupt selection register (EXTI_EXTICR4) 466
16.6.17 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1) 468
16.6.18 EXTI CPU wake-up with event mask register (EXTI_EMR1) 468
16.6.19 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2) 469

16.6.20 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2) 469
16.6.21 EXTI register map 470

17 Cyclic redundancy check calculation unit (CRC) 472
17.1 CRC introduction 472
17.2 CRC main features 472
17.3 CRC functional description 473
17.3.1 CRC block diagram 473
17.3.2 CRC internal signals 473
17.3.3 CRC operation 473
17.4 CRC registers 475
17.4.1 CRC data register (CRC_DR) 475
17.4.2 CRC independent data register (CRC_IDR) 475
17.4.3 CRC control register (CRC_CR) 476
17.4.4 CRC initial value (CRC_INIT) 477
17.4.5 CRC polynomial (CRC_POL) 477
17.4.6 CRC register map 478

18 CORDIC coprocessor (CORDIC) 479
18.1 CORDIC introduction 479
18.2 CORDIC main features 479
18.3 CORDIC functional description 479
18.3.1 General description 479
18.3.2 CORDIC functions 479
18.3.3 Fixed point representation 486
18.3.4 Scaling factor 486
18.3.5 Precision 487
18.3.6 Zero-overhead mode 490
18.3.7 Polling mode 491
18.3.8 Interrupt mode 492
18.3.9 DMA mode 492
18.4 CORDIC registers 493
18.4.1 CORDIC control/status register (CORDIC_CSR) 493
18.4.2 CORDIC argument register (CORDIC_WDATA) 495
18.4.3 CORDIC result register (CORDIC_RDATA) 496
18.4.4 CORDIC register map 496

19 Extended-SPI interface (XSPI) 497
19.1 XSPI introduction 497
19.2 XSPI main features 497
19.3 XSPI implementation 498
19.4 XSPI functional description 499
19.4.1 XSPI block diagram 499
19.4.2 XSPI pins and internal signals 500
19.4.3 XSPI interface to memory modes 501
19.4.4 XSPI regular-command protocol 501
19.4.5 XSPI regular-command protocol signal interface 504
19.4.6 HyperBus protocol 508
19.4.7 Specific features 511
19.4.8 XSPI operating mode introduction 512
19.4.9 XSPI indirect mode 512
19.4.10 XSPI automatic status-polling mode 514
19.4.11 XSPI memory-mapped mode 515
19.4.12 XSPI configuration introduction 516
19.4.13 XSPI system configuration 516
19.4.14 XSPI device configuration 517
19.4.15 XSPI regular-command mode configuration 519
19.4.16 XSPI HyperBus protocol configuration 522
19.4.17 XSPI error management 523
19.4.18 XSPI BUSY and ABORT 523
19.4.19 XSPI reconfiguration or deactivation 524
19.4.20 NCS behavior 524
19.4.21 Software control of two external memories 524
19.4.22 Hardware-controlled extended memory support 525
19.5 Address alignment and data number 526
19.6 XSPI interrupts 527
19.7 XSPI registers 528
19.7.1 XSPI control register (XSPI_CR) 528
19.7.2 XSPI device configuration register 1 (XSPI_DCR1) 531
19.7.3 XSPI device configuration register 2 (XSPI_DCR2) 533
19.7.4 XSPI device configuration register 3 (XSPI_DCR3) 534
19.7.5 XSPI device configuration register 4 (XSPI_DCR4) 534
19.7.6 XSPI status register (XSPI_SR) 535

19.7.7 XSPI flag clear register (XSPI_FCR) . . . 536
19.7.8 XSPI data length register (XSPI_DLR) . . . 537
19.7.9 XSPI address register (XSPI_AR) . . . 537
19.7.10 XSPI data register (XSPI_DR) . . . 538
19.7.11 XSPI polling status mask register (XSPI_PSMKR) . . . 538
19.7.12 XSPI polling status match register (XSPI_PSMAR) . . . 539
19.7.13 XSPI polling interval register (XSPI_PIR) . . . 539
19.7.14 XSPI communication configuration register (XSPI_CCR) . . . 540
19.7.15 XSPI timing configuration register (XSPI_TCR) . . . 542
19.7.16 XSPI instruction register (XSPI_IR) . . . 542
19.7.17 XSPI alternate bytes register (XSPI_ABR) . . . 543
19.7.18 XSPI low-power timeout register (XSPI_LPTR) . . . 543
19.7.19 XSPI wrap communication configuration register (XSPI_WPCCR) . . . 544
19.7.20 XSPI wrap timing configuration register (XSPI_WPTCR) . . . 546
19.7.21 XSPI wrap instruction register (XSPI_WPIR) . . . 546
19.7.22 XSPI wrap alternate bytes register (XSPI_WPABR) . . . 547
19.7.23 XSPI write communication configuration register (XSPI_WCCR) . . . 547
19.7.24 XSPI write timing configuration register (XSPI_WTCR) . . . 549
19.7.25 XSPI write instruction register (XSPI_WIR) . . . 550
19.7.26 XSPI write alternate bytes register (XSPI_WABR) . . . 550
19.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR) . . . 551
19.7.28 XSPI register map . . . 551

20 Delay block (DLYB) . . . 555

20.1 DLYB introduction . . . 555
20.2 DLYB main features . . . 555
20.3 DLYB functional description . . . 555
20.3.1 DLYB diagram . . . 555
20.3.2 DLYB pins and internal signals . . . 556
20.3.3 General description . . . 556
20.3.4 Delay line length configuration procedure . . . 557
20.3.5 Output clock phase configuration procedure . . . 558
20.4 DLYB registers . . . 558
20.4.1 DLYB control register (DLYB_CR) . . . 558
20.4.2 DLYB configuration register (DLYB_CFGR) . . . 559

20.4.3 DLYB register map 560

21 Analog-to-digital converters (ADC) 561

21.1 ADC introduction 561

21.2 ADC main features 561

21.3 ADC implementation 563

21.4 ADC functional description 564

21.4.1 ADC block diagram 564

21.4.2 ADC pins and internal signals 565

21.4.3 ADC clocks 567

21.4.4 ADC connectivity 569

21.4.5 Slave AHB interface 571

21.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) 572

21.4.7 Calibration (ADCAL, ADC_CALFACT) 572

21.4.8 ADC on-off control (ADEN, ADDIS, ADRDY) 574

21.4.9 Constraints when writing the ADC control bits 575

21.4.10 Channel selection (ADC_SQRy, ADC_JSQR) 576

21.4.11 Channel preselection register (ADC_PCSEL) 576

21.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2) 577

21.4.13 Single conversion mode 578

21.4.14 Continuous conversion mode (CONT = 1) 579

21.4.15 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) 580

21.4.16 Starting conversions (ADSTART, JADSTART) 581

21.4.17 Timing 582

21.4.18 Stopping an ongoing conversion (ADSTP, JADSTP) 583

21.4.19 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) 585

21.4.20 Injected channel management 586

21.4.21 Programmable resolution (RES) - fast conversion mode 587

21.4.22 End of conversion and end of sampling phase (EOC, JEOC, EOSMP) 588

21.4.23 End of conversion sequence (EOS, JEOS) 588

21.4.24 Timing diagram examples (single/continuous modes, hardware/software triggers) 589

21.4.25 Data management 591

21.4.26 Dynamic low-power features 599

21.4.27 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT) . . . 603
21.4.28 Oversampler . . . 608
21.4.29 Dual ADC modes . . . 614
21.4.30 Temperature sensor . . . 629
21.4.31 Monitoring the internal voltage reference . . . 631
21.5 ADC interrupts . . . 633
21.6 ADC registers (for each ADC) . . . 634
21.6.1 ADC interrupt and status register (ADC_ISR) . . . 634
21.6.2 ADC interrupt enable register (ADC_IER) . . . 636
21.6.3 ADC control register (ADC_CR) . . . 638
21.6.4 ADC configuration register (ADC_CFGR1) . . . 641
21.6.5 ADC configuration register 2 (ADC_CFGR2) . . . 644
21.6.6 ADC sample time register 1 (ADC_SMPR1) . . . 647
21.6.7 ADC sample time register 2 (ADC_SMPR2) . . . 648
21.6.8 ADC channel preselection register (ADC_PCSEL) . . . 649
21.6.9 ADC regular sequence register 1 (ADC_SQR1) . . . 649
21.6.10 ADC regular sequence register 2 (ADC_SQR2) . . . 650
21.6.11 ADC regular sequence register 3 (ADC_SQR3) . . . 651
21.6.12 ADC regular sequence register 4 (ADC_SQR4) . . . 652
21.6.13 ADC regular data register (ADC_DR) . . . 653
21.6.14 ADC injected sequence register (ADC_JSQR) . . . 653
21.6.15 ADC offset y configuration register (ADC_OFCFGRy) . . . 655
21.6.16 ADC offset y register (ADC_OFRy) . . . 656
21.6.17 ADC gain compensation register (ADC_GCOMP) . . . 657
21.6.18 ADC injected channel y data register (ADC_JDRy) . . . 658
21.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . 658
21.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . 659
21.6.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR) . 659
21.6.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR) . . . 660
21.6.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR) . 660
21.6.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR) . . . 661
21.6.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR) . 661
21.6.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR) . . . 661

21.6.27 ADC calibration factors (ADC_CALFACT) 662
21.6.28 ADC option register (ADC_OR) 663
21.7 ADC common registers 663
21.7.1 ADC common status register (ADCC_CSR) 663
21.7.2 ADC common control register (ADCC_CCR) 665
21.7.3 ADC common regular data register for dual mode (ADCC_CDR) 666
21.7.4 ADC common regular data register for dual mode (ADCC_CDR2) 667
21.8 ADC register map 667

22 Digital-to-analog converter (DAC) 672
22.1 DAC introduction 672
22.2 DAC main features 672
22.3 DAC implementation 673
22.4 DAC functional description 674
22.4.1 DAC block diagram 674
22.4.2 DAC pins and internal signals 675
22.4.3 DAC clocks 676
22.4.4 DAC channel enable 676
22.4.5 DAC data format 677
22.4.6 DAC conversion 678
22.4.7 DAC output voltage 679
22.4.8 DAC trigger selection 680
22.4.9 DMA requests 680
22.4.10 Noise generation 681
22.4.11 Triangle-wave generation 682
22.4.12 DAC channel modes 683
22.4.13 DAC channel buffer calibration 686
22.4.14 DAC channel conversion modes 687
22.4.15 Dual DAC channel conversion modes (if dual channels are available) 688
22.5 DAC in low-power modes 692
22.6 DAC interrupts 692
22.7 DAC registers 693
22.7.1 DAC control register (DAC_CR) 693
22.7.2 DAC software trigger register (DAC_SWTRGR) 696

22.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) . . . 697
22.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) . . . 698
22.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) . . . 698
22.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) . . . 699
22.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) . . . 699
22.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) . . . 700
22.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) . . . 700
22.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) . . . 701
22.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) . . . 701
22.7.12 DAC channel1 data output register (DAC_DOR1) . . . 702
22.7.13 DAC channel2 data output register (DAC_DOR2) . . . 702
22.7.14 DAC status register (DAC_SR) . . . 703
22.7.15 DAC calibration control register (DAC_CCR) . . . 704
22.7.16 DAC mode control register (DAC_MCR) . . . 705
22.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1) . . . 706
22.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2) . . . 707
22.7.19 DAC sample and hold time register (DAC_SHHR) . . . 707
22.7.20 DAC sample and hold refresh time register (DAC_SHRR) . . . 708
22.7.21 DAC register map . . . 709

23 Comparator (COMP) . . . 711

23.1 Introduction . . . 711
23.2 COMP main features . . . 711
23.3 COMP implementation . . . 712
23.3.1 COMP implementation for STM32C55x/562 and STM32C59x/5A3 devices . . . 712
23.3.2 COMP implementation for STM32C53x/542 device . . . 712
23.4 COMP functional description . . . 713
23.4.1 COMP block diagram . . . 713

23.4.2 COMP pins and internal signals . . . 713
23.4.3 COMP reset and clocks . . . 715
23.4.4 Comparator LOCK mechanism . . . 715
23.4.5 Window comparator (optional feature) . . . 715
23.4.6 Hysteresis . . . 716
23.4.7 Comparator output blanking function . . . 716
23.4.8 COMP startup stabilization time . . . 717
23.4.9 COMP power and speed modes . . . 717
23.5 COMP low-power modes . . . 717
23.6 COMP interrupts . . . 718
23.6.1 COMP wake-up interrupt through EXTI . . . 718
23.6.2 COMP NVIC interrupt through EXTI . . . 718
23.6.3 COMP interrupt through NVIC of the CPU . . . 719
23.7 SCALER function . . . 719
23.8 COMP registers . . . 720
23.8.1 Comparator status register (COMP_SR) . . . 720
23.8.2 Comparator interrupt clear flag register (COMP_ICFR) . . . 721
23.8.3 Comparator configuration register 1 (COMP_CFGR1) . . . 721
23.8.4 Comparator configuration register 2 (COMP_CFGR2) . . . 723
23.8.5 COMP register map . . . 725

24 Operational amplifier (OPAMP) . . . 726
24.1 Introduction . . . 726
24.2 OPAMP main features . . . 726
24.3 OPAMP implementation . . . 726
24.4 OPAMP functional description . . . 727
24.4.1 OPAMP block diagram . . . 727
24.4.2 OPAMP pins and internal signals . . . 728
24.4.3 OPAMP reset and clocks . . . 729
24.4.4 OPAMP output redirection to an internal COMP channel . . . 729
24.4.5 OPAMP initial configuration . . . 730
24.4.6 OPAMP operational amplifier speed . . . 730
24.4.7 OPAMP enable . . . 730
24.4.8 OPAMP operating modes . . . 730
24.4.9 OPAMP programmable gain . . . 736
24.4.10 OPAMP operational amplifier offset compensation . . . 737

24.4.11 OPAMP calibration . . . 737
24.4.12 Timer-controlled operation . . . 739
24.5 OPAMP in low-power modes . . . 740
24.6 OPAMP registers . . . 741
24.6.1 OPAMP control/status register (OPAMP_CSR) . . . 741
24.6.2 OPAMP timer-controlled mode register (OPAMP_TCMR) . . . 743
24.6.3 OPAMP register map . . . 746

25 Coupling and chaining bridge (CCB) . . . 747
25.1 CCB introduction . . . 747
25.2 CCB main features . . . 747
25.3 CCB implementation . . . 748
25.4 CCB functional description . . . 748
25.4.1 CCB block diagram . . . 748
25.4.2 CCB internal signals . . . 749
25.4.3 CCB reset and clocks . . . 749
25.4.4 CCB coupling and chaining . . . 749
25.4.5 CCB error management . . . 752
25.5 CCB coupling and chaining operations . . . 752
25.5.1 Preliminary steps before any protected operation . . . 754
25.5.2 ECDSA signature . . . 755
25.5.3 ECC scalar multiplication . . . 762
25.5.4 RSA modular exponentiation . . . 765
25.6 CCB interrupts . . . 767
25.7 CCB registers . . . 768
25.7.1 CCB control register (CCB_CR) . . . 768
25.7.2 CCB status register (CCB_SR) . . . 769
25.7.3 CCB reference tag register (CCB_REFTAGRx) . . . 771
25.7.4 CCB register map . . . 772

26 True random number generator (RNG) . . . 773
26.1 RNG introduction . . . 773
26.2 RNG main features . . . 773
26.3 RNG functional description . . . 774
26.3.1 RNG block diagram . . . 774
26.3.2 RNG internal signals . . . 774

26.3.3 Random number generation . . . 775
26.3.4 RNG initialization . . . 777
26.3.5 RNG operation . . . 778
26.3.6 RNG clocking . . . 780
26.3.7 Error management . . . 780
26.3.8 RNG low-power use . . . 782
26.4 RNG interrupts . . . 782
26.5 RNG processing time . . . 783
26.6 RNG entropy source validation . . . 783
26.6.1 Introduction . . . 783
26.6.2 Validation conditions . . . 783
26.7 RNG registers . . . 785
26.7.1 RNG control register (RNG_CR) . . . 785
26.7.2 RNG status register (RNG_SR) . . . 787
26.7.3 RNG data register (RNG_DR) . . . 788
26.7.4 RNG noise source control register (RNG_NSCR) . . . 789
26.7.5 RNG health test control register x (RNG_HTCRx) . . . 789
26.7.6 RNG health test status register 0 (RNG_HTSR0) . . . 790
26.7.7 RNG health test status register 1 (RNG_HTSR1) . . . 790
26.7.8 RNG noise source mask register (RNG_NSMR) . . . 791
26.7.9 RNG register map . . . 792

27 AES hardware accelerator (AES) . . . 793
27.1 AES introduction . . . 793
27.2 AES main features . . . 793
27.3 AES implementation . . . 794
27.4 AES functional description . . . 794
27.4.1 AES block diagram . . . 794
27.4.2 AES internal signals . . . 795
27.4.3 AES reset and clocks . . . 795
27.4.4 AES symmetric cipher implementation . . . 795
27.4.5 AES encryption or decryption typical usage . . . 796
27.4.6 AES authenticated encryption, decryption, and cipher-based message authentication . . . 799
27.4.7 AES ciphertext stealing and data padding . . . 799
27.4.8 AES suspend and resume operations . . . 800

27.4.9 AES basic chaining modes (ECB, CBC) 800
27.4.10 AES counter (CTR) mode 804
27.4.11 AES Galois/counter mode (GCM) 806
27.4.12 AES Galois message authentication code (GMAC) 811
27.4.13 AES counter with CBC-MAC (CCM) 812
27.4.14 AES key sharing with secure AES co-processor 817
27.4.15 AES data registers and data swapping 818
27.4.16 AES key registers 820
27.4.17 AES initialization vector registers 820
27.4.18 AES error management 821
27.5 AES interrupts 822
27.6 AES DMA requests 822
27.7 AES processing latency 823
27.8 AES registers 825
27.8.1 AES control register (AES_CR) 825
27.8.2 AES status register (AES_SR) 827
27.8.3 AES data input register (AES_DINR) 828
27.8.4 AES data output register (AES_DOUTR) 829
27.8.5 AES key register 0 (AES_KEYR0) 829
27.8.6 AES key register 1 (AES_KEYR1) 830
27.8.7 AES key register 2 (AES_KEYR2) 830
27.8.8 AES key register 3 (AES_KEYR3) 830
27.8.9 AES initialization vector register 0 (AES_IVR0) 831
27.8.10 AES initialization vector register 1 (AES_IVR1) 831
27.8.11 AES initialization vector register 2 (AES_IVR2) 831
27.8.12 AES initialization vector register 3 (AES_IVR3) 832
27.8.13 AES key register 4 (AES_KEYR4) 832
27.8.14 AES key register 5 (AES_KEYR5) 832
27.8.15 AES key register 6 (AES_KEYR6) 833
27.8.16 AES key register 7 (AES_KEYR7) 833
27.8.17 AES suspend registers (AES_SUSPRx) 833
27.8.18 AES interrupt enable register (AES_IER) 834
27.8.19 AES interrupt status register (AES_ISR) 835
27.8.20 AES interrupt clear register (AES_ICR) 836
27.8.21 AES register map 836

28 Secure AES coprocessor (SAES) 839

28.1 SAES introduction . . . 839
28.2 SAES main features . . . 839
28.3 SAES implementation . . . 840
28.4 SAES functional description . . . 840
28.4.1 SAES block diagram . . . 840
28.4.2 SAES internal signals . . . 841
28.4.3 SAES reset and clocks . . . 842
28.4.4 SAES symmetric cipher implementation . . . 842
28.4.5 SAES encryption or decryption typical usage . . . 843
28.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication . . . 845
28.4.7 SAES ciphertext stealing and data padding . . . 846
28.4.8 SAES suspend and resume operations . . . 846
28.4.9 SAES basic chaining modes (ECB, CBC) . . . 847
28.4.10 SAES counter (CTR) mode . . . 851
28.4.11 SAES Galois/counter mode (GCM) . . . 853
28.4.12 SAES Galois message authentication code (GMAC) . . . 857
28.4.13 SAES counter with CBC-MAC (CCM) . . . 859
28.4.14 SAES operation with wrapped keys . . . 864
28.4.15 SAES operation with shared keys . . . 868
28.4.16 SAES data registers and data swapping . . . 869
28.4.17 SAES key registers . . . 872
28.4.18 SAES initialization vector registers . . . 873
28.4.19 SAES error management . . . 873
28.5 SAES interrupts . . . 875
28.6 SAES DMA requests . . . 876
28.7 SAES processing latency . . . 876
28.8 SAES registers . . . 878
28.8.1 SAES control register (SAES_CR) . . . 878
28.8.2 SAES status register (SAES_SR) . . . 881
28.8.3 SAES data input register (SAES_DINR) . . . 882
28.8.4 SAES data output register (SAES_DOUTR) . . . 882
28.8.5 SAES key register 0 (SAES_KEYR0) . . . 883
28.8.6 SAES key register 1 (SAES_KEYR1) . . . 883
28.8.7 SAES key register 2 (SAES_KEYR2) . . . 884
28.8.8 SAES key register 3 (SAES_KEYR3) . . . 884

28.8.9 SAES initialization vector register 0 (SAES_IVR0) 884
28.8.10 SAES initialization vector register 1 (SAES_IVR1) 885
28.8.11 SAES initialization vector register 2 (SAES_IVR2) 885
28.8.12 SAES initialization vector register 3 (SAES_IVR3) 885
28.8.13 SAES key register 4 (SAES_KEYR4) 886
28.8.14 SAES key register 5 (SAES_KEYR5) 886
28.8.15 SAES key register 6 (SAES_KEYR6) 886
28.8.16 SAES key register 7 (SAES_KEYR7) 887
28.8.17 SAES suspend registers (SAES_SUSPRx) 887
28.8.18 SAES interrupt enable register (SAES_IER) 887
28.8.19 SAES interrupt status register (SAES_ISR) 888
28.8.20 SAES interrupt clear register (SAES_ICR) 890
28.8.21 SAES register map 890

29 Hash processor (HASH) 893

29.1 HASH introduction 893
29.2 HASH main features 893
29.3 HASH implementation 894
29.4 HASH functional description 894
29.4.1 HASH block diagram 894
29.4.2 HASH internal signals 895
29.4.3 About secure hash algorithms 895
29.4.4 Message data feeding 895
29.4.5 Message digest computing 897
29.4.6 Message padding 899
29.4.7 HMAC operation 901
29.4.8 HASH suspend/resume operations 902
29.4.9 HASH DMA interface 904
29.4.10 HASH error management 905
29.4.11 HASH processing time 905
29.5 HASH interrupts 906
29.6 HASH registers 906
29.6.1 HASH control register (HASH_CR) 906
29.6.2 HASH data input register (HASH_DIN) 908
29.6.3 HASH start register (HASH_STR) 909
29.6.4 HASH digest registers 910

29.6.5 HASH interrupt enable register (HASH_IMR) 912
29.6.6 HASH status register (HASH_SR) 912
29.6.7 HASH context swap registers 913
29.6.8 HASH register map 914

30 Public key accelerator (PKA) 916
30.1 PKA introduction 916
30.2 PKA main features 916
30.3 PKA functional description 917
30.3.1 PKA block diagram 917
30.3.2 PKA internal signals 917
30.3.3 PKA reset and clocks 918
30.3.4 PKA public key acceleration 918
30.3.5 Typical applications for PKA 920
30.3.6 PKA procedure to perform an operation 922
30.3.7 Private keys protection using CCB 923
30.3.8 PKA error management 927
30.4 PKA operating modes 927
30.4.1 Introduction 927
30.4.2 Montgomery parameter computation 929
30.4.3 Modular addition 929
30.4.4 Modular subtraction 930
30.4.5 Modular and Montgomery multiplication 930
30.4.6 Modular exponentiation 931
30.4.7 Modular inversion 933
30.4.8 Modular reduction 933
30.4.9 Arithmetic addition 934
30.4.10 Arithmetic subtraction 934
30.4.11 Arithmetic multiplication 935
30.4.12 Arithmetic comparison 935
30.4.13 RSA CRT exponentiation 935
30.4.14 Point on elliptic curve Fp check 936
30.4.15 ECC Fp scalar multiplication 937
30.4.16 ECDSA sign 938
30.4.17 ECDSA verification 940
30.4.18 ECC complete addition 941
30.4.19 ECC double base ladder 941

30.4.20 ECC projective to affine 942
30.5 Example of configurations and processing times 943
30.5.1 Supported elliptic curves 943
30.5.2 Computation times 945
30.6 PKA interrupts 947
30.7 PKA registers 948
30.7.1 PKA control register (PKA_CR) 948
30.7.2 PKA status register (PKA_SR) 950
30.7.3 PKA clear flag register (PKA_CLRFR) 952
30.7.4 PKA RAM 953
30.7.5 PKA register map 954

31 Advanced-control timers (TIM1/TIM8) 955
31.1 TIM1/TIM8 introduction 955
31.2 TIM1/TIM8 main features 955
31.3 TIM1/TIM8 functional description 956
31.3.1 Block diagram 956
31.3.2 TIM1/TIM8 pins and internal signals 957
31.3.3 Time-base unit 961
31.3.4 Counter modes 964
31.3.5 Repetition counter 976
31.3.6 External trigger input 977
31.3.7 Clock selection 978
31.3.8 Capture/compare channels 982
31.3.9 Input capture mode 985
31.3.10 PWM input mode 986
31.3.11 Forced output mode 987
31.3.12 Output compare mode 987
31.3.13 PWM mode 988
31.3.14 Asymmetric PWM mode 997
31.3.15 Combined PWM mode 1000
31.3.16 Combined 3-phase PWM mode 1003
31.3.17 Complementary outputs and dead-time insertion 1004
31.3.18 Clock dividers for dead-time generators and digital filters 1007
31.3.19 Using the break function 1007
31.3.20 Delayed break 1013

31.3.21 Bidirectional break inputs . . . 1015
31.3.22 Output idle state override . . . 1017
31.3.23 Clearing the tim_ocxref signal on an external event . . . 1018
31.3.24 6-step PWM generation . . . 1021
31.3.25 One-pulse mode . . . 1022
31.3.26 Retriggerable One-pulse mode . . . 1024
31.3.27 Pulse on compare mode . . . 1025
31.3.28 Encoder interface mode . . . 1027
31.3.29 Encoder with built-in debouncer . . . 1044
31.3.30 Direction bit output . . . 1046
31.3.31 Clock drift measurement . . . 1046
31.3.32 UIF bit remapping . . . 1047
31.3.33 Timer input XOR function . . . 1047
31.3.34 Interfacing with Hall sensors . . . 1049
31.3.35 Timer synchronization . . . 1050
31.3.36 ADC triggers . . . 1055
31.3.37 ADC synchronization . . . 1055
31.3.38 DMA burst mode . . . 1056
31.3.39 TIM1/TIM8 DMA requests . . . 1057
31.3.40 Debug mode . . . 1058
31.4 TIM1/TIM8 low-power modes . . . 1058
31.5 TIM1/TIM8 interrupts . . . 1058
31.6 TIM1/TIM8 registers . . . 1060
31.6.1 TIM control register 1 (TIM_CR1) . . . 1060
31.6.2 TIM control register 2 (TIM_CR2) . . . 1062
31.6.3 TIM slave mode control register (TIM_SMCR) . . . 1067
31.6.4 TIM DMA/interrupt enable register (TIM_DIER) . . . 1071
31.6.5 TIM status register (TIM_SR) . . . 1072
31.6.6 TIM event generation register (TIM_EGR) . . . 1076
31.6.7 TIM capture/compare mode register 1 (TIM_CCMR1) . . . 1077
31.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 1079
31.6.9 TIM capture/compare mode register 2 (TIM_CCMR2) . . . 1083
31.6.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2) . . . 1085
31.6.11 TIM capture/compare enable register (TIM_CCER) . . . 1088
31.6.12 TIM counter (TIM_CNT) . . . 1092
31.6.13 TIM prescaler (TIM_PSC) . . . 1092
31.6.14 TIM autoreload register (TIM_ARR) . . . 1093

31.6.15 TIM repetition counter register (TIM_RCR) 1093
31.6.16 TIM capture/compare register 1 (TIM_CCR1) 1094
31.6.17 TIM capture/compare register 2 (TIM_CCR2) 1094
31.6.18 TIM capture/compare register 3 (TIM_CCR3) 1095
31.6.19 TIM capture/compare register 4 (TIM_CCR4) 1096
31.6.20 TIM break and dead-time register (TIM_BDTR) 1097
31.6.21 TIM capture/compare register 5 (TIM_CCR5) 1101
31.6.22 TIM capture/compare register 6 (TIM_CCR6) 1103
31.6.23 TIM capture/compare mode register 3 (TIM_CCMR3) 1104
31.6.24 TIM timer deadtime register 2 (TIM_DTR2) 1107
31.6.25 TIM timer encoder control register (TIM_ECR) 1108
31.6.26 TIM timer input selection register (TIM_TISEL) 1109
31.6.27 TIM alternate function option register 1 (TIM_AF1) 1110
31.6.28 TIM alternate function register 2 (TIM_AF2) 1115
31.6.29 TIM capture/compare register 7 (TIM_CCR7) 1119
31.6.30 TIM capture/compare mode register 4 (TIM_CCMR4) 1120
31.6.31 TIM multilevel protection register 1 (TIM_MPR1) 1122
31.6.32 TIM multilevel protection register 2 (TIM_MPR2) 1124
31.6.33 TIM output override register (TIM_OOR) 1125
31.6.34 TIM DMA control register (TIM_DCR) 1127
31.6.35 TIM DMA address for full transfer (TIM_DMAR) 1128
31.6.36 TIM1/TIM8 register map 1129

32 General-purpose timers(TIM2/TIM3/TIM4/TIM5) 1133

32.1 TIM2/TIM3/TIM4/TIM5 introduction 1133
32.2 TIM2/TIM3/TIM4/TIM5 main features 1133
32.3 TIM2/TIM3/TIM4/TIM5 implementation 1134
32.4 TIM2/TIM3/TIM4/TIM5 functional description 1135
32.4.1 Block diagram 1135
32.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals 1136
32.4.3 Time-base unit 1140
32.4.4 Counter modes 1142
32.4.5 External trigger input 1153
32.4.6 Clock selection 1154
32.4.7 Capture/compare channels 1158
32.4.8 Input capture mode 1160
32.4.9 PWM input mode 1161

32.4.10 Forced output mode 1162
32.4.11 Output compare mode 1162
32.4.12 PWM mode 1164
32.4.13 Asymmetric PWM mode 1172
32.4.14 Combined PWM mode 1174
32.4.15 Clock dividers for dead-time generators and digital filters 1175
32.4.16 Clearing the tim_ocxref signal on an external event 1176
32.4.17 One-pulse mode 1179
32.4.18 Retriggerable one-pulse mode 1181
32.4.19 Pulse on compare mode 1182
32.4.20 Encoder interface mode 1184
32.4.21 Encoder with built-in debouncer 1202
32.4.22 Direction bit output 1204
32.4.23 Clock drift measurement 1204
32.4.24 UIF bit remapping 1205
32.4.25 Timer input XOR function 1205
32.4.26 Timers and external trigger synchronization 1207
32.4.27 Timer synchronization 1210
32.4.28 ADC triggers 1215
32.4.29 ADC synchronization 1217
32.4.30 DMA burst mode 1217
32.4.31 TIM2/TIM3/TIM4/TIM5 DMA requests 1218
32.4.32 Debug mode 1219
32.4.33 TIM2/TIM3/TIM4/TIM5 low-power modes 1219
32.4.34 TIM2/TIM3/TIM4/TIM5 interrupts 1220
32.5 TIM2/TIM3/TIM4/TIM5 registers 1221
32.5.1 TIM control register 1 (TIM_CR1) 1221
32.5.2 TIM control register 2 (TIM_CR2) 1223
32.5.3 TIM slave mode control register (TIM_SMCR) 1226
32.5.4 TIM DMA/interrupt enable register (TIM_DIER) 1230
32.5.5 TIM status register (TIM_SR) 1231
32.5.6 TIM event generation register (TIM_EGR) 1234
32.5.7 TIM capture/compare mode register 1 (TIM_CCMR1) 1235
32.5.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 1237
32.5.9 TIM capture/compare mode register 2 (TIM_CCMR2) 1239
32.5.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2) . . . 1240
32.5.11 TIM capture/compare enable register (TIM_CCER) 1243

32.5.12 TIM counter (TIM_CNT) 1245
32.5.13 TIM prescaler (TIM_PSC) 1245
32.5.14 TIM autoreload register (TIM_ARR) 1246
32.5.15 TIM capture/compare register 1 (TIM_CCR1) 1246
32.5.16 TIM capture/compare register 2 (TIM_CCR2) 1247
32.5.17 TIM capture/compare register 3 (TIM_CCR3) 1248
32.5.18 TIM capture/compare register 4 (TIM_CCR4) 1249
32.5.19 TIM timer encoder control register (TIM_ECR) 1250
32.5.20 TIM timer input selection register (TIM_TISEL) 1251
32.5.21 TIM alternate function register 1 (TIM_AF1) 1252
32.5.22 TIM alternate function register 2 (TIM_AF2) 1253
32.5.23 TIM DMA control register (TIM_DCR) 1254
32.5.24 TIM DMA address for full transfer (TIM_DMAR) 1255
32.5.25 TIM2/TIM3/TIM4/TIM5 register map 1256

33 Basic timers (TIM6/TIM7) 1259

33.1 TIM6/TIM7 introduction 1259
33.2 TIM6/TIM7 main features 1259
33.3 TIM6/TIM7 functional description 1260
33.3.1 TIM6/TIM7 block diagram 1260
33.3.2 TIM6/TIM7 internal signals 1260
33.3.3 TIM6/TIM7 clocks 1261
33.3.4 Time-base unit 1261
33.3.5 Counting mode 1263
33.3.6 UIF bit remapping 1270
33.3.7 ADC triggers 1271
33.3.8 ADC synchronization 1271
33.3.9 TIM6/TIM7 DMA requests 1272
33.3.10 Debug mode 1272
33.3.11 TIM6/TIM7 low-power modes 1272
33.3.12 TIM6/TIM7 interrupts 1272
33.4 TIM6/TIM7 registers 1273
33.4.1 TIM control register 1 (TIM_CR1) 1273
33.4.2 TIM control register 2 (TIM_CR2) 1275
33.4.3 TIM DMA/interrupt enable register (TIM_DIER) 1275
33.4.4 TIM status register (TIM_SR) 1277
33.4.5 TIM event generation register (TIM_EGR) 1277

33.4.6 TIM counter (TIM_CNT) 1278
33.4.7 TIM prescaler (TIM_PSC) 1278
33.4.8 TIM autoreload register (TIM_ARR) 1279
33.4.9 TIM6/TIM7 register map 1280

34 General-purpose timers (TIM12) 1281

34.1 TIM12 introduction 1281
34.2 TIM12 main features 1281
34.3 TIM12 functional description 1282
34.3.1 Block diagram 1282
34.3.2 TIM12 pins and internal signals 1282
34.3.3 Time-base unit 1284
34.3.4 Counter modes 1286
34.3.5 Clock selection 1290
34.3.6 Capture/compare channels 1292
34.3.7 Input capture mode 1294
34.3.8 PWM input mode 1295
34.3.9 Forced output mode 1296
34.3.10 Output compare mode 1297
34.3.11 PWM mode 1298
34.3.12 Combined PWM mode 1303
34.3.13 Clock dividers for digital filters 1304
34.3.14 One-pulse mode 1305
34.3.15 Retriggerable one pulse mode 1306
34.3.16 UIF bit remapping 1307
34.3.17 Timer input XOR function 1307
34.3.18 TIM12 external trigger synchronization 1309
34.3.19 Slave mode – combined reset + trigger mode 1312
34.3.20 Slave mode – combined reset + gated mode 1312
34.3.21 Timer synchronization 1312
34.3.22 ADC triggers 1312
34.3.23 ADC synchronization 1313
34.3.24 Debug mode 1314
34.4 TIM12 low-power modes 1314
34.5 TIM12 interrupts 1314
34.6 TIM12 registers 1314

34.6.1 TIM control register 1 (TIM_CR1) 1315
34.6.2 TIM control register 2 (TIM_CR2) 1316
34.6.3 TIM slave mode control register (TIM_SMCR) 1318
34.6.4 TIM interrupt enable register (TIM_DIER) 1320
34.6.5 TIM status register (TIM_SR) 1321
34.6.6 TIM event generation register (TIM_EGR) 1322
34.6.7 TIM capture/compare mode register 1 (TIM_CCMR1) 1323
34.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 1324
34.6.9 TIM capture/compare enable register (TIM_CCER) 1327
34.6.10 TIM counter (TIM_CNT) 1328
34.6.11 TIM prescaler (TIM_PSC) 1329
34.6.12 TIM autoreload register (TIM_ARR) 1329
34.6.13 TIM capture/compare register 1 (TIM_CCR1) 1330
34.6.14 TIM capture/compare register 2 (TIM_CCR2) 1330
34.6.15 TIM timer input selection register (TIM_TISEL) 1331
34.6.16 TIM12 register map 1332

35 General purpose timers (TIM15/TIM16/TIM17) 1334

35.1 TIM15/TIM16/TIM17 introduction 1334
35.2 TIM15 main features 1334
35.3 TIM16/TIM17 main features 1335
35.4 TIM15/TIM16/TIM17 implementation 1335
35.5 TIM15/TIM16/TIM17 functional description 1336
35.5.1 Block diagram 1336
35.5.2 TIM15/TIM16/TIM17 pins and internalsignals 1337
35.5.3 Time-base unit 1341
35.5.4 Counter modes 1343
35.5.5 Repetition counter 1355
35.5.6 Clock selection 1357
35.5.7 Capture/compare channels 1359
35.5.8 Input capture mode 1361
35.5.9 PWM input mode (only for TIM15) 1362
35.5.10 Forced output mode 1363
35.5.11 Output compare mode 1364
35.5.12 PWM mode 1365
35.5.13 Asymmetric PWM mode (TIM15 only) 1372
35.5.14 Combined PWM mode (TIM15 only) 1373

35.5.15 Complementary outputs and dead-time insertion 1374
35.5.16 Clock dividers for dead-time generators and digital filters 1377
35.5.17 Using the break function 1377
35.5.18 Delayed break 1382
35.5.19 Bidirectional break input 1384
35.5.20 Output idle state override 1386
35.5.21 Clearing the tim_ocxref signal on an external event 1386
35.5.22 6-step PWM generation 1388
35.5.23 One-pulse mode 1390
35.5.24 Retriggerable one pulse mode (TIM15 only) 1391
35.5.25 Encoder interface mode 1392
35.5.26 Encoder with built-in debouncer (TIM15 only) 1399
35.5.27 Clock drift measurement 1400
35.5.28 UIF bit remapping 1401
35.5.29 Timer input XOR function (TIM15 only) 1401
35.5.30 External trigger synchronization (TIM15 only) 1403
35.5.31 Slave mode – combined reset + trigger mode (TIM15 only) 1406
35.5.32 Slave mode – combined reset + gated mode (TIM15 only) 1406
35.5.33 Timer synchronization (TIM15 only) 1407
35.5.34 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 1407
35.5.35 ADC triggers (TIM15 only) 1407
35.5.36 ADC synchronization (TIM15 only) 1408
35.5.37 DMA burst mode 1408
35.5.38 TIM15/TIM16/TIM17 DMA requests 1409
35.5.39 Debug mode 1410
35.6 TIM15/TIM16/TIM17 low-power modes 1410
35.7 TIM15/TIM16/TIM17 interrupts 1410
35.8 TIM15 registers 1412
35.8.1 TIM control register 1 (TIM_CR1) 1412
35.8.2 TIM control register 2 (TIM_CR2) 1414
35.8.3 TIM slave mode control register (TIM_SMCR) 1417
35.8.4 TIM DMA/interrupt enable register (TIM_DIER) 1420
35.8.5 TIM status register (TIM_SR) 1421
35.8.6 TIM event generation register (TIM_EGR) 1423
35.8.7 TIM capture/compare mode register 1 (TIM_CCMR1) 1424
35.8.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 1426
35.8.9 TIM capture/compare enable register (TIM_CCER) 1429

35.8.10 TIM counter (TIM_CNT) 1432
35.8.11 TIM prescaler (TIM_PSC) 1432
35.8.12 TIM autoreload register (TIM_ARR) 1432
35.8.13 TIM repetition counter register (TIM_RCR) 1433
35.8.14 TIM capture/compare register 1 (TIM_CCR1) 1433
35.8.15 TIM capture/compare register 2 (TIM_CCR2) 1435
35.8.16 TIM break and dead-time register (TIM_BDTR) 1435
35.8.17 TIM timer deadtime register 2 (TIM_DTR2) 1438
35.8.18 TIM input selection register (TIM_TISEL) 1439
35.8.19 TIM alternate function register 1 (TIM_AF1) 1440
35.8.20 TIM alternate function register 2 (TIM_AF2) 1444
35.8.21 TIM multilevel protection register 1 (TIM_MPR1) 1445
35.8.22 TIM multilevel protection register 2 (TIM_MPR2) 1445
35.8.23 TIM output override register (TIM_OOR) 1446
35.8.24 TIM DMA control register (TIM_DCR) 1447
35.8.25 TIM DMA address for full transfer (TIM_DMAR) 1448
35.8.26 TIM15 register map 1449
35.9 TIM16/TIM17 registers 1452
35.9.1 TIM control register 1 (TIM_CR1) 1452
35.9.2 TIM control register 2 (TIM_CR2) 1454
35.9.3 TIM DMA/interrupt enable register (TIM_DIER) 1455
35.9.4 TIM status register (TIM_SR) 1456
35.9.5 TIM event generation register (TIM_EGR) 1457
35.9.6 TIM capture/compare mode register 1 (TIM_CCMR1) 1458
35.9.7 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 1459
35.9.8 TIM capture/compare enable register (TIM_CCER) 1461
35.9.9 TIM counter (TIM_CNT) 1464
35.9.10 TIM prescaler (TIM_PSC) 1464
35.9.11 TIM auto-reautoreload register (TIM_ARR) 1465
35.9.12 TIM repetition counter register (TIM_RCR) 1465
35.9.13 TIM capture/compare register 1 (TIM_CCR1) 1466
35.9.14 TIM break and dead-time register (TIM_BDTR) 1467
35.9.15 TIM timer deadtime register 2 (TIM_DTR2) 1470
35.9.16 TIM input selection register (TIM_TISEL) 1471
35.9.17 TIM alternate function register 1 (TIM_AF1) 1471
35.9.18 TIM alternate function register 2 (TIM_AF2) 1475
35.9.19 TIM multilevel protection register 1 (TIM_MPR1) 1476

35.9.20 TIM multilevel protection register 2 (TIM_MPR2) 1477
35.9.21 TIM output override register (TIM_OOR) 1477
35.9.22 TIM DMA control register (TIM_DCR) 1478
35.9.23 TIM DMA address for full transfer (TIM_DMAR) 1479
35.9.24 TIM16/TIM17 register map 1480

36 Low-power timer (LPTIM) 1483
36.1 LPTIM introduction 1483
36.2 LPTIM main features 1483
36.3 LPTIM implementation 1483
36.4 LPTIM functional description 1485
36.4.1 LPTIM block diagram 1485
36.4.2 LPTIM pins and internal signals 1485
36.4.3 LPTIM input and trigger mapping 1487
36.4.4 LPTIM reset and clocks 1488
36.4.5 Glitch filter 1488
36.4.6 Prescaler 1489
36.4.7 Trigger multiplexer 1489
36.4.8 Operating mode 1490
36.4.9 Timeout function 1492
36.4.10 Waveform generation 1492
36.4.11 Register update 1493
36.4.12 Counter mode 1494
36.4.13 Timer enable 1495
36.4.14 Timer counter reset 1495
36.4.15 Encoder mode 1496
36.4.16 Repetition counter 1497
36.4.17 Capture/compare channels 1498
36.4.18 Input capture mode 1499
36.4.19 PWM mode 1501
36.4.20 DMA requests 1503
36.4.21 Debug mode 1504
36.5 LPTIM low-power modes 1504
36.6 LPTIM interrupts 1504
36.7 LPTIM registers 1505
36.7.1 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR) 1506

36.7.2 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR) 1508
36.7.3 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR) 1510
36.7.4 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR) 1511
36.7.5 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER) 1513
36.7.6 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER) 1514
36.7.7 LPTIM configuration register (LPTIM_CFGR) 1516
36.7.8 LPTIM control register (LPTIM_CR) 1518
36.7.9 LPTIM compare register 1 (LPTIM_CCR1) 1519
36.7.10 LPTIM autoreload register (LPTIM_ARR) 1520
36.7.11 LPTIM counter register (LPTIM_CNT) 1520
36.7.12 LPTIM configuration register 2 (LPTIM_CFGR2) 1521
36.7.13 LPTIM repetition register (LPTIM_RCR) 1522
36.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) 1523
36.7.15 LPTIM compare register 2 (LPTIM_CCR2) 1525
36.7.16 LPTIM register map 1526

37 Independent watchdog (IWDG) 1528

37.1 IWDG introduction 1528
37.2 IWDG main features 1528
37.3 IWDG implementation 1528
37.4 IWDG functional description 1529
37.4.1 IWDG block diagram 1529
37.4.2 IWDG internal signals 1530
37.4.3 Software and hardware watchdog modes 1530
37.4.4 Window option 1531
37.4.5 Debug 1534
37.4.6 Register access protection 1534
37.5 IWDG low power modes 1535
37.6 IWDG interrupts 1535
37.7 IWDG registers 1537
37.7.1 IWDG key register (IWDG_KR) 1538
37.7.2 IWDG prescaler register (IWDG_PR) 1538
37.7.3 IWDG reload register (IWDG_RLR) 1539
37.7.4 IWDG status register (IWDG_SR) 1539
37.7.5 IWDG window register (IWDG_WINR) 1541
37.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 1541

37.7.7 IWDG interrupt clear register (IWDG_ICR) 1542
37.7.8 IWDG register map 1543

38 System window watchdog (WWDG) 1544
38.1 WWDG introduction 1544
38.2 WWDG main features 1544
38.3 WWDG implementation 1544
38.4 WWDG functional description 1545
38.4.1 WWDG block diagram 1545
38.4.2 WWDG internal signals 1545
38.4.3 Enabling the watchdog 1546
38.4.4 Controlling the down-counter 1546
38.4.5 How to program the watchdog timeout 1546
38.4.6 Debug mode 1547
38.5 WWDG interrupts 1548
38.6 WWDG registers 1548
38.6.1 WWDG control register (WWDG_CR) 1548
38.6.2 WWDG configuration register (WWDG_CFR) 1549
38.6.3 WWDG status register (WWDG_SR) 1550
38.6.4 WWDG register map 1550

39 Real-time clock (RTC) 1551
39.1 RTC introduction 1551
39.2 RTC main features 1551
39.3 RTC functional description 1551
39.3.1 RTC block diagram 1551
39.3.2 RTC pins and internal signals 1553
39.3.3 GPIOs controlled by the RTC and TAMP 1554
39.3.4 RTC privilege protection modes 1556
39.3.5 Clock and prescalers 1557
39.3.6 Real-time clock and calendar 1558
39.3.7 Calendar ultra-low power mode 1559
39.3.8 Programmable alarms 1559
39.3.9 Periodic auto-wake-up 1559
39.3.10 RTC initialization and configuration 1560
39.3.11 Reading the calendar 1563

39.3.12 Resetting the RTC 1564
39.3.13 RTC synchronization 1565
39.3.14 RTC reference clock detection 1565
39.3.15 RTC smooth digital calibration 1566
39.3.16 Timestamp function 1568
39.3.17 Calibration clock output 1569
39.3.18 Tamper and alarm output 1569
39.4 RTC low-power modes 1570
39.5 RTC interrupts 1570
39.6 RTC registers 1571
39.6.1 RTC time register (RTC_TR) 1571
39.6.2 RTC date register (RTC_DR) 1572
39.6.3 RTC subsecond register (RTC_SSR) 1573
39.6.4 RTC initialization control and status register (RTC_ICSR) 1573
39.6.5 RTC prescaler register (RTC_PRER) 1575
39.6.6 RTC wake-up timer register (RTC_WUTR) 1576
39.6.7 RTC control register (RTC_CR) 1576
39.6.8 RTC privilege mode control register (RTC_PRIVCFGR) 1580
39.6.9 RTC write protection register (RTC_WPR) 1582
39.6.10 RTC calibration register (RTC_CALR) 1582
39.6.11 RTC shift control register (RTC_SHIFTR) 1584
39.6.12 RTC timestamp time register (RTC_TSTR) 1585
39.6.13 RTC timestamp date register (RTC_TSDR) 1586
39.6.14 RTC timestamp subsecond register (RTC_TSSSR) 1586
39.6.15 RTC alarm A register (RTC_ALRMAR) 1587
39.6.16 RTC alarm A subsecond register (RTC_ALRMASSR) 1588
39.6.17 RTC alarm B register (RTC_ALRMBR) 1589
39.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR) 1590
39.6.19 RTC status register (RTC_SR) 1591
39.6.20 RTC masked interrupt status register (RTC_MISR) 1592
39.6.21 RTC status clear register (RTC_SCR) 1593
39.6.22 RTC alarm A binary mode register (RTC_ALRABINR) 1594
39.6.23 RTC alarm B binary mode register (RTC_ALRBBINR) 1594
39.6.24 RTC register map 1595

40 Tamper and backup registers (TAMP) 1597

40.1 TAMP introduction 1597

40.2 TAMP main features . . . 1597
40.3 TAMP functional description . . . 1598
40.3.1 TAMP block diagram . . . 1598
40.3.2 TAMP pins and internal signals . . . 1599
40.3.3 GPIOs controlled by the RTC and TAMP . . . 1600
40.3.4 TAMP register write protection . . . 1600
40.3.5 Backup registers protection zones . . . 1601
40.3.6 TAMP privilege protection modes . . . 1601
40.3.7 Tamper detection . . . 1602
40.3.8 TAMP backup registers and other device secrets erase . . . 1602
40.3.9 Tamper detection configuration and initialization . . . 1603
40.4 TAMP low-power modes . . . 1606
40.5 TAMP interrupts . . . 1606
40.6 TAMP registers . . . 1606
40.6.1 TAMP control register 1 (TAMP_CR1) . . . 1606
40.6.2 TAMP control register 2 (TAMP_CR2) . . . 1608
40.6.3 TAMP control register 3 (TAMP_CR3) . . . 1610
40.6.4 TAMP filter control register (TAMP_FLTCR) . . . 1611
40.6.5 TAMP configuration register (TAMP_CFGR) . . . 1612
40.6.6 TAMP privilege configuration register (TAMP_PRIVCFGR) . . . 1613
40.6.7 TAMP interrupt enable register (TAMP_IER) . . . 1614
40.6.8 TAMP status register (TAMP_SR) . . . 1616
40.6.9 TAMP masked interrupt status register (TAMP_MISR) . . . 1617
40.6.10 TAMP status clear register (TAMP_SCR) . . . 1619
40.6.11 TAMP option register (TAMP_OR) . . . 1620
40.6.12 TAMP backup x register (TAMP_BKPxR) . . . 1621
40.6.13 TAMP register map . . . 1622

41 Inter-integrated circuit interface (I2C) . . . 1624
41.1 I2C introduction . . . 1624
41.2 I2C main features . . . 1624
41.3 I2C implementation . . . 1625
41.4 I2C functional description . . . 1625
41.4.1 I2C block diagram . . . 1626
41.4.2 I2C pins and internal signals . . . 1626
41.4.3 I2C clock requirements . . . 1627

41.4.4 I2C mode selection . . . 1627
41.4.5 I2C initialization . . . 1628
41.4.6 I2C reset . . . 1632
41.4.7 I2C data transfer . . . 1633
41.4.8 I2C target mode . . . 1635
41.4.9 I2C controller mode . . . 1644
41.4.10 I2C_TIMINGR register configuration examples . . . 1655
41.4.11 SMBus specific features . . . 1657
41.4.12 SMBus initialization . . . 1659
41.4.13 SMBus I2C_TIMEOUTR register configuration examples . . . 1661
41.4.14 SMBus target mode . . . 1662
41.4.15 SMBus controller mode . . . 1665
41.4.16 Wake-up from Stop mode on address match . . . 1668
41.4.17 Error conditions . . . 1669
41.5 I2C in low-power modes . . . 1671
41.6 I2C interrupts . . . 1671
41.7 I2C DMA requests . . . 1672
41.7.1 Transmission using DMA . . . 1672
41.7.2 Reception using DMA . . . 1672
41.8 I2C debug modes . . . 1672
41.9 I2C registers . . . 1673
41.9.1 I2C control register 1 (I2C_CR1) . . . 1673
41.9.2 I2C control register 2 (I2C_CR2) . . . 1676
41.9.3 I2C own address 1 register (I2C_OAR1) . . . 1678
41.9.4 I2C own address 2 register (I2C_OAR2) . . . 1678
41.9.5 I2C timing register (I2C_TIMINGR) . . . 1679
41.9.6 I2C timeout register (I2C_TIMEOUTR) . . . 1680
41.9.7 I2C interrupt and status register (I2C_ISR) . . . 1681
41.9.8 I2C interrupt clear register (I2C_ICR) . . . 1684
41.9.9 I2C PEC register (I2C_PECR) . . . 1685
41.9.10 I2C receive data register (I2C_RXDR) . . . 1685
41.9.11 I2C transmit data register (I2C_TXDR) . . . 1686
41.9.12 I2C register map . . . 1687

42 Improved inter-integrated circuit (I3C) . . . 1688

42.1 I3C introduction . . . 1688

42.2 I3C main features 1688
42.3 I3C implementation 1690
42.3.1 I3C instantiation 1690
42.3.2 I3C wake-up from low-power mode(s) 1690
42.3.3 I3C FIFOs 1690
42.3.4 I3C triggers 1690
42.3.5 I3C interrupt(s) 1690
42.3.6 I3C MIPI® support 1691
42.4 I3C block diagram 1692
42.5 I3C pins and internal signals 1692
42.6 I3C reset and clocks 1693
42.6.1 I3C reset 1693
42.6.2 I3C clocks and requirements 1693
42.7 I3C peripheral state and programming 1695
42.7.1 I3C peripheral state 1695
42.7.2 I3C controller state and programming sequence 1696
42.7.3 I3C target state and programming sequence 1701
42.8 I3C registers and programming 1705
42.8.1 I3C register set, as controller/target 1705
42.8.2 I3C registers and fields use versus peripheral state, as controller . . 1706
42.8.3 I3C registers and fields usage versus peripheral state, as target . . . 1709
42.9 I3C bus transfers and programming 1711
42.9.1 I3C command set (CCCs), as controller/target 1711
42.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller 1715
42.9.3 I3C broadcast ENTDAA CCC transfer, as controller 1717
42.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller 1717
42.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target 1719
42.9.6 I3C broadcast ENTDAA CCC transfer, as target 1721
42.9.7 I3C broadcast DEFTGTS CCC transfer, as target 1722
42.9.8 I3C broadcast DEFGRPA CCC transfer, as target 1723
42.9.9 I3C direct GETSTATUS CCC response, as target 1724
42.9.10 I3C private read/write transfer, as controller 1725
42.9.11 I3C private read/write transfer, as target 1726
42.9.12 Legacy I2C read/write transfer, as controller 1727
42.9.13 I3C IBI transfer, as controller/target 1728

42.9.14 I3C hot-join request transfer, as controller/target 1729
42.9.15 I3C controller-role request transfer, as controller/target 1730
42.10 I3C FIFOs management, as controller 1731
42.10.1 C-FIFO management, as controller 1731
42.10.2 TX-FIFO management, as controller 1732
42.10.3 RX-FIFO management, as controller 1735
42.10.4 S-FIFO management, as controller 1737
42.11 I3C FIFOs management, as target 1739
42.11.1 RX-FIFO management, as target 1739
42.11.2 TX-FIFO management, as target 1740
42.12 I3C error management 1743
42.12.1 Controller error management 1743
42.12.2 Target error management 1745
42.13 I3C wake-up from low-power mode(s) 1746
42.13.1 Wake-up from Stop 1746
42.14 I3C in low-power modes 1749
42.15 I3C interrupts 1750
42.16 I3C registers 1750
42.16.1 I3C message control register (I3C_CR) 1750
42.16.2 I3C message control register [alternate] (I3C_CR) 1752
42.16.3 I3C configuration register (I3C_CFGR) 1754
42.16.4 I3C receive data byte register (I3C_RDR) 1759
42.16.5 I3C receive data word register (I3C_RDWR) 1760
42.16.6 I3C transmit data byte register (I3C_TDR) 1761
42.16.7 I3C transmit data word register (I3C_TDWR) 1762
42.16.8 I3C IBI payload data register (I3C_IBIDR) 1763
42.16.9 I3C target transmit configuration register (I3C_TGTTDR) 1764
42.16.10 I3C status register (I3C_SR) 1765
42.16.11 I3C status error register (I3C_SER) 1766
42.16.12 I3C received message register (I3C_RMR) 1768
42.16.13 I3C event register (I3C_EVR) 1769
42.16.14 I3C interrupt enable register (I3C_IER) 1773
42.16.15 I3C clear event register (I3C_CEVR) 1775
42.16.16 I3C masked interrupt status register (I3C_MISR) 1777
42.16.17 I3C own device characteristics register (I3C_DEVR0) 1778
42.16.18 I3C device x characteristics register (I3C_DEVRx) 1780

42.16.19 I3C maximum read length register (I3C_MAXRLR) 1782
42.16.20 I3C maximum write length register (I3C_MAXWLR) 1783
42.16.21 I3C timing register 0 (I3C_TIMINGR0) 1784
42.16.22 I3C timing register 1 (I3C_TIMINGR1) 1785
42.16.23 I3C timing register 2 (I3C_TIMINGR2) 1787
42.16.24 I3C bus characteristics register (I3C_BCR) 1788
42.16.25 I3C device characteristics register (I3C_DCR) 1789
42.16.26 I3C get capability register (I3C_GETCAPR) 1790
42.16.27 I3C controller-role capability register (I3C_CRCAPR) 1791
42.16.28 I3C get max data speed register (I3C_GETMXDSR) 1792
42.16.29 I3C extended provisioned ID register (I3C_EPIDR) 1794
42.16.30 I3C register map 1795

43 Universal synchronous/asynchronous receiver transmitter (USART/UART) 1798

43.1 USART introduction 1798
43.2 USART main features 1798
43.3 USART extended features 1799
43.4 USART implementation 1799
43.5 USART functional description 1801
43.5.1 USART block diagram 1801
43.5.2 USART pins and internal signals 1801
43.5.3 USART clocks 1803
43.5.4 USART character description 1803
43.5.5 USART FIFOs and thresholds 1806
43.5.6 USART transmitter 1806
43.5.7 USART receiver 1809
43.5.8 USART baud rate generation 1816
43.5.9 Tolerance of the USART receiver to clock deviation 1818
43.5.10 USART auto baud rate detection 1819
43.5.11 USART multiprocessor communication 1821
43.5.12 USART Modbus communication 1823
43.5.13 USART parity control 1824
43.5.14 USART LIN (local interconnection network) mode 1825
43.5.15 USART synchronous mode 1827
43.5.16 USART single-wire half-duplex communication 1831
43.5.17 USART receiver timeout 1831

43.5.18 USART smartcard mode 1832
43.5.19 USART IrDA SIR ENDEC block 1836
43.5.20 Continuous communication using USART and DMA 1839
43.5.21 RS232 hardware flow control and RS485 driver enable 1841
43.5.22 USART low-power management 1844
43.6 USART in low-power modes 1847
43.7 USART interrupts 1847
43.8 USART registers 1850
43.8.1 USART control register 1 (USART_CR1) 1850
43.8.2 USART control register 1 [alternate] (USART_CR1) 1854
43.8.3 USART control register 2 (USART_CR2) 1857
43.8.4 USART control register 3 (USART_CR3) 1861
43.8.5 USART control register 3 [alternate] (USART_CR3) 1865
43.8.6 USART baud rate register (USART_BRR) 1868
43.8.7 USART guard time and prescaler register (USART_GTPR) 1869
43.8.8 USART receiver timeout register (USART_RTOR) 1870
43.8.9 USART request register (USART_RQR) 1871
43.8.10 USART interrupt and status register (USART_ISR) 1872
43.8.11 USART interrupt and status register [alternate] (USART_ISR) 1878
43.8.12 USART interrupt flag clear register (USART_ICR) 1883
43.8.13 USART receive data register (USART_RDR) 1884
43.8.14 USART transmit data register (USART_TDR) 1885
43.8.15 USART prescaler register (USART_PRESC) 1885
43.8.16 USART register map 1886

44 Low-power universal asynchronous receiver transmitter (LPUART) 1888

44.1 LPUART introduction 1888
44.2 LPUART main features 1888
44.3 LPUART implementation 1889
44.4 LPUART functional description 1891
44.4.1 LPUART block diagram 1891
44.4.2 LPUART pins and internal signals 1892
44.4.3 LPUART clocks 1893
44.4.4 LPUART character description 1893
44.4.5 LPUART FIFOs and thresholds 1895

44.4.6 LPUART transmitter . . . 1895
44.4.7 LPUART receiver . . . 1899
44.4.8 LPUART baud rate generation . . . 1903
44.4.9 Tolerance of the LPUART receiver to clock deviation . . . 1904
44.4.10 LPUART multiprocessor communication . . . 1905
44.4.11 LPUART parity control . . . 1907
44.4.12 LPUART single-wire half-duplex communication . . . 1908
44.4.13 Continuous communication using DMA and LPUART . . . 1908
44.4.14 RS232 hardware flow control and RS485 driver enable . . . 1911
44.4.15 LPUART low-power management . . . 1913
44.5 LPUART in low-power modes . . . 1916
44.6 LPUART interrupts . . . 1917
44.7 LPUART registers . . . 1918
44.7.1 LPUART control register 1 (LPUART_CR1) . . . 1918
44.7.2 LPUART control register 1 [alternate] (LPUART_CR1) . . . 1921
44.7.3 LPUART control register 2 (LPUART_CR2) . . . 1924
44.7.4 LPUART control register 3 (LPUART_CR3) . . . 1926
44.7.5 LPUART control register 3 [alternate] (LPUART_CR3) . . . 1929
44.7.6 LPUART baud rate register (LPUART_BRR) . . . 1931
44.7.7 LPUART request register (LPUART_RQR) . . . 1931
44.7.8 LPUART interrupt and status register (LPUART_ISR) . . . 1932
44.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 1937
44.7.10 LPUART interrupt flag clear register (LPUART_ICR) . . . 1940
44.7.11 LPUART receive data register (LPUART_RDR) . . . 1941
44.7.12 LPUART transmit data register (LPUART_TDR) . . . 1941
44.7.13 LPUART prescaler register (LPUART_PRESC) . . . 1942
44.7.14 LPUART register map . . . 1943

45 Serial peripheral interface (SPI) . . . 1945
45.1 SPI introduction . . . 1945
45.2 SPI main features . . . 1945
45.3 SPI implementation . . . 1946
45.4 SPI functional description . . . 1947
45.4.1 SPI block diagram . . . 1947
45.4.2 SPI pins and internal signals . . . 1948
45.4.3 SPI communication general aspects . . . 1949

45.4.4 Communications between one master and one slave 1949
45.4.5 Standard multislave communication 1952
45.4.6 Multimaster communication 1955
45.4.7 Slave select (NSS pin) management 1956
45.4.8 Ready pin (RDY) management 1960
45.4.9 Communication formats 1960
45.4.10 Configuring the SPI 1962
45.4.11 Enabling the SPI 1963
45.4.12 SPI data transmission and reception procedures 1964
45.4.13 Disabling the SPI 1968
45.4.14 Communication using DMA (direct memory addressing) 1969
45.5 SPI specific modes and control 1971
45.5.1 TI mode 1971
45.5.2 SPI error flags 1971
45.5.3 CRC computation 1975
45.6 SPI in low-power modes 1976
45.7 SPI interrupts 1976
45.8 I2S main features 1978
45.9 I2S functional description 1978
45.9.1 I2S general description 1978
45.9.2 Pin sharing with SPI function 1979
45.9.3 Bitfields usable in I2S/PCM mode 1979
45.9.4 Slave and master modes 1980
45.9.5 Supported audio protocols 1980
45.9.6 Additional serial interface flexibility 1986
45.9.7 Startup sequence 1988
45.9.8 Stop sequence 1990
45.9.9 Clock generator 1991
45.9.10 Internal FIFOs 1993
45.9.11 FIFO status flags 1994
45.9.12 Handling of underrun situation 1994
45.9.13 Handling of overrun situation 1995
45.9.14 Frame error detection 1996
45.9.15 DMA interface 1998
45.9.16 Programing examples 1998
45.10 I2S interrupts 2000

45.11 SPI/I2S registers 2001
45.11.1 SPI/I2S control register 1 (SPI_CR1) 2001
45.11.2 SPI/I2S control register 2 (SPI_CR2) 2003
45.11.3 SPI/I2S configuration register 1 (SPI_CFG1) 2003
45.11.4 SPI/I2S configuration register 2 (SPI_CFG2) 2007
45.11.5 SPI/I2S interrupt enable register (SPI_IER) 2009
45.11.6 SPI/I2S status register (SPI_SR) 2010
45.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) 2013
45.11.8 SPI/I2S transmit data register (SPI_TXDR) 2014
45.11.9 SPI/I2S receive data register (SPI_RXDR) 2014
45.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) 2015
45.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) 2015
45.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) 2016
45.11.13 SPI/I2S underrun data register (SPI_UDRDR) 2017
45.11.14 SPI/I2S configuration register (SPI_I2SCFGR) 2017
45.11.15 SPI/I2S register map 2019

46 FD controller area network (FDCAN) 2021
46.1 FDCAN introduction 2021
46.2 FDCAN main features 2023
46.3 FDCAN implementation 2023
46.4 FDCAN functional description 2024
46.4.1 FDCAN block diagram 2024
46.4.2 FDCAN pins and internal signals 2025
46.4.3 Bit timing 2026
46.4.4 Operating modes 2027
46.4.5 Error management 2036
46.4.6 Message RAM 2037
46.4.7 FIFO acknowledge handling 2046
46.4.8 FDCAN Rx FIFO element 2046
46.4.9 FDCAN Tx buffer element 2048
46.4.10 FDCAN Tx event FIFO element 2050
46.4.11 FDCAN standard message ID filter element 2051
46.4.12 FDCAN extended message ID filter element 2052
46.5 FDCAN registers 2054
46.5.1 FDCAN core release register (FDCAN_CREL) 2054

46.5.2 FDCAN endian register (FDCAN_ENDN) 2054
46.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 2054
46.5.4 FDCAN test register (FDCAN_TEST) 2055
46.5.5 FDCAN RAM watchdog register (FDCAN_RWD) 2056
46.5.6 FDCAN CC control register (FDCAN_CCCR) 2057
46.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 2058
46.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 2060
46.5.9 FDCAN timestamp counter value register (FDCAN_TSCV) 2060
46.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 2061
46.5.11 FDCAN timeout counter value register (FDCAN_TOCV) 2062
46.5.12 FDCAN error counter register (FDCAN_ECR) 2062
46.5.13 FDCAN protocol status register (FDCAN_PSR) 2063
46.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 2065
46.5.15 FDCAN interrupt register (FDCAN_IR) 2065
46.5.16 FDCAN interrupt enable register (FDCAN_IE) 2068
46.5.17 FDCAN interrupt line select register (FDCAN_ILS) 2070
46.5.18 FDCAN interrupt line enable register (FDCAN_ILE) 2071
46.5.19 FDCAN global filter configuration register (FDCAN_RXGFC) 2071
46.5.20 FDCAN extended ID and mask register (FDCAN_XIDAM) 2073
46.5.21 FDCAN high-priority message status register (FDCAN_HPMS) 2073
46.5.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 2074
46.5.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 2075
46.5.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 2075
46.5.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 2076
46.5.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) 2076
46.5.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 2077
46.5.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 2077
46.5.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) 2078
46.5.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 2079
46.5.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 2079
46.5.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 2080
46.5.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 2080
46.5.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 2081
46.5.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 2081
46.5.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 2082
46.5.37 FDCAN CFG clock divider register (FDCAN_CKDIV) 2082

46.5.38 FDCAN register map 2083

47 Universal serial bus full-speed host/device interface (USB) 2087

47.1 USB introduction 2087
47.2 USB main features 2087
47.3 USB implementation 2087
47.4 USB functional description 2088
47.4.1 USB block diagram 2088
47.4.2 USB pins and internal signals 2088
47.4.3 USB reset and clocks 2089
47.4.4 General description and Device mode functionality 2089
47.4.5 Description of USB blocks used in both Device and Host modes . . . 2090
47.4.6 Description of host frame scheduler (HFS) specific to Host mode . . 2091
47.5 Programming considerations for Device and Host modes 2092
47.5.1 Generic USB Device programming 2092
47.5.2 System and power-on reset 2092
47.5.3 Double-buffered endpoints and usage in Device mode 2099
47.5.4 Double buffered channels: usage in Host mode 2101
47.5.5 Isochronous transfers in Device mode 2102
47.5.6 Isochronous transfers in Host mode 2103
47.5.7 Suspend/resume events 2104
47.6 USB registers 2107
47.6.1 USB control register (USB_CNTR) 2108
47.6.2 USB interrupt status register (USB_ISTR) 2111
47.6.3 USB frame number register (USB_FNR) 2115
47.6.4 USB Device address (USB_DADDR) 2115
47.6.5 USB LPM control and status register (USB_LPMCSR) 2116
47.6.6 USB battery charging detector (USB_BCDR) 2117
47.6.7 USB endpoint/channel n register (USB_CHEPnR) 2118
47.7 USBSRAM registers 2127
47.7.1 Channel/endpoint transmit buffer descriptor n (USBSRAM_CHEP_TXRXBD_n) 2128
47.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USBSRAM_CHEP_TXRXBD_n) 2128
47.7.3 Channel/endpoint receive buffer descriptor n (USBSRAM_CHEP_RXTXBD_n) 2130

47.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USBSRAM_CHEP_RXTXBD_n) 2131
47.7.5 USBSRAM register map 2132

48 Ethernet (ETH): media access control (MAC) with DMA controller 2133

48.1 Ethernet introduction 2133
48.2 Ethernet main features 2133
48.2.1 Standard compliance 2133
48.2.2 MAC features 2133
48.2.3 Transaction layer (MTL) features 2135
48.2.4 DMA block features 2136
48.2.5 Bus interface features 2136
48.3 Ethernet pins and internal signals 2137
48.4 Ethernet architecture 2139
48.4.1 DMA controller 2140
48.4.2 MTL 2148
48.4.3 MAC 2148
48.5 Ethernet functional description: MAC 2153
48.5.1 Double VLAN processing 2153
48.5.2 Source address and VLAN insertion, replacement, or deletion 2154
48.5.3 Packet filtering 2156
48.5.4 IEEE 1588 timestamp support 2163
48.5.5 Checksum offload engine 2187
48.5.6 TCP segmentation offload 2193
48.5.7 IPv4 ARP offload 2199
48.5.8 Loopback 2200
48.5.9 Flow control 2201
48.5.10 MAC management counters 2203
48.5.11 Interrupts generated by the MAC 2205
48.5.12 MAC and MMC register descriptions 2205
48.6 Ethernet functional description: PHY interfaces 2206
48.6.1 Station management agent (SMA) 2206
48.6.2 Media independent interface (MII) 2213
48.6.3 Reduced media independent interface (RMII) 2214
48.7 Ethernet low-power modes 2217
48.7.1 Low-power management 2217

48.7.2 Energy-efficient Ethernet (EEE) . . . 2223
48.8 Ethernet interrupts . . . 2229
48.8.1 DMA interrupts . . . 2229
48.8.2 MTL interrupts . . . 2232
48.8.3 MAC interrupts . . . 2232
48.9 Ethernet programming model . . . 2233
48.9.1 DMA initialization . . . 2234
48.9.2 MTL initialization . . . 2235
48.9.3 MAC initialization . . . 2235
48.9.4 Performing normal receive and transmit operation . . . 2236
48.9.5 Stopping and starting transmission . . . 2237
48.9.6 Programming guidelines for switching to new descriptor list in RxDMA . . . 2237
48.9.7 Programming guidelines for switching the AHB clock frequency . . . 2237
48.9.8 Programming guidelines for MII link state transitions . . . 2238
48.9.9 Programming guidelines for IEEE 1588 timestamping . . . 2239
48.9.10 Programming guidelines for PTP offload feature . . . 2240
48.9.11 Programming guidelines for Energy Efficient Ethernet (EEE) . . . 2244
48.9.12 Programming guidelines for flexible pulse-per-second (PPS) output 2246
48.9.13 Programming guidelines for IEEE 1588 auxiliary snapshot . . . 2248
48.9.14 Programming guidelines for TSO . . . 2248
48.9.15 Programming guidelines to perform VLAN filtering on the receiver . . 2249
48.9.16 Programming guidelines for receive idle and receive interrupt watchdog timers . . . 2250
48.10 Descriptors . . . 2251
48.10.1 Descriptor overview . . . 2251
48.10.2 Descriptor structure . . . 2251
48.10.3 Transmit descriptor . . . 2255
48.10.4 Receive descriptor . . . 2267
48.11 Ethernet registers . . . 2280
48.11.1 Ethernet register maps . . . 2280
48.11.2 Ethernet DMA registers . . . 2280
48.11.3 Ethernet MTL registers . . . 2302
48.11.4 Ethernet MAC and MMC registers . . . 2311

49 Debug support (DBG) . . . 2413

49.1 DBG introduction . . . 2413

49.2 DBG functional description 2414
49.2.1 DBG block diagram 2414
49.2.2 DBG pins and internal signals 2414
49.2.3 DBG reset and clocks 2415
49.2.4 DBG power domains 2415
49.2.5 Debug and low-power modes 2415
49.2.6 Security 2416
49.2.7 OEMKEY and BSKEY password management 2417
49.3 Serial-wire and JTAG debug port (SWJ-DP) 2419
49.3.1 JTAG debug port 2420
49.3.2 Serial-wire debug port 2422
49.3.3 Debug port registers 2423
49.4 Access ports 2431
49.4.1 Access port registers 2431
49.5 ROM tables 2438
49.5.1 System ROM table registers 2441
49.5.2 MCU ROM table registers 2445
49.5.3 Processor ROM table registers 2450
49.6 Data watchpoint and trace unit (DWT) 2455
49.6.1 DWT registers 2456
49.7 Instrumentation trace macrocell (ITM) 2472
49.7.1 ITM registers 2472
49.8 Breakpoint unit (BPU) 2481
49.8.1 BPU registers 2481
49.9 Embedded trace macrocell (ETM) 2488
49.9.1 ETM registers 2488
49.10 Trace port interface unit (TPIU) 2514
49.10.1 TPIU registers 2515
49.11 Cross-trigger interface (CTI) 2526
49.11.1 CTI registers 2527
49.12 Microcontroller debug unit (DBGMCU) 2539
49.12.1 Device ID 2540
49.12.2 Low-power mode emulation 2540
49.12.3 Peripheral clock freeze 2540
49.12.4 DBGMCU registers 2541
49.13 References 2557

50 Device electronic signature 2558
50.1 Unique device ID register (96 bits) 2558
50.2 Flash size data register 2559
50.3 Package data register 2559

51 Important security notice 2561

52 Revision history 2562

List of tables

Table 1. Implementation of masters on STM32C5 series . . . 87
Table 2. Implementation of slaves on STM32C5 series . . . 87
Table 3. Memory map and peripheral register boundary addresses . . . 94
Table 4. SRAM sizes . . . 98
Table 5. Internal tampers in TAMP . . . 102
Table 6. Effect of low-power modes on TAMP . . . 103
Table 7. Accelerated cryptographic operations . . . 105
Table 8. Main product life cycle transitions . . . 107
Table 9. Typical product life cycle phases . . . 108
Table 10. OEM RDP unlocking methods . . . 109
Table 11. BS RDP locking methods . . . 109
Table 12. Summary of protected assets and protection mechanism . . . 110
Table 13. STM32C5 boot modes . . . 112
Table 14. SRAM density (Kbytes) . . . 114
Table 15. Internal SRAMs features . . . 115
Table 16. SRAM with ECC memory map . . . 116
Table 17. Effect of low-power modes on RAMCFG . . . 117
Table 18. RAMCFG interrupt requests . . . 117
Table 19. RAMCFG register map and reset values . . . 124
Table 20. FLASH recommended number of wait states and programming delay . . . 132
Table 21. OTP first page organization . . . 139
Table 22. Flash memory OTP organization . . . 140
Table 23. Read-only public data organization . . . 141
Table 24. Memory map and the swapping option (STM32C53x/542 devices) . . . 143
Table 25. Memory map and the swapping option (STM32C55x/562 devices) . . . 144
Table 26. Memory map and the swapping option (STM32C59x/5A3 devices) . . . 145
Table 27. Recommended reactions to FLASH_OPSR contents . . . 148
Table 28. Option-byte organization . . . 151
Table 29. Write-protection management . . . 153
Table 30. Specific OB modifying rules overview . . . 154
Table 31. OBs modifiable in L2 product . . . 154
Table 32. Secure hide protection . . . 157
Table 33. HDP protections summary . . . 157
Table 34. FLASH interface register protection summary . . . 158
Table 35. RDP levels description . . . 160
Table 36. RDP levels transitions . . . 161
Table 37. OTP/RO access constraints . . . 163
Table 38. C defined macro for HDP services . . . 163
Table 39. EXITHDPlib interface function list . . . 164
Table 40. Effect of low-power modes on the FLASH . . . 165
Table 41. Locating ECC failure . . . 169
Table 42. FLASH interrupt request . . . 172
Table 43. FLASH register map and reset values . . . 202
Table 44. ICACHE features for STM32C5 devices . . . 208
Table 45. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 210
Table 46. TAG memory dimensioning parameters for direct-mapped cache mode . . . 211
Table 47. ICACHE cacheability for AHB transaction . . . 213

Table 48. Memory configurations . . . 213
Table 49. ICACHE remap region size, base address, and remap address . . . 214
Table 50. ICACHE interrupts . . . 218
Table 51. ICACHE register map and reset values . . . 222
Table 52. PWR input/output pins . . . 224
Table 53. PWR internal input/output signals . . . 225
Table 54. Low-power mode summary . . . 230
Table 55. Functionalities depending on the working mode . . . 231
Table 56. Sleep mode . . . 235
Table 57. Memory power down block selection . . . 235
Table 58. Stop 0 mode . . . 237
Table 59. Stop 1 mode . . . 239
Table 60. Standby mode . . . 241
Table 61. Power mode output states versus MCU power modes . . . 242
Table 62. PWR interrupt requests . . . 243
Table 63. PWR register map and reset values . . . 249
Table 64. RCC input/output signals connected to package pins or balls . . . 251
Table 65. Reset source identification (RCC_RSR) . . . 253
Table 66. PSI frequency settings . . . 258
Table 67. PSIK and HSIK output frequency . . . 260
Table 68. HSIKERON and PSIKERON behavior . . . 263
Table 69. Kernel clock distribution overview . . . 265
Table 70. Interrupt sources and control . . . 273
Table 71. RCC register map and reset values . . . 322
Table 72. CRS features . . . 327
Table 73. CRS internal input/output signals . . . 328
Table 74. CRS interconnection . . . 329
Table 75. Effect of low-power modes on CRS . . . 332
Table 76. Interrupt control bits . . . 332
Table 77. CRS register map and reset values . . . 337
Table 78. Port bit configuration . . . 340
Table 79. GPIO register map and reset values . . . 355
Table 80. SBS internal input/output signals . . . 358
Table 81. HDPL encoded values . . . 360
Table 82. NEXT-HDPL logic . . . 361
Table 83. SBS register map and reset values . . . 369
Table 84. Peripherals interconnect matrix . . . 372
Table 85. peripheral interconnect matrix details . . . 374
Table 86. LPDMA1/2 channels implementation . . . 383
Table 87. LPDMA1 channels implementation . . . 383
Table 88. LPDMA2 channels implementation . . . 384
Table 89. LPDMA1/2 autonomous mode and wake-up in low-power modes . . . 384
Table 90. Programmed LPDMA1/2 request . . . 384
Table 91. Programmed LPDMA1/2 request as a block request . . . 387
Table 92. LPDMA1/2 channel with peripheral early termination . . . 387
Table 93. Programmed LPDMA1/2 request with peripheral early termination . . . 388
Table 94. Programmed LPDMA1/2 trigger . . . 388
Table 95. Programmed LPDMA source/destination single . . . 408
Table 96. Programmed data handling . . . 409
Table 97. Effect of low-power modes on LPDMA . . . 420
Table 98. LPDMA interrupt requests . . . 420
Table 99. LPDMA register map and reset values . . . 438

Table 100. STM32C5 vector table . . . 440
Table 101. EXTI signals . . . 446
Table 102. EVG signals . . . 447
Table 103. EXTI line connections . . . 447
Table 104. Masking functionality . . . 451
Table 105. Register protection overview . . . 451
Table 106. EXTI register map sections . . . 452
Table 107. EXTI register map and reset values . . . 470
Table 108. CRC internal input/output signals . . . 473
Table 109. CRC register map and reset values . . . 478
Table 110. CORDIC functions . . . 480
Table 111. Cosine parameters . . . 480
Table 112. Sine parameters . . . 481
Table 113. Phase parameters . . . 481
Table 114. Modulus parameters . . . 482
Table 115. Arctangent parameters . . . 483
Table 116. Hyperbolic cosine parameters . . . 483
Table 117. Hyperbolic sine parameters . . . 484
Table 118. Hyperbolic arctangent parameters . . . 484
Table 119. Natural logarithm parameters . . . 485
Table 120. Natural log scaling factors and corresponding ranges . . . 485
Table 121. Square root parameters . . . 486
Table 122. Square root scaling factors and corresponding ranges . . . 486
Table 123. Precision vs. number of iterations . . . 489
Table 124. CORDIC register map and reset value . . . 496
Table 125. XSPI implementation . . . 498
Table 126. XSPI input/output pins . . . 500
Table 127. XSPI internal signals . . . 500
Table 128. Command/address phase description . . . 508
Table 129. OctaRAM command address bit assignment (based on 64 Mb OctaRAM) . . . 519
Table 130. Address alignment cases . . . 526
Table 131. XSPI interrupt requests . . . 527
Table 132. XSPI register map and reset values . . . 551
Table 133. DLYB internal input/output signals . . . 556
Table 134. Delay block control . . . 557
Table 135. DLYB register map and reset values . . . 560
Table 136. ADC features . . . 563
Table 137. ADC input/output pins . . . 565
Table 138. ADC internal input/output signals . . . 565
Table 139. ADC1/2/3 interconnection . . . 565
Table 140. Configuring the trigger polarity for regular external triggers . . . 585
Table 141. Configuring the trigger polarity for injected external triggers . . . 585
Table 142. TSAR timings depending on resolution . . . 588
Table 143. Offset computation versus data resolution . . . 592
Table 144. 12-bit data formats . . . 594
Table 145. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0) . . . 595
Table 146. Analog window watchdog features . . . 603
Table 147. Analog watchdog 1 channel selection . . . 604
Table 148. Analog watchdog 1, 2, 3 comparison . . . 605
Table 149. Maximum output results versus N and M . . . 610
Table 150. Oversampler operating mode summary . . . 614

Table 151. DELAY bits versus ADC resolution . . . 620
Table 152. ADC interrupts . . . 633
Table 153. ADC register map and reset values for each ADC . . . 667
Table 154. ADC register map and reset values (master and slave ADC common registers) . . . 670
Table 155. DAC features . . . 673
Table 156. DAC input/output pins . . . 675
Table 157. DAC internal input/output signals . . . 675
Table 158. DAC interconnection . . . 676
Table 159. Data format (case of 12-bit data) . . . 678
Table 160. HFSEL description . . . 679
Table 161. Sample and refresh timings . . . 684
Table 162. Channel output modes summary . . . 686
Table 163. Effect of low-power modes on DAC . . . 692
Table 164. DAC interrupts . . . 692
Table 165. DAC register map and reset values . . . 709
Table 166. COMP implementation for STM32C55x/562 and STM32C59x/5A3 . . . 712
Table 167. COMP implementation non-inverting input assignment for STM32C55x/562 and STM32C59x/5A3 . . . 712
Table 168. COMP implementation for STM32C53x/542 . . . 712
Table 169. COMP implementation non-inverting input assignment for STM32C53x/542 . . . 712
Table 170. Window paired comparators . . . 713
Table 171. COMP input/output internal signals . . . 714
Table 172. COMP inverting input assignment . . . 714
Table 173. COMP blanking source assignment . . . 714
Table 174. Comparator behavior in low-power modes . . . 717
Table 175. COMP wake-up interrupt through EXTI control bits . . . 718
Table 176. COMP NVIC interrupt through EXTI control bits . . . 718
Table 177. COMP interrupt through NVIC of the CPU control bits . . . 719
Table 178. COMP register map and reset values . . . 725
Table 179. OPAMP pin and signal assignment per instance . . . 727
Table 180. OPAMP pins . . . 728
Table 181. OPAMP internal input/output signals . . . 729
Table 182. Calibration loops . . . 738
Table 183. Timer-controlled input selection . . . 739
Table 184. Timer-controlled programmable gain selection . . . 740
Table 185. Effect of low-power modes on OPAMP . . . 740
Table 186. OPAMP register map and reset values . . . 746
Table 187. Peripherals connected by CCB interconnect . . . 748
Table 188. CCB internal input/output signals . . . 749
Table 189. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0) . . . 756
Table 190. Blob use steps for ECDSA signature (CCOP = 0b11000011) . . . 759
Table 191. Blob use steps for signing public key computation (CCOP = 0b10000001) . . . 761
Table 192. Blob creation steps for scalar multiplication (CCOP = 0b100000<i>0) . . . 763
Table 193. Blob use steps for scalar multiplication (CCOP = 0b10000001) . . . 764
Table 194. Blob creation steps for modular exponentiation (CCOP = 0b01000100) . . . 765
Table 195. Blob use steps for modular exponentiation (CCOP = 0b01000101) . . . 766
Table 196. CCB register map and reset values . . . 772
Table 197. RNG internal input/output signals . . . 774
Table 198. RNG interrupt requests . . . 783
Table 199. RNG configurations . . . 784

Table 200. Additional health test configurations . . . 784
Table 201. Configuration selection . . . 784
Table 202. RNG register map and reset map . . . 792
Table 203. AES versus SAES features . . . 794
Table 204. AES internal input/output signals . . . 795
Table 205. AES approved symmetric key functions . . . 796
Table 206. Counter mode initialization vector definition . . . 805
Table 207. Initialization of IV registers in GCM mode . . . 808
Table 208. GCM last block definition . . . 808
Table 209. Initialization of IV registers in CCM mode . . . 815
Table 210. AES data swapping example . . . 818
Table 211. Key endianness in AES_KEYRx registers (128/256-bit keys) . . . 820
Table 212. IVI bitfield spread over AES_IVRx registers . . . 821
Table 213. AES interrupt requests . . . 822
Table 214. Processing latency for ECB, CBC and CTR . . . 823
Table 215. Processing latency for GCM and CCM (in clock cycles) . . . 824
Table 216. AES register map and reset values . . . 836
Table 217. AES versus SAES features . . . 840
Table 218. SAES internal input/output signals . . . 841
Table 219. SAES approved symmetric key functions . . . 842
Table 220. Counter mode initialization vector definition . . . 852
Table 221. Initialization of IV registers in GCM mode . . . 855
Table 222. GCM last block definition . . . 855
Table 223. Initialization of IV registers in CCM mode . . . 861
Table 224. AES data swapping example . . . 870
Table 225. Key endianness in SAES_KEYRx registers (128/256-bit keys) . . . 872
Table 226. IVI bitfield spread over SAES_IVRx registers . . . 873
Table 227. SAES interrupt requests . . . 875
Table 228. Processing latency for ECB, CBC and CTR . . . 876
Table 229. Processing latency for GCM and CCM (in SAES kernel clock cycles) . . . 877
Table 230. SAES register map and reset values . . . 890
Table 231. HASH implementation on different devices . . . 894
Table 232. HASH internal input/output signals . . . 895
Table 233. Information on supported hash algorithms . . . 895
Table 234. Hash processor outputs . . . 899
Table 235. Processing time (in clock cycle) . . . 905
Table 236. HASH interrupt requests . . . 906
Table 237. HASH1 register map and reset values . . . 914
Table 238. Internal input/output signals . . . 917
Table 239. PKA integer arithmetic functions list . . . 918
Table 240. PKA prime field (Fp) elliptic curve functions list . . . 919
Table 241. Chaining mode to/from PKA . . . 924
Table 242. Integrity protected parameters list . . . 926
Table 243. Confidentiality protected parameter list . . . 926
Table 244. Example of ‘a’ curve coefficient for ECC Fp scalar . . . 928
Table 245. Montgomery parameter computation . . . 929
Table 246. Modular addition . . . 929
Table 247. Modular subtraction . . . 930
Table 248. Montgomery multiplication . . . 931
Table 249. Modular exponentiation (normal mode) . . . 932
Table 250. Modular exponentiation (fast mode) . . . 932
Table 251. Modular exponentiation (protected mode) . . . 933

Table 252. Modular inversion . . . 933
Table 253. Modular reduction . . . 934
Table 254. Arithmetic addition . . . 934
Table 255. Arithmetic subtraction . . . 934
Table 256. Arithmetic multiplication . . . 935
Table 257. Arithmetic comparison . . . 935
Table 258. CRT exponentiation . . . 936
Table 259. Point on elliptic curve Fp check . . . 937
Table 260. ECC Fp scalar multiplication . . . 937
Table 261. ECDSA sign - Inputs . . . 939
Table 262. ECDSA sign - Outputs . . . 939
Table 263. Extended ECDSA sign - additional outputs . . . 940
Table 264. ECDSA verification - inputs . . . 940
Table 265. ECDSA verification - outputs . . . 941
Table 266. ECC complete addition . . . 941
Table 267. ECC double base ladder . . . 942
Table 268. ECC projective to affine . . . 943
Table 269. Family of supported curves for ECC operations . . . 944
Table 270. Modular exponentiation . . . 945
Table 271. ECC scalar multiplication . . . 945
Table 272. ECDSA signature average computation time . . . 946
Table 273. ECDSA verification average computation times . . . 946
Table 274. ECC double base ladder average computation times . . . 946
Table 275. ECC projective to affine average computation times . . . 946
Table 276. ECC complete addition average computation times . . . 946
Table 277. Point on elliptic curve Fp check average computation times . . . 946
Table 278. Montgomery parameters average computation times . . . 947
Table 279. PKA interrupt requests . . . 947
Table 280. PKA register map and reset values . . . 954
Table 281. TIM input/output pins . . . 957
Table 282. TIM internal input/output signals . . . 957
Table 283. Interconnect to the tim_ti1 input multiplexer . . . 959
Table 284. Interconnect to the tim_ti2 input multiplexer . . . 959
Table 285. Interconnect to the tim_ti3 input multiplexer . . . 959
Table 286. Interconnect to the tim_ti4 input multiplexer . . . 959
Table 287. Internal trigger connection . . . 959
Table 288. Interconnect to the tim_etr input multiplexer . . . 960
Table 289. Interconnect to the tim_bkr_cmp multiplexer . . . 960
Table 290. Timer break2 interconnect . . . 961
Table 291. Interconnect to tim_ocref_clr multiplexer . . . 961
Table 292. System break interconnect . . . 961
Table 293. CCR and ARR register change dithering pattern . . . 995
Table 294. CCR register change dithering pattern in center-aligned PWM mode . . . 996
Table 295. Asymmetric PWM modes . . . 998
Table 296. Combined PWM mode overview . . . 1001
Table 297. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 1012
Table 298. Break protection disarming conditions . . . 1016
Table 299. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 1028
Table 300. Counting direction versus encoder signals and polarity settings . . . 1032
Table 301. DMA request . . . 1057
Table 302. Effect of low-power modes on TIM1/TIM8 . . . 1058
Table 303. Interrupt requests . . . 1059

Table 304. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 1091
Table 305. TIM1/TIM8 register map and reset values . . . 1129
Table 306. Timer instance implementation on STM32C5 . . . 1134
Table 307. STM32C5 general purpose timers . . . 1134
Table 308. TIM input/output pins . . . 1136
Table 309. TIM internal input/output signals . . . 1136
Table 310. Interconnect to the tim_ti1 input multiplexer . . . 1137
Table 311. Interconnect to the tim_ti2 input multiplexer . . . 1137
Table 312. Interconnect to the tim_ti3 input multiplexer . . . 1138
Table 313. Interconnect to the tim_ti4 input multiplexer . . . 1138
Table 314. TIMx internal trigger connection . . . 1138
Table 315. Interconnect to the tim_etr input multiplexer . . . 1139
Table 316. Interconnect to the tim_ocref_clr input multiplexer . . . 1139
Table 317. CCR and ARR register change dithering pattern . . . 1171
Table 318. CCR register change dithering pattern in center-aligned PWM mode . . . 1172
Table 319. Asymmetric PWM modes . . . 1173
Table 320. Combined PWM mode overview . . . 1174
Table 321. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 1185
Table 322. Counting direction versus encoder signals and polarity settings . . . 1190
Table 323. DMA request . . . 1219
Table 324. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5 . . . 1219
Table 325. Interrupt requests . . . 1220
Table 326. Output control bit for standard tim_ocx channels . . . 1244
Table 327. TIM2/TIM3/TIM4/TIM5 register map and reset values . . . 1256
Table 328. TIM internal input/output signals . . . 1260
Table 329. TIM_ARR register change dithering pattern . . . 1270
Table 330. DMA request . . . 1272
Table 331. Effect of low-power modes on TIM6/TIM7 . . . 1272
Table 332. Interrupt request . . . 1272
Table 333. TIM6/TIM7 register map and reset values . . . 1280
Table 334. TIM input/output pins . . . 1282
Table 335. TIM internal input/output signals . . . 1283
Table 336. Interconnect to the tim_ti1 input multiplexer . . . 1283
Table 337. Interconnect to the tim_ti2 input multiplexer . . . 1283
Table 338. TIMx internal trigger connection . . . 1284
Table 339. CCR and ARR register change dithering pattern . . . 1302
Table 340. Effect of low-power modes on TIM12 . . . 1314
Table 341. Interrupt requests . . . 1314
Table 342. Output control bit for standard tim_ocx channels . . . 1328
Table 343. TIM12 register map and reset values . . . 1332
Table 344. Timer instance implementation on STM32C5 . . . 1335
Table 345. TIM input/output pins . . . 1337
Table 346. TIM internal input/output signals . . . 1338
Table 347. Interconnect to the tim_ti1 input multiplexer . . . 1339
Table 348. Interconnect to the tim_ti2 input multiplexer . . . 1339
Table 349. TIMx internal trigger connection . . . 1339
Table 350. Interconnect to the tim_brk_cmp multiplexer . . . 1340
Table 351. Interconnect to the ocref_clr input multiplexer . . . 1340
Table 352. System break interconnect . . . 1340
Table 353. CCR and ARR register change dithering pattern . . . 1371
Table 354. CCR register change dithering pattern in center-aligned PWM mode . . . 1372

Table 355. Asymmetric PWM modes . . . 1373
Table 356. Break protection disarming conditions . . . 1385
Table 357. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 1393
Table 358. Counting direction versus encoder signals and polarity settings . . . 1397
Table 359. DMA request . . . 1410
Table 360. Effect of low-power modes on TIM15/TIM16/TIM17 . . . 1410
Table 361. Interrupt requests . . . 1411
Table 362. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) . . . 1431
Table 363. TIM15 register map and reset values . . . 1449
Table 364. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 1463
Table 365. TIM16/TIM17 register map and reset values . . . 1480
Table 366. STM32C5 LPTIM features . . . 1484
Table 367. LPTIM1 input/output pins . . . 1485
Table 368. LPTIM1 internal signals . . . 1486
Table 369. LPTIM1 external trigger connections . . . 1487
Table 370. LPTIM1 input 1 connections . . . 1487
Table 371. LPTIM1 input 2 connections . . . 1487
Table 372. LPTIM1 input capture 1 connections . . . 1487
Table 373. LPTIM1 input capture 2 connections . . . 1488
Table 374. Prescaler division ratios . . . 1489
Table 375. Encoder counting scenarios . . . 1496
Table 376. Input capture Glitch filter latency (in counter step unit) . . . 1500
Table 377. Effect of low-power modes on the LPTIM . . . 1504
Table 378. Interrupt events . . . 1505
Table 379. LPTIM register map and reset values . . . 1526
Table 380. IWDG features . . . 1528
Table 381. IWDG delays versus actions . . . 1529
Table 382. IWDG internal input/output signals . . . 1530
Table 383. Effect of low power modes on IWDG . . . 1535
Table 384. IWDG interrupt request . . . 1537
Table 385. IWDG register map and reset values . . . 1543
Table 386. WWDG features . . . 1544
Table 387. WWDG internal input/output signals . . . 1545
Table 388. WWDG interrupt requests . . . 1548
Table 389. WWDG register map and reset values . . . 1550
Table 390. RTC input/output pins . . . 1553
Table 391. RTC internal input/output signals . . . 1553
Table 392. RTC interconnection . . . 1554
Table 393. RTC pin configuration . . . 1554
Table 394. RTC_OUT mapping . . . 1556
Table 395. Effect of low-power modes on RTC . . . 1570
Table 396. RTC pins functionality over modes . . . 1570
Table 397. Interrupt requests . . . 1570
Table 398. RTC register map and reset values . . . 1595
Table 399. TAMP input/output pins . . . 1599
Table 400. TAMP internal input/output signals . . . 1599
Table 401. TAMP interconnection . . . 1600
Table 402. Effect of low-power modes on TAMP . . . 1606
Table 403. TAMP pins functionality over modes . . . 1606
Table 404. Interrupt requests . . . 1606

Table 405. TAMP register map and reset values . . . 1622
Table 406. I2C implementation . . . 1625
Table 407. I2C input/output pins . . . 1626
Table 408. I2C internal input/output signals . . . 1627
Table 409. Comparison of analog and digital filters . . . 1629
Table 410. I²C-bus and SMBus specification data setup and hold times . . . 1631
Table 411. I2C configuration . . . 1635
Table 412. I²C-bus and SMBus specification clock timings . . . 1646
Table 413. Timing settings for f_{I2CCLK} of 8 MHz . . . 1656
Table 414. Timing settings for f_{I2CCLK} of 16 MHz . . . 1656
Table 415. SMBus timeout specifications . . . 1658
Table 416. SMBus with PEC configuration . . . 1660
Table 417. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 1661
Table 418. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 1661
Table 419. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 1661
Table 420. Effect of low-power modes to I2C . . . 1671
Table 421. I2C interrupt requests . . . 1671
Table 422. I2C register map and reset values . . . 1687
Table 423. I3C instantiation . . . 1690
Table 424. I3C wake-up . . . 1690
Table 425. I3C FIFOs implementation . . . 1690
Table 426. I3C interrupt(s) . . . 1690
Table 427. I3C peripheral controller/target features versus MIPI v1.1 . . . 1691
Table 428. I3C input/output pins . . . 1692
Table 429. I3C internal input/output signals . . . 1692
Table 430. I3C register usage . . . 1705
Table 431. I3C registers/fields usage versus controller state . . . 1706
Table 432. I3C registers/fields usage versus target state . . . 1709
Table 433. List of supported I3C CCCs, as controller/target . . . 1712
Table 434. I3C controller error management . . . 1743
Table 435. I3C target error management . . . 1745
Table 436. Effect of low-power modes . . . 1749
Table 437. I3C register map and reset values . . . 1795
Table 438. Instance implementation on STM32C5 . . . 1799
Table 439. USART/LPUART features . . . 1799
Table 440. USART/UART input/output pins . . . 1802
Table 441. USART internal input/output signals . . . 1803
Table 442. Noise detection from sampled data . . . 1815
Table 443. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 1819
Table 444. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 1819
Table 445. USART frame formats . . . 1824
Table 446. Effect of low-power modes on the USART . . . 1847
Table 447. USART interrupt requests . . . 1848
Table 448. USART register map and reset values . . . 1886
Table 449. Instance implementation on STM32C5 . . . 1889
Table 450. USART/LPUART features . . . 1889
Table 451. LPUART input/output pins . . . 1892
Table 452. LPUART internal input/output signals . . . 1892
Table 453. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 1903
Table 454. Tolerance of the LPUART receiver . . . 1904
Table 456. Effect of low-power modes on the LPUART . . . 1916
Table 457. LPUART interrupt requests . . . 1917

Table 458. LPUART register map and reset values . . . 1943
Table 459. SPI features . . . 1946
Table 460. SPI/I2S input/output pins . . . 1949
Table 461. SPI internal input/output signals . . . 1949
Table 462. Effect of low-power modes on the SPI . . . 1976
Table 463. SPI wake-up and interrupt requests . . . 1977
Table 464. Bitfields usable in PCM/I2S mode . . . 1979
Table 465. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 1988
Table 466. Serial data line swapping . . . 1988
Table 467. CLKGEN programming examples for usual I2S frequencies . . . 1992
Table 468. I2S interrupt requests . . . 2001
Table 469. SPI register map and reset values . . . 2019
Table 470. FDCAN features . . . 2023
Table 471. CAN subsystem I/O signals . . . 2025
Table 472. CAN subsystem I/O pins . . . 2026
Table 473. DLC coding in FDCAN . . . 2029
Table 474. Possible configurations for frame transmission . . . 2043
Table 475. Rx FIFO element . . . 2046
Table 476. Rx FIFO element description . . . 2046
Table 477. Tx buffer and FIFO element . . . 2048
Table 478. Tx buffer element description . . . 2048
Table 479. Tx event FIFO element . . . 2050
Table 480. Tx event FIFO element description . . . 2050
Table 481. Standard message ID filter element . . . 2051
Table 482. Standard message ID filter element field description . . . 2052
Table 483. Extended message ID filter element . . . 2052
Table 484. Extended message ID filter element field description . . . 2053
Table 485. FDCAN register map and reset values . . . 2083
Table 486. STM32C5 USB implementation . . . 2087
Table 487. USB input/output pins . . . 2088
Table 488. Double-buffering buffer flag definition . . . 2100
Table 489. Bulk double-buffering memory buffers usage (Device mode) . . . 2100
Table 490. Bulk double-buffering memory buffers usage (Host mode) . . . 2102
Table 491. Isochronous memory buffers usage . . . 2103
Table 492. Isochronous memory buffers usage . . . 2104
Table 493. Resume event detection . . . 2106
Table 494. Resume event detection for host . . . 2107
Table 495. Reception status encoding . . . 2125
Table 496. Endpoint/channel type encoding . . . 2125
Table 497. Endpoint/channel kind meaning . . . 2125
Table 498. Transmission status encoding . . . 2125
Table 499. USB register map and reset values . . . 2126
Table 500. Definition of allocated buffer memory . . . 2129
Table 501. USBSRAM register map and reset values . . . 2132
Table 502. Ethernet peripheral pins . . . 2137
Table 503. Internal Ethernet input/output signals . . . 2138
Table 504. Ethernet trigger connections . . . 2138
Table 505. Priority scheme for Tx DMA and Rx DMA . . . 2147
Table 506. Double VLAN processing features in Tx path . . . 2153
Table 507. Double VLAN processing in Rx path . . . 2154
Table 508. VLAN insertion or replacement based on VLTI bit . . . 2155
Table 509. Destination address filtering . . . 2158

Table 510. Source address filtering . . . 2159
Table 511. VLAN match status . . . 2160
Table 512. Ordinary clock: PTP messages for snapshot . . . 2164
Table 513. End-to-end transparent clock: PTP messages for snapshot . . . 2164
Table 514. Peer-to-peer transparent clock: PTP messages for snapshot . . . 2165
Table 515. Egress and ingress latency for PHY interfaces . . . 2169
Table 516. Minimum PTP clock frequency example . . . 2169
Table 517. Message format defined in IEEE 1588-2008 . . . 2170
Table 518. Message format defined in IEEE 1588-2008 . . . 2171
Table 519. IPv6-UDP PTP packet fields required for control and status . . . 2172
Table 520. Ethernet PTP packet fields required for control and status . . . 2173
Table 521. Timestamp Snapshot Dependency on ETH_MACTSCR bits . . . 2174
Table 522. PTP message generation criteria . . . 2180
Table 523. Common PTP message header fields . . . 2182
Table 524. MAC transmit PTP mode and one-step timestamping operation . . . 2185
Table 525. Transmit checksum offload engine functions for different packet types . . . 2189
Table 526. Receive checksum offload engine functions for different packet types . . . 2192
Table 527. TSO: TCP and IP header fields . . . 2196
Table 528. Pause packet fields . . . 2201
Table 529. Tx MAC flow control . . . 2202
Table 530. Rx MAC flow control . . . 2202
Table 531. Size of the maximum receive packet . . . 2205
Table 532. MCD clock selection . . . 2207
Table 533. MDIO Clause 45 frame structure . . . 2208
Table 534. MDIO Clause 22 frame structure . . . 2209
Table 535. Remote wake-up packet filter register . . . 2220
Table 536. Description of the remote wake-up filter fields . . . 2220
Table 537. Remote wake-up packet and PMT interrupt generation . . . 2222
Table 538. Transfer complete interrupt behavior . . . 2231
Table 539. TDES0 normal descriptor (read format) . . . 2255
Table 540. TDES1 normal descriptor (read format) . . . 2256
Table 541. TDES2 normal descriptor (read format) . . . 2256
Table 542. TDES3 normal descriptor (read format) . . . 2257
Table 543. TDES0 normal descriptor (write-back format) . . . 2260
Table 544. TDES1 normal descriptor (write-back format) . . . 2260
Table 545. TDES2 normal descriptor (write-back format) . . . 2261
Table 546. TDES3 normal descriptor (write-back format) . . . 2261
Table 547. TDES0 context descriptor . . . 2265
Table 548. TDES1 context descriptor . . . 2265
Table 549. TDES2 context descriptor . . . 2265
Table 550. TDES3 context descriptor . . . 2266
Table 551. RDES0 normal descriptor (read format) . . . 2269
Table 552. RDES1 normal descriptor (read format) . . . 2269
Table 553. RDES2 normal descriptor (read format) . . . 2269
Table 554. RDES3 normal descriptor (read format) . . . 2270
Table 555. RDES0 normal descriptor (write-back format) . . . 2271
Table 556. RDES1 normal descriptor (write-back format) . . . 2272
Table 557. RDES2 normal descriptor (write-back format) . . . 2274
Table 558. RDES3 normal descriptor (write-back format) . . . 2275
Table 559. RDES0 context descriptor . . . 2278
Table 560. RDES1 context descriptor . . . 2279
Table 561. RDES2 context descriptor . . . 2279

Table 562. RDES3 context descriptor . . . 2279
Table 563. ETH_DMA common register map and reset values . . . 2300
Table 564. ETH_DMA_CH0 register map and reset values . . . 2300
Table 565. ETH_MTL register map and reset values . . . 2310
Table 566. Giant Packet Status based on S2KP and JE Bits . . . 2316
Table 567. Packet Length based on the CST and ACS bits . . . 2316
Table 568. Ethernet MAC register map and reset values . . . 2403
Table 569. JTAG/Serial-wire debug port pins . . . 2414
Table 570. Trace port pins . . . 2414
Table 571. Single-wire trace port pins . . . 2415
Table 572. Authentication signal states of the CPU . . . 2416
Table 573. Authentication signal states of the DAP . . . 2416
Table 574. Lifecycle state and debug states . . . 2417
Table 575. JTAG-DP data registers . . . 2421
Table 576. Packet request . . . 2423
Table 577. ACK response . . . 2423
Table 578. Data transfer . . . 2423
Table 579. Debug port registers . . . 2424
Table 580. Debug port register map and reset values . . . 2430
Table 581. MEM-AP registers . . . 2431
Table 582. Access port register map and reset values . . . 2437
Table 583. System ROM table . . . 2438
Table 584. MCU ROM table . . . 2439
Table 585. Processor ROM table . . . 2439
Table 586. System ROM table register map and reset values . . . 2444
Table 587. MCU ROM table register map and reset values . . . 2449
Table 588. CPU ROM table register map and reset values . . . 2454
Table 589. DWT register map and reset values . . . 2469
Table 590. ITM register map and reset values . . . 2480
Table 591. BPU register map and reset values . . . 2487
Table 592. ETM register map and reset values . . . 2511
Table 593. TPIU register map and reset values . . . 2524
Table 594. CTI inputs . . . 2526
Table 595. CTI outputs . . . 2527
Table 596. CTI register map and reset values . . . 2538
Table 597. Peripheral clock-freeze control bits . . . 2540
Table 598. Peripheral behavior in debug mode . . . 2541
Table 599. DBGMCU register map and reset values . . . 2554
Table 600. Document revision history . . . 2562

List of figures

Figure 1. System architecture . . . 88
Figure 2. Memory map for STM32C53x/542 . . . 91
Figure 3. Memory map for STM32C55x/562 . . . 92
Figure 4. Memory map for STM32C59x/5A3 . . . 93
Figure 5. Flash memory HDP area . . . 100
Figure 6. Key management principle . . . 104
Figure 7. Device life cycle security . . . 107
Figure 8. RDP level transition scheme . . . 108
Figure 9. FLASH block diagram (simplified) . . . 127
Figure 10. Embedded flash memory organization . . . 129
Figure 11. FLASH use . . . 130
Figure 12. FLASH data area memory map . . . 142
Figure 13. FLASH bank swapping sequence . . . 147
Figure 14. HDP in user flash memory . . . 156
Figure 15. Protection attributes in case of bank swap illustration . . . 158
Figure 16. RDP transitions . . . 159
Figure 17. ICACHE block diagram (master2 port only on STM32C59x/5A3 devices) . . . 209
Figure 18. ICACHE TAG and data memories functional view . . . 211
Figure 19. ICACHE remapping address mechanism . . . 214
Figure 20. Power supply . . . 225
Figure 21. Power-on (POR) / power-down (PDR) reset waveform . . . 228
Figure 22. PVD thresholds . . . 229
Figure 23. I/O states in Standby mode . . . 240
Figure 24. Simplified diagram of the reset circuit . . . 252
Figure 25. Clock tree . . . 255
Figure 26. HSE/ LSE clock sources . . . 256
Figure 27. Clock distribution for ETH . . . 267
Figure 28. Clock distribution for Crypton subsystem . . . 269
Figure 29. CRS block diagram . . . 328
Figure 30. CRS counter behavior . . . 330
Figure 31. Structure of 3- or 5-V tolerant GPIO (TT or FT) . . . 340
Figure 32. Input floating/pull-up/pull-down configurations . . . 344
Figure 33. Output configuration . . . 345
Figure 34. Alternate function configuration . . . 346
Figure 35. High-impedance analog configuration . . . 346
Figure 36. SBS block diagram . . . 357
Figure 37. Compensation cell management . . . 359
Figure 38. Compensation cell use . . . 359
Figure 39. SBS hardware secure storage control . . . 360
Figure 40. LPDMA block diagram . . . 390
Figure 41. LPDMA channel direct programming without linked-list (LPDMA_CxLLR = 0) . . . 391
Figure 42. LPDMA channel suspend and resume sequence . . . 392
Figure 43. LPDMA channel abort and restart sequence . . . 393
Figure 44. Static linked-list data structure (all Uxx = 1) of channel x . . . 394
Figure 45. LPDMA dynamic linked-list data structure of an addressing channel x . . . 395
Figure 46. LPDMA channel execution and linked-list programming in run-to-completion mode (LPDMA_CxCR.LSM = 0) . . . 397
Figure 47. Inserting a LLIn with an auxiliary LPDMA channel y . . . 399

Figure 48. LPDMA channel execution and linked-list programming in link step mode (LPDMA_CxCR.LSM = 1) . . . 401
Figure 49. Building LLIn+1: LPDMA dynamic linked-lists in link step mode . . . 402
Figure 50. Replace with a new LLIn' in register file in link step mode . . . 403
Figure 51. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 404
Figure 52. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 405
Figure 53. LPDMA channel execution and linked-list programming . . . 407
Figure 54. LPDMA arbitration policy . . . 411
Figure 55. Trigger hit, memorization and overrun waveform . . . 415
Figure 56. LPDMA circular buffer programming: update of the memory start address . . . 416
Figure 57. Shared LPDMA channel with circular buffering: update of the memory start address . . . 417
Figure 58. EXTI block diagram . . . 446
Figure 59. Configurable event trigger logic CPU wake-up . . . 449
Figure 60. EXTI direct events . . . 450
Figure 61. EXTI mux GPIO selection . . . 450
Figure 62. CRC calculation unit block diagram . . . 473
Figure 63. CORDIC convergence for trigonometric functions . . . 487
Figure 64. CORDIC convergence for hyperbolic functions . . . 488
Figure 65. CORDIC convergence for square root . . . 489
Figure 66. XSPI block diagram for octal configuration . . . 499
Figure 67. XSPI block diagram for quad configuration . . . 499
Figure 68. XSPI block diagram for dual-quad configuration . . . 500
Figure 69. SDR read command in octal configuration . . . 501
Figure 70. DTR read in octal-SPI mode with DQS (Macronix mode) example . . . 504
Figure 71. SDR write command in octal-SPI mode example . . . 506
Figure 72. DTR write in octal-SPI mode (Macronix mode) example . . . 507
Figure 73. Example of HyperBus read operation (8-bit data mode) . . . 508
Figure 74. HyperBus write operation with initial latency . . . 509
Figure 75. HyperBus read operation with additional latency . . . 510
Figure 76. HyperBus write operation with additional latency . . . 510
Figure 77. HyperBus write operation with no latency (register write) . . . 511
Figure 78. HyperBus read operation page crossing with latency . . . 511
Figure 79. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access . . . 518
Figure 80. OctaRAM read operation with reverse data ordering D1/D0 . . . 518
Figure 81. NCS when CKMODE = 0 (T = CLK period) . . . 524
Figure 82. Example of software control of two external memories (octal mode) . . . 525
Figure 83. Example of hardware-controller extended memory support (octal mode) . . . 526
Figure 84. DLYB block diagram . . . 556
Figure 85. ADC block diagram . . . 564
Figure 86. ADC clock scheme . . . 568
Figure 87. ADC1 connectivity . . . 569
Figure 88. ADC2 connectivity . . . 570
Figure 89. ADC3 connectivity . . . 571
Figure 90. ADC calibration . . . 573
Figure 91. Updating the ADC calibration factor . . . 574
Figure 92. Enabling/disabling the ADC . . . 575
Figure 93. Bulb mode timing diagram . . . 578
Figure 94. Analog-to-digital conversion time . . . 583
Figure 95. Stopping ongoing regular conversions . . . 584
Figure 96. Stopping ongoing regular and injected conversions . . . 584
Figure 97. Triggers shared between ADC master and slave . . . 586

Figure 98. Injected conversion latency during ongoing regular conversion . . . 587
Figure 99. Single conversions of a sequence, software trigger . . . 589
Figure 100. Continuous conversion of a sequence, software trigger . . . 589
Figure 101. Single conversions of a sequence, hardware trigger . . . 590
Figure 102. Continuous conversions of a sequence, hardware trigger . . . 590
Figure 103. Right alignment (offset disabled, unsigned value) . . . 592
Figure 104. Right alignment (offset enabled, signed value) . . . 593
Figure 105. Left alignment (offset disabled, unsigned value) . . . 593
Figure 106. Left alignment (offset enabled, signed value) . . . 594
Figure 107. Example of overrun (OVRMOD = 0) . . . 597
Figure 108. Example of overrun (OVRMOD = 1) . . . 597
Figure 109. AUTDLY = 1, regular conversion in continuous mode, software trigger . . . 600
Figure 110. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 601
Figure 111. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . 602
Figure 112. AUTDLY = 1, regular continuous conversions interrupted by injected conversions . . . 602
Figure 113. AUTDLY = 1 in autoinjection mode (JAUTO = 1) . . . 603
Figure 114. Analog watchdog guarded area . . . 604
Figure 115. ADC_AWDy_OUT signal generation (on all regular channels) . . . 606
Figure 116. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software) . . . 607
Figure 117. ADC_AWDy_OUT signal generation (on a single regular channel) . . . 607
Figure 118. ADC_AWDy_OUT signal generation (on all injected channels) . . . 608
Figure 119. 12-bit result oversampling with 10-bit right shift and rounding . . . 609
Figure 120. Triggered regular oversampling mode (TROVS bit = 1) . . . 611
Figure 121. Regular oversampling modes (4x ratio) . . . 612
Figure 122. Regular and injected oversampling modes used simultaneously . . . 613
Figure 123. Triggered regular oversampling with injection . . . 613
Figure 124. Oversampling in autoinjection mode . . . 614
Figure 125. Dual ADC block diagram . . . 616
Figure 126. Injected simultaneous mode on four channels: Dual ADC mode . . . 617
Figure 127. Regular simultaneous mode on 16 channels: dual ADC mode . . . 619
Figure 128. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode . . . 621
Figure 129. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode . . . 621
Figure 130. Interleaved conversion with injection . . . 622
Figure 131. Alternate trigger: injected group of each ADC . . . 623
Figure 132. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 624
Figure 133. Alternate + regular simultaneous . . . 624
Figure 134. Case of trigger occurring during injected conversion . . . 625
Figure 135. Interleaved single channel CH0 with injected sequence CH11, CH12 . . . 625
Figure 136. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first . . . 626
Figure 137. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first . . . 626
Figure 138. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00 . . . 627
Figure 139. DMA requests in interleaved mode when DAMDF = 0b10 . . . 628
Figure 140. Temperature sensor channel block diagram . . . 630
Figure 141. VREFINT channel block diagram . . . 631
Figure 142. Dual-channel DAC block diagram . . . 674
Figure 143. Data registers in single DAC channel mode . . . 677

Figure 144. Data registers in dual DAC channel mode . . . 678
Figure 145. Timing diagram for conversion with trigger disabled TEN = 0 . . . 679
Figure 146. DAC LFSR register calculation algorithm . . . 681
Figure 147. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 682
Figure 148. DAC triangle wave generation . . . 682
Figure 149. DAC conversion (SW trigger enabled) with triangle wave generation . . . 683
Figure 150. DAC sample and hold mode phase diagram . . . 685
Figure 151. Comparator functional block diagram . . . 713
Figure 152. Window comparators functional block diagram . . . 715
Figure 153. Comparator hysteresis . . . 716
Figure 154. Comparator output blanking . . . 717
Figure 155. Scaler block diagram . . . 720
Figure 156. OPAMP block diagram . . . 728
Figure 157. Standalone operational amplifier . . . 731
Figure 158. Follower . . . 732
Figure 159. Noninverting PGA . . . 733
Figure 160. Noninverting PGA with filtering . . . 734
Figure 161. Noninverting PGA with external bias, or inverting PGA . . . 735
Figure 162. Inverting or noninverting PGA with external bias or/and filtering . . . 736
Figure 163. Simplified view . . . 736
Figure 164. Timer-controlled operation for a dual motor application . . . 739
Figure 165. CCB block diagram . . . 748
Figure 166. CCB operation CPU → PKA + SAES (write) . . . 750
Figure 167. CCB operation PKA → SAES . . . 750
Figure 168. CCB operation SAES → PKA . . . 750
Figure 169. CCB operation SAES → CCB . . . 751
Figure 170. CCB operation RNG → PKA (write) . . . 751
Figure 171. CCB operation RNG → SAES (write) . . . 752
Figure 172. Blob creation scenarios . . . 753
Figure 173. RNG block diagram . . . 774
Figure 174. NIST SP800-90B entropy source model . . . 775
Figure 175. RNG initialization overview . . . 778
Figure 176. AES block diagram . . . 795
Figure 177. Encryption/ decryption typical usage . . . 796
Figure 178. Typical operation with authentication . . . 799
Figure 179. Example of suspend mode management . . . 800
Figure 180. ECB encryption . . . 801
Figure 181. ECB decryption . . . 801
Figure 182. CBC encryption . . . 801
Figure 183. CBC decryption . . . 802
Figure 184. Message construction in CTR mode . . . 804
Figure 185. CTR encryption . . . 805
Figure 186. Message construction in GCM . . . 806
Figure 187. GCM authenticated encryption . . . 808
Figure 188. Message construction in GMAC mode . . . 811
Figure 189. GMAC authentication mode . . . 812
Figure 190. Message construction in CCM mode . . . 813
Figure 191. CCM mode authenticated encryption . . . 814
Figure 192. 128-bit block construction according to the data type . . . 819
Figure 193. SAES block diagram . . . 841
Figure 194. Encryption/ decryption typical usage . . . 843
Figure 195. Typical operation with authentication . . . 845

Figure 196. Example of suspend mode management . . . 846
Figure 197. ECB encryption . . . 847
Figure 198. ECB decryption . . . 847
Figure 199. CBC encryption . . . 848
Figure 200. CBC decryption . . . 848
Figure 201. Message construction in CTR mode . . . 851
Figure 202. CTR encryption . . . 852
Figure 203. Message construction in GCM . . . 853
Figure 204. GCM authenticated encryption . . . 855
Figure 205. Message construction in GMAC mode . . . 858
Figure 206. GMAC authentication mode . . . 858
Figure 207. Message construction in CCM mode . . . 859
Figure 208. CCM mode authenticated encryption . . . 861
Figure 209. Operation with wrapped keys for SAES in ECB and CBC modes . . . 864
Figure 210. Operation with wrapped keys for SAES in CTR mode . . . 867
Figure 211. Usage of Shared-key mode . . . 868
Figure 212. 128-bit block construction according to the data type . . . 871
Figure 213. HASH block diagram . . . 894
Figure 214. Message data swapping feature . . . 897
Figure 215. HASH suspend/resume mechanism . . . 903
Figure 216. PKA block diagram . . . 917
Figure 217. Chaining of PKA with SAES or RNG . . . 924
Figure 218. Advanced-control timer block diagram . . . 956
Figure 219. Counter timing diagram with prescaler division change from 1 to 2 . . . 963
Figure 220. Counter timing diagram with prescaler division change from 1 to 4 . . . 963
Figure 221. Counter timing diagram, internal clock divided by 1 . . . 965
Figure 222. Counter timing diagram, internal clock divided by 2 . . . 965
Figure 223. Counter timing diagram, internal clock divided by 4 . . . 966
Figure 224. Counter timing diagram, internal clock divided by N . . . 966
Figure 225. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 967
Figure 226. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 968
Figure 227. Counter timing diagram, internal clock divided by 1 . . . 969
Figure 228. Counter timing diagram, internal clock divided by 2 . . . 970
Figure 229. Counter timing diagram, internal clock divided by 4 . . . 970
Figure 230. Counter timing diagram, internal clock divided by N . . . 971
Figure 231. Counter timing diagram, update event when repetition counter is not used . . . 971
Figure 232. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 . . . 973
Figure 233. Counter timing diagram, internal clock divided by 2 . . . 973
Figure 234. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 . . . 974
Figure 235. Counter timing diagram, internal clock divided by N . . . 974
Figure 236. Counter timing diagram, update event with ARPE = 1 (counter underflow) . . . 975
Figure 237. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 976
Figure 238. Update rate examples depending on mode and TIM_RCR register settings . . . 977
Figure 239. External trigger input block . . . 978
Figure 240. Control circuit in normal mode, internal clock divided by 1 . . . 979
Figure 241. tim_ti2 external clock connection example . . . 979
Figure 242. Control circuit in external clock mode 1 . . . 980
Figure 243. External trigger input block . . . 981
Figure 244. Control circuit in external clock mode 2 . . . 982
Figure 245. Capture/compare channel (example: channel 1 input stage) . . . 982

Figure 246. Capture/compare channel 1 main circuit . . . 983
Figure 247. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) . . . 984
Figure 248. Output stage of capture/compare channel (channel 5, idem ch. 6 and 7) . . . 984
Figure 249. PWM input mode timing . . . 986
Figure 250. Output compare mode, toggle on tim_oc1 . . . 988
Figure 251. Edge-aligned PWM waveforms (ARR = 8) . . . 990
Figure 252. Center-aligned PWM waveforms (ARR = 8) . . . 991
Figure 253. Dithering principle . . . 992
Figure 254. Data format and register coding in dithering mode . . . 993
Figure 255. PWM resolution vs frequency . . . 994
Figure 256. PWM dithering pattern . . . 995
Figure 257. Dithering effect on duty cycle in center-aligned PWM mode . . . 996
Figure 258. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 997
Figure 259. Asymmetric mode example for single-shunt current sensing . . . 1000
Figure 260. Asymmetric mode 8 example . . . 1000
Figure 261. Combined PWM mode examples . . . 1002
Figure 262. Combined PWM 3 example . . . 1003
Figure 263. 3-phase combined PWM signals with multiple trigger pulses per period . . . 1004
Figure 264. Complementary output with symmetrical dead-time insertion . . . 1005
Figure 265. Asymmetrical deadtime . . . 1006
Figure 266. Dead-time waveforms with delay greater than the negative pulse . . . 1006
Figure 267. Dead-time waveforms with delay greater than the positive pulse . . . 1006
Figure 268. Clock prescalers for dead-time and digital filters . . . 1007
Figure 269. Break and Break2 circuitry overview . . . 1009
Figure 270. Various output behavior in response to a break event on tim_brk (OSSI = 1) . . . 1011
Figure 271. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) . . . 1012
Figure 272. PWM output state following tim_brk assertion (OSSI = 0) . . . 1013
Figure 273. Immediate break and Delayed 1/2 break modes . . . 1013
Figure 274. Pulse termination during delayed break . . . 1014
Figure 275. Delayed break with deadtime insertion . . . 1015
Figure 276. Output redirection (tim_brk2 request not represented) . . . 1017
Figure 277. tim_ocref_clr input selection multiplexer . . . 1018
Figure 278. Clearing TIMx tim_ocxref . . . 1019
Figure 279. Asymmetric PWM mode 1 . . . 1019
Figure 280. Asymmetric PWM mode 2 . . . 1020
Figure 281. Asymmetric PWM mode 7 (OC1refc = oc1ref OR oc5ref) . . . 1020
Figure 282. Combined PWM mode 1 . . . 1021
Figure 283. 6-step generation, COM example (OSSR = 1) . . . 1022
Figure 284. Example of one pulse mode. . . . 1023
Figure 285. Retriggerable one-pulse mode . . . 1024
Figure 286. Pulse generator circuitry . . . 1025
Figure 287. Pulse generation on compare event, for edge-aligned and encoder modes . . . 1026
Figure 288. Extended pulsewidth in case of concurrent triggers . . . 1027
Figure 289. Example of counter operation in encoder interface mode. . . . 1029
Figure 290. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 1029
Figure 291. Quadrature encoder counting modes . . . 1030
Figure 292. Direction plus clock encoder mode . . . 1031
Figure 293. Directional clock encoder mode (CC1P = CC2P = 0) . . . 1031
Figure 294. Directional clock encoder mode (CC1P = CC2P = 1) . . . 1032
Figure 295. Index gating options . . . 1033
Figure 296. Jittered Index signals . . . 1033
Figure 297. Index generation for IPOS[1:0] = 11 . . . 1034

Figure 298. Counter reading with index gated on channel A (IPOS[1:0] = 11) 1034
Figure 299. Counter reading with index ungated (IPOS[1:0] = 00) 1035
Figure 300. Counter reading with index gated on channel A and B 1035
Figure 301. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) 1036
Figure 302. Counter reset Narrow index pulse (closer view, ARR = 0x07) 1037
Figure 303. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) 1038
Figure 304. Directional index sensitivity 1038
Figure 305. Counter reset as function of FIDX bit setting 1039
Figure 306. Index blanking 1039
Figure 307. Index behavior in clock + direction mode, IPOS[0] = 1 1040
Figure 308. Index behavior in directional clock mode, IPOS[0] = 1 1040
Figure 309. State diagram for quadrature encoded signals 1041
Figure 310. Up-counting encoder error detection 1042
Figure 311. Down-counting encode error detection 1043
Figure 312. Encoder mode change with preload transferred on update (SMSPS = 0) 1044
Figure 313. Encoder with built-in debouncer, x2 mode 1045
Figure 314. Encoder with built-in debouncer, x4 mode 1045
Figure 315. Clock drift measure using directional clock mode 1046
Figure 316. Measuring time interval between edges on three signals 1047
Figure 317. XOR gate connection with TI1S = 1 and XORPS = 0 1048
Figure 318. XOR gate connection with TI1S = 1 and XORPS = 1 1048
Figure 319. Example of Hall sensor interface 1050
Figure 320. Control circuit in reset mode 1051
Figure 321. Control circuit in Gated mode 1052
Figure 322. Control circuit in trigger mode 1053
Figure 323. Control circuit in external clock mode 2 + trigger mode 1054
Figure 324. tim_trgo2 postscaler example 1055
Figure 325. General-purpose timer block diagram 1135
Figure 326. Counter timing diagram with prescaler division change from 1 to 2 1141
Figure 327. Counter timing diagram with prescaler division change from 1 to 4 1141
Figure 328. Counter timing diagram, internal clock divided by 1 1142
Figure 329. Counter timing diagram, internal clock divided by 2 1143
Figure 330. Counter timing diagram, internal clock divided by 4 1143
Figure 331. Counter timing diagram, internal clock divided by N 1144
Figure 332. Counter timing diagram, Update event when ARPE = 0 (TIM_ARR not preloaded) . . . 1144
Figure 333. Counter timing diagram, Update event when ARPE = 1 (TIM_ARR preloaded) 1145
Figure 334. Counter timing diagram, internal clock divided by 1 1146
Figure 335. Counter timing diagram, internal clock divided by 2 1147
Figure 336. Counter timing diagram, internal clock divided by 4 1147
Figure 337. Counter timing diagram, internal clock divided by N 1148
Figure 338. Counter timing diagram, Update event 1148
Figure 339. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 1150
Figure 340. Counter timing diagram, internal clock divided by 2 1150
Figure 341. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 1151
Figure 342. Counter timing diagram, internal clock divided by N 1151
Figure 343. Counter timing diagram, Update event with ARPE = 1 (counter underflow) 1152
Figure 344. Counter timing diagram, Update event with ARPE = 1 (counter overflow) 1153
Figure 345. External trigger input block 1153
Figure 346. Control circuit in normal mode, internal clock divided by 1 1155
Figure 347. tim_ti2 external clock connection example 1155
Figure 348. Control circuit in external clock mode 1 1156
Figure 349. External trigger input block 1157

Figure 350. Control circuit in external clock mode 2 . . . 1158
Figure 351. Capture/compare channel (example: channel 1 input stage) . . . 1158
Figure 352. Capture/compare channel 1 main circuit . . . 1159
Figure 353. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4) . . . 1159
Figure 354. PWM input mode timing . . . 1162
Figure 355. Output compare mode, toggle on tim_oc1 . . . 1164
Figure 356. Edge-aligned PWM waveforms (ARR = 8) . . . 1165
Figure 357. Center-aligned PWM waveforms (ARR = 8) . . . 1166
Figure 358. Dithering principle . . . 1167
Figure 359. Data format and register coding in dithering mode . . . 1168
Figure 360. PWM resolution vs frequency (16-bit mode) . . . 1169
Figure 361. PWM resolution vs frequency (32-bit mode) . . . 1169
Figure 362. PWM dithering pattern . . . 1170
Figure 363. Dithering effect on duty cycle in center-aligned PWM mode . . . 1171
Figure 364. Generation of two phase-shifted PWM signals with 50% duty cycle . . . 1173
Figure 365. Combined PWM mode on channels 1 and 3 . . . 1175
Figure 366. Clock prescalers for dead-time and digital filters . . . 1176
Figure 367. OCREF_CLR input selection multiplexer . . . 1176
Figure 368. Clearing TIMx tim_ocxref . . . 1177
Figure 369. Asymmetric PWM mode 1 . . . 1178
Figure 370. Asymmetric PWM mode 2 . . . 1178
Figure 371. Combined PWM mode 1 . . . 1179
Figure 372. Example of one-pulse mode . . . 1180
Figure 373. Retriggerable one-pulse mode . . . 1181
Figure 374. Pulse generator circuitry . . . 1182
Figure 375. Pulse generation on compare event, for edge-aligned and encoder modes . . . 1183
Figure 376. Extended pulse width in case of concurrent triggers . . . 1184
Figure 377. Example of counter operation in encoder interface mode . . . 1186
Figure 378. Example of encoder interface mode with tim_ti1fp1 polarity inverted . . . 1186
Figure 379. Quadrature encoder counting modes . . . 1187
Figure 380. Direction plus clock encoder mode . . . 1188
Figure 381. Directional clock encoder mode (CC1P = CC2P = 0) . . . 1189
Figure 382. Directional clock encoder mode (CC1P = CC2P = 1) . . . 1189
Figure 383. Index gating options . . . 1191
Figure 384. Jittered Index signals . . . 1191
Figure 385. Index generation for IPOS[1:0] = 11 . . . 1192
Figure 386. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 1192
Figure 387. Counter reading with index ungated (IPOS[1:0] = 00) . . . 1193
Figure 388. Counter reading with index gated on channel A and B . . . 1193
Figure 389. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 1194
Figure 390. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 1195
Figure 391. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 1196
Figure 392. Directional index sensitivity . . . 1196
Figure 393. Counter reset as function of FIDX bit setting . . . 1197
Figure 394. Index blanking . . . 1197
Figure 395. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 1198
Figure 396. Index behavior in directional clock mode, IPOS[0] = 1 . . . 1198
Figure 397. State diagram for quadrature encoded signals . . . 1199
Figure 398. Up-counting encoder error detection . . . 1200
Figure 399. Down-counting encode error detection . . . 1201
Figure 400. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 1202
Figure 401. Encoder with built-in debouncer, x2 mode . . . 1203

Figure 402. Encoder with built-in debouncer, x4 mode . . . 1203
Figure 403. Clock drift measure using directional clock mode . . . 1204
Figure 404. Measuring time interval between edges on three signals . . . 1205
Figure 405. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 1206
Figure 406. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 1206
Figure 407. Control circuit in reset mode . . . 1207
Figure 408. Control circuit in gated mode . . . 1208
Figure 409. Control circuit in trigger mode . . . 1209
Figure 410. Control circuit in external clock mode 2 + trigger mode . . . 1210
Figure 411. Master/Slave timer example . . . 1211
Figure 412. Master/slave connection example with 1 channel only timers . . . 1211
Figure 413. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 1212
Figure 414. Gating TIM_slv with Enable of TIM_mstr . . . 1213
Figure 415. Triggering TIM_slv with update of TIM_mstr . . . 1214
Figure 416. Triggering TIM_slv with Enable of TIM_mstr . . . 1214
Figure 417. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 1215
Figure 418. Basic timer block diagram . . . 1260
Figure 419. Control circuit in normal mode, internal clock divided by 1 . . . 1261
Figure 420. Counter timing diagram with prescaler division change from 1 to 2 . . . 1262
Figure 421. Counter timing diagram with prescaler division change from 1 to 4 . . . 1263
Figure 422. Counter timing diagram, internal clock divided by 1 . . . 1264
Figure 423. Counter timing diagram, internal clock divided by 2 . . . 1264
Figure 424. Counter timing diagram, internal clock divided by 4 . . . 1265
Figure 425. Counter timing diagram, internal clock divided by N . . . 1265
Figure 426. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 1266
Figure 427. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 1267
Figure 428. Dithering principle . . . 1268
Figure 429. Data format and register coding in dithering mode . . . 1268
Figure 430. FCnt resolution vs frequency . . . 1269
Figure 431. PWM dithering pattern . . . 1269
Figure 432. General-purpose timer block diagram (TIM12) . . . 1282
Figure 433. Counter timing diagram with prescaler division change from 1 to 2 . . . 1285
Figure 434. Counter timing diagram with prescaler division change from 1 to 4 . . . 1286
Figure 435. Counter timing diagram, internal clock divided by 1 . . . 1287
Figure 436. Counter timing diagram, internal clock divided by 2 . . . 1287
Figure 437. Counter timing diagram, internal clock divided by 4 . . . 1288
Figure 438. Counter timing diagram, internal clock divided by N . . . 1288
Figure 439. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 1289
Figure 440. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 1290
Figure 441. Control circuit in normal mode, internal clock divided by 1 . . . 1291
Figure 442. tim_ti2 external clock connection example . . . 1291
Figure 443. Control circuit in external clock mode 1 . . . 1292
Figure 444. Capture/compare channel 1 input stage (TIM12) . . . 1293
Figure 445. Capture/compare channel 1 main circuit . . . 1293
Figure 446. Output stage of capture/compare channel 1 . . . 1294
Figure 447. PWM input mode timing . . . 1296
Figure 448. Output compare mode, toggle on tim_oc1 . . . 1298
Figure 449. Edge-aligned PWM waveforms (ARR = 8) . . . 1299

Figure 450. Dithering principle . . . 1300
Figure 451. Data format and register coding in dithering mode . . . 1300
Figure 452. PWM resolution vs frequency . . . 1301
Figure 453. PWM dithering pattern . . . 1302
Figure 454. Combined PWM mode on channel 1 and 2 . . . 1304
Figure 455. Clock prescalers for digital filters . . . 1305
Figure 456. Example of one pulse mode . . . 1305
Figure 457. Retriggerable one pulse mode . . . 1307
Figure 458. Measuring time interval between edges on 2 signals . . . 1308
Figure 459. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 1308
Figure 460. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 1309
Figure 461. Control circuit in reset mode . . . 1310
Figure 462. Control circuit in gated mode . . . 1311
Figure 463. Control circuit in trigger mode . . . 1311
Figure 464. TIM15 block diagram . . . 1336
Figure 465. TIM16/TIM17 block diagram . . . 1337
Figure 466. Counter timing diagram with prescaler division change from 1 to 2 . . . 1342
Figure 467. Counter timing diagram with prescaler division change from 1 to 4 . . . 1342
Figure 468. Counter timing diagram, internal clock divided by 1 . . . 1344
Figure 469. Counter timing diagram, internal clock divided by 2 . . . 1344
Figure 470. Counter timing diagram, internal clock divided by 4 . . . 1345
Figure 471. Counter timing diagram, internal clock divided by N . . . 1345
Figure 472. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 1346
Figure 473. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 1347
Figure 474. Counter timing diagram, internal clock divided by 1 . . . 1348
Figure 475. Counter timing diagram, internal clock divided by 2 . . . 1349
Figure 476. Counter timing diagram, internal clock divided by 4 . . . 1349
Figure 477. Counter timing diagram, internal clock divided by N . . . 1350
Figure 478. Counter timing diagram, update event when repetition counter is not used . . . 1350
Figure 479. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 . . . 1352
Figure 480. Counter timing diagram, internal clock divided by 2 . . . 1352
Figure 481. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 . . . 1353
Figure 482. Counter timing diagram, internal clock divided by N . . . 1353
Figure 483. Counter timing diagram, update event with ARPE = 1 (counter underflow) . . . 1354
Figure 484. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 1355
Figure 485. Update rate examples depending on mode and TIM_RCR register settings . . . 1356
Figure 486. Control circuit in normal mode, internal clock divided by 1 . . . 1357
Figure 487. tim_ti2 external clock connection example . . . 1358
Figure 488. Control circuit in external clock mode 1 . . . 1359
Figure 489. Capture/compare channel (example: channel 1 input stage) . . . 1359
Figure 490. Capture/compare channel 1 main circuit . . . 1360
Figure 491. Output stage of capture/compare channel (channel 1) . . . 1360
Figure 492. Output stage of capture/compare channel (channel 2 for TIM15) . . . 1361
Figure 493. PWM input mode timing . . . 1363
Figure 494. Output compare mode, toggle on tim_oc1 . . . 1365
Figure 495. Edge-aligned PWM waveforms (ARR = 8) . . . 1366
Figure 496. Center-aligned PWM waveforms (ARR = 8) . . . 1367
Figure 497. Dithering principle . . . 1368
Figure 498. Data format and register coding in dithering mode . . . 1369
Figure 499. PWM resolution vs frequency . . . 1370

Figure 500. PWM dithering pattern . . . 1370
Figure 501. Dithering effect on duty cycle in center-aligned PWM mode . . . 1371
Figure 502. Generation of a phase-shifted PWM signal with 50% duty cycle . . . 1372
Figure 503. Combined PWM mode on channel 1 and 2 . . . 1374
Figure 504. Complementary output with symmetrical dead-time insertion. . . . 1375
Figure 505. Asymmetrical deadtime . . . 1376
Figure 506. Dead-time waveforms with delay greater than the negative pulse. . . . 1376
Figure 507. Dead-time waveforms with delay greater than the positive pulse. . . . 1376
Figure 508. Clock prescalers for dead-time and digital filters . . . 1377
Figure 509. Break circuitry overview . . . 1379
Figure 510. Output behavior in response to a break event on tim_brk . . . 1381
Figure 511. Immediate break and Delayed 1/2 break modes . . . 1382
Figure 512. Pulse termination during delayed break . . . 1383
Figure 513. Delayed break with deadtime insertion . . . 1383
Figure 514. Output redirection . . . 1385
Figure 515. tim_ocref_clr input selection multiplexer . . . 1386
Figure 516. Asymmetric PWM mode 1 . . . 1387
Figure 517. Asymmetric PWM mode 2 . . . 1387
Figure 518. Combined PWM mode 1 . . . 1388
Figure 519. 6-step generation, COM example (OSSR = 1) . . . 1389
Figure 520. Example of one pulse mode . . . 1390
Figure 521. Retriggerable one pulse mode . . . 1392
Figure 522. Example of counter operation in encoder interface mode. . . . 1394
Figure 523. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 1394
Figure 524. Quadrature encoder counting modes . . . 1395
Figure 525. Direction plus clock encoder mode . . . 1396
Figure 526. Directional clock encoder mode (CC1P = CC2P = 0) . . . 1396
Figure 527. Directional clock encoder mode (CC1P = CC2P = 1) . . . 1397
Figure 528. State diagram for quadrature encoded signals . . . 1398
Figure 529. Encoder mode change with preload transferred on update . . . 1398
Figure 530. Encoder with built-in debouncer, x2 mode . . . 1400
Figure 531. Encoder with built-in debouncer, x4 mode . . . 1400
Figure 532. Clock drift measure using directional clock mode . . . 1401
Figure 533. Measuring time interval between edges on 2 signals . . . 1402
Figure 534. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 1402
Figure 535. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 1403
Figure 536. Control circuit in reset mode . . . 1404
Figure 537. Control circuit in gated mode . . . 1405
Figure 538. Control circuit in trigger mode . . . 1406
Figure 539. LPTIM1 block diagram(1) . . . 1485
Figure 540. Glitch filter timing diagram . . . 1489
Figure 541. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 1491
Figure 542. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 1491
Figure 543. LPTIM output waveform, Continuous counting mode configuration . . . 1492
Figure 544. Waveform generation . . . 1493
Figure 545. Encoder mode counting sequence . . . 1497
Figure 546. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1). . . . 1498
Figure 547. Capture/compare input stage (channel 1) . . . 1499
Figure 548. Capture/compare output stage (channel 1) . . . 1499

Figure 549. Edge-aligned PWM mode (PRELOAD = 1) . . . 1501
Figure 550. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 1502
Figure 551. PWM mode with immediate update versus preloaded update . . . 1503
Figure 552. Independent watchdog block diagram . . . 1529
Figure 553. Reset timing due to timeout . . . 1531
Figure 554. Reset timing due to refresh in the not allowed area . . . 1532
Figure 555. Changing PR, RL, and performing a refresh(1) . . . 1533
Figure 556. Window comparator update(1) . . . 1534
Figure 557. Independent watchdog interrupt timing diagram . . . 1536
Figure 558. Early wake-up comparator update(1) . . . 1537
Figure 559. Watchdog block diagram . . . 1545
Figure 560. Window watchdog timing diagram . . . 1547
Figure 561. RTC block diagram . . . 1552
Figure 562. TAMP block diagram . . . 1598
Figure 563. Backup registers protection zones . . . 1601
Figure 564. Tamper sampling with precharge pulse . . . 1605
Figure 565. Low level detection with precharge and filtering . . . 1605
Figure 566. Block diagram . . . 1626
Figure 567. I²C-bus protocol . . . 1628
Figure 568. Setup and hold timings . . . 1630
Figure 569. I2C initialization flow . . . 1632
Figure 570. Data reception . . . 1633
Figure 571. Data transmission . . . 1634
Figure 572. Target initialization flow . . . 1637
Figure 573. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 1639
Figure 574. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 1640
Figure 575. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 1641
Figure 576. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 1642
Figure 577. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 1643
Figure 578. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 1643
Figure 579. Controller clock generation . . . 1645
Figure 580. Controller initialization flow . . . 1647
Figure 581. 10-bit address read access with HEAD10R = 0 . . . 1647
Figure 582. 10-bit address read access with HEAD10R = 1 . . . 1648
Figure 583. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 1649
Figure 584. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 1650
Figure 585. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 1651
Figure 586. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 1653
Figure 587. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 1654
Figure 588. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 1655
Figure 589. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 1659
Figure 590. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 1662
Figure 591. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 1663
Figure 592. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 1664
Figure 593. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . 1665
Figure 594. Bus transfer diagrams for SMBus controller transmitter . . . 1666
Figure 595. Bus transfer diagrams for SMBus controller receiver . . . 1668
Figure 596. I3C block diagram . . . 1692
Figure 597. I3C (primary) controller state and programming sequence diagram . . . 1697

Figure 598. I3C target state and programing sequence diagram 1702
Figure 599. I3C CCC messages, as controller 1716
Figure 600. I3C broadcast ENTDAA CCC, as controller 1717
Figure 601. I3C broadcast, direct read and direct write RSTACT CCC, as controller 1718
Figure 602. I3C CCC messages, as target 1720
Figure 603. I3C broadcast ENTDAA CCC, as target 1721
Figure 604. I3C broadcast DEFTGTS CCC, as target 1722
Figure 605. I3C broadcast DEFGRPA CCC, as target 1723
Figure 606. I3C private read/write messages, as controller 1725
Figure 607. I3C private read/write messages, as target 1726
Figure 608. Legacy I2C read/write messages, as controller 1727
Figure 609. IBI transfer, as controller/target 1728
Figure 610. Hot-join request transfer, as controller/target 1729
Figure 611. Controller-role request transfer, as controller/target 1730
Figure 612. C-FIFO management, as controller 1731
Figure 613. TX-FIFO management, as controller 1733
Figure 614. RX-FIFO management, as controller 1735
Figure 615. S-FIFO management, as controller 1738
Figure 616. RX-FIFO management, as target on the I3C bus 1739
Figure 617. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus 1741
Figure 618. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target 1743
Figure 619. USART block diagram 1801
Figure 620. Word length programming 1805
Figure 621. Configurable stop bits 1807
Figure 622. TC/TXE behavior when transmitting 1809
Figure 623. Start bit detection when oversampling by 16 or 8 1810
Figure 624. usart_ker_ck clock divider block diagram 1813
Figure 625. Data sampling when oversampling by 16 1814
Figure 626. Data sampling when oversampling by 8 1815
Figure 627. Mute mode using Idle line detection 1822
Figure 628. Mute mode using address mark detection 1823
Figure 629. Break detection in LIN mode (11-bit break length - LBDL bit is set) 1826
Figure 630. Break detection in LIN mode vs. Framing error detection 1827
Figure 631. USART example of synchronous master transmission 1828
Figure 632. USART data clock timing diagram in synchronous master mode (M bits = 00) 1828
Figure 633. USART data clock timing diagram in synchronous master mode (M bits = 01) 1829
Figure 634. USART data clock timing diagram in synchronous slave mode (M bits = 00) 1830
Figure 635. ISO 7816-3 asynchronous protocol 1832
Figure 636. Parity error detection using the 1.5 stop bits 1834
Figure 637. IrDA SIR ENDEC block diagram 1838
Figure 638. IrDA data modulation (3/16) - normal mode 1838
Figure 639. Transmission using DMA 1840
Figure 640. Reception using DMA 1841
Figure 641. Hardware flow control between two USARTs 1841
Figure 642. RS232 RTS flow control 1842
Figure 643. RS232 CTS flow control 1843
Figure 644. Wake-up event verified (wake-up event = address match, FIFO disabled) 1846
Figure 645. Wake-up event not verified (wake-up event = address match,

FIFO disabled) . . . 1846
Figure 646. LPUART block diagram . . . 1891
Figure 647. LPUART word length programming . . . 1894
Figure 648. Configurable stop bits . . . 1896
Figure 649. TC/TXE behavior when transmitting . . . 1898
Figure 650. lpuart_ker_ck clock divider block diagram . . . 1902
Figure 651. Mute mode using Idle line detection . . . 1906
Figure 652. Mute mode using address mark detection . . . 1907
Figure 653. Transmission using DMA . . . 1909
Figure 654. Reception using DMA . . . 1910
Figure 655. Hardware flow control between two LPUARTs . . . 1911
Figure 656. RS232 RTS flow control . . . 1911
Figure 657. RS232 CTS flow control . . . 1912
Figure 658. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 1915
Figure 659. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 1915
Figure 660. SPI/I2S block diagram . . . 1947
Figure 661. Full-duplex single master/ single slave application . . . 1950
Figure 662. Half-duplex single master/ single slave application . . . 1951
Figure 663. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) . . . 1952
Figure 664. Master and three independent slaves connected in star topology . . . 1953
Figure 665. Master and three slaves connected in circular (daisy chain) topology . . . 1955
Figure 666. Multimaster application . . . 1956
Figure 667. Scheme of NSS control logic . . . 1958
Figure 668. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 1958
Figure 669. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 1959
Figure 670. Data clock timing diagram . . . 1961
Figure 671. Data alignment when data size is not equal to 8, 16 or 32 bits . . . 1962
Figure 672. TI mode transfer . . . 1971
Figure 673. Optional configurations of the slave behavior when an underrun condition is detected . 1973
Figure 674. Waveform examples . . . 1981
Figure 675. Master I2S Philips protocol waveforms (16/32-bit full accuracy) . . . 1982
Figure 676. I2S Philips standard waveforms . . . 1982
Figure 677. Master MSB-justified 16- or 32-bit full-accuracy length . . . 1983
Figure 678. Master MSB-justified 16- or 24-bit data length . . . 1983
Figure 679. Slave MSB-justified 16-, 24- or 32-bit data length . . . 1984
Figure 680. LSB-justified 16 or 24-bit data length . . . 1984
Figure 681. Master PCM when the frame length is equal the data length . . . 1985
Figure 682. Master PCM standard waveforms (16 or 24-bit data length) . . . 1985
Figure 683. Slave PCM waveforms . . . 1986
Figure 684. Startup sequence, I2S Philips standard, master . . . 1989
Figure 685. Startup sequence, I2S Philips standard, slave . . . 1990
Figure 686. Stop sequence, I2S Philips standard, master . . . 1990
Figure 687. I^2S clock generator architecture . . . 1991
Figure 688. Data Format . . . 1993
Figure 689. Handling of underrun situation . . . 1995
Figure 690. Handling of overrun situation . . . 1996
Figure 691. Frame error detection, with FIXCH = 0 . . . 1997
Figure 692. Frame error detection, with FIXCH = 1 . . . 1997
Figure 693. CAN subsystem . . . 2022

Figure 694. FDCAN block diagram . . . 2024
Figure 695. Bit timing . . . 2026
Figure 696. Transceiver delay measurement . . . 2031
Figure 697. Pin control in bus monitoring mode . . . 2032
Figure 698. Pin control in loop-back mode . . . 2035
Figure 699. CAN error state diagram . . . 2036
Figure 700. Message RAM configuration . . . 2037
Figure 701. Standard message ID filter path . . . 2040
Figure 702. Extended message ID filter path . . . 2041
Figure 703. USB peripheral block diagram . . . 2088
Figure 704. Packet buffer areas with examples of buffer description table locations . . . 2094
Figure 705. High-level Ethernet block diagram . . . 2139
Figure 706. DMA transmission flow (standard mode) . . . 2142
Figure 707. DMA transmission flow (OSP mode) . . . 2144
Figure 708. Receive DMA flow . . . 2146
Figure 709. Overview of MAC transmission flow . . . 2150
Figure 710. MAC reception flow . . . 2152
Figure 711. Packet filtering sequence . . . 2156
Figure 712. Networked time synchronization . . . 2165
Figure 713. Propagation delay calculation in clocks supporting peer-to-peer path correction . . . 2166
Figure 714. System time update using fine correction method . . . 2176
Figure 715. TCP segmentation offload overview . . . 2193
Figure 716. TCP segmentation offload flow . . . 2194
Figure 717. Header and payload fields of segmented packets . . . 2197
Figure 718. Supported PHY interfaces . . . 2206
Figure 719. SMA Interface block . . . 2206
Figure 720. MDIO packet structure (Clause 45) . . . 2207
Figure 721. MDIO packet structure (Clause 22) . . . 2208
Figure 722. SMA write operation flow . . . 2210
Figure 723. Write data packet . . . 2211
Figure 724. Read data packet . . . 2211
Figure 725. Media independent interface (MII) signals . . . 2213
Figure 726. RMII block diagram . . . 2215
Figure 727. Transmission bit order . . . 2216
Figure 728. Receive bit order . . . 2217
Figure 729. LPI transitions (Transmit, 100 Mbds) . . . 2225
Figure 730. LPI Tx clock gating (when LPITCSE = 1) . . . 2226
Figure 731. LPI transitions (receive, 100 Mbit/s) . . . 2227
Figure 732. Descriptor ring structure . . . 2251
Figure 733. DMA descriptor ring . . . 2253
Figure 734. Descriptor tail pointer example 1 . . . 2253
Figure 735. Descriptor tail pointer example 2 . . . 2254
Figure 736. Transmit descriptor (read format) . . . 2255
Figure 737. Transmit descriptor write-back format . . . 2260
Figure 738. Transmit context descriptor format . . . 2264
Figure 739. Receive normal descriptor (read format) . . . 2268
Figure 740. Receive normal descriptor (write-back format) . . . 2271
Figure 741. Receive context descriptor . . . 2278
Figure 742. Generation of ETH_DMAISR flags . . . 2293
Figure 743. Block diagram of debug support infrastructure . . . 2414
Figure 744. Product life cycle states and regression management . . . 2417

Figure 745. Product life cycle states and transition management 2419
Figure 746. JTAG TAP state machine 2420
Figure 747. CoreSight topology 2440
Figure 748. Trace port interface unit (TPIU) 2515
Figure 749. Embedded cross trigger 2526

1 Documentation conventions

1.1 General information

The STM32C5 devices have an Arm® Cortex®-M33 core.

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

1.2 List of abbreviations for registers

The following abbreviations[a] are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

a. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.

1.3 Register reset value

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is undefined are marked as X.

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is unmodified are marked as U.

1.4 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **AHB**: advanced high-performance bus.

2 Memory and bus architecture

2.1 System architecture

The STM32C5 architecture relies on an Arm Cortex-M33 core, optimized for execution thanks to an instruction cache (ICACHE) that has a direct access to the embedded flash memory (FLASH).

This architecture also features a 32-bit multilayer AHB bus matrix with the interconnections details in *Table 1* and *Table 2*.

Table 1. Implementation of masters on STM32C5 series

Master	Comment	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
Cortex-M33 Fast C-bus	Connecting Cortex-M33 (with FPU) to the internal SRAMs and flash memory through ICACHE	X	X	X
Cortex-M33 Slow C-bus	Connecting Cortex-M33 (with FPU) to the external memories through ICACHE	-	-	X
Cortex-M33 S-bus	Connecting the Cortex-M33 (with FPU) to internal SRAMs without latency	2 masters		
LPDMA1		1 master		
LPDMA2		1 master		
Ethernet MAC		-	-	X(1)

1. Not available on STM32C591xx devices.

Table 2. Implementation of slaves on STM32C5 series

Slave	Comment	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
Flash memory	-	X	X	X
SRAM1	-	X	X	X
SRAM2	-	X	X	X
AHB1	Peripherals including AHB and APB bridge, and APB peripherals (connected to APB1 and APB2)	X	X	X
AHB2	Peripherals	X	X	X
XSPI1	-	-	-	X

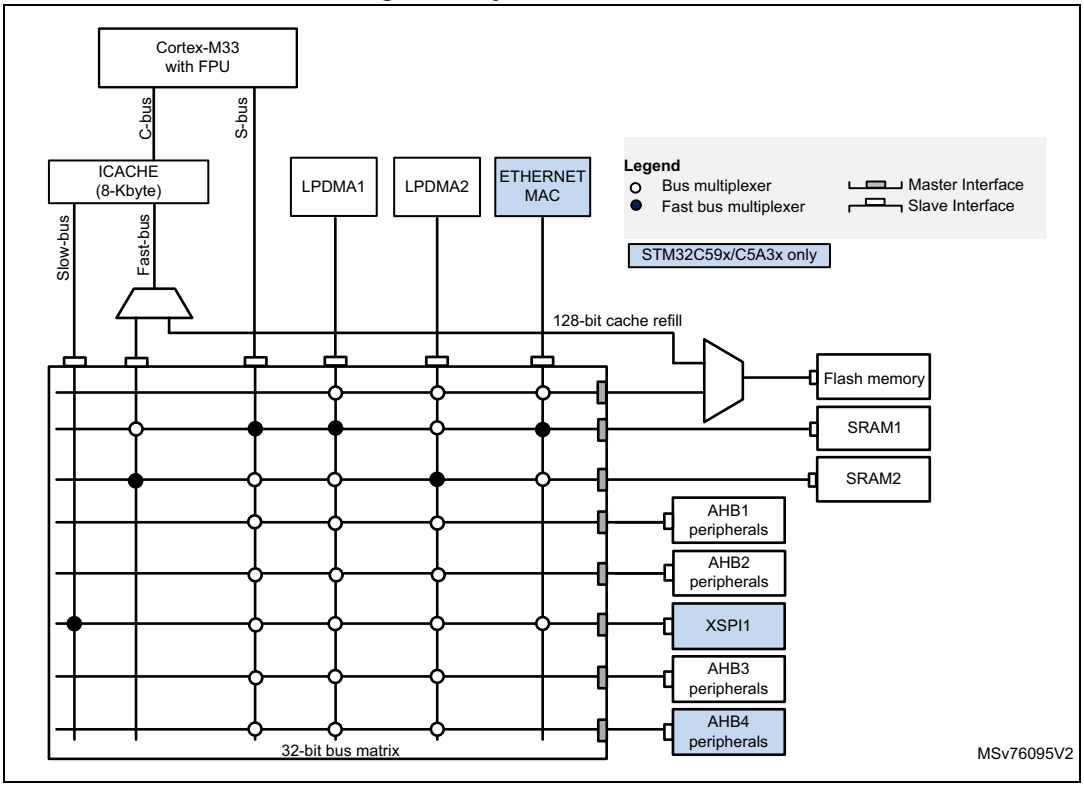
Table 2. Implementation of slaves on STM32C5 series (continued)

Slave	Comment	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
AHB3	Including AHB to APB bridge, and APB peripherals (connected to APB3)	X	X	X
AHB4	Peripherals	-	-	X

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation, even when several high-speed peripherals work simultaneously.

This architecture is shown in *Figure 1*.

Figure 1. System architecture



2.1.1 Fast C-bus

This bus connects the C-bus of the Cortex-M33 core to the FLASH and to the bus matrix through the ICACHE. This bus is used for instruction fetch and data access to the internal memories mapped in code region. This bus targets the FLASH and the internal SRAMs (SRAM1/2).

SRAM1 and SRAM2 are accessible on this bus with a continuous mapping.

2.1.2 Slow C-bus

This bus connects the C-bus of the Cortex-M33 core to the bus matrix through the instruction cache. This bus is used for instruction fetch and data access to the external memories mapped in code region. This bus targets the external memories (XSPI).

2.1.3 S-bus

This bus connects the system bus of the Cortex-M33 core to the bus matrix. This bus is used by the core to access data located in a peripheral or SRAM area. This bus targets the internal SRAMs (SRAM1/2), the AHB1 peripherals (including APB1/2), AHB2, and AHB3 peripherals.

SRAM1 and SRAM2 are accessible on this bus with a continuous mapping.

2.1.4 LPDMA1 and LPDMA2 buses

These buses connect the two AHB master interfaces of LPDMA1/2 to the bus matrix. These buses target the FLASH, the internal SRAMs (SRAM1/2), the AHB1 peripherals (including APB1/2), AHB2, AHB3, and AHB4 peripherals.

2.1.5 Bus matrix

The bus matrix manages the access arbitration between masters. The arbitration uses a Round-Robin algorithm. This bus matrix features a fast bus multiplexer used to connect each master to a given slave without latency (see *Figure 1*). For the same master, other slaves undergo a latency of at least one cycle at each new access.

2.1.6 AHB/APB bridges

The three AHB/APB bridges provide full synchronous connections between AHB and APB buses, allowing flexible selection of the peripheral frequency.

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to these bridges.

After each device reset, all peripheral clocks are disabled (except for internal SRAMs and FLASH). Before using a peripheral, its clock must be enabled in RCC_AHBxENR and RCC_APBxENR registers.

Note: When an 8- or 16-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 8- or 16-bit data to feed the 32-bit vector.

2.2 Memory organization

2.2.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

2.2.2 Memory map and register boundary addresses

Figure 2. Memory map for STM32C53x/542

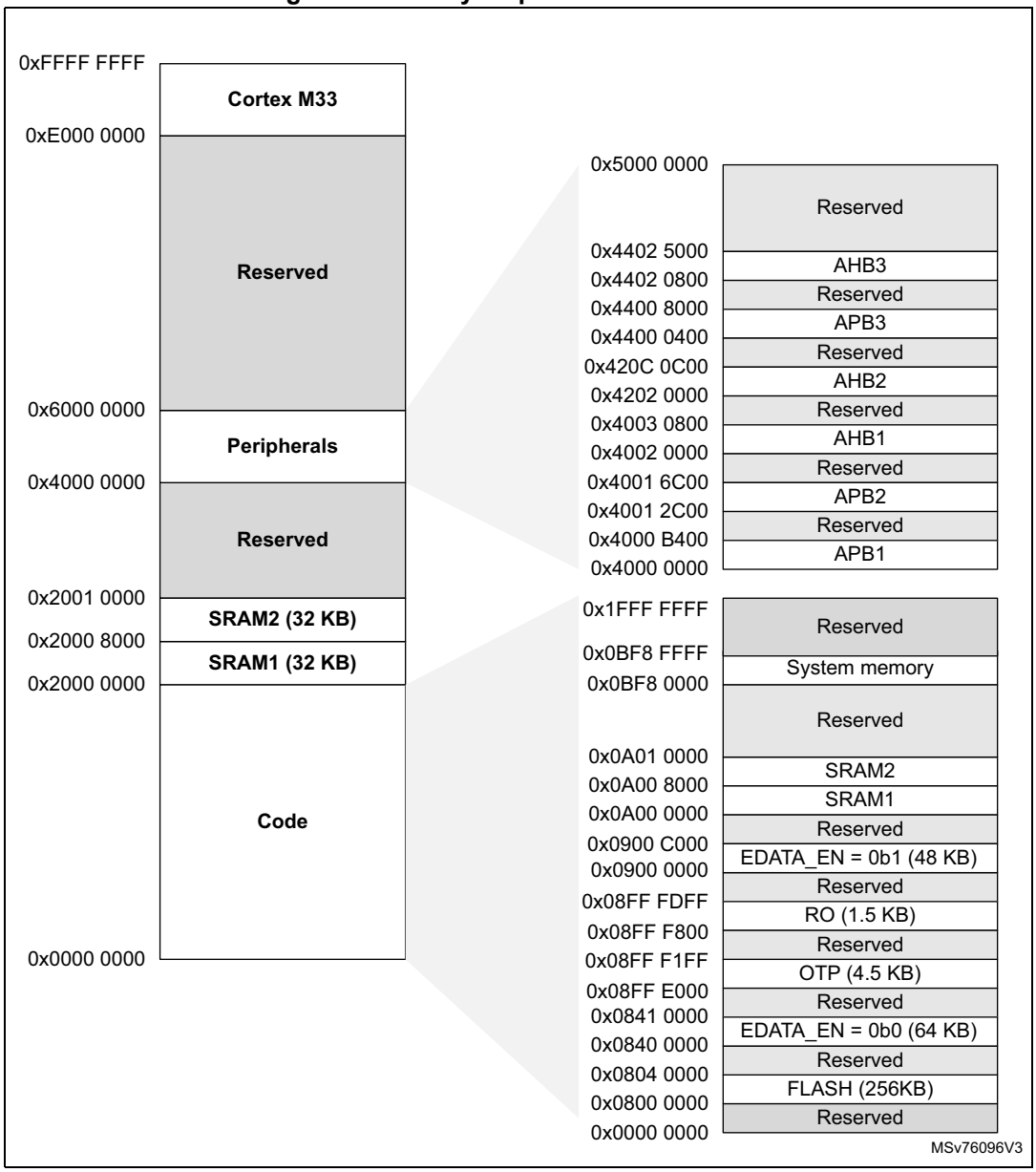


Figure 3. Memory map for STM32C55x/562

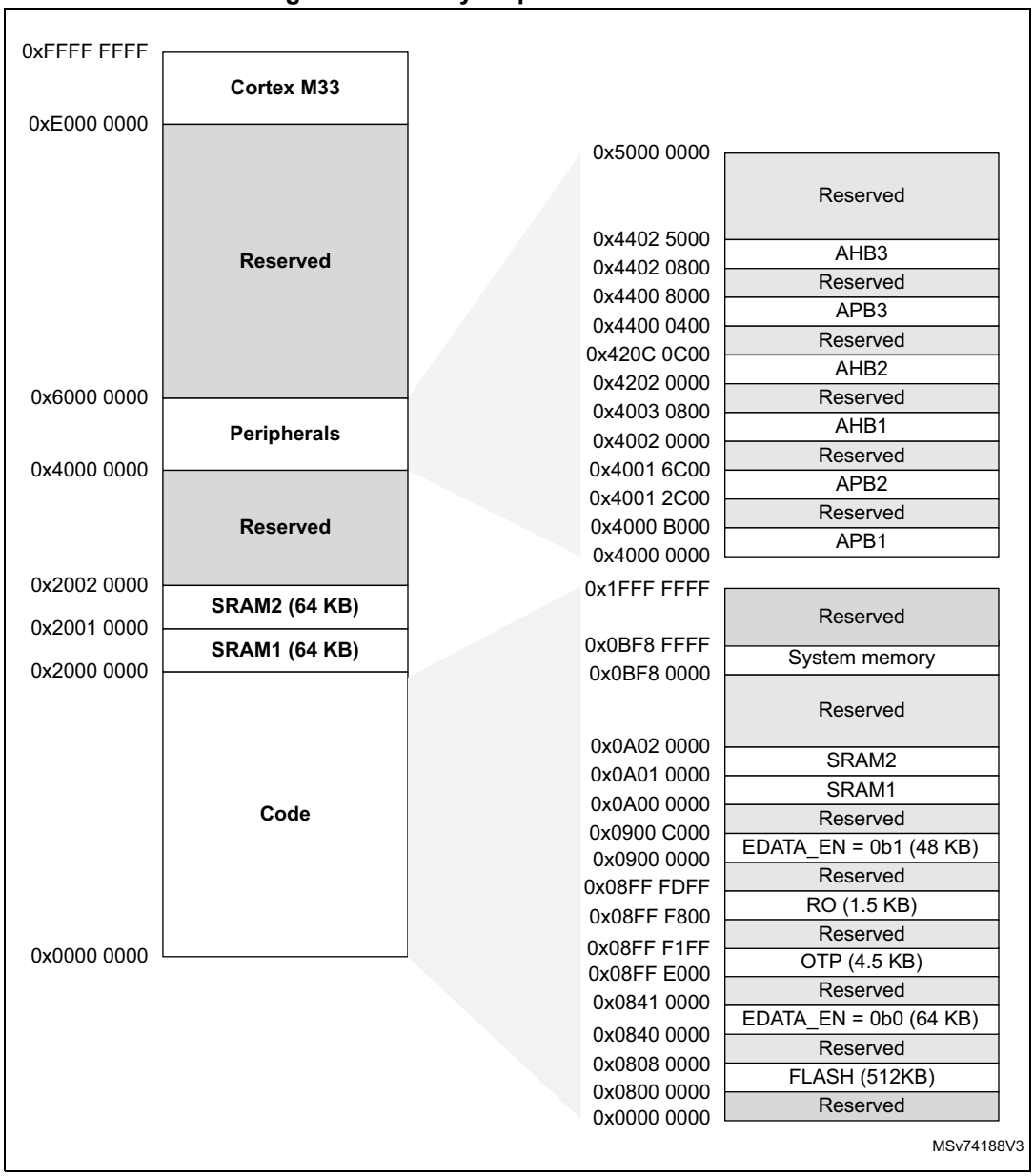
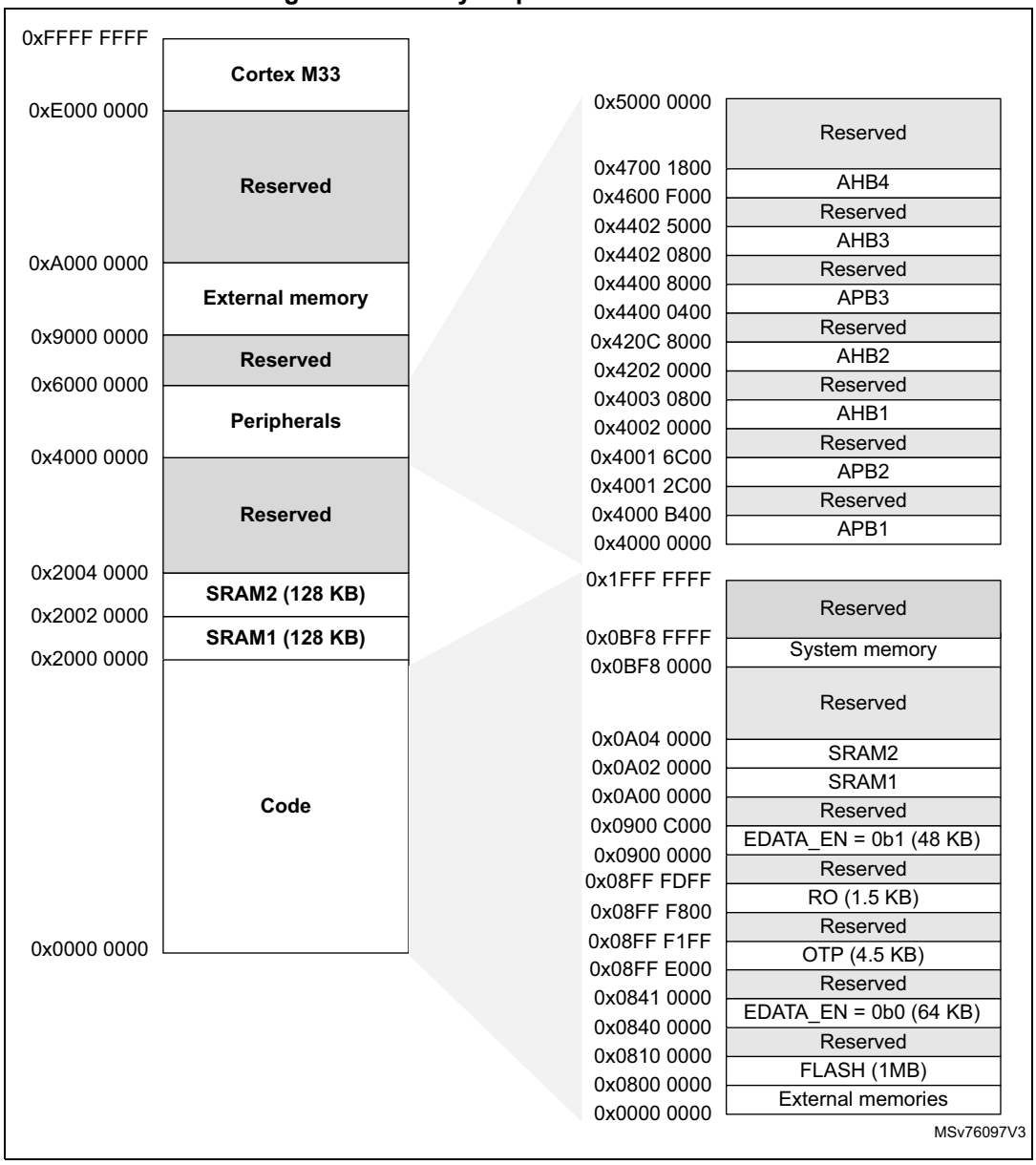


Figure 4. Memory map for STM32C59x/5A3



All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. The following table gives the boundary addresses of the peripherals available in the devices.

Table 3. Memory map and peripheral register boundary addresses

Bus	Boundary address	Peripheral	Peripheral register map	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
AHB4	0x4700 1800 - 0x47FF FFFF	Reserved	-	-	-	-
	0x4700 1400 - 0x4700 17FF	XSPI1	XSPI register map	-	-	X
	0x4600 F400 - 0x4700 13FF	Reserved	-	-	-	-
	0x4600 F000 - 0x4600 F3FF	DLYBXS1	DLYB register map	-	-	X
	0x4600 0000 - 0x4600 EFFF	Reserved	-	-	-	-
AHB3	0x4402 5000 - 0x44FF FFFF	Reserved	-	-	-	-
	0x4402 4000 - 0x4402 4FFF	DEBUG	DEBUG register map	X	X	X
	0x4402 2400 - 0x4402 3FFF	Reserved	-	-	-	-
	0x4402 2000 - 0x4402 23FF	EXTI	EXTI register map	X	X	X
	0x4402 1000 - 0x4402 1FFF	Reserved	-	-	-	-
	0x4402 0C00 - 0x4402 0FFF	RCC	RCC register map	X	X	X
	0x4402 0800 - 0x4402 0BFF	PWR	PWR register map	X	X	X
	0x4402 0000 - 0x4402 07FF	Reserved	-	-	-	-
APB3	0x4400 8000 - 0x4400 FFFF	Reserved	-	-	-	-
	0x4400 7C00 - 0x4400 7FFF	TAMP	TAMP register map	X	X	X
	0x4400 7800 - 0x4400 7BFF	RTC	RTC register map	X	X	X
	0x4400 4800 - 0x4400 77FF	Reserved	-	-	-	-
	0x4400 4400 - 0x4400 47FF	LPTIM1	LPTIM1 register map	X	X	X
	0x4400 2800 - 0x4400 43FF	Reserved	-	-	-	-
	0x4400 2400 - 0x4400 27FF	LPUART1	LPUART register map	X	X	X
	0x4400 0800 - 0x4400 23FF	Reserved	-	-	-	-
	0x4400 0400 - 0x4400 07FF	SBS	SBS register map	X	X	X
	0x4400 0000 - 0x4400 03FF	Reserved	-	-	-	-

Table 3. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Peripheral	Peripheral register map	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
AHB2	0x420C 8000 - 0x43FF FFFF	Reserved	-	-	-	-
	0x420C 7C00 - 0x420C 7FFF	CCB	CCB register map	-	-	X
	0x420C 4000 - 0x420C 7BFF	Reserved	-	-	-	-
	0x420C 2000 - 0x420C 3FFF	PKA	PKA register map	-	-	X
	0x420C 1000 - 0x420C 1FFF	Reserved	-	-	-	-
	0x420C 0C00 - 0x420C 0FFF	SAES	SAES register map	-	-	X
	0x420C 0800 - 0x420C 0BFF	RNG	RNG register map	X	X	X
	0x420C 0400 - 0x420C 07FF	HASH	HASH register map	X	X	X
	0x420C 0000 - 0x420C 03FF	AES	AES register map	X	X	X
	0x4202 DC00 - 0x420B FFFF	Reserved	-	-	-	-
	0x4202 DB00 - 0x4202 DBFF	ADCC3	ADC register map	-	-	X
	0x4202 D900 - 0x4202 DAFF	Reserved	-	-	-	-
	0x4202 D800 - 0x4202 D8FF	ADC3	ADC register map	-	-	X
	0x4202 8800 - 0x420B D7FF	Reserved	-	-	-	-
	0x4202 8400 - 0x4202 87FF	DAC1	DAC register map	X	X	X
	0x4202 8300 - 0x4202 83FF	ADCC12	ADC register map	X	X	X
	0x4202 8200 - 0x4202 82FF	Reserved	-	-	-	-
	0x4202 8100 - 0x4202 81FF	ADC2	ADC register map	-	X	X
	0x4202 8000 - 0x4202 80FF	ADC1	ADC register map	X	X	X
	0x4202 2000 - 0x4202 7FFF	Reserved	-	-	-	-
	0x4202 1C00 - 0x4202 1FFF	GPIOH	GPIO register map	X	X	X
	0x4202 1800 - 0x4202 1BFF	GPIOG	GPIO register map	-	-	X
	0x4202 1400 - 0x4202 17FF	GPIOF	GPIO register map	-	-	X
	0x4202 1000 - 0x4202 13FF	GPIOE	GPIO register map	X	X	X
	0x4202 0C00 - 0x4202 0FFF	GPIOD	GPIO register map	X	X	X
	0x4202 0800 - 0x4202 0BFF	GPIOC	GPIO register map	X	X	X
	0x4202 0400 - 0x4202 07FF	GPIOB	GPIO register map	X	X	X
	0x4202 0000 - 0x4202 03FF	GPIOA	GPIO register map	X	X	X

Table 3. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Peripheral	Peripheral register map	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
AHB1	0x4003 0800 - 0x41FF FFFF	Reserved	-	-	-	-
	0x4003 0400 - 0x4003 07FF	ICACHE	ICACHE register map	X	X	X
	0x4002 9400 - 0x4003 03FF	Reserved	-	-	-	-
	0x4002 8000 - 0x4002 93FF	ETHERNET MAC	ETHERNET register map	-	-	X
	0x4002 7000 - 0x4002 7FFF	Reserved	-	-	-	-
	0x4002 6000 - 0x4002 6FFF	RAMCFG	RAMCFG register map	X	X	X
	0x4002 3C00 - 0x4002 5FFF	Reserved	-	-	-	-
	0x4002 3800 - 0x4002 3BFF	CORDIC	CORDIC register map	X	X	X
	0x4002 3400 - 0x4002 37FF	Reserved	-	-	-	-
	0x4002 3000 - 0x4002 33FF	CRC	CRC register map	X	X	X
	0x4002 2400 - 0x4002 2FFF	Reserved	-	-	-	-
	0x4002 2000 - 0x4002 23FF	FLASH	FLASH register map	X	X	X
	0X4002 1000 - 0x4002 1FFF	LPDMA2	LPDMA register map	X	X	X
	0x4002 0000 - 0x4002 0FFF	LPDMA1		X	X	X
APB2	0x4001 6C00 - 0x4001 FFFF	Reserved	-	-	-	-
	0x4001 6400 - 0x4001 6BFF	USB RAM	USB register map	X	X	X
	0x4001 6000 - 0x4001 63FF	USB		X	X	X
	0x4001 4C00 - 0x4001 5FFF	Reserved	-	-	-	-
	0x4001 4800 - 0x4001 4BFF	TIM17	TIM17 register map	-	X	X
	0x4001 4400 - 0x4001 47FF	TIM16	TIM16 register map	-	X	X
	0x4001 4000 - 0x4001 43FF	TIM15	TIM15 register map	X	X	X
	0x4001 3C00 - 0x4001 3FFF	Reserved	-	-	-	-
	0x4001 3800 - 0x4001 3BFF	USART1	USART register map	X	X	X
	0x4001 3400 - 0x4001 37FF	TIM8	TIM8 register map	X	X	X
	0x4001 3000 - 0x4001 33FF	SPI1 / I2S1	SPI register map	X	X	X
	0x4001 2C00 - 0x4001 2FFF	TIM1	TIM1 register map	X	X	X
	0x4001 0000 - 0x4001 2BFF	Reserved	-	-	-	-

Table 3. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Peripheral	Peripheral register map	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
APB1	0x4000 B400 - 0x4000 FFFF	Reserved	-	-	-	-
	0x4000 AC00 - 0x4000 B3FF	FDCAN SRAM(1)	FDCAN register map	X	X	X
	0x4000 A800 - 0x4000 ABFF	FDCAN2	FDCAN register map	X	-	X
	0x4000 A400 - 0x4000 A7FF	FDCAN1	FDCAN register map	X	X	X
	0x4000 7C00 - 0x4000 A3FF	Reserved	-	-	-	-
	0x4000 7800 - 0x4000 7BFF	UART7	USART register map	-	-	X
	0x4000 6800 - 0x4000 77FF	Reserved	-	-	-	-
	0x4000 6400 - 0x4000 67FF	USART6	USART register map	-	-	X
	0x4000 6000 - 0x4000 63FF	CRS	CRS register map	X	X	X
	0x4000 5C00 - 0x4000 5FFF	I3C1	I3C register map	X	X	X
	0x4000 5800 - 0x4000 5BFF	I2C2	I2C register map	-	X	X
	0x4000 5400 - 0x4000 57FF	I2C1		X	X	X
	0x4000 5000 - 0x4000 53FF	UART5	USART register map	X	X	X
	0x4000 4C00 - 0x4000 4FFF	UART4		X	X	X
	0x4000 4800 - 0x4000 4BFF	USART3		-	X	X
	0x4000 4400 - 0x4000 47FF	USART2		X	X	X
	0x4000 4000 - 0x4000 43FF	COMP12(2)	COMP register map	X	X	X
	0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3	SPI register map	-	X	X
	0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		X	X	X
	0x4000 3400 - 0x4000 37FF	OPAMP1	OPAMP register map	X	-	-
	0x4000 3000 - 0x4000 33FF	IWDG	IWDG hardware configuration register (IWDG_HWCFGR)	X	X	X
	0x4000 2C00 - 0x4000 2FFF	WWDG	WWDG register map	X	X	X
	0x4000 1C00 - 0x4000 2BFF	Reserved	-	-	-	-
	0x4000 1800 - 0x4000 1BFF	TIM12	TIM12 register map	X	X	X
	0x4000 1400 - 0x4000 17FF	TIM7	TIM6/TIM7 register map	X	X	X
	0x4000 1000 - 0x4000 13FF	TIM6		X	X	X
	0x4000 0C00 - 0x4000 0FFF	TIM5	TIM5 register map	-	X	X
	0x4000 0800 - 0x4000 0BFF	TIM4	TIM4 register map	-	-	X
	0x4000 0400 - 0x4000 07FF	TIM3	TIM3 register map	-	-	X
	0x4000 0000 - 0x4000 03FF	TIM2	TIM2 register map	X	X	X

1. For STM32C55x/562 devices, FDCAN RAM boundary end address is 0x4000 AFFF.

2. The COMP2 peripheral is only available in STM32C53x/542.

2.2.3 Embedded SRAM

Table 4. SRAM sizes

SRAM	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
SRAM1	32 Kbytes	64 Kbytes	128 Kbytes
SRAM2	32 Kbytes with ECC	64 Kbytes with ECC	128 Kbytes with ECC

These SRAMs can be accessed as bytes, half-words (16 bits), or full words (32 bits). These memories can be addressed both by CPU and DMAs.

The CPU can access the SRAM1 and SRAM2 through the system bus or through the C-bus depending on the selected address. SRAM features are detailed in *Section 5.3.1: Internal SRAMs features*.

2.2.4 Flash memory overview

The flash memory is composed of two distinct physical areas:

- The main flash memory block that contains the application program and user data.
- The information block that is composed of the following parts:
  - option bytes for hardware and memory protection user configuration
  - system memory that contains ST proprietary code
  - OTP (one-time programmable) area

The flash memory interface implements instruction access and data access based on the AHB protocol. It also implements the logic necessary to carry out the flash memory operations (program/erase) controlled through the flash memory registers plus security access control features. Refer to *Section 6: Embedded flash memory (FLASH)* for more details.

3 System security

The STM32C5 is designed with a comprehensive set of security features.

These security features simplify the process of evaluating IoT devices against security standards. They also significantly reduce the cost and complexity of software development for OEM and third-party developers, by facilitating the reuse, improving the interoperability, and minimizing the API fragmentation.

This section explains the different security features available on the STM32C5 devices.

3.1 Key security features

- Resource isolation using privilege mode of Cortex-M33
- Temporal isolation: boot levels are isolated thanks to the HDPL (hide protect level) monotonic counter
- General purpose cryptographic acceleration
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes
  - Secure AES 256-bit security coprocessor, supporting ECB, CBC, CTR, GCM, and CCM chaining modes with side-channel counter-measures and mitigations on STM32C59x and STM32C5Ax.
  - HASH processor, supporting SHA-1 checksums and SHA-2 secure hash (SHA2-224, SHA2-256) for STM32C53x/C54x/C55x/C56x, and SHA2-384, SHA2-512 in addition for STM32C59x/C5Ax.
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures and mitigations when manipulating secrets on STM32C59x/C5Ax.
  - True random number generator (RNG), NIST SP800-90B pre-certified
  - Coupling and chaining bridge (CCB) for special coupling and chaining operations (involving PKA, SAES, and RNG peripherals) to protect private keys used in PKA-protected operations on STM32C59x/C5Ax.
- Secure storage, featuring:
  - Nonvolatile on-chip secure storage, protected with HDP areas
  - Battery-powered volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in the AES engines
  - On-chip enhance storage technology, using hardware secret nonvolatile derived hardware unique keys (DHUK), and application-defined volatile boot hardware key (BHK), both loadable by hardware to the DPA-resistant SAES engine
- Device 96-bit unique ID
- Readout protection-based life cycle scheme
  - Debug protection, depending on the RDP level
  - Password-based RDP level regressions
- Tamper and protection against frequency attacks

3.2 Resource isolation

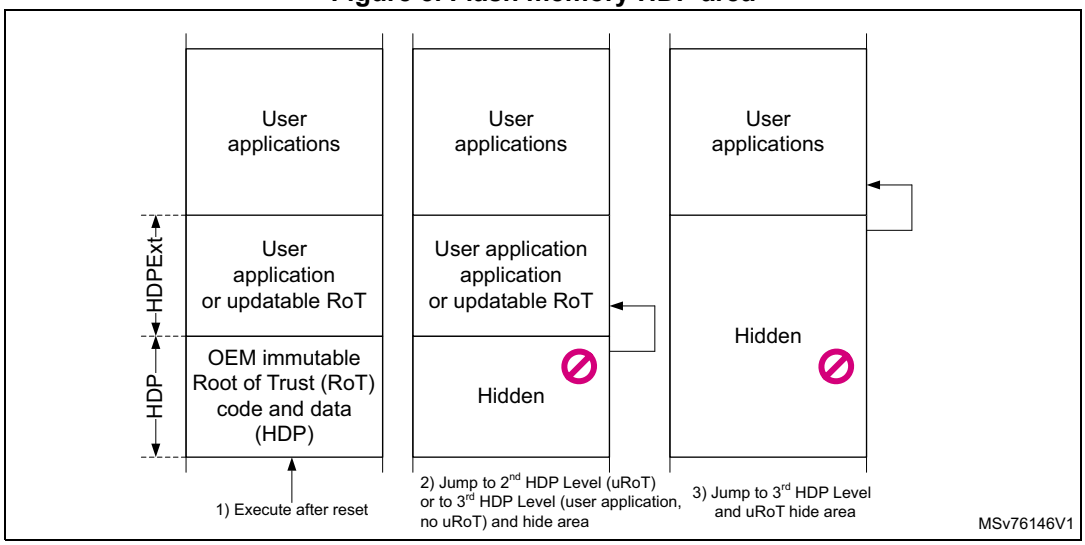
3.2.1 Temporal isolation using secure hide protection (HDP)

The embedded flash memory allows an HDP (high-density protection) area to be defined for each watermarked-secure area of each bank, with a granularity of 8-Kbyte sectors.

The code executed in this HDP area, with its related data and keys, can be hidden after boot until the next system reset.

The following figure shows the hidden protection principle.

Figure 5. Flash memory HDP area



The activation of the HDP area in the user flash memory is related to HDPL1. It means that as soon as HDPL becomes equal or greater than 2, data read, write, and instruction fetch on the area defined by HDPx_PSTRT and HDPx_PEND in FLASH_HDPxR_PRG option bytes are denied until the next device reset.

The END of the HDPx areas can be extended dynamically by the application using the FLASH_HDPEXTR register. This extension prevents access to protected sectors under HDPL3.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area).

3.2.2 Resource isolation using Cortex privileged mode

The hardware and software resources of STM32C5 devices can be partitioned to restrict access to software running in Cortex privileged mode.

Thanks to this hardware isolation technology, a critical code or data can be protected against intentional or unintentional tampering from the more exposed unprivileged code.

Memory and peripheral privileged allocation using MPU

The Cortex-M33 MPU divides the unified memory into eight regions, each aligned to a multiple of 32 bytes. Each memory region can be programmed to generate faults when accessed inappropriately by unprivileged software.

3.3 Secure execution

Through a mix of special software and hardware features, the devices ensure the correct operation of their functions against abnormal situations caused by programmer errors, software attacks through network access or local attempt for tampering code execution.

This section describes the hardware features specifically designed for secure execution.

3.3.1 Memory protection unit (MPU)

The Cortex-M33 includes a memory protection unit (MPU) that can restrict the read and write accesses to memory regions (including regions mapped to peripherals), based on one or more of the following parameters:

- Cortex-M33 operating mode (privileged, unprivileged)
- Data/instruction fetch

The memory map and the programming of the MPU split memory into up to eight regions.

3.3.2 Embedded flash memory write protection

The write protection (WRP) prevents illegal or unwanted write/erase to selected sectors of the embedded flash memory user area (system area is permanently write protected).

Write-protected sectors can be modified through option byte changes only when RDP_LEVEL = L0.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area). For more information, see Section 6.5.5: Write protection.

3.3.3 Tamper detection and response

Principle

The devices include protection of critical security assets against attacks, with the following features:

- Erase of device secrets upon tamper detection
- Improved guarantee of safe execution for the CPU and its associated security peripherals, including:
  - Out-of-range clocking (LSE) detection
  - Security watchdog IWDG clocked by the internal oscillator LSI
  - Possible selection of internal oscillator HSI as system clock

See *Section 40: Tamper and backup registers (TAMP)*, for more details.

Tamper detection sources

The devices support three input tampers that can be mapped on five different pins.

Detection time is programmable, and a digital filtering is available (tamper triggered after two false comparisons in four consecutive comparison samples).

Note: Timestamps are automatically generated when a tamper event occurs.

The internal tamper sources are listed in the table below.

Table 5. Internal tampers in TAMP

Tamper input	Tamper source
itamp3	LSE monitoring(1)
itamp4	HSE monitoring
itamp5	RTC calendar overflow (rtc_calovf)
itamp6	JTAG/SWD access
itamp9	Fault generation for cryptographic peripherals (SAES, PKA, AES, RNG)(2)
itamp11	IWDG timeout and potential tamper (IWDG reset when at least one enabled tamper flag is set)

1. LSE missing or over frequency detection (> 2 MHz), glitch filter (> 2 MHz).
2. Only RNG fault on STM32C53x/C54x/C55x/C56x.

Response to tampers

Each source of tamper in the device can be configured to trigger the following events:

- Generate an interrupt, capable of waking up the device from Stop and Standby modes (see TAMPxMSK bits in TAMP_CR2 register).
- Generate a hardware trigger for the low-power timers.
- Erase device secrets if the corresponding TAMPxPOM bit is cleared in TAMP_CR2 (for tamper pins) or TAMP_CR3 (for internal tamper). These erasable secrets are:
  - Symmetric keys stored in backup registers (x32), and in HASH
  - Other secrets stored in SRAM2 and CPU instruction cache memory

Read/write accesses by software to all these secrets can be blocked, by setting the BKBLOCK bit in TAMP_CR2. The device secrets access is possible only when BKBLOCK is cleared, and no tamper flag is set for any enabled tamper source.

Note: Device secret erase is also triggered by setting the BKERASE bit in TAMP_CR2, or by performing a RDP_LEVEL regression as defined in Section 3.7.1.

Device secrets are not reset by system reset or when the device wakes up from Standby mode.

Software filtering mechanism

Each tamper source can be configured not to launch an immediate erase, by setting the corresponding TAMPxPOM bit in TAMP_CR2 (for external tamper pin) or TAMP_CR3 (for internal tamper).

In such a situation, when the tamper flag is raised, access to below secrets is blocked until all tamper flags are cleared:

- Backup registers, SRAM2: read-as-zero, write-ignored
- AES, SAES, CCB, HASH, PKA peripherals : automatically reset by RCC
- The CPU instruction cache (ICACHE) is erased

Once the application, notified by the tamper event, analyzes the situation, there are two possible cases:

- The application launches secrets erase with a software command (confirmed tamper).
- The application clears the flags to release the blocking of secrets (false tamper).

Note: If the tamper software fails to react to such a tamper flag, the tamper input associated to IWDG reset must trigger automatically the erase of secrets.

Tamper detection and low-power modes

The effect of low-power modes on a tamper detection is summarized in the following table.

Table 6. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect on tamper detection features TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on tamper detection features, except for level detection with filtering that remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Stop mode.
Standby	No effect on tamper detection features, except for level detection with filtering that remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Standby mode.

3.4 Secure storage

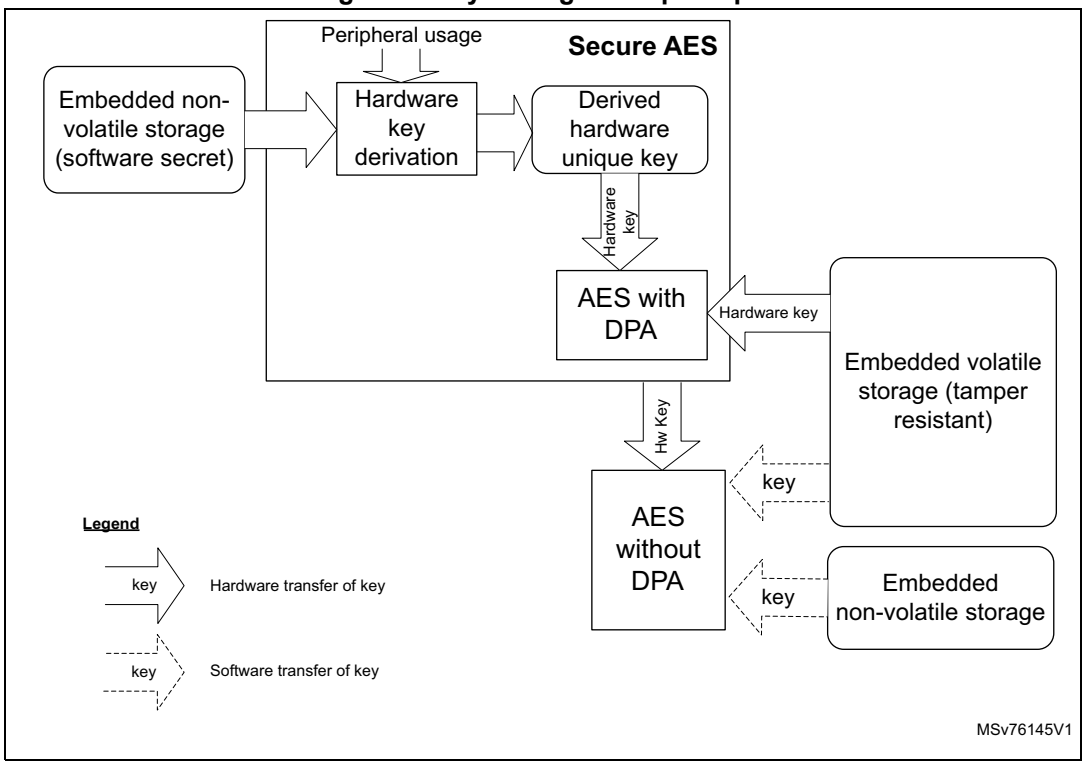
This section applies only to STM32C59xx and STM32C5Axx devices.

A critical feature of security systems is how long-term keys are stored, protected, and provisioned. Such keys are typically used for loading a boot image or handling critical user data.

Figure 6 shows how the key management service application can use the AES engine, for example, to compute external image decryption keys. Nonvolatile keys are stored in the flash memory in a dedicated area with access control (*Section 3.2.1: Temporal isolation using secure hide protection (HDP)*), while volatile key storage consists of tamper-protected SRAM.

Figure 6 also shows keys that are manipulated by software, or keys that are managed only by hardware (such as DHUK). More information about those hardware keys can be found in *Section 3.4.1: Hardware secret key management*.

Figure 6. Key management principle



3.4.1 Hardware secret key management

As shown in the previous figure, the devices provide better protection for application keys by using hardware secret keys. This AES key can be made usable by the application, without exposing it in clear text (unencrypted). Such keys also become immediately unusable in case of tampering.

STM32C59xx/C5Axx devices embed a unique source of hardware secret key:

- **DHUK**: derived keys based on a 256-bit nonvolatile device-unique secret in flash memory. The flash memory provides a value provisioned during product manufacturing (called RHUK). The generation of DHUK key considers the NEXT-HDPL (temporal isolation counter) and key use state (KMOD).
- BHK: 256-bit application key stored in tamper-resistant volatile storage in TAMP. This key is written at boot time, then read/write locked to application until next reset.
- XORK: result of an XOR of BHK and DHUK

These keys can be used in the following modes:

- As a normal key, loaded into write-only key registers (software key mode).
- As an encryption/decryption key for another key, used in the DPA-resistant SAES (wrapped key mode).
- As an encryption or decryption key for another key, used in a faster AES engine (shared key mode)

3.5 Unique ID

The devices store a 96-bit ID that is unique to each device (see *Section 50.1: Unique device ID register (96 bits)*).

Application services can use this unique identity key to identify the product in the cloud network, or make it difficult for counterfeit devices or clones to inject untrusted data into the network.

Alternatively, the 256-bit device unique key (DHUK) can be used (*Section 3.4.1: Hardware secret key management*).

3.6 Crypto engines

Refer to the product datasheet to identify the cryptographic engines embedded in the selected devices.

The devices implement advanced cryptographic algorithms featuring key sizes and computing protection as recommended by national security agencies such as NIST for the United States, BSI for Germany, and ANSSI for France. These algorithms support privacy, authentication, integrity, entropy, and identity attestation.

The cryptographic engines embedded in STM32 reduce the weaknesses in the implementation of critical cryptographic functions. They prevent the use of weak cryptographic algorithms and key sizes. They enable lower processing times, and lower power consumption during cryptographic operations. These engines offload computations from Cortex-M33, particularly true for asymmetric cryptography.

For product certification purposes, ST can provide certified device information detailing how these security functions are implemented and validated.

For more information about cryptographic engine processing times, refer to their respective sections in the reference manual.

3.6.1 Cryptographic engines features

Figure 7 lists the accelerated cryptographic operations available in the devices. Two AES accelerators are supported.

Note: Additional operations can be added by using the firmware.

The PKA can accelerate asymmetric cryptographic operations (such as key pair generation, ECC scalar multiplication, point on curve check). See *Section 30: Public key accelerator (PKA)* for more details.

Table 7. Accelerated cryptographic operations

Operations	Algorithm	Specification	Key lengths (in bit)	Modes
Get entropy	RNG	NIST SP800-90B(1)	N/A	Software and hardware(2) modes running in parallel

Table 7. Accelerated cryptographic operations (continued)

Operations	Algorithm	Specification	Key lengths (in bit)	Modes
Encryption, decryption	AES(3)	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D	128, 256	GCM, CCM
Cipher-based message authentication code		NIST SP800-38D	128, 256	GMAC
Checksum	SHA-1	FIPS PUB 180-4	N/A	Digest 160-bit
Cryptographic hash	SHA-2			SHA-224, SHA-256, SHA-384, SHA-51
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	Short, long (> 64 bytes)	-
Encryption/decryption key-pair generation	RSA(4)	IETF RFC 8017	Up to 4160	RSAES-OAEP
		NIST SP800-56B		
Signature with hashing	RSA(4)	IETF RFC 8017 FIPS PUB 186-4	Up to 4160	PKCS1-v1_5, PSS
Signature verification	ECDSA	ANSI X9.62 IETF RFC 7027 FIPS PUB 186-4	Up to 640	Refer to *Table 269: Family of supported curves for ECC operations*t for details.
Key agreement	ECDH	ANSI X9.42		

1. Certifiable by using STMicroelectronics-reviewed documents.
2. Random number distribution to SAES and PKA through a dedicated hardware bus.
3. ECB and CBC chaining modes are protected against side-channel and timing attacks in SAES (see *Section 3.6.2: Secure AES co-processor (SAES)*.
4. Private key cryptography is protected against side-channel and timing attacks.

3.6.2 Secure AES co-processor (SAES)

The devices provide an additional on-chip hardware AES encryption and decryption engine, that implements countermeasures and mitigations against power and electromagnetic side-channel attacks.

Clocked by the system clock, the SAES delivers excellent performance for a DPA-resistant hardware accelerator. The SAES engine supports 128-bit or 256-bit keys in electronic code book (ECB), cipher block chaining (CBC), (CTR), (GCM), (CCM), (GMAC) modes.

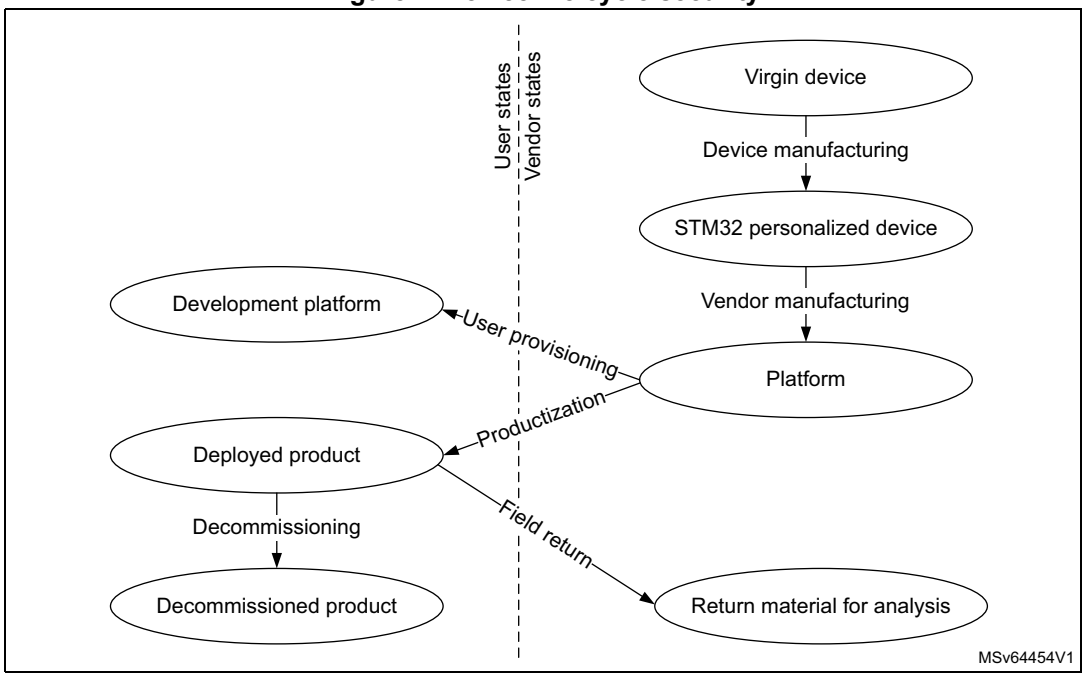
As shown in *Section 3.4: Secure storage*, the SAES can be used for highly secure on-chip storage for sensitive information. It can also be made secure-only.

For more information, refer to *Section 27: AES hardware accelerator (AES)*.

3.7 Product life cycle

A typical IoT device life cycle is summarized in the figure below. For each step, the devices propose secure life cycle management mechanisms embedded in the hardware.

Figure 7. Device life cycle security



More details on the various phases and associated transitions, found either at the vendor or end-user premises, are summarized in the table below.

Table 8. Main product life cycle transitions

Transitions	Description
Device manufacturing	STMicroelectronics creates new STM32 devices, always checking for manufacturing defects. During this process STM32 is provisioned with ROM firmware.
Vendor manufacturing	One (or more) vendor is responsible for the platform assembly, initialization, and provisioning before delivery to the end user. This end user can use the final product (“production” transition) or he/she can use the platform for software development (“user provisioning” transition).
Production	The end-user gets a product ready for use. All security functions of the platform are enabled, the debugging/testing features are restricted/disabled, and unique boot entry to immutable code is enforced.
User provisioning	Platform vendor prepares an individual platform for development, not to be connected to a production cloud network.
Field return or decommissioning	These are one-way transitions, with devices kept in user premises or returned to the manufacturer. In both cases, all data including user data is destroyed, therefore the devices lose the ability to operate securely (like connecting to a managed IoT network).

The features described hereafter contribute to secure the device life cycle.

3.7.1 Life cycle management with readout protection (RDP)

The readout protection mechanism (full hardware feature) controls the access to the devices debug, test and provisioned secrets, as summarized in the table below.

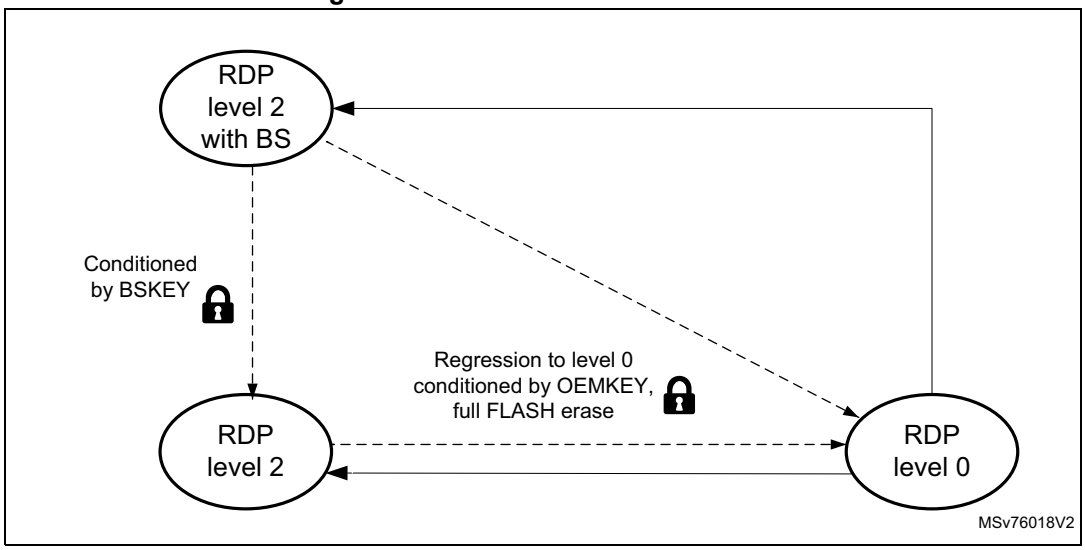
Table 9. Typical product life cycle phases

RDP protection level		Debug	Comments
Level 0	Device open	Available	OEM and BS unlocking/locking keys can be provisioned in the flash memory user options. The keys programming, while not readable, can be verified under reset through debugger.
Level 2 with boundary scan	Device closed	None (boundary scan available under reset)	Boot address must target the user flash memory. Option-byte modification limited, RDP_LEVEL not modifiable hence RDP level 2 with boundary scan cannot be changed, unless the OEM unlocking key is activated or the BS locking key to progress to level 2 (see *Table 10*).
Level 2	Device closed	None (JTAG fuse)	Boot address must target the user flash memory. Option-byte modification limited, RDP_LEVEL not modifiable hence RDP level 2 cannot be changed, unless the OEM unlocking key is activated (see *Table 10*).

Note: OEMKEY option byte can be modified when RDP = 0 only.

The supported transitions, summarized in the following figure can be requested (when available) through the debug interface, or via the system bootloader to go to RDP L2_wBS or L2. Additionally, through JTAG/SWD under reset, once in L2_wBS or L2, it is possible to either progress to L2 or regress to L0.

Figure 8. RDP level transition scheme



As shown in the previous figure, the user flash memory is automatically erased, during an RDP regression. These regressions can be conditioned on dedicated 128-bit password keys, if provisioned by the OEM (see the next subsection). During the regression from RDP L2 or L2_wBS to RDP L0, the FLASH is erased. The OTP area in the flash memory is kept, all SRAMS and targeted device secrets are erased. Hence, no secrets must be stored in the OTP as they are revealed after a regression to RDP L0. These secrets, also erased as response to tamper, are defined in *Section 3.3.3: Tamper detection and response*.

For more details on RDP, refer to *Section 6: Embedded flash memory (FLASH)*.

RDP unlocking sequences

The use of the OEM and BS password keys described in *Figure 8* is described hereafter.

Note: The devices support password-based RDP L2 (or L2_wBS) regression to L0. This L2 regression erases the application code.

Details on the password-based regression can be found in the tables below.

Table 10. OEM RDP unlocking methods

OEM password options		
OEM LOCK	**RDP level**	**RDP regression**
1	L2_wBS	Automatic regression to L0 triggered upon successful OEM unlock sequence
	L2	
0	L2_wBS	Regression to L0 impossible. Possibility to transition to RDP L2.
	L2	Regression to L0 impossible. RDP L2 remains a permanent state.

Table 11. BS RDP locking methods

BS password options	
Initial RDP level	**RDP regression**
L2_wBS, with BSKEY not provisioned	Automatic progression to L2 triggered upon successful BS lock sequence using default key
L2_wBS, with BSKEY provisioned	Automatic progression to L2 triggered upon successful BS lock sequence using custom key provisioned under RDP L0

- OEM unlock sequence, starting at RDP L2 or L2_wBS:
  - Shift the password key through JTAG/SWD under reset.
  - If this key matches the OEMKEY provisioned in the device, the device automatically triggers a regression sequence to level 0. After the regression is completed, a power-on reset has to be performed by the user.
  - In case of mismatch value, the RDP regression is blocked, and RDP L2 protections are enforced until the next power-on reset.
- BS lock sequence, starting at RDP L2_wBS:
  - Shift the password key through JTAG/SWD under reset (see the note below). A default key is provisioned in ST factory 0xAAAAAAAA. The user can provision a custom one under RDP L0.

– If this key matches the BSKEY provisioned in the device, the device automatically triggers a progression sequence to L2. After the progression is completed, a power-on reset has to be performed by the user.
– In case of a mismatch value, the RDP progression is blocked until the next power-on reset.

Note: Unlocking the device with a password is possible only once per power cycle.

Shifting the OEM password key through JTAG/SWD corresponds to writing four 32-bit key words: AUTH_KEY[31:0], then AUTH_KEY[63:32], AUTH_KEY[95:64], then AUTH_KEY[127:96] in the DBGMCU_DBG_AUTH_HOST register.

Shifting the BS password key through JTAG/SWD corresponds to writing a 32-bit key word: BS_KEY[31:0] in the DBG_BSKEY_PWD register.

JTAG 32-bit device specific ID

Unless the JTAG port is deactivated (OEMLOCK = 0 and RDP L2), a 32-bit device specific quantity can always be read through the JTAG port. This information is stored in DBGMCU_DBG_AUTH_DEVICE.

The OEM can use this 32-bit information to derive the expected OEM password keys to unlock this specific device.

3.8 Summary of protection mechanism

The different security mechanism may reset, block or erase device secrets or sensitive peripherals. To be more explicit on those effects, please refer to the below table:

Table 12. Summary of protected assets and protection mechanism(1)

Assets	Regression to RDP L0	Potential tamper	Confirmed tamper	POR	System reset	RTC domain reset
SRAM1	E	-	-	-	Erase if SRAM1_RST bit is ‘0’ (erase done at OPT activation)	-
SRAM2	E	B	E	E	Erase if SRAM2_RST bit is ‘0’ (erase done at OPT activation)	-
FLASH	E	-	-	-	-	-
FLASH EDATA page	E	-	-	-	-	-
Flash OTP	-	-	-	-	-	-
BKUP registers	R	B(2)	R/B(2)	R(3)	-	R
ICACHE	E	E	E	E	E	-
PKA RAM	E	E/B(4)	E/B(4)	E	E	-
FDCAN RAM	-	-	-	-	-	-
USB RAM	-	-	-	-	-	-
ETH RAM	-	-	-	-	-	-
AES/HASH/SAES/ CCB registers	R	R	R	R	R	-

Table 12. Summary of protected assets and protection mechanism(1)

Assets	Regression to RDP L0	Potential tamper	Confirmed tamper	POR	System reset	RTC domain reset
PKA registers	R	R	R	R	R	-
RNG registers	R	-	-	R	R	-

1. E = Erase, B = Access blocked, R = Reset, N.A. = Not applicable, “-” = No effect
2. Bocked up to clearing of TAMP status register flag.
3. Erase after POR
4. Blocked during erasing.

4 Boot modes

At startup, the BOOT0 pin or option bit (based on the value of the BOOT_SEL option bit) and BOOTADD[31:8] option bytes are used to select the boot memory address that includes:

- Boot from any address in user flash memory, excluding EDATA sectors.
- Boot from system memory
  - Bootloader

When boot from user flash is selected, the boot address is defined by BOOTADD. This address can be locked thanks to BOOT_LOCK.

Embedded bootloader

The embedded bootloader is located in the system memory and is programmed by ST during production. It is used to reprogram the flash memory using USART, SPI, FDCAN, or USB in device mode through the DFU (device firmware upgrade).

Refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).

4.1 STM32C5 boot modes

The following table provides the details of the boot mode for the STM32C5 devices.

Table 13. STM32C5 boot modes

RDP_LEVEL	BOOT_SEL option bit	BOOT0 option bit	BOOT0 pin	EMPTY flag value(1)	Boot area
Level 0	1	x	0	x	Boot address defined by user option byte BOOTADD[31:8]
			1		Bootloader
	0	0	x	0	Boot address defined by user option byte BOOTADD[31:8]
				1	Bootloader
		1		x	Bootloader
Level 2 with boundary scan and level 2	x	x	x	x	Boot address defined by user option byte BOOTADD[31:8]

1. The boot address, defined by BOOTADD option byte, is considered empty only if the FLASH word is equal to 0xFFFF FFFF. In this case, at option byte loading, the EMPTY bit is set in FLASH_ACR.

Empty check

During the option byte loading phase, after loading all options, the flash memory interface checks whether the boot location in the main memory, indicated by BOOTADD[31:8], is programmed. The result of this check, with the boot0 information, is used to determine where the system should boot from. It prevents the system from booting from the main flash memory area when, for instance, no user code has been programmed.

The main flash memory empty check status can be read from the EMPTY bit in the FLASH access control register (FLASH_ACR). Software can modify it by writing an appropriate value to the EMPTY bit.

The BOOT_LOCK, which locks modifications of option bytes involved in BOOT selection, has no impact on this empty check mechanism

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. It is used to reprogram the flash memory by using USART, SPI, FDCAN, or USB in device mode through the DFU (device firmware upgrade).

For further information, refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).

5 RAMs configuration controller (RAMCFG)

5.1 RAMCFG introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1 and SRAM2).

5.2 RAMCFG main features

The internal SRAM supports some of the features listed hereafter, configured in RAMCFG:

- Error code correction (ECC):
  - Single-error detection and correction with interrupt generation
  - Double-error detection with interrupt or NMI generation
  - Status with failing address
- Write protection (1-Kbyte granularity)
- SRAM software erase

5.3 RAMCFG functional description

5.3.1 Internal SRAMs features

Two SRAMs are embedded in the device, each with specific features:

- SRAM1 and SRAM2 are the main SRAMs (see *Table 15*), made of several blocks that can be powered down in Stop mode to reduce consumption
- SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the flash memory user option bytes. SRAM1 is erased when a system reset occurs if the SRAM1_RST option bit is selected in the flash memory user option bytes. Refer to *Section 6: Embedded flash memory (FLASH)* for more details.
- SRAM2 is protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. They are also erased by hardware in case of a backup domain reset. Refer to *Section 40: Tamper and backup registers (TAMP)* for more details.
- The RAMCFG embeds registers related to the internal SRAMs ECC, write protection, and software erase.

Table 14. SRAM density (Kbytes)

SRAM feature	SRAM1	SRAM2
STM32C53x/542	32	32
STM32C55x/562	64	64
STM32C59x/5A3	128	128

The table below summarizes the features supported by each internal SRAM

Table 15. Internal SRAMs features

SRAM feature	SRAM1	SRAM2
Optional retention in Standby mode	-	-
Optional retention in Stop mode	X	X
Erased with RDP level regression	X	X
Optionally erased with tamper detection	-	X
Optionally erased with system reset	X	X
Software erase	X	X
ECC	-	X
Write protection	-	X

5.3.2 Error code correction (SRAM2)

The ECC is supported by SRAM2 when enabled with the SRAM2_ECC user option bits. Refer to *Section 6: Embedded flash memory (FLASH)* for more details.

Seven ECC bits are added per 32 bits of SRAM, allowing 2-bit error detection and 1-bit error correction on memory read access.

As the ECC is calculated and checked for a 32-bit word, byte and half-word write accesses are managed by the SRAM interface by first reading the whole word, then write the word again with the new byte/half-word value. ECC double errors are also detected during these bytes or half-word AHB write accesses (read/modify/write done by interface). The byte or half-word write access latency is two AHB clock cycles.

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM two AHB clock cycles after the SRAM AHB is released (when the SRAM is no more accessed).

Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDC/CSEDC bits are set in RAMCFG_MxISR and RAMCFG_MxICR respectively. An interrupt is generated if enabled by the SEIE bit in RAMCFG_MxIER. The failing address is stored in RAMCFG_MxSEAR if the ALE bit is set in RAMCFG_MxCR.

Caution: Single errors cannot be detected when the SEDC bit is set.

When a double error is detected, the DED and CDED bits are set in RAMCFG_MxISR and RAMCFG_MxICR respectively. An interrupt or NMI is generated if enabled by the DEIE or ECCNMI bit in RAMCFG_MxIER. The failing address is stored in RAMCFG_MxDEAR if the ALE bit is set in RAMCFG_MxCR.

Caution: Double errors cannot be detected when the DED bit is set.

SRAM2 with ECC memory map

When the ECC is enabled for SRAM2, the full SRAM2 has ECC.

The figure below shows the SRAM areas, when ECC is enabled for both SRAM1 and SRAM2.

Table 16. SRAM with ECC memory map

Product	SRAM	Address over C-bus	Address over S-bus	SRAM size	
STM32C53x/542	SRAM1	0x0A00 0000 - 0x0A00 7FFF	0x2000 0000 - 0x2000 7FFF	32 Kbytes	
	SRAM2	0x0A00 8000 - 0x0A00 FFFF	0x2000 8000 - 0x2000 FFFF	32 Kbytes	ECC
STM32C55x/562	SRAM1	0x0A00 0000 - 0x0A00 FFFF	0x2000 0000 - 0x2000 FFFF	64 Kbytes	
	SRAM2	0x0A01 0000 - 0x0A01 FFFF	0x2001 0000 - 0x2001 FFFF	64 Kbytes	ECC
STM32C59x/5A3	SRAM1	0x0A00 0000 - 0x0A01 FFFF	0x2000 0000 - 0x2001 FFFF	128 Kbytes	
	SRAM2	0x0A00 0002 - 0x0A03 FFFF	0x2002 0000 - 0x2003 FFFF	128 Kbytes	ECC

When ECC is enabled by user option bits, the ECCE bit is automatically set after system reset in the related RAMCFG_MxCR.

The ECC can be deactivated by executing the following software sequence:

1. Write 0xAE in RAMCFG_MxECCKEYR.
2. Write 0x75 in RAMCFG_MxECCKEYR.
3. Write 0 in RAMCFG_MxCR.

Testing ECC mechanism

To test the ECC mechanism, 1 or 2 bits by word for single- or double-error test respectively.

The procedure to check ECC is the following:

1. On an erased memory, write data with ECC ON.
2. Disable ECC.
3. Write the same data with 1- or 2-bit modification (for single or double error test respectively).
4. Enable ECC.
5. Read the data. Enabled interrupt is generated because of a single or double error.

5.3.3 Write protection (SRAM2)

SRAM2 is made up of 1-Kbyte pages, each of which can be write-protected by setting its corresponding PxWP bit in RAMCFG_M2WPR1, RAMCFG_M2WPR2, RAMCFG_M2WPR3[(a)], and RAMCFG_M2WPR4[(a)].

5.3.4 Software erase

SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in RAMCFG_MxERKEYR.
2. Write 0x53 in RAMCFG_MxERKEYR.
3. Write 1 to the SRAMER bit in RAMCFG_MxCR.

a. This register is available only on STM32C59x/5A3 devices. If not present, consider this register as reserved, and keep its bits at reset value.

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is ongoing.

The total duration of each SRAM erase is N AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read or written while an erase is ongoing, wait states are inserted on the AHB bus until the end of the erase operation.

5.4 RAMCFG low-power modes

Table 17. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit Sleep mode.
Stop	The content of RAMCFG registers is kept.
Standby	The RAMCFG peripheral is powered down and must be reinitialized after exiting Standby mode.

5.5 RAMCFG interrupts

The table below gives the list of RAMCFG interrupt requests.

Table 18. RAMCFG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
RAMCFG	ECC single-error detection and correction	SEDC	SEIE	Write 1 in CSEDC	Yes	No	No
	ECC double-error detection	DED	DEIE = 1 and ECCNMI = 0	Write 1 in CDED	Yes	No	No
NMI	ECC double-error detection	DED	ECCNMI	Write 1 in CDED	Yes	No	No

5.6 RAMCFG registers

In the registers described below, x refers to:

- SRAM1 when x = 1
- SRAM2 when x = 2

5.6.1 RAMCFG memory x control register (RAMCFG_MxCR)

Address offset: 0x000 + 0x40 * (x - 1), (x = 1, 2)

Reset value: 0x0000 000X

ECCE reset value depends on the ECC enable user option bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
							rs				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_MxERKEYR register. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: Address latch enable

0: Failing address not stored in the SRAMx ECC single/double error address registers

1: Failing address stored in the SRAMx ECC single/double error address registers

Note: This bit is reserved and must be kept at reset value in SRAM1 control register.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable.

This bit reset value is defined by the user option bit configuration. When set, it can be cleared by software only after writing the unlock sequence in the RAMCFG_MxECCKEYR register.

0: ECC disabled

1: ECC enabled

Note: This bit is reserved and must be kept at reset value in SRAM1 control register.

5.6.2 RAMCFG memory x interrupt status register (RAMCFG_MxISR)

Address offset: 0x008 + 0x40 * (x - 1), (x = 1, 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
							r							r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation ongoing

1: Erase operation ongoing

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option bit is enabled, tamper detection or RDP level regression. Refer to Table 15: Internal SRAMs features.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DED**: ECC double error detected

0: No double error

1: Double error detected

Note: This bit is reserved and must be kept at reset value in SRAM1 interrupt status register.

Bit 0 **SEDC**: ECC single error detected and corrected

0: No single error

1: Single error detected and corrected

Note: This bit is reserved and must be kept at reset value in SRAM1 interrupt status register.

5.6.3 RAMCFG memory x erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x40 * (x - 1), (x = 1, 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_MxCR.

1) Write 0xCA into ERASEKEY[7:0].

2) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

5.6.4 RAMCFG memory 2 interrupt enable register (RAMCFG_M2IER)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
												rs		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCNMI**: Double error NMI

This bit is set by software and cleared only by a global RAMCFG reset.

0: NMI not generated in case of ECC double error

1: NMI generated in case of ECC double error

Note: if ECCNMI is set, the RAMCFG maskable interrupt is not generated whatever DEIE bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DEIE**: ECC double error interrupt enable

0: Double error interrupt disabled

1: Double error interrupt enabled

Bit 0 **SEIE**: ECC single error interrupt enable

0: Single error interrupt disabled

1: Single error interrupt enabled

5.6.5 RAMCFG memory 2 ECC single error address register (RAMCFG_M2SEAR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESEA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ESEA[31:0]**: ECC single error address

When the ALE bit is set in RAMCFG_M2CR, this bitfield is updated with the address corresponding to the ECC single error.

5.6.6 RAMCFG memory 2 ECC double error address register (RAMCFG_M2DEAR)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EDEA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EDEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EDEA[31:0]**: ECC double error address

When the ALE bit is set in RAMCFG_M2CR, this bitfield is updated with the address corresponding to the ECC double error.

5.6.7 RAMCFG memory 2 interrupt clear register (RAMCFG_M2ICR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CDED**: Clear ECC double error detected

Writing 1 to this flag clears the DED bit in RAMCFG_M2ISR. Reading this flag returns the DED value.

Bit 0 **CSEDC**: Clear ECC single error detected and corrected

Writing 1 to this flag clears the SEDC bit in RAMCFG_M2ISR. Reading this flag returns the SEDC value.

5.6.8 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 31 to 0)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

5.6.9 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 63 to 32)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

5.6.10 RAMCFG memory 2ECC key register (RAMCFG_M2ECCKEYR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ECCKEY[7:0]**: ECC write protection key

The following steps are required to unlock the write protection of ECCE in RAMCFG_M2CR.

1) Write 0xAE into ECCKEY[7:0].

2) Write 0x75 into ECCKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

5.6.11 RAMCFG memory 2 write protection register 3 (RAMCFG_M2WPR3)

Address offset: 0x070

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P95WP	P94WP	P93WP	P92WP	P91WP	P90WP	P89WP	P88WP	P87WP	P86WP	P85WP	P84WP	P83WP	P82WP	P81WP	P80WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P79WP	P78WP	P77WP	P76WP	P75WP	P74WP	P73WP	P72WP	P71WP	P70WP	P69WP	P68WP	P67WP	P66WP	P65WP	P64WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 95 to 64)

These bits are set by software and cleared only by a global RAMCFG reset.

0: Write protection of SRAM2 1-Kbyte page y is disabled.

1: Write protection of SRAM2 1-Kbyte page y is enabled.

5.6.12 RAMCFG memory 2 write protection register 4 (RAMCFG_M2WPR4)

Address offset: 0x074

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P127WP	P126WP	P125WP	P124WP	P123WP	P122WP	P121WP	P120WP	P119WP	P118WP	P117WP	P116WP	P115WP	P114WP	P113WP	P112WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P111WP	P110WP	P109WP	P108WP	P107WP	P106WP	P105WP	P104WP	P103WP	P102WP	P101WP	P100WP	P99WP	P98WP	P97WP	P96WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 127 to 96)

These bits are set by software and cleared only by a global RAMCFG reset.

0: Write protection of SRAM2 1-Kbyte page y is disabled.

1: Write protection of SRAM2 1-Kbyte page y is enabled.

5.6.13 RAMCFG register map

Table 19. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RAMCFG_M1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x004	Reserved																																
0x008	RAMCFG_M1ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x00C - 0x024	Reserved																																
0x028	RAMCFG_M1ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C - 0x03C	Reserved																																
0x040	RAMCFG_M2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x044	RAMCFG_M2IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x048	RAMCFG_M2ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x04C	RAMCFG_M2SEAR	ESEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	RAMCFG_M2DEAR	EDEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	RAMCFG_M2ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x058	RAMCFG_M2WPR1	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x05C	RAMCFG_M2WPR2	P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP	P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	Reserved																																
0x064	RAMCFG_M2ECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x068	RAMCFG_M2ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x070	RAMCFG_M2WPR3	P95WP	P94WP	P93WP	P92WP	P91WP	P90WP	P89WP	P88WP	P87WP	P86WP	P85WP	P84WP	P83WP	P82WP	P81WP	P80WP	P79WP	P78WP	P77WP	P76WP	P75WP	P74WP	P73WP	P72WP	P71WP	P70WP	P69WP	P68WP	P67WP	P66WP	P65WP	P64WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	RAMCFG_M2WPR4	P127WP	P126WP	P125WP	P124WP	P123WP	P122WP	P121WP	P120WP	P119WP	P118WP	P117WP	P116WP	P115WP	P114WP	P113WP	P112WP	P111WP	P110WP	P109WP	P108WP	P107WP	P106WP	P105WP	P104WP	P103WP	P102WP	P101WP	P100WP	P99WP	P98WP	P97WP	P96WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

6 Embedded flash memory (FLASH)

6.1 FLASH introduction

The embedded flash memory (FLASH) manages the accesses of any master to the up to 1 Mbyte of embedded nonvolatile memory. It implements the read, program and erase operations, error corrections, as well as various integrity and confidentiality protection mechanisms.

The FLASH manages the automatic loading of nonvolatile user option bytes at power-on reset, and implements the dynamic update of these options.

The FLASH also features a one-time-programmable (OTP) area and a read-only area provisioned by STMicroelectronics during the product manufacturing.

6.2 FLASH main features

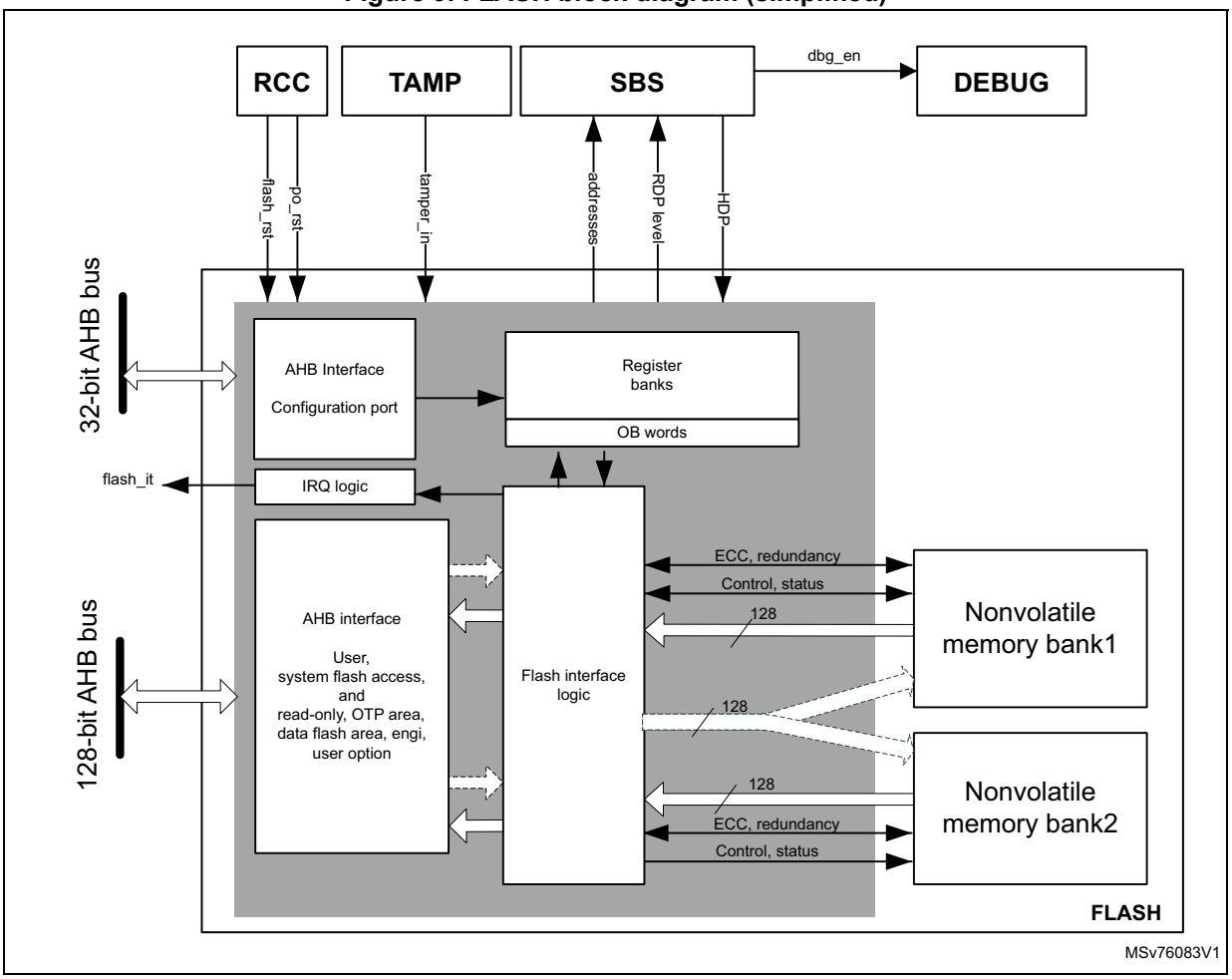
- Up to 1 Mbyte of nonvolatile memory divided into two banks
- Flash memory read operations supporting multiple lengths: 128, 64, 32, 16 bits, or 1 byte
- Flash memory programming by 128 bits in user area and 32 or 16 bits for OTP and flash data area
- 8-Kbyte page erase, bank erase and dual-bank mass erase
- 16 EDATA pages per bank that can be configured
  - to extend user area, 2-Kbyte pages, 128-bit programming
  - to offer user data flash memory, 1.5-Kbyte pages, 16- or 32-bit programming
- Dual-bank organization supporting:
  - Simultaneous operations: read-while-write (program and erase) is supported.
  - The two banks share the same interface. Write and erase can not be performed in parallel (write-while-write is not supported).
  - Bank swapping: the address mapping of the user flash memory of each bank can be swapped, along with the corresponding registers. Security flags remain valid for the physical bank. So the data are not reveled by swapping to a bank with lower security configuration.
- Error code correction (ECC): one error detection/correction or two error detections per 128-bit FLASH word using 9 ECC bits
- User configurable nonvolatile option bytes
- Flash memory enhanced protections, activated by option bytes
  - Read out protection for protecting memory content from debug access
  - Page group write-protection (WRPG)
  - HDP protection providing temporal isolation for startup code
- 4.5-Kbyte one-time programmable (OTP) area
- Read-only area provisioned by STMicroelectronics
- Prefetch is reading the next sequential instruction from flash memory

6.3 FLASH functional description

6.3.1 FLASH block diagram

The following figure shows the simplified FLASH block diagram.

Figure 9. FLASH block diagram (simplified)



6.3.2 FLASH signals

The flash memory has two AHB connections: the flash AHB register interface and the main AHB interface.

FLASH AHB register interface

- Data size is 32 bits.
- Except for some registers (FLASH_KEYR and FLASH_OPTKEYR that are used to insert unlock sequences for control and option registers and can be written in 32 bits), it is possible to read and write all registers in 8, 16, and 32 bits.
- When an unlock sequence for control and option registers is incorrect, a bus error is raised. Otherwise, no read or write errors are generated on the bus.

Main AHB interface

The AHB data bus size is 128 bits. This interface is used to handle two different targets:

- Code placed in user and system memory. 128-bit data are protected by 9 bits of ECC.
- OTP, user data flash, and read only memory. 16-bit data are protected by 6 bits of ECC.

The main AHB interface is implemented as follows:

- User and system memory:
  - Supports multiple length: 128-, 64-, 32-, 16-, and 8-bit data width.
  - There is a read buffer of 128 bits for each bank where the last data read is stored. If the data is available in the read buffer, no read access is done to the flash memory. Buffer is flushed when write access, OTP access, flash data area access user option change request or erase operation occurs
  - There is a prefetch of the same size as the read buffer.
  - A 9-bit ECC is associated to each 128-bit data flash memory word
- OTP, user data flash and read only
  - Two dedicated data buffer of 137 bits are used to manage 16-bit data with 6-bit ECC.
  - Reading two times the same address triggers two FLASH read accesses.
  - During a read access, two wait states are added in addition of the memory wait states. These wait states are necessary to parse the data buffer
  - Each write access triggers a write in the flash memory.
  - 6-bit ECC is associated to each 16-bit data.

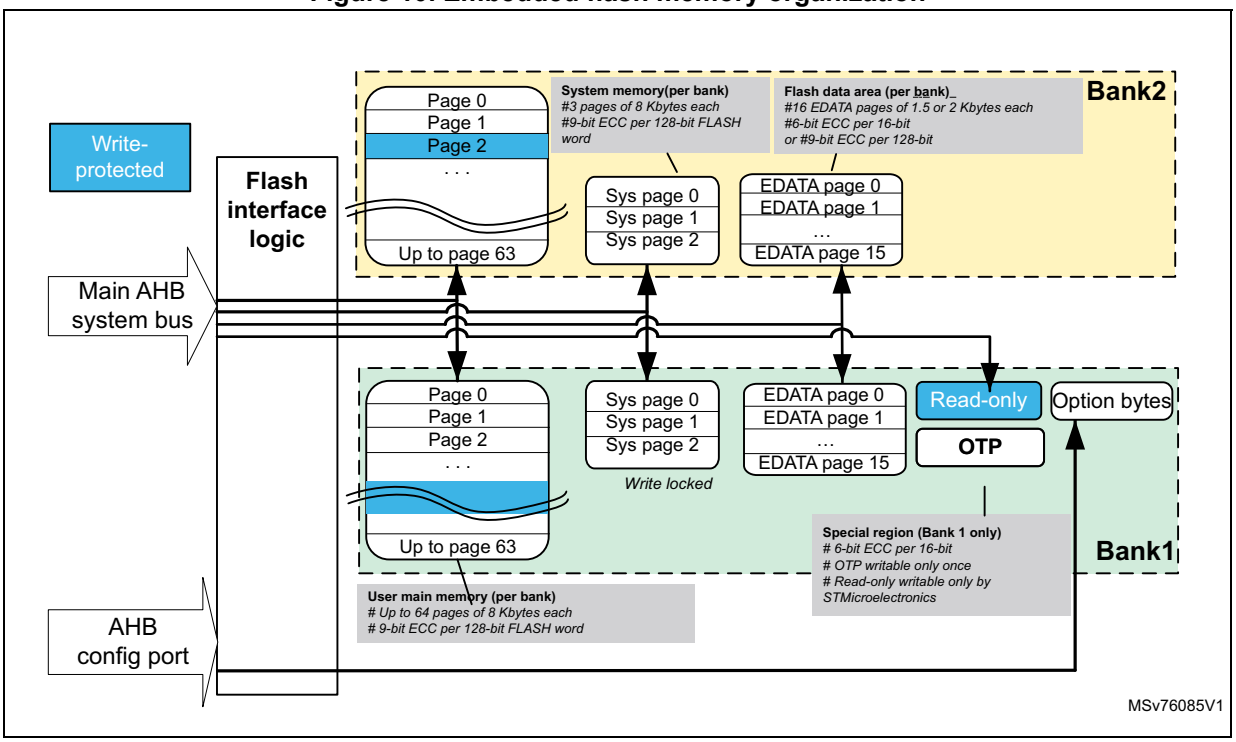
Caution: By default, all the AHB memory range is cacheable. For regions where caching is not permitted (OTP, RO, data area), MPU has to be used to disable local cacheability. An attempt to cache these regions will generate a Hard Fault.

6.3.3 Flash memory architecture and usage

Flash memory architecture

Figure 10 shows the nonvolatile memory organization supported by the FLASH.

Figure 10. Embedded flash memory organization



The embedded flash nonvolatile memory is composed of:

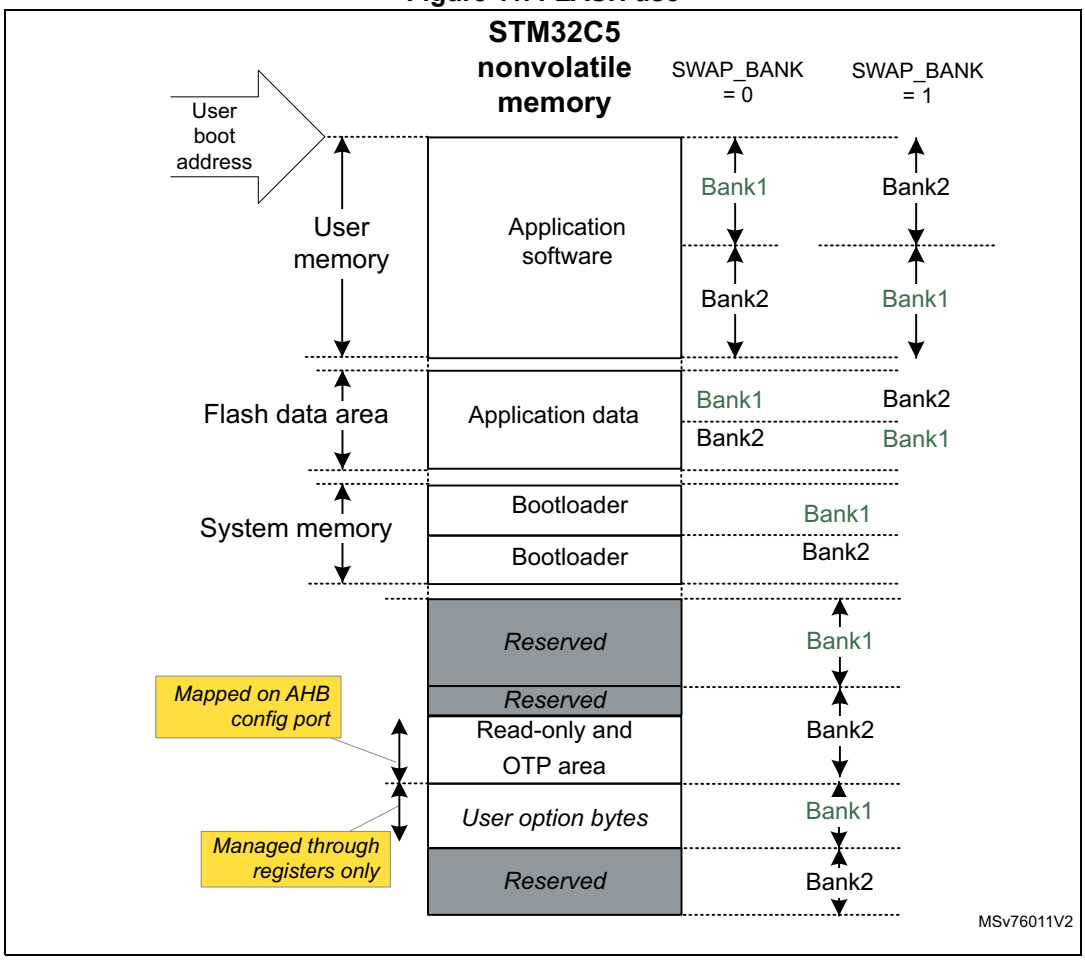
- A main memory block is organized into two banks. Each bank is divided in pages of 8 Kbytes each and features flash-word rows of 128 bits + 9 bits of ECC per word.
- A system memory block of 48 Kbytes is divided into two 24-Kbyte banks. Each bank is divided in three 8-Kbyte pages. The system flash memory is ECC protected (9-bit ECC per 128-bit word).
- A set of nonvolatile option bytes loaded at reset by the FLASH, and accessible by the application software only through the AHB configuration register interface.
- A 4.5-Kbyte OTP area that can be written only once by the application software
- A 1.5-Kbyte read-only area . It contains a unique device ID and product information.
- Up to 32 pages of user flash memory for data, 16 pages of 2 Kbytes per bank if EDATA_EN bit is reset, or 16 pages of 1.5 Kbytes per bank if EDATA_EN bit is set.

The overall flash memory architecture and its corresponding access interface is summarized in *Table 24*.

Partition usage

The figure below shows how the FLASH is used by both STMicroelectronics and the application software.

Figure 11. FLASH use



User and system memories are used differently according to RDP level and other option-byte settings.

The user memory contains the application code and associated constant data. The flash data area contains application nonvolatile data.The system memory contains the STM32 bootloader. When a reset occurs, the core jumps to the boot address configured through the BOOT pin (or BOOT option bit), the BOOTADD option bytes, and the RDP level.

Note: For more information on option-byte setup for boot, refer to Section 6.4.8.

6.3.4 FLASH read operations

Read operation overview

Read access to main flash memory and system flash memory operates as follows:

- There is a 128-bit read data buffer associated to each bank, which stores the last data read. If several consecutive read accesses request data belonging to the same FLASH data word (128 bits), the data are read directly from the current data read buffer, without triggering additional FLASH read operations. This mechanism occurs each time a read access is granted. When a read access is rejected for security reasons, the corresponding read error response is issued by the FLASH and no read operation to flash memory is triggered
- The read data buffer is disabled when write access or OTP access, user option change request or other erase operation occurs

Read access to OTP, RO, and user data flash area operates as follows:

1. A FLASH data word of 137 bits is read and stored in a temporary buffer.
2. The interface parses the 137-bit data word, and selects the 16- or 32-bit data requested.
3. While parsing the 137-bit data word, two wait states are added and the AHB bus is stalled.
4. If the application reads an OTP data or user data FLASH not previously written, a double ECC error is reported, and only word full of set bits is returned (see *Section 6.3.9* for details). The read data (in 16 bits) is stored in FLASH_ECCDR, so that the user can identify if the double ECC error is due to a virgin data or a real ECC error.
5. Reading two times the same address triggers two reads in the flash memory.
6. For 8-bit accesses, an AHB bus error is generated.

Note: The FLASH can perform single-error correction and double-error detection while read operations are being executed (see Section 6.3.8).

Instruction prefetch

The Cortex-M33 fetches instructions and literal pools (constants/data) over the C-Bus and through the instruction cache if it is enabled. The prefetch block aims at increasing the efficiency of C-Bus accesses when the ICACHE is enabled, by reducing the cache refill latency.
Prefetch is efficient in case of sequential code; prefetch in the flash memory allows the next sequential instruction line to be read from the flash memory, while the current instruction line is being filled in the ICACHE and executed by the CPU.
Prefetch is enabled by setting the PRFTEN bit in FLASH_ACR. PRFTEN must be set only if at least one wait state is needed to access the flash memory.

Adjusting read-timing constraints

The FLASH clock must be enabled and running before reading data from a nonvolatile memory.

To correctly read data from the flash memory, the number of wait states (LATENCY) must be correctly programmed in FLASH_ACR, according to the FLASH main AHB interface clock frequency.

The table below shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ) and the FLASH clock frequency.

Table 20. FLASH recommended number of wait states and programming delay

Number of wait states (LATENCY)	Programming delay (WRHIGHFREQ)	Interface clock frequency
0 WS (1 FLASH clock cycle)	00	[0 MHz; 34 MHz]
1 WS (2 FLASH clock cycles)		[34 MHz; 68 MHz]
2 WS (3 FLASH clock cycles)	01	[68 MHz; 102 MHz]
3 WS (4 FLASH clock cycles)		[102 MHz; 136 MHz]
4 WS (5 FLASH clock cycles)	10	[136 MHz; 144 MHz]

Adjusting system frequency

After power-on, the FLASH is clocked by the divided by 3 high speed internal oscillator HSIDIV3 at 48 MHz. As a result, a conservative 7 wait-state latency is specified in FLASH_ACR register (see the previous table).

When changing the bus frequency, the application software must follow the sequence described below, in order to tune the number of wait states required to access the nonvolatile memory.

To increase the CPU frequency:

1. If necessary, program LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in *Table 20.*
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the FLASH clock source and/or the clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new FLASH clock source and/or the new AHB clock prescaler value are taken in account by reading back the FLASH clock source status and/or the prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the CPU frequency:

1. Modify the FLASH clock source and/or the clock prescaler in the RCC_CFGR register of reset and clock controller (RCC).
2. Check that the FLASH new clock source and/or the new clock prescaler value are taken in account by reading back the FLASH clock source status and/or the AHB interface prescaler value in the RCC_CFGR register of reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in FLASH_ACR register, as described in *Table 20.*
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

Error code correction (ECC)

The embedded flash memory embeds an error correction mechanism. Single-error correction and double-error detection are performed for each read operation. For more details, refer to *Section 6.3.8*.

Read errors

When the ECC mechanism is not able to correct the read operation, the embedded flash memory reports read errors as described in *Section 6.8.5*

Read interrupts

See *Section 6.9* for details.

6.3.5 FLASH program operations

Program operation overview

Program operation consists in issuing write commands. The FLASH supports the execution of only one write-command at a time. Write-while-write are not supported. Nothing prevents overwriting a nonvirgin FLASH word but this is not recommended. The result may lead to invalid data and inconsistent ECC code.

User flash memory

For the user flash memory, 9-bit ECC is associated to each 128-bit data FLASH word. In this case, the FLASH must always perform write operations to nonvolatile memory with a 128-bit word granularity. Once the write buffer is full (128 bits), the busy flag BSY is set and a programming operation is triggered.

There is a write buffer common to bank1 and 2, which support multiple write-access types (128, 64, 32, 16, or 8 bits). The application can decide to write as little as 8- to a 128-bit FLASH word. In this case, a force-write mechanism to the 128 bits + ECC is used (see FW bit in FLASH_CR register).

When the write request is issued to the memory any new write request stalls the main AHB bus.

Moreover, while a write operation is ongoing, any new read request to the same bank stalls the main AHB bus.

OTP, RO and flash data area

When the target memory is OTP, RO, and flash data area pages, 6-bit ECC code is associated to each 16-bit data FLASH word. The FLASH supports 16- or 32-bit write operations (8-bit write operations are not supported). For 8-bit accesses, write accesses are ignored. There is no write data buffer. Each write access triggers a write in the flash memory.

Note: The OTP area is typically write-protected on the final product, as described in Section 6.3.9.

The write protection check is performed at the reception of the write request (during address phase). Write protection is not checked anymore at the output of the write buffer. If a write protection violation is detected, the write operation is canceled and write protection error (WRPERR) is raised in FLASH_SR register.

Note: For write protections of main FLASH, flash data area, and OTP, see Section 6.5.

Monitoring ongoing write operations

The application software can use a status flag located in FLASH_SR to monitor ongoing write operations. Since only one operation is possible at a time, this flag indicates if any operation (write, erase, option change) is ongoing, whatever the banks.

- **BSY**: This bit indicates that an effective write, erase, option-byte change is ongoing in the nonvolatile memory. This flag is not dedicated to a specific bank. It is set when an operation is starting on the memory, whatever the bank. An operation is triggered by:
  - An erase (FLASH_CR.STRT)
  - A write (FLASH_CR.PG + AHB write)
  - An option modification (FLASH_OPTCR.OPTSTRT)

  It is cleared when the current operation is ending or in case of error.

Note: BSY is also set when an OBL launch is triggered (OBL_LAUNCH in FLASH_OPTCR).

- **WBNE**: This bit indicates that the FLASH is waiting for new data to complete the 128-bit write buffer. In this state, the write buffer is not empty. It is reset as soon as the application software fills the write buffer, or forces the writes by using FW bit in FLASH_CR, or an error is detected. When WBNE is high, it is not possible to launch an erase or an option modification on flash memory.
- **DBNE**: This bit indicates that the data buffer for parsing 16-bits data is not empty:
  - 16-bit data write access is received and that the data buffer is being filled. It is set at the receipt of the a valid write access and it is reset as soon as the write request preparation has been processed.

Note: If the flash memory is busy at the receipt of the AHB write request, the CPU execution is stalled.

Enabling write operations

Before programming the user flash memory in bank1 or bank2, the application software must make sure that the PG bit is set to 1 in FLASH_CR. If it is not the case, an unlock sequence must be used (see *Section 6.5.4*), and the PG bit must be set.

When the option bytes must be modified, the application software must make sure that FLASH_OPTCR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 6.5.4*).

Note: The application software must not unlock an already unlocked register, otherwise this register remains locked until the next system reset.

If needed, the application software can update the programming delay, as described in *Adjusting programming timing constraints*.

Writing to FLASH_CR and FLASH_OPTCR

The FLASH_CR and FLASH_OPTCR registers are not accessible in write mode when the BSY bit is set. Any attempt to write these registers while the BSY bit is set causes the AHB bus to stall until BSY bit is cleared.

Single-write sequence

The recommended single-write sequence is the following:

1. Make sure protection mechanism does not prevent to attempt programming.
2. Check that no flash memory operation is ongoing by checking the BSY bit in FLASH_SR and DBNE bit in FLASH_SR. Check that the write buffer is empty by checking the WBNE bit in FLASH_SR.
3. Check and clear all the error flags due to previous programming/erase operation.
4. Unlock the FLASH_CR register, as described in *Section 6.5.4* (only if register is not already unlocked).
5. Enable write operations by setting PG bit in FLASH_CR.
6. Write one FLASH word at aligned address.

Note: WBNE flag indicates if the 128-bit write buffer is waiting for new data.

No erase request or options change request is allowed between the first write to the write buffer and the completion of the FLASH write operation.

7. Wait for the BSY bit to be cleared in the corresponding FLASH_SR register.
8. Clear PG bit in FLASH_CR if there are not any more programming requests.

If step 6 is executed incrementally (for example byte per byte), the write buffer can become partially filled. In this case, the application software can decide to force-write what is stored in the write buffer by using FW bit in FLASH_CR. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.

Note: Using a force-write operation prevents the application from updating later the missing bits with a value different from 1, which is likely to lead to a unexpected or inconsistent data or ECC.

Adjusting programming timing constraints

Program operation timing constraints depend of the FLASH clock frequency, which directly impacts the performance. If timing constraints are too tight, the nonvolatile memory does not operate correctly. If they are too lax, the programming speed is not optimal.

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to *Table 20* in *Section 6.3.4* for the recommended programming delay depending on the FLASH clock frequency. FLASH_ACR configuration register is common to both banks.

The application software must check that no program/erase operation is ongoing before modifying WRHIGHFREQ parameter.

Caution: Modifying WRHIGHFREQ while programming/erasing the flash memory may corrupt the flash memory content.

Programming errors

When a program operation fails, an error can be reported as described in *Section 6.8*.

Programming interrupts

See *Section 6.9* for details.

6.3.6 FLASH erase operations

Erase operation overview

The FLASH can perform erase operations on 8-Kbyte or 2-Kbyte user pages, on one user flash memory bank, or on two user flash memory banks (for example mass erase).

For more details in user flash memory, user data flash, user options, and OTP erase protection, see *Section 6.5*.

Erase commands are issued through the AHB configuration interface. Since the FLASH supports one operation at a time, if it receives simultaneously a write and an erase request, an error flag is raised and both operation are canceled (see *Section 6.8* for details).

After successful completion of erase, all the bytes in the erased flash sectors are set to the flash default 0xFF value.

Erase and WRP

If the application software attempts to erase a user page that is write protected, the page erase operation is aborted, and the WRPERR flag is raised in FLASH_SR, as described in *Section 6.8.1*.

FLASH busy

Busy signals are described in *Monitoring ongoing write operations*.

Writing to FLASH_CR and FLASH_OPTCR

Refer to *Writing to FLASH_CR and FLASH_OPTCR*.

Enabling erase operations

Before erasing a page, the application software must make sure that FLASH_CR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 6.5.4*).

Note: The application software must not unlock a register that is already unlocked, otherwise this register remains locked until next system reset. This can be used to deliberately lock-out a register from further accesses.

Similar constraints apply to bank erase requests.

FLASH page-erase sequence

To erase an 8-Kbyte, a 2-Kbyte user page, or a 1.5K-byte if in data flash without security protections, proceed as follows:

1. Make sure protection mechanism does not prevent to attempt page erase (WRP, HDP).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in FLASH_SR, and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the error flags due to previous programming/erase operation (refer to *Section 6.8* for details).
4. Unlock the FLASH_CR register, as described in *Section 6.5.4* (only if register is not already unlocked).

5. Set BKSEL bit, PER bit, EDATASEL bit, and PNB bitfield in FLASH_CR.
   - BKSEL indicates in which physical bank page has to be erased.
   - PER indicates a page erase operation.
   - EDATASEL indicates if the page targeted is a 2-Kbyte EDATA page or a standard page.
   - PNB contains the target page number.
6. Set the STRT bit in FLASH_CR.
7. Wait for the BSY bit to be cleared in FLASH_SR.
8. STRT bit is automatically cleared at the end of the page erase or in case of error.
9. Clear PER in FLASH_CR if there are not anymore page erase request to be issued.

Note: If another erase flag is requested simultaneously to the page erase, an PGSERR error is generated.

Standard flash memory bank-erase sequence

To erase a bank , proceed as follows:

1. Make sure protection mechanism does not prevent to attempt bank erase.
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in FLASH_SR, and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the error flags due to previous programming/erase operation (refer to *Section 6.8* for details).
4. Unlock the FLASH_CR register, as described in *Section 6.5.4* (only if register is not already unlocked).
5. Set the BKSEL bit and the BER bit in FLASH_CR to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in FLASH_CR to start the bank erase operation. Then wait until the BSY bit is cleared in FLASH_SR.
7. STRT bit is automatically cleared at the end of erase sequence or in case of error.
8. Clear BER in FLASH_CR if there is not other bank erase request to be issued.

FLASH mass-erase sequence

The application software can set the MER bit to 1 in FLASH_CR, as described below:

1. Make sure protection mechanisms do not prevent to attempt mass erase (*Section 6.5*)
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in FLASH_SR, and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the error flags due to previous programming/erase operation (refer to *Section 6.8* for details).
4. Unlock the FLASH_CR register as described in *Section 6.5.4* (only if the registers are not already unlocked).
5. Set the MER bit to 1 in FLASH_CR.
6. Set the STRT bit in FLASH_CR. Then wait until BSY bit is cleared in FLASH_SR.
7. STRT bit is cleared automatically at the end of the erase sequence or in case of error.
8. Clear MER in FLASH_CR.

Note: Mass and bank erase also erase flash data area pages from the erased bank.

6.3.7 FLASH parallel operations

Since the nonvolatile memory is divided into two independent banks but encapsulated in the same macro, the embedded flash memory interface only supports reading in one bank while writing or erasing is executed in the other bank. This feature is called read-while-write capability (RWW). It does not support write-while-write or read-while-read. Same is valid for the flash data area.

In all cases, the sequences are described in *Section 6.3.4*, *Section 6.3.5*, and *Section 6.3.6*.

6.3.8 Flash memory error protections

Error correction codes (ECC)

The FLASH supports an error correction code (ECC) mechanism. It is based on the SECDED algorithm in order to correct single errors and detect double errors.

This mechanism uses 9 ECC bits per 128-bit FLASH word, and applies to user and system memory.

For read-only, OTP, and user flash data area, a stronger 6 ECC bits per 16-bit word is used. A double ECC error is generated for an OTP or flash data area virgin word (for example a word with 22 bits at 1). When this OTP or flash data area word is no more virgin, the ECC error disappears.

More specifically, during each read operation from a 128-bit FLASH word, the FLASH retrieves the 9-bit ECC information, computes the ECC of the FLASH word, and compares the result with the reference value. If they do not match, the corresponding ECC error is raised as described in *Section 6.8.5*.

During each program operation, a 9-bit ECC code is associated to each 128-bit data FLASH word, and the resulting 137-bit FLASH word information is written in nonvolatile memory.

A similar mechanism applies to read-only, OTP, and user flash data area but with 6-bit ECC for 16-bit data.

6.3.9 OTP and RO memory access

OTP and RO memory are accessed through the main AHB interface. The OTP is accessible at the address 0x08FF E000 to 0x08FF F1FF, and the read only section is accessible from 0x08FF F800 to 0x08FF FDFF.

They are stored in a distinct way as data or user flash, filling first 4 bytes of a page, then 5th to 8th and finally 9th to 12th as columns. It does not impact addressing nor locking mechanism but requires caution to decode ECC double error or operation interruption. Indeed for OTP or RO the ECC address returned in ADDR_ECC bitfield indicate FLASH line.

In case of double error detection, the corrupted access can be identified thanks to DATA_ADDR_ECC bitfield returning OTP position on the line.

They are stored in a different way than user data, each sector being filled vertically 32-bit wise as described in the table below.

Table 21. OTP first page organization

FLASH line	OTP position on FLASH line					
	0b000	**0b001**	**0b010**	**0b011**	**0b100**	**0b101**
0x00	OTP0 0x08FF E000	OTP1 0x08FF E002	OTP256 0x08FF E200	OTP257 0x08FF E202	OTP512 0x08FF E400	OTP513 0x08FF E402
0x01	OTP2 0x08FF E004	OTP3 0x08FF E006	OTP258 0x08FF E204	OTP259 0x08FF E206	OTP514 0x08FF E404	OTP515 0x08FF E406
...	...		...		...	
0x7E	OTP252 0x08FF E1F8	OTP253 0x08FF E1FA	OTP508 0x08FF E3F8	OTP509 0x08FF E3FA	OTP764 0x08FF E5F8	OTP765 0x08FF E5FA
0x7F	OTP254 0x08FF E1FC	OTP255 0x08FF E1FE	OTP510 0x08FF E3FC	OTP511 0x08FF E3FE	OTP766 0x08FF E5FC	OTP767 0x08FF E5FE

FLASH one-time programmable area

The FLASH offers a 4608-byte memory area dedicated to application nonconfidential, one-time programmable data (OTP). This area is composed of 2304 words of 16 bits (plus 6 bits of ECC). It cannot be erased, and can be written only once. The OTP area can be accessed through the main AHB interface from address 0x08FF E000 to 0x08FF F1FE (last whole readable 16-bit word).

OTP data can be programmed by the application software by chunks of 16 bits. Overwriting an already programmed 16-bit half-word can lead to data and ECC errors and is therefore not supported.

Note: The OTP area is virgin when the device is delivered by STMicroelectronics.

When reading OTP data with a single error corrected or a double error detected, the FLASH reports read errors, as described in *Section 6.8.5*.

When reading OTP data not written by the application software (such as virgin OTP), the ECC correction reports a double-error detection (ECCD), and the data are to be found in FLASH_ECCDR register. ECCD implies a NMI raised, unless setting in SBS prevents that.

OTP write protection

OTP data are organized as 24 blocks of 96 OTP words, as shown in *Table 22*. An entire OTP block can be protected (locked) from write accesses by setting the LOCKBLi (i = 0 to 23) bit corresponding to each OTP block (i = 0 to 23) in FLASH_OTPBLR. A block can be write-protected whether or not it has been programmed (even partially).

The OTP block locking operation is irreversible and independent from the product life state.

Note: The OTP area can only be accessed in read mode.

Table 22. Flash memory OTP organization

OTP block	AHB address	AHB word		Lock bit
		[31:16]	[15:0]	
Block 0	0x08FF E000	OTP0001	OTP0000	LOCKBL0
	0x08FF E004	OTP0003	OTP0002	
	...			
	0x08FF E0BC	OTP0095	OTP0094	
Block 1	0x08FF E0C0	OTP0097	OTP0096	LOCKBL1
	0x08FF E0C4	OTP0099	OTP0098	
	...			
	0x08FF E17C	OTP0191	OTP0190	
Block 2	0x08FF E180	OTP0193	OTP0192	LOCKBL2
	0x08FF E184	OTP0195	OTP0194	
	...			
	0x08FF E23C	OTP0287	OTP0286	
...				
Block 23	0x08FF F140	OTP2209	OTP2208	LOCKBL23
	0x08FF F144	OTP2211	OTP2210	
	...			
	0x08FF F1FC	OTP2303	OTP2302	

OTP write sequence

Follow the sequence below to write an OTP word:

1. Check that no flash memory operation is ongoing by checking the BSY bit in FLASH_SR register, and that the data buffer is empty by checking the DBNE bit in the same register.
2. Check and clear all the error flags due to previous programming/erase operation.
3. Set PG bit in FLASH_CR.
4. Check the protection status of the target OTP word (see *Table 22*). The corresponding LOCKBLi bit must not be set to 1.
5. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in *Table 22*. Alternatively, the application software can program separately the 16-bit MSB or 16-bit LSB. In this case the first 16-bit write operation starts immediately without waiting for the second one
6. Wait for the BSY bit to be cleared in FLASH_SR.
7. Clear PG bit in FLASH_CR if there is not any programming request anymore in the bank.
8. Optionally, lock the OTP block using LOCKBLi to prevent further data changes.

Note: Do not write twice an OTP 16-bit word, otherwise an ECC error can be generated.

Writing OTP data at byte level is not supported and generates a bus error.

To avoid data corruption, it is important to complete the OTP write process (for example by reading back the OTP value), before starting an option change.

FLASH read-only area

The FLASH offers a 1.5-Kbyte memory area to store read-only data, also referred to as RO. This area is mapped as described in *Table 23*. It can be accessed through the AHB main port. This read-only area is protected by a robust ECC scheme, as explained in *Section 6.3.8*.

The read-only information that can be used by the application software are described in the table below. This information is programmed by STMicroelectronics.

Table 23. Read-only public data organization

Read-only data name	Address	Comment
Unique device ID	0x08FF F800	U_ID[31:0]
	0x08FF F804	U_ID[63:32]
	0x08FF F808	U_ID[96:64]
Flash memory size/ package	0x08FF F80C	Flash memory size[15:0] || Package code[15:0]
Factory calibration	0x08FF F810	Vrefint calibration value[31:0]
	0x08FF F814	TS_CAL1[31:0]
	0x08FF F818	TS_CAL2[31:0]
Reserved	0x08FF F81C to 0x08FF FDFF	Reserved information

6.3.10 FLASH data area

The FLASH offers 64 Kbytes to store extra code, data or emulate EEPROM in 16 EDATA pages per bank. Based on value of EDATA_EN option bit this area is one of the following:

- Protected by a robust 6-bit ECC, which allows a write granularity of 16 or 32 bits at the expense of having page size shrunk to 1.5 Kbytes. This area is the FLASH data area and the memory can be accessed through the AHB system port from address 0x0900 0000 to 0x0900 BFFF.
- Page size of 2 Kbytes with write granularity of 128 bits protected by a robust 9-bit ECC. This area is part of user main flash memory and instructions can be fetched from addresses 0x0840 0000 to 0x0840 FFFF (see *Figure 13*).

Caution: This area cannot be used as a boot location in BOOTADD.

For greater efficiency, it is always recommended to use page on the other bank for the FLASH data area, so that the application benefits from RWW capability of the dual-bank arrangement.

When SWAP_BANK feature is enabled, the banks are swapped: the flash data area in bank 2 are accessible from 0x0900 000 to 0x0900 5FFF, and the data in bank 1 are accessible from 0x0900 6000 to 0x0900 BFFF.

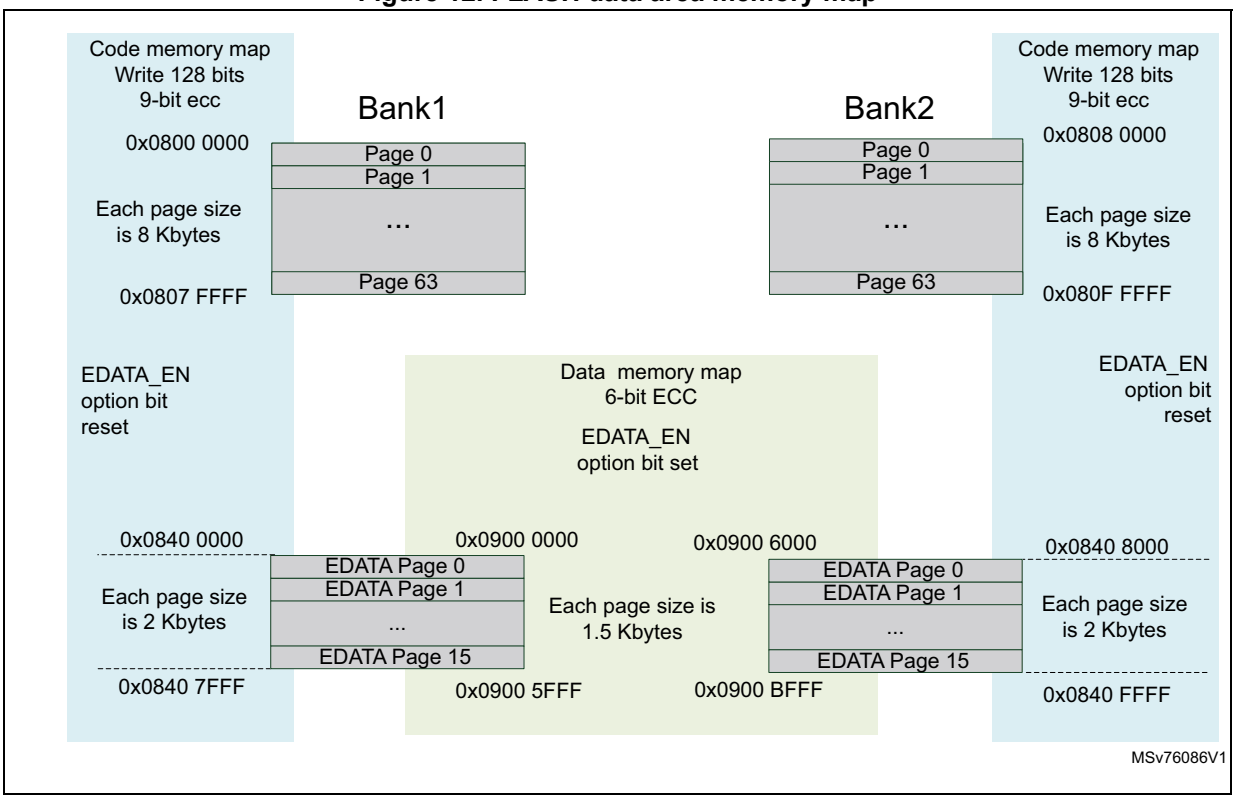
A bus error is generated on:

- Attempt to access one of the following address
  - Between 0x0900 0000 to 0x0900 BFFF, and this address is not a data FLASH valid page.
  - Between 0x0840 0000 to 0x0840 FFFF, and this address is not a user memory valid page.
- Attempt to fetch instructions from data FLASH area (fetch from 0x0900 0000 to 0x0900 BFFF result in HardFault).

Erasing these EDATA pages is possible by setting the EDATASEL bit and following a page erase request. The EDATA page erasure is not taking account EDATA_EN bit.

Protections and security of data FLASH area are detailed in *Section 6.5*.

Figure 12. FLASH data area memory map



6.3.11 FLASH bank swapping

The FLASH bank1 and bank2 can be swapped for the user FLASH. This feature can be used after a firmware upgrade to restart the device on the new firmware. Bank swapping is a user option-byte flag controlled by the SWAP_BANK bit in FLASH_OPTCR register.

Bank specific settings for data area and security attributes follow the original bank and its contents. The control bit BKSEL always refers to physical bank, not the SWAP_BANK setting.

Table 24 shows the memory map that can be accessed depending on the SWAP_BANK bit configuration.

Table 24. Memory map and the swapping option (STM32C53x/542 devices)

Flash memory area	Flash memory corresponding bank		Start address	End address	Size (bytes)	Region name
	SWAP_ BANK = 0	SWAP_ BANK = 1				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8 K	Page 0
			0x0800 2000	0x0800 3FFF	8 K	Page 1
			...	...	...	...
			0x0801 E000	0x0801 FFFF	8 K	Page 15
	Bank2	Bank1	0x0802 0000	0x0802 1FFF	8 K	Page 0
			0x0802 2000	0x0802 3FFF	8 K	Page 1
			...	...	...	...
			0x0803 E000	0x0803 FFFF	8 K	Page 15
User flash memory / EDATA_EN cleared	Bank1	Bank2	0x0840 0000	0x0840 07FF	2 K	EDATA Page 0
			0x0840 0800	0x0840 0FFF	2 K	EDATA Page 1
			...	...	...	...
			0x0840 7800	0x0840 7FFF	2 K	EDATA Page 15
	Bank2	Bank1	0x0840 8000	0x0840 87FF	2 K	EDATA Page 0
			0x0840 8800	0x0840 8FFF	2 K	EDATA Page 1
			...	...	...	...
			0x0840 F800	0x0840 FFFF	2 K	EDATA Page 15
Data flash memory / EDATA_EN set	Bank1	Bank2	0x0900 0000	0x0900 05FF	1.5K	EDATA Page 0
			0x0900 0600	0x0900 0BFF	1.5K	EDATA Page 1
			...	...	...	...
			0x0900 5A00	0x0900 5FFF	1.5K	EDATA Page 15
	Bank2	Bank1	0x0900 6000	0x0900 65FF	1.5K	EDATA Page 0
			0x0900 6600	0x0900 6BFF	1.5K	EDATA Page 1
			...	...	...	...
			0x0900 B800	0x0900 BFFF	1.5K	EDATA Page 15
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8 K	System 1 Page 0
			0x0BF8 2000	0x0BF8 3FFF	8 K	System 1 Page 1
			0x0BF8 4000	0x0BF8 5FFF	8 K	System 1 Page 2
	Bank2		0x0BF8 6000	0x0BF8 7FFF	8 K	System 2 Page 0
			0x0BF8 8000	0x0BF8 9FFF	8 K	System 2 Page 1
			0x0BF8 A000	0x0BF8 BFFF	8 K	System 2 Page 2

Table 25. Memory map and the swapping option (STM32C55x/562 devices)

Flash memory area	Flash memory corresponding bank		Start address	End address	Size (bytes)	Region name
	SWAP_ BANK = 0	SWAP_ BANK = 1				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8 K	Page 0
			0x0800 2000	0x0800 3FFF	8 K	Page 1
			...	...	...	...
			0x0803 E000	0x0803 FFFF	8 K	Page 31
	Bank2	Bank1	0x0804 0000	0x0804 1FFF	8 K	Page 0
			0x0804 2000	0x0804 3FFF	8 K	Page 1
			...	...	...	...
			0x0807 E000	0x0807 FFFF	8 K	Page 31
User flash memory / EDATA_EN cleared	Bank1	Bank2	0x0840 0000	0x0840 07FF	2 K	EDATA Page 0
			0x0840 0800	0x0840 0FFF	2 K	EDATA Page 1
			...	...	...	...
			0x0840 7800	0x0840 7FFF	2 K	EDATA Page 15
	Bank2	Bank1	0x0840 8000	0x0840 87FF	2 K	EDATA Page 0
			0x0840 8800	0x0840 8FFF	2 K	EDATA Page 1
			...	...	...	...
			0x0840 F800	0x0840 FFFF	2 K	EDATA Page 15
Data flash memory / EDATA_EN set	Bank1	Bank2	0x0900 0000	0x0900 05FF	1.5K	EDATA Page 0
			0x0900 0600	0x0900 0BFF	1.5K	EDATA Page 1
			...	...	...	...
			0x0900 5A00	0x0900 5FFF	1.5K	EDATA Page 15
	Bank2	Bank1	0x0900 6000	0x0900 65FF	1.5K	EDATA Page 0
			0x0900 6600	0x0900 6BFF	1.5K	EDATA Page 1
			...	...	...	...
			0x0900 B800	0x0900 BFFF	1.5K	EDATA Page 15
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8 K	System 1 Page 0
			0x0BF8 2000	0x0BF8 3FFF	8 K	System 1 Page 1
			0x0BF8 4000	0x0BF8 5FFF	8 K	System 1 Page 2
	Bank2		0x0BF8 6000	0x0BF8 7FFF	8 K	System 2 Page 0
			0x0BF8 8000	0x0BF8 9FFF	8 K	System 2 Page 1
			0x0BF8 A000	0x0BF8 BFFF	8 K	System 2 Page 2

Table 26. Memory map and the swapping option (STM32C59x/5A3 devices)

Flash memory area	Flash memory corresponding bank		Start address	End address	Size (bytes)	Region name
	SWAP_ BANK = 0	SWAP_ BANK = 1				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8 K	Page 0
			0x0800 2000	0x0800 3FFF	8 K	Page 1
			...	...	...	...
			0x0807 E000	0x0807 FFFF	8 K	Page 63
	Bank2	Bank1	0x0808 0000	0x0808 1FFF	8 K	Page 0
			0x0808 2000	0x0808 3FFF	8 K	Page 1
			...	...	...	...
			0x080F E000	0x080F FFFF	8 K	Page 63
User flash memory / EDATA_EN cleared	Bank1	Bank2	0x0840 0000	0x0840 07FF	2 K	EDATA Page 0
			0x0840 0800	0x0840 0FFF	2 K	EDATA Page 1
			...	...	...	...
			0x0840 7800	0x0840 7FFF	2 K	EDATA Page 15
	Bank2	Bank1	0x0840 8000	0x0840 87FF	2 K	EDATA Page 0
			0x0840 8800	0x0840 8FFF	2 K	EDATA Page 1
			...	...	...	...
			0x0840 F800	0x0840 FFFF	2 K	EDATA Page 15
Data flash memory / EDATA_EN set	Bank1	Bank2	0x0900 0000	0x0900 05FF	1.5K	EDATA Page 0
			0x0900 0600	0x0900 0BFF	1.5K	EDATA Page 1
			...	...	...	...
			0x0900 5A00	0x0900 5FFF	1.5K	EDATA Page 15
	Bank2	Bank1	0x0900 6000	0x0900 65FF	1.5K	EDATA Page 0
			0x0900 6600	0x0900 6BFF	1.5K	EDATA Page 1
			...	...	...	...
			0x0900 B800	0x0900 BFFF	1.5K	EDATA Page 15
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8 K	System 1 Page 0
			0x0BF8 2000	0x0BF8 3FFF	8 K	System 1 Page 1
			0x0BF8 4000	0x0BF8 5FFF	8 K	System 1 Page 2
	Bank2		0x0BF8 6000	0x0BF8 7FFF	8 K	System 2 Page 0
			0x0BF8 8000	0x0BF8 9FFF	8 K	System 2 Page 1
			0x0BF8 A000	0x0BF8 BFFF	8 K	System 2 Page 2

The SWAP_BANK bit in FLASH_OPTCR register is loaded from the SWAP_BANK option bit only after system reset or POR.

To change the SWAP_BANK bit (for example to apply a new firmware update), respect the sequence below:

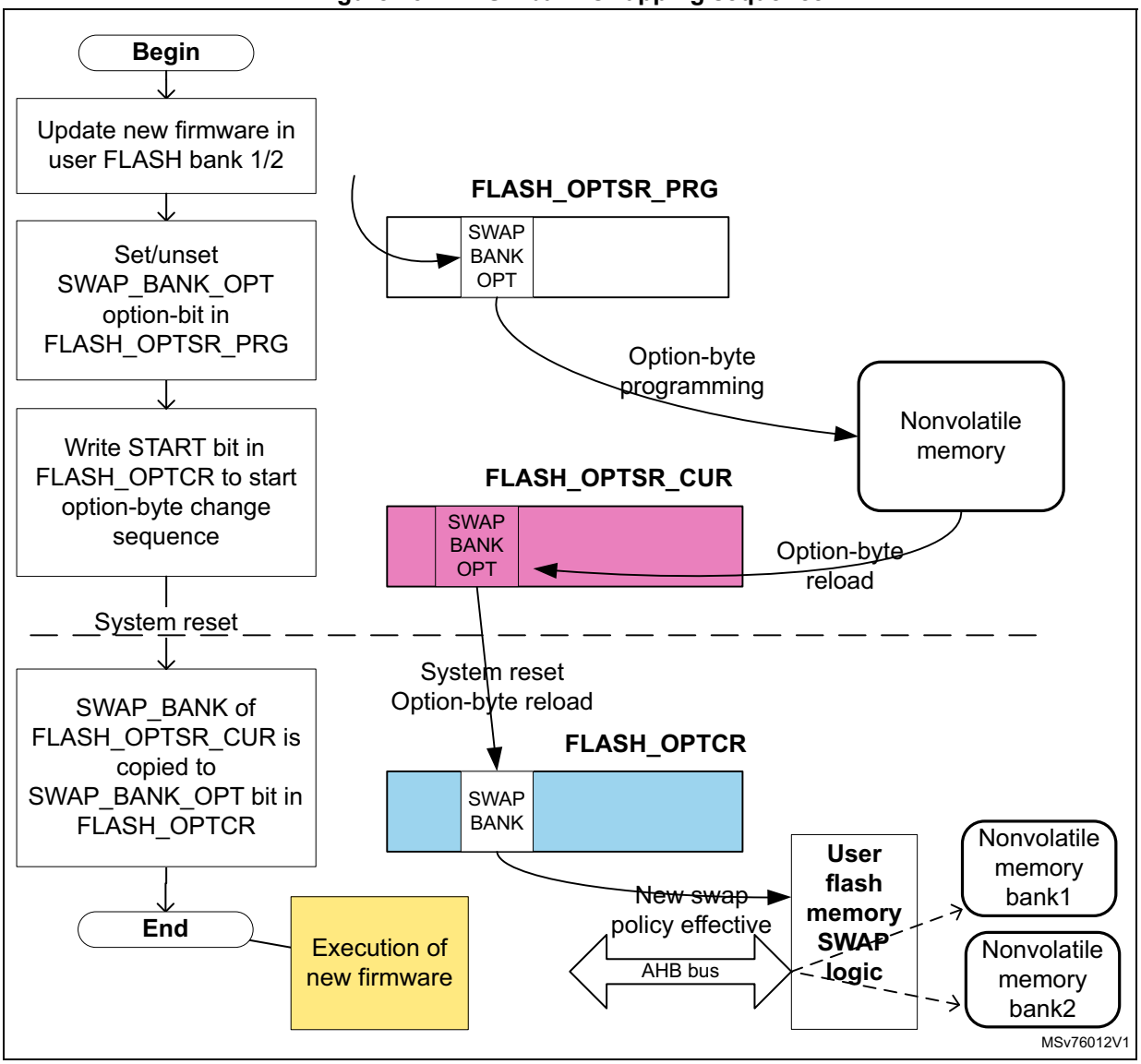
1. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in FLASH_SR, and that the write buffer is empty by checking the WBNE bit in the same register.
2. Clear all error flags due to a previous operation.
3. Unlock OPTLOCK bit, if not already unlocked.
4. Set the new desired SWAP_BANK value in the FLASH_OPTSR_PRG register.
5. Start the option-byte change sequence by setting the OPTSTRT bit in FLASH_OPTCR.
6. Once the option-byte change has completed, FLASH_OPTSR_CUR contains the expected SWAP_BANK value, but SWAP_BANK bit in FLASH_OPTCR has not yet been modified, and the bank swapping is not yet effective
7. Force a system reset or a POR. When the reset rises up, the bank swapping is effective (SWAP_BANK value updated in FLASH_OPTCR) and the new firmware must be executed.

Note: The SWAP_BANK bit in FLASH_OPTCR is read-only and cannot be modified by the application software.

The SWAP_BANK option bit in FLASH_OPTSR_PRG can be modified whatever the RDP level. Instead of being locked by RDP_LEVEL, it is locked by BOOT_LOCK User OB.

The figure below gives an overview of the bank swapping sequence.

Figure 13. FLASH bank swapping sequence



6.3.12 FLASH reset and clocks

Reset management

The FLASH can be reset by a core domain reset, driven by the reset and clock control (RCC). The main effects of this reset are the following:

- All registers, except for option-byte registers, are cleared, including read and write latencies. If the bank swapping option is changed, it is applied.
- Most control registers are automatically protected against write operations. To unprotect them, new unlock sequences must be used as described in *Section 6.5.4*.

The FLASH can be reset by a power-on core domain reset, driven by the RCC. When the reset falls, all option-byte registers are reset. When the reset rises up, the option bytes are

loaded, potentially applying new features. During this loading sequence, the device remains under reset and the FLASH is not accessible.

Reset occurring during FLASH operation

If a reset occurs during a FLASH operation (programing, erase, or option change), the content of the flash memory is not guaranteed. It is mandatory for flash memory integrity than user restarts the operation. The status register FLASH_OPSR gives information if any FLASH operation has been interrupted by a reset.

FLASH_OPSR.CODE_OP gives opcode of operation. The table below indicates how to use FLASH_OPSR and which operation is required.

Table 27. Recommended reactions to FLASH_OPSR contents

CODE_OP	Operation interrupted	OTP_OP	BK_OP	DATA_OP	Flash memory area	ADDR_OP min	ADDR_OP max	Recommended action
0b000	No operation ongoing while reset	0	0	0	-(1)	-	-	No extra action
0b001	Write operation	0	0/1	0	User flash	0x0000	0x7FFF(2)	Erase page and rewrite
		1	1	0	OTP(3)	0x0200	0x037F	
		0	0/1	1	EDATA Page(4)	0x0000	0x07FF(2)	
0b011	Page erase(5)	0	0/1	0	User flash	0x0000	0x7FFF	Relaunch page erase
		0	0/1	1	EDATA Page	0x0000	0x07FF	
0b100	Bank erase	0	0/1	0	User flash	-	-	Relaunch bank erase
0b101	Mass erase	0	0	0		-	-	Relaunch mass erase
0b110	Option change	0	0	0	User configuration	-	-	New attempt on option change

1. '-' = does not matter.
2. ADDR_OP maximum depends on the flash memory size of the device.
3. OTP are organized per column, refer to *Section 6.3.9*.
4. Depends on EDATA setting in the OB.
5. Addresses indicated are aligned to page start, data area pages are erased by erase request to corresponding user flash memory page.

Clock management

The FLASH uses the microcontroller AHB interface clock HCLK.

6.4 FLASH option bytes

6.4.1 About option bytes

The FLASH includes a set of nonvolatile option bytes. They are loaded at power-on reset and can be read and modified only through configuration registers.

This section documents:

- When option bytes are loaded.
- How application software can modify them.
- The detailed list of option bytes, together with their initial values (before the first option byte change, user default configuration).

6.4.2 Option-byte loading

There are multiple ways of loading the option bytes into the FLASH:

1. Power-on wake-up

   When the device is first powered, the FLASH automatically loads all the option bytes. During the option-byte loading (OBL) sequence, the device remains under reset, and the FLASH cannot be accessed.

2. Wake-up from system Standby mode

   When the core power domain, which contains the FLASH, is switched from Standby mode to Run mode, the FLASH behaves as during a power-on sequence. During OBL time, the device is not under reset, unlike power-on sequence.

3. Dedicated option-byte reloading by the application

   When the user application successfully modifies the option-byte content through the FLASH registers, the nonvolatile option bytes are programmed, and the FLASH automatically reloads all option bytes to update the option registers.

Note: The option-byte read sequence is protected by error correction code.

In case of error, the option bytes are loaded with default values (see Section 6.4.3). This value differs from the initial values (user default configuration), and tends to be more restrictive.

6.4.3 Option-byte modification

Changing user option bytes

A user-option-byte change operation can be used to modify the configuration and the protection settings saved in the nonvolatile option byte area.

There are numerous rules enforced when attempting to change a user option byte, they are all summarized in *Section 6.4.9*. Failing to stick to these rules usually results in error, described in *Section 6.8.6*.

The FLASH features two sets of option-byte registers:

- The first register set contains the current values of the option bytes. Their names have the _CUR extension. All "_CUR" registers are read-only. Their values are automatically loaded from the nonvolatile memory after power-on reset, wake-up from system Standby mode, or after an option-byte change operation.
- The second register set allows the modification of the option bytes. Their names contain the _PRG extension. All "_PRG" registers can be accessed in read/write mode.

When the OPTLOCK bit in FLASH_OPTCR is set, modifying the FLASH_XXX_PRG registers is not possible.

When OPTSTRT bit is set to 1, the FLASH checks the programming sequence (PGSERR), and the conditions described in *Section 6.4.9* (OPTCHANGEERR). If no error has been detected (PGSERR/OPTCHANGEERR), the FLASH launches the option-byte modification in its nonvolatile memory, and updates the option-byte registers with _CUR extension.

If one of the condition described in *Section 6.4.9*, *Section 6.8.6*, or *Section 6.8.2* is not respected, the FLASH aborts the option-byte change operation. In this case, FLASH_XXX_PRG registers are not overwritten by current option value. The user application can check what was wrong in their configuration.

Unlocking the option-byte modification

After reset, the OPTLOCK bit is set to 1 and FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option bytes. FLASH_OPTCR unlock sequence is described in *Section 6.5.4*.

Option-byte modification sequence

To modify user option bytes, follow the sequence below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in FLASH_SR, and that the write buffer is empty by checking the WBNE bit in the same register.
2. Check the data buffer is empty (DBNE = 0) in FLASH_SR.
3. Clear all error flags due to a previous operation.
4. Unlock FLASH_OPTCR register as described in *Section 6.5.4*, unless the register is already unlocked.
5. Write the desired new option byte values in the corresponding option registers (FLASH_XXX_PRG).
6. Set the option byte start change OPTSTRT bit to 1 in FLASH_OPTCR.
7. Wait until BSY bit is cleared in FLASH_SR.
8. OPTSTRT bit is cleared automatically at the end of the sequence (or in case of error).

9. Perform a system reset if the RDP level, SWAP_BANK, or BOOTADD options have been changed. A system reset is also recommended if other option bytes have been modified.

Note: If a reset or a power-down occurs while the option-byte modification is ongoing, the original option-byte value is kept. A new option-byte modification sequence is required to program the new value

6.4.4 Option-byte overview

Table 28 lists all the user option bytes managed through the FLASH registers, as well as initial values before the first option-byte change (user default configuration).

Table 28. Option-byte organization

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_OPTSR	SWAP_BANK	SINGLE_BANK	EDATA_EN	Res.	Res.	Res.	Res.	Res.	BOOT0	BOOT_SEL	IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.	RDP_LEVEL								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
	0	0	1						0	0	1	1					1	1	1	0	1	1	0	1	1	1		1	1			
FLASH_OPTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2_ECC	Res.	Res.	SRAM2_RST	SRAM1_RST
																												1			1	1
FLASH_BOOTR	BOOTADD[31:0]																								BOOT_LOCK							
	0x0800_00																								0xC3							
FLASH_ WRPSG1R	WRPG[31:0]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_ WRPSG2R	WRPG[31:0]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_OTPBLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKBL																							
									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_BL_ COM_CFG	BL_COM_CFG[31:0]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_HDP1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT							
									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	1
FLASH_HDP2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT							
									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	1
FLASH_OEMKEY R1	OEMKEY[31:0]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_OEMKEY R2	OEMKEY[63:32]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 28. Option-byte organization (continued)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_ OEMKEYR3	OEMKEY[95:64]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_ OEMKEYR4	OEMKEY[127:96]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_BSKEY	BSKEY[31:0]																															
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

6.4.5 Description of user and system option bytes

The general-purpose option bytes that can be used by the application are listed below:

- Bank management
  - SWAP_BANK: bank swapping option, set to 1 to swap user FLASH banks after boot (see *Section 6.3.11*).
  - SINGLE_BANK: this bit is applicable for:
    - STM32C53x/542 with 128 Kbytes of flash
    - STM32C55x/562 with 256 Kbytes of flash
    - STM32C59x/5A3 with 512 Kbytes of flash

    If SINGLE_BANK = 1, the available user memory is located in a single bank. Otherwise, the device works in dual-bank mode. In any case, the address range is kept linear.
- Watchdog management
  - IWDG_STOP: independent watchdog IWDG counter active in Stop mode if 1 (stop counting or freeze if 0)
  - IWDG_STDBY: independent watchdog IWDG counter active in Standby mode if 1 (stop counting or freeze if 0)
  - IWDG_SW and WWDG_SW: hardware (0) or software (1) IWDG watchdog control selection

Note: If the hardware watchdog "control selection" feature is enabled (set to 0), the watchdog is automatically enabled at power-on, thus generating a reset unless the watchdog key register is written to or the down-counter is reloaded before the end-of-count is reached.

Depending on the configuration of IWDG_STOP and IWDG_STDBY options, the IWDG can continue counting (1) or not (0) when the device is in Stop or Standby mode, respectively. When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

- Reset management
  - NRST_STDBY: generates a reset when entering Standby mode if cleared to 0.
  - NRST_STOP: generates a reset when entering Stop mode if cleared to 0.

Note: Whenever a Standby (respectively Stop) mode entry sequence is successfully executed, the device is reset instead of entering Standby (respectively Stop) mode if NRST_STDBY (respectively NRST_STOP) is cleared to 0.

- Bootloader configuration

  After reset, if the STMicroelectronics bootloader is selected as boot location, it reads the contents of BL_COM_CFG, and only attempts to establish communication on selected interfaces. This setting allows using the bootloader safely in applications where it otherwise may cause trouble by reconfiguring some GPIOs.

When STMicroelectronics delivers the device, the values programmed in the general-purpose option bytes are the following:

- Watchdog management
  - IWDG active in Standby and Stop modes (option value = 0x1)
  - IWDG not automatically enabled at power-on (option byte value = 0x1)
- Reset management:
  - A reset is not generated when the device enters Standby or Stop low-power mode (option byte value = 0x1)

Refer to *Section 6.10* for details.

6.4.6 Description of data protection option bytes

The option bytes that can be used to enhance data protection are listed below:

- RDP_LEVEL[7:0]: product lifecycle state (see *Section 6.5.8* for details)
- WRP1/2: write-protection option of pages in bank1 (respectively bank2). It is active low. Refer to *Section 6.5.5* for details.

Table 29. Write-protection management

WRP bit number	Write-protected pages	
	STM32C53x/542/55x/562	STM32C59x/5A3
Bit 0	Page 0	Page [0 to 1]
Bit 1	Page 1	Page [2 to 3]
…	…	…
Bit N	Page N	Page [2 x N to 2 x N +1]

- HDPx: HDPL exclusive area control in the user flash memory.

When factory programmed, values of the data-protection option bytes are the following:

- Read-out protection level: L0 (Open)
- Write protection disabled (all option byte bits set to 1)

Refer to *Section 6.10* for details.

6.4.7 Description of data FLASH option bytes

The option bytes that can be used to address the data FLASH pages are:

- EDATA_EN: activate data FLASH area for this device (see *Section 6.3.10* for details).

6.4.8 Description of boot-address option byte

Below the list of option bytes that can be used to configure the appropriate boot address for an application:

- RDP_LEVEL: Influences the boot sequence indirectly.
- BOOTADD: Selects default boot address.
- BOOT_LOCK: Protects the boot configuration from further modification attempts.
- SWAP_BANK: bank swapping option, set to 1 to swap user FLASH banks after boot (see *Section 6.3.11*). If BOOT_LOCK is active, the value in SWAP_BANK is fixed, read-only.
- BOOT0 management
  - BOOT0: This option bit is the value of BOOT0 used depending on BOOT_SEL to select boot location. BOOTLOADER (1) or User code(0).
  - BOOT_SEL: BOOT0 value is taken from option bit (0) or from BOOT0 pin (1)

When STMicroelectronics delivers the device, the RDP_LEVEL is L0, BOOT_LOCK is not set (0xC3). Address BOOTADD is 0x0800 0000.

Refer to *Section 6.10* for details.

6.4.9 Specific rules for modifying option bytes

Besides the OPTLOCK bit and register access rules, there are additional protection means for selected security-sensitive option byte fields.

Multiple option bytes can be modified simultaneously, but if they rely on each other for protection, both states are checked.

With few exceptions listed in this section, failing to uphold the rules results in raising OPTCHANGEERR flag (*Section 6.8.6* for additional details).

Table 30. Specific OB modifying rules overview

OB	HDPL	Value	RDP level
RDP_LEVEL	-	Set of possible transitions	Set of possible transitions (*Table 36*)
HDP	1(1)	-	-
BOOTADD	-	-	L0 open
LOCKBL	-	One way switch(1)	-
SWAP_BANK	-	BOOT_LOCK can restrict this change	-
BOOT_LOCK	-	-	L0 open

1. No OPTCHANGEERR is raised in violation of this.

Even in the closed RDP_LEVEL progression, some OBs can still be modified, if all the other constraints are satisfied. An overview is presented in the table below.

Table 31. OBs modifiable in L2 product

RDP_LEVEL	OBs that can be modified
L2_wBS or L2	SWAP_BANK, LOCKBL, RDP_LEVEL

Specific rules must be respected to update the following OBs:

- **RDP_LEVEL:** its transitions follow a state machine with two types of transition. Locking down the product is allowed without restriction. Opening the product is only possible under the condition that OEM key were provisioned ahead. The regression to L0 RDP_LEVEL results in the erasing of the protected content (see *Section 6.5.8*).
- **HDP:** Can only be modified in HDPL0 and HDPL1.
- **BOOTADD:** Can only be changed in Open RDP L0 (open).
- **LOCKBL:** Can only be changed freely in one direction. A permanent irreversible switch.
- **SWAP_BANK:** Cannot be modified when BOOT_LOCK is active (0xB4).
- **BOOT_LOCK:** Can only be changed freely in locking direction. Unlock is possible in L0.

Note: For all user option bytes above, default values are loaded and Tamper is signaled when a double ECC error occurs during OBL.

6.5 FLASH security and protections

Since sensitive information are stored in the flash memory, it is important to protect it against unwanted operations such as reading confidential areas, illegal programming of immutable pages, or malicious flash memory erasing.

For that purpose, the flash memory implements the following protection mechanisms:

- Temporal isolation protection (HDP)
- Configuration protection
- User FLASH write protection
- Device boot state management
- OTP locking

This section provides a detailed description of all these security mechanisms.

The flash memory interface evaluates the access restrictions in the following order:

1. Write protection (for write access)
2. HDP

6.5.1 Hide protection (HDP)

A nonvolatile hide-protection (HDP) area per bank can be defined with a page granularity. Access to the hide-protection area can be denied by progressing the HDPL level in the SBS.

When the HDPL = 1, no user FLASH is protected by the HDP. With HDPL ≥ 2, the user OB defined HDP area in each bank is closed. No read, write, fetch, or erase is allowed in the HDP area.

The HDP area can be extended. The extension setting is only a register value, not stored in the OB. With HDPL = 3, the HDP area remains activated and the extension is added. If an extension is added while the base user OB defined area is not active, the extension covers one more page, because HDPx_END page is also protected.

The HDPL level can be only cleared by a system reset, there is no means to deactivate the HDP area.

The protected HDP area is defined by setting its size using start and end pages .

The size of the inaccessible area can be extended using register FLASH_HDPEXTR values HDPx_EXT. The value HDPx_EXT is a number of pages added to the HDP area (past the HDPx_END page). The volatile HDPx_EXT value in the register can be increased, but cannot decrement.

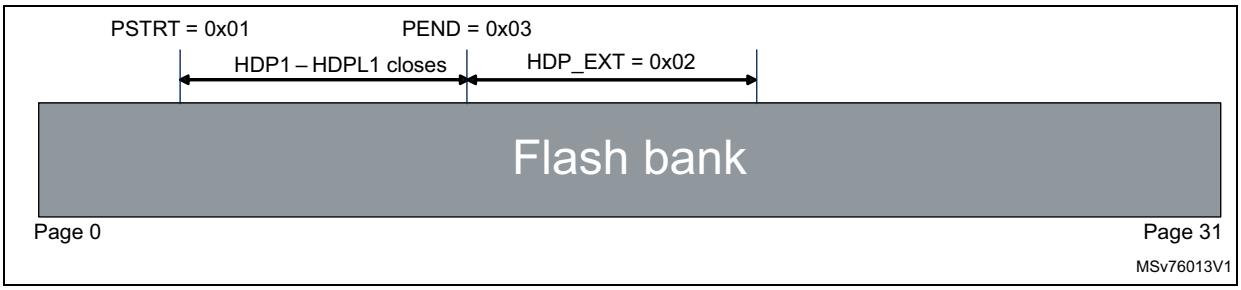
In case of HDPx_STRT > HDPx_END, the extension covers (HDPx_EXT + 1) size area between pages marked by HDPx_END and HDPx_END + HDPx_EXT.

By default, the HDPx_END is set to zero and HDPx_STRT is set to one. This means that the HDP area in user FLASH is zero size, and extends from the FLASH start address (first page).

For example, to protect area by HDP from 0x0800 2000 (included) to 0x0800 7FFF (included):

- For physical bank1, the option-byte registers must be programmed with:
  - HDP1_STRT = 0x01,
  - HDP1_END = 0x03.

Figure 14. HDP in user flash memory



- Once the HDPL is incremented to 2, the marked area becomes inaccessible.
  The HDPL = 2 code then sets the extension size to 2. Once the HDPL is incremented to 3, the extension area becomes also inaccessible.

  Alternatively, if:
  - HDP1_END = 0
  - HDP1_START = 1
  - HDPx_EXT = 1

  Page 0 and page 1 are HDP protected.

  Or a rather unusual example:
  - HDP1_END = 2
  - HDP1_START = 7
  - HDPx_EXT = 3
- Pages 2 until 5 are HDP protected. Here the user must know that, if the application needs pages 2 until 4 only as HDP area, HDPx_EXT must be programmed at 2 instead of 3.
- If the two banks are swapped, the protection defined to physical bank1 remains on the physical bank1, unaffected by swapping. Separate protection applies to physical bank2 and the option bytes registers must also be programmed with:
  - HDP2_STRT = 0x01
  - HDP2_END = 0x03

Note: For more details on the bank-swapping mechanism, refer to Section 6.5.3.

Table 32. Secure hide protection

HDPx watermark option byte values (x = 1, 2)		Hide protection area
HDPL = 1	-	No inaccessible HDP area in user flash memory
HDPL = 2	HDPx_END < HDPx_STRT	No inaccessible HDP area in user flash memory
	HDPx_END = HDPx_STRT	Exactly one page is protected by HDP
	HDPx_END > HDPx_STRT	The area between HDPx_STRT and HDPx_END is HDP protected.
HDPL = 3	HDPx_END < HDPx_STRT and HDPx_EXT = 0	No inaccessible HDP area in user flash memory
	HDPx_END < HDPx_STRT and HDPx_EXT > 0	The area betwen (HDPx_END + HDPx_EXT) is HDP protected
	HDPx_END ≥ HDPx_STRT and HDPx_EXT > 0	The area between min (HDPx_STRT, HDPx_END) and (HDPx_END + HDPx_EXT) is HDP protected

Table 33. HDP protections summary

HDP protection	User flash memory			
	Access to OB HDP area		Access to EXT HDP area	
	HDPL 2 or 3	HDPL1	HDPL3	HDPL1 or HDPL 2
Fetch	BUS ERROR	OK	BUS ERROR	OK
Read	RAZ		RAZ	
Write	WI, WRPERR	If WRP disabled: OK else: WI, WRPERR	WI, WRPERR	If not WRP: OK Else: WI, WRPERR
Erase (mass erase)				

6.5.2 Flash memory register privileged and unprivileged modes

The FLASH registers can be read and written by privileged and unprivileged accesses, depending on the PRIV bit in FLASH_PRIVCFGR.

- When the PRIV bit is reset, all flash memory registers can be read and written by both privileged or unprivileged access.
- When the PRIV bit is set, all flash memory registers can be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

6.5.3 Flash memory banks attributes in case of bank swap

The SWAP_BANK option bit modifies the address of each bank in the memory map. When SWAP_BANK is reset, FLASH bank1 is mapped at the lower address range. When SWAP_BANK is set, FLASH bank1 is mapped at the higher address range. FLASH

bank attributes follow their bank contents so there is no need to modify their setting registers when swapping banks:

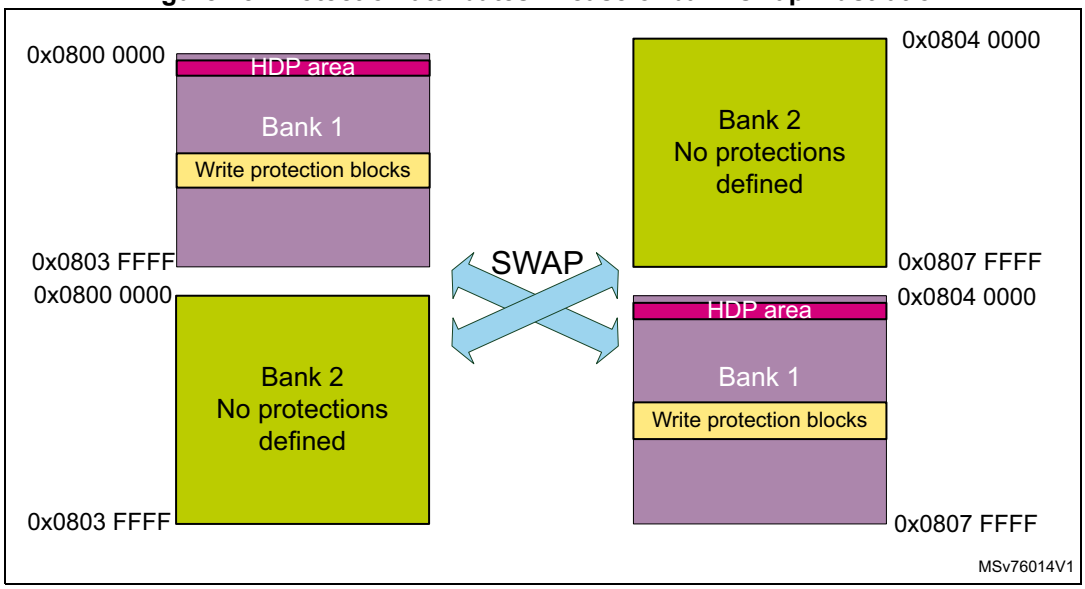
- FLASH write protection page FLASH_WRPG (refer to *Section 6.5.5*)
- Hide protection in FLASH_HDPx registers

The SWAP_BANK is rendered immutable by setting BOOT_LOCK.

Note: The BK_ECC bit in FLASH_OEMKEYR2_CUR and FLASH_ECCDETR, BKSEL bit in FLASH_CR always refers to bank1 (respectively bank2) when it is low (respectively high), regardless of the SWAP_BANK value.

The figure below shows how security attributes and protections behave in case of bank swap.

Figure 15. Protection attributes in case of bank swap illustration



6.5.4 Flash memory configuration protection

The FLASH uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (such as software bugs):

- Option-byte changes
- Write operations
- Erase commands
- Interrupt masking

The FLASH configuration registers protection is summarized in *Table 34*. Registers not present in this table are not protected by a key.

Table 34. FLASH interface register protection summary

Register name	Unlocking register	Protected asset
FLASH_CR	FLASH_KEYR	Write/erase control
FLASH_OPTCR	FLASH_OPTKEYR	FLASH bank option-byte words change

6.5.5 Write protection

The purpose of FLASH write protection is to prevent unwanted modifications to embedded nonvolatile code and/or data.

A write-protected group of pages can neither be erased nor programmed. As a result, a bank erase cannot be performed if some page is write-protected, unless a RDP level transition to L0 is triggered (erasing the whole user flash memory).

Note: Write protection errors are documented in Section 6.8.

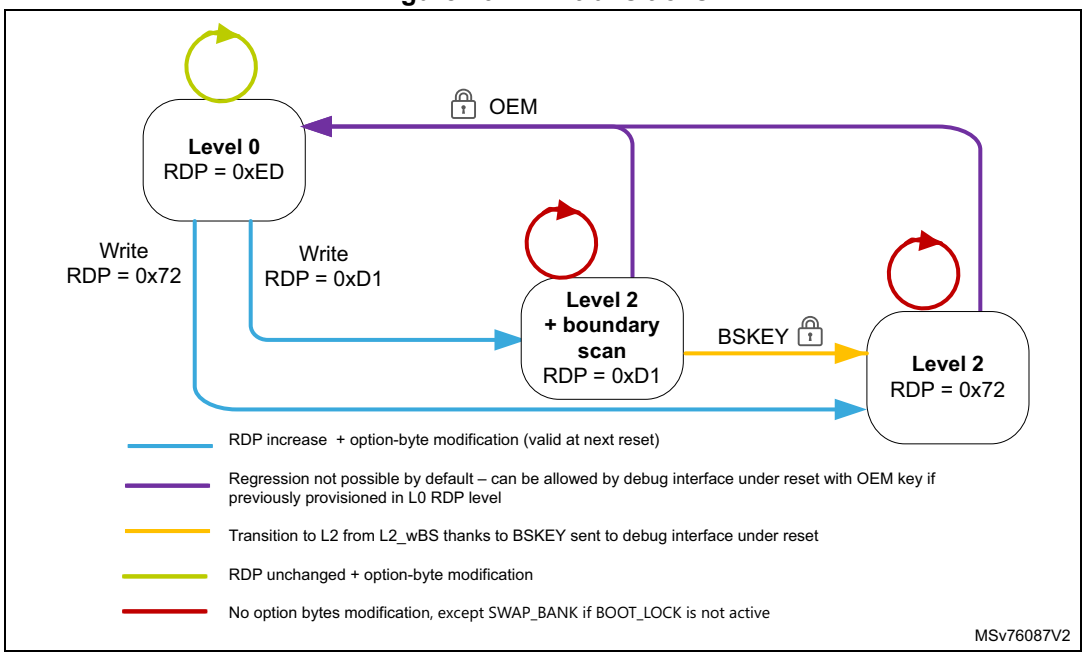
6.5.6 FLASH data area protections

The FLASH can be configured to have 64-Kbyte (or 48-Kbyte if EDATA_EN is set) memory area to store data and emulate EEPROM (see *Section 6.3.10: FLASH data area*). These pages can not be protected by HDP or WRPGS whatever the setting of EDATA_EN.

6.5.7 RDP protection and life cycle management

Nonvolatile states, or debug states, are determined by the RDP levels set by user option bytes. Debug possibility and possibility to change selected security settings are related to this state. While the state is stored in the OB, it is the SBS that controls the debug policy.

Figure 16. RDP transitions



HDP works equally regardless of the RDP level.

Table 35. RDP levels description

RDP_LEVEL	Code in OB	Description
L0 (Open)	0xED	– Full debug capabilities – Read/Write/Erase Flash is possible – Read/Modify the option bytes – Boot in bootloader or user flash based on BOOT0 pin or option bytes – Boundary scan available under reset
L2_wBS (Closed + boundary scan)	0xD1	– No debug capabilities – Read/Write/Erase Flash is possible – All option bytes Read by user code – SWAP_BANK (depending on BOOT_LOCK) & LOCKBL (for OTP) option bytes can be modified – Boot in user flash only – Boundary Scan available under reset
L2 (Closed)	0x72	– No debug capabilities – Read/Write/Erase Flash is possible – All option bytes Read by user code – SWAP_BANK (depending on BOOT_LOCK) & LOCKBL (for OTP) option bytes can be modified – Boot in user flash only

Caution: STMicroelectronics cannot functionally analyze parts with level 2 protection enabled. Regress parts to RDP level 0 before returning them for analysis (refer to OEM RDP lock mechanism).

6.5.8 RDP level transitions

Progressing to more closed levels is the normal product life cycle, that does not require security measures. Transition in direction to open level (RDP Level 0) is a regression, and is possible only thanks to the OEM Key.

There is no restriction on reading the current RDP level.

Transitions to L2 with boundary scan (L2_wBS) or L2 (L2) closed states

Transition from L0 to L2 with or without boundary scan is managed either by software running on the device, or directly, using a debug interface. By software, it is assumed that transitions are triggered by the bootloader, but generally any software running on the device can do a progress transition.

Caution: The new RDP Level will only be valid after next system or power reset.

Transition from L2 with boundary scan to L2 is only feasible thanks the BSKEY option-byte key:

- If the BSKEY key have not been modified under L0 state, the default value apply.
- If the BSKEY key have been modified under L0 state, the user must apply the key populated.

This key must be written under reset and L2 with boundary scan thanks to JTAG in order to progress to L2 RDP level.

Regression to L0 open debug level

This transition is a full regression, possible from either L2_wBS or L2 if an OEM key has been provisioned. The debug tools are used to provide the OEM key, if the key matches the one provisioned under L0, the device regresses securely in RDP L0.

The regression has the following consequences:

- RDP level updated
- User options updated (security settings reset)
- Flash memory mass erase (even write protected pages)

Table 36. RDP levels transitions

From	To(1)		
	L0	L2_wBS	L2
L0	-	OB update thanks to debug or bootloader	
L2_wBS	OEM key regression	-	BSKEY
L2	OEM key regression	X	-

1. ‘X’ = transitions that are not permitted (OPTCHANGEERR raised). ‘-’ = no change.

An attempt to make a illegal transition results in the OPTCHANGEERR.

An incorrect RDP_LEVEL (on OBL) is interpreted as L2.

6.5.9 OEM key management

The OEM 128-bit key (OEMKEY) can be defined in order to permit the RDP regression.

OEM lock activation

The 128-bit key is coded on four registers: FLASH_OEMKEYR1 to FLASH_OEMKEYR4. OEMKEY and BSKEY cannot be read through these registers. They are read as zero.

Caution: OEMKEY can only be modified in readout protection level 0.

In order to activate OEM regression mechanism, the following steps are needed:

1. Check that the RDP is at L0.
2. Write a 128-bit key in FLASH_OEMKEYR1 to FLASH_OEMKEYR4.
3. Launch option modification by setting the OPTSTRT bit in FLASH_CR.
4. Wait for the BSY bit to be cleared and check that OPTWERR is not set.
5. Check that OEMLOCK is set.

Note: Once the OEM KEY has been programmed a first time, the OEMLOCK bit remains set. The user can reprogram the key, but cannot erase it.

OEM key validation mechanism

The OEM RDP lock mechanism is active when the OEMLOCK bit is set. The OEMKEY, can

be validated while in RDP L0.

- Shift OEMKEY[31:0], then OEMKEY[63:32], then OEMKEY[95:64], then OEMKEY[127:96] through JTAG or SWD under reset in DBGMCU_DBG_AUTH_HOST register.
- If this key matches the OEMKEY value:
  - A status bit in DBGMCU_DBG_VALR indicates that the adequate key has been successfully entered, a power sequence is required to clear the validation status bit. Omitting this step will prevent to enter RDP Level 2.
- If the key does not match the OEMKEY value, a power sequence is required to clear the validation status bit and retry.

OEM RDP regression mechanism

The OEM RDP lock mechanism is active when the OEMLOCK bit is set. It authorizes RDP L2 to RDP L0 regression.

- Shift OEMKEY[31:0,] then OEMKEY[63:32], then OEMKEY[95:64], then OEMKEY[127:96] through JTAG or SWD under reset in DBGMCU_DBG_AUTH_HOST register.
- If this key matches the OEMKEY value:
  - The RDP regression is launched by hardware when the reset is released (it is not possible to execute instructions when the key is matching).
  - A power-on reset must be applied (cycle V_{DD} power supply off and on).
- If the key does not match the OEMKEY value, the RDP regression is blocked until next power-on reset.

6.5.10 BS key management

The BS key is used to transition from RDP L2_wBS to L2. This key is coded in FLASH_BSKEYR, and can be modified only under RDP L0. During production, the value 0xAAAA AAAA is preprogrammed in BSKEY option bytes.
In order to personalize BSKEY transition mechanism, the following steps are needed:

1. Check that the RDP is at L0.
2. Write the desired key in FLASH_BSKEYR.
3. Launch option modification by setting the OPTSTRT bit in FLASH_CR.
4. Wait for the BSY bit to be cleared and check that OPTWERR is not set.
5. Check that BSLOCK bit has been set.

BSKEY RDP transition mechanism

In order to transition from RDP L2_wBS to L2 follow these steps:

1. Provide BSKEY[31:0] through JTAG or SWD under reset in DBGMCU_BSKEY_PWD. The key is 0xAAAA AAAA if not updated under L0, otherwise it is the value provisioned in BSKEYR.
2. If this key matches the BSKEY value:
   - The RDP transition is launched by hardware (it is not possible to execute instructions when the key is matching).
   - A power-on reset is applied (cycle V_{DD} power supply off and on).

if the key does not match the BSKEY value, the RDP transition and any access to the flash memory are blocked until a next power-on reset.

6.5.11 One-time-programmable and read-only memory protections

OTP/RO in the flash memory are detailed in *Section 6.3.9*. There is no protection provided by the FLASH interface other than the dedicated write protection.

No write is possible to the RO area, once it was established in manufacturing. *OTP write protection* describes a method of locking out the OTP area.

The access conditions are summarized in the table below.

Table 37. OTP/RO access constraints

-	**Undefined addresses**	**4.5-Kbyte page dedicated to OTP**	**1.5-Kbyte page dedicated to RO**
	0x08FF_F200 - F7FF 0x08FF_FE00 - FFFF	**0x08FF_E000 - F1FF OTP**	**0x08FF_F800 - FDFF RO**
X	Bus error	Bus error	
R		If correct size OK, else bus error	
W		Bus error if incorrect size. WRPERR if block is protected by LOCKBL, else OK.	If correct size WI, else bus error
erase	Not applicable		

6.6 System memory

6.6.1 System memory introduction

System memory stores bootloader firmware that is programmed by ST during production. The bootloader provides runtime services to user firmware. These services are described hereafter in this section.

6.6.2 Bootloader user functions

As other microcontroller peripherals features and mapping, the bootloader library functions are exposed to user within the CMSIS device header file provided by the STM32CubeC5 firmware package. Refer to the online HAL/LL drivers documentation to get more details regarding STM32CubeC5 firmware package. Bootloader library provides runtime services through prefixed EXITHDP functions.

Table 38. C defined macro for HDP services

C defined macro	Location in flash memory
EXITHDP_PFUNC	0x0BF883E0

EXITHDPLIB

The user firmware calls EXITHDPLIB functions using EXITHDPLIB_PFUNC C defined macro, that points to a location within system memory.

Table 39. EXITHDPlib interface function list

Library	Function
EXITHDP_PFUNC	JumpHDPLvl2
	JumpHDPLvl3

The EXITHDPLIB functions are described within section hereafter.

a) JumpHDPLvl2

Prototype:

```
uint32_t JumpHDPLvl2(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
EXITHDPLIB_PFUNC->JumpHDPLvl2((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:

Input parameter, address of the next vector table to apply.

The vector table format is the one used by the Cortex-M33 core.

- MPUIndex:

Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the Reset Handler of the vector table. The vector table Reset Handler function shall belong to the MPU region.

Descriptions:

User calls JumpHDPLvl2 to:

- Close user Flash HDPL1 area by incrementing HDPL to 2
- Jump to the reset handler embedded within the Vector Table which address is passed as input parameter

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed Vector Table and jumps to the Reset Handler function supported by the Vector Table too. JumpHDPLvl2 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

Errors:

In case of failure (bad input parameter value), EXITHDPLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

b) JumpHDPLvl3

Prototype

```
uint32_t JumpHDPLvl3(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
EXITHDPLIB_PFUNC->JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  Input parameter, address of the next vector table to apply.
  The vector table format is the one used by the Cortex-M33 core.
- MPUIndex:
  Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the Reset Handler of the vector table. The vector table Reset Handler function shall belong to the MPU region.

Descriptions:

User calls JumpHDPLvl3 to:

- Close user Flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3
- Then jump to the reset handler embedded within the Vector Table which address is passed as input parameter

After closing HDPL1/2, JumpHDPLvl3 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed Vector Table and jumps to the Reset Handler function supported by the Vector Table too. JumpHDPLvl3 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

Errors:

In case of failure (bad input parameter value), JumpHDPLvl3 returns 0xF5F5F5F5.

6.7 FLASH low-power modes

The table below summarizes the FLASH behavior in STM32 low-power modes. The FLASH belongs to the core domain.

Table 40. Effect of low-power modes on the FLASH

Power mode	Allowed if FLASH busy	FLASH power mode
Run/Sleep	Yes	Run
Stop (clock stopped)	No	Clock gated (in normal or low-power mode)
Standby	No	Off

The procedure to switch the FLASH into various power mode (run, clock gated, stopped, off) is described hereafter.

Note: For more information on microcontroller power states, refer to Section 8: Power control (PWR).

Managing FLASH domain switching to Stop or Standby mode

As explain in *Table 40*, if the FLASH informs the RCC that it is busy (BSY, DBNE, WBNE is set), the microcontroller cannot switch the core domain to Stop or Standby mode.

There are two ways to release the FLASH:

- Reset the WBNE busy flag in FLASH_SR by any of the following actions:
  - Complete the write buffer with missing data.
  - Force the write operation without filling the missing data by activating the FW bit in FLASH_CR. This forces all missing data “high”.
- Poll BSY busy bits in FLASH_SR until they are cleared. This indicates that all recorded write, erase, and option change operations are complete.

The microcontroller can then switch the domain to Stop or Standby mode.

6.8 FLASH error management

The FLASH automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- Write protection error (WRPERR) (*Section 6.8.1*)
- Programming sequence error (PGSERR) (*Section 6.8.2*)
- Error correction code error (ECCC/ECCD) (*Section 6.8.3*)
- Option-byte change error (OPTCHANGEERR) (*Section 6.8.4*)

The application software can individually enable the interrupt for each error, as detailed in *Section 6.9*.

Note: For all errors, the application software must clear the error flag before attempting a new modify operation.

Since there is just one write buffer and only one operation is allowed at a time, the write control bits and status bits are shared by both banks:

- Write control bits (PG, FW, EOPIE, WRPERRIE, PGSERRIE, STRBERRIE, INCERRIE, CLR_EOP, CLR_WRPERR, CLR_STRBERR, CLR_INCERR, CLR_PGSERR) controls bank1 and 2 at the same time.
- Write status flag reports (WBNE, DBNE, EOP, WRPERR, PGSERR, STRBERR, INCERR, BSY) error for bank1 and 2 at the same time.

6.8.1 Write protection error (WRPERR)

When an illegal erase/program operation is attempted to the nonvolatile memory, the FLASH sets the write protection error flag WRPERR in FLASH_SR.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A page write-protected with WRP.
- An HDP area while the HDPL has made it inaccessible.

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- The system flash memory
- A user page write-protected with WRP
- An OTP block, locked with LOCKBL
- A reserved area

When WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank.

If this error is detected the write buffer is invalidated.

Note: WRPERR flag has to be cleared before any erase/program operation.

WRPERR flag is cleared by setting CLR_WRPERR bit to 1 in FLASH_CCR.

If WRPERRIE bit in FLASH_CR is set to 1, an interrupt is generated when WRPERR flag is raised (see *Section 6.9* for details).

6.8.2 Programming sequence error (PGSERR)

When the programming sequence is incorrect, the FLASH interface sets the programming sequence error flag PGSERR in FLASH_SR.

More specifically, PGSERR flag is set when one of below conditions is met:

- An error like INCERR, WRPERR, PGSERR, STRBERR, OPTCHANGEERR have not been cleared before requesting a new write or erase operation or options change.
- For erase, PGSERR is set:
  - If missing erase operation (FLASH_CR): STRT = 1 with (MER = 0, PER = 0, and BER = 0)
  - If page and bank erase requested at the same time (STRT and more than one of MER, PER and BER)
  - If PG is set during erase operation (it ensures that no erase request and write request happen at the same time): (STRT = 1) with (PG = 1)
  - If an erase operation is started while write buffer is waiting for next data: (STRT = 1) with WBNE = 1.
  - If an erase operation is started while DBNE = 1.

- For programming, PGSERR is set:
  - If missing write flag: A write operation is requested but the program enable bit PG has not been set in FLASH_CR prior to the request.
  - If a write operation to flash data area is requested but PG = 0.
  - If an AHB write request is received and (PER = 1, BER = 1, or MER = 1)
  - If a 16-bit data access requested while WBNE = 1.
- For options change, PGSERR is set:
  - If an option change is started while write buffer is waiting for next data: OPTSTRT = 1 with WBNE = 1
  - If OPTSTRT is set with DBNE = 1.

When PGSERR flag is raised as consequence of failed write attempt (FLASH programming), the current program operation is aborted, and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

Note: When PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. Hence it is recommended to generate interrupts on PGSERR, and to verify in the interrupt handler if the last write operation has been successful by reading back the value in the flash memory.

The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

PGSERR flag is cleared by setting CLR_PGSERR bit to 1 in FLASH_CCR.

If PGSERRIE bit in FLASH_CR is set to 1, an interrupt is generated when PGSERR flag is raised (see *Section 6.9* for details).

6.8.3 Strobe error (STRBERR)

When the application software writes several times to the same byte in the write buffer, the FLASH sets the strobe error flag STRBERR in FLASH_SR, whatever the target bank of the write access.

When STRBERR is raised, the current program operation is aborted and the write buffer is invalidated.

STRBERR is cleared by setting CLR_STRBERR bit to 1 in FLASH_CCR.

If STRBERRIE bit in FLASH_CR is set to 1, an interrupt is generated when STRBERR flag is raised (see *Section 6.9* for details).

6.8.4 Inconsistency error (INCERR)

When a programming inconsistency is detected, the FLASH sets the inconsistency error flag INCERR in FLASH_SR.

More specifically, INCERR flag is set when the following condition is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.

Note: INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow the sequence below to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when INCERR flag is raised.
2. Stop all write requests to FLASH.
3. Clear INCERR bit.
4. Restart the write operations where they have been interrupted.

INCERR flag is cleared by setting CLR_INCERR bit to 1 in FLASH_CCR.

If INCERRIE bit in FLASH_CR register is set to 1, an interrupt is generated when INCERR flag is raised (see *Section 6.9* for details).

6.8.5 Error correction code error (ECCC/ECCD)

When a single-error correction is detected during a read, the FLASH interface sets the single-error correction flag ECCC in FLASH_ECCCORR.

When two ECC errors are detected during a read, the FLASH interface sets the double error detection flag ECCD in FLASH_ECCDETR.

When ECCC flag is raised, the corrected read data are returned. Hence the application can ignore the error and request new read operations. When ECCD flag is raised a NMI is generated. The software must invalidate the instruction cache (CACHEINV = 1) in the NMI interrupt service routine when the ECCD flag is set. This NMI can be masked in SBS registers *Section 12.5.12: SBS ECC NMI mask register (SBS_ECCNMIR)* for data access (OTP, data area, RO data).

When ECCC or ECCD flag is raised, the address of the FLASH word that generated the error is saved in the FLASH_ECCCORR (FLASH_ECCDETR). If the address corresponds to a read-only area, to a EDATA page (in user mode or in FLASH data area mode) or to an OTP area, the OTP_ECC or SYSF_ECC or EDATA_ECC bit is also set to 1 in FLASH_ECCCORRR (FLASHBS_ECCDETR). This register is automatically cleared when the associated flag that generated the error is reset.

A BK_ECC flag indicates in which FLASH bank the error occurred.

A SYSF_ECC flag indicates error detected in the system FLASH area.

Table 41. Locating ECC failure

OTP_ECC	SYSF_ECC	BK_ECC	DATA_ECC	FLASH area	ADDR_ECC min	ADDR_ECC max
0	0	0/1	0	User FLASH	0x0000	0x7FFF(1)
0	1	0/1	0	System FLASH	0x0000	0x0FFF
1	0	1	0	OTP	0x0200	0x037F
0	0	0/1	1	EDATA page 0	0x0000	0x007F
0	0	0/1	1	EDATA page 1	0x0080	0x00FF
0	0	0/1	1	EDATA page 2	0x0100	0x017F

Table 41. Locating ECC failure (continued)

OTP_ECC	SYSF_ECC	BK_ECC	DATA_ECC	FLASH area	ADDR_ECC min	ADDR_ECC max
0	0	0/1	1	EDATA page 3	0x0180	0x01FF
0	0	0/1	1	EDATA page 4	0x0200	0x027F
0	0	0/1	1	EDATA page 5	0x0280	0x02FF
0	0	0/1	1	EDATA page 6	0x0300	0x037F
0	0	0/1	1	EDATA page 7	0x0380	0x03FF
0	0	0/1	1	EDATA page 8	0x0400	0x047F
0	0	0/1	1	EDATA page 9	0x0480	0x04FF
0	0	0/1	1	EDATA page 10	0x0500	0x057F
0	0	0/1	1	EDATA page 11	0x0580	0x05FF
0	0	0/1	1	EDATA page 12	0x0600	0x067F
0	0	0/1	1	EDATA page 13	0x0680	0x06FF
0	0	0/1	1	EDATA page 14	0x0700	0x077F
0	0	0/1	1	EDATA page 15	0x0780	0x07FF

1. ADDR_ECC maximum depends on the flash memory size of the product.

Note: In case of successive single correction or double detection errors, only the address corresponding to the first error is stored in FLASH_ECCCORR (FLASH_ECCDETR).

It is mandatory to clear ECCC or ECCD flags before starting a new read operation.

As the ECC interface is shared by the two banks, if the same error is registered simultaneously, a physical bank1 error is registered. Both errors are registered if one bank reports ECCD and the other bank ECCC.

ADDR_ECC value indicates the FLASH line where the ECC error occurred.

If the EDATA_EN option bit is set, the EDATA sectors are programmed and read as 16-bit accesses, (memory row being 137 bits long: 128-bit + 9-bit ECC). Consequently, the arrangement of the memory is modified. The first 32 bits of each row of a sector are filled first, then the bits from 32 to 63 and finally from 64 to 96 (the rest of the memory row contains ECC code). The ADDR_ECC indicates the memory line holding the failure. To identify the data corrupted on the line, DATA_ADDR_ECC bitfield value can be used to

discriminate which column holds the failed data (only in case of error relative to data area or OTP).

Example: ADDR_ECC = 0x0312 and DATA_ADDR_ECC = 0x3.

- With ADDR_ECC between 0x0300 and 0x037F, the address of failure is in page 6, starting at 0x0900 2400.
- DATA_ADDR_ECC = 0b01X indicates the concerned column is the second, starting from 0x0900 2600.
- ADDR_ECC ending with 0xXX12 means the concerned row in memory is 0x12. With 4 bytes per column, this gives the 32-bit address as 0x0900 2648.
- DATA_ADDR_ECC = 0b0X1 indicates the concerned final address is the second 16-bit data, so 0x0900 264A.

ECCC (respectively ECCD) flag is cleared by setting to 1 the ECCC bit (respectively ECCD bit) in FLASH_ECCCORR (FLASH_ECCDETR).

If ECCC bit in FLASH_ECCCORR is set to 1, an interrupt is generated when ECCC flag is raised. Only NMI is generated for the ECCD (see *Section 6.9* for details).

6.8.6 Option-byte change error (OPTCHANGEERR)

When the FLASH finds an error during an option-byte change operation, it aborts the operation, and sets the option-byte change error flag OPTCHANGEERR in FLASH_SR.

The error is raised after OPTSTRT bit set if one of the following occurs:

- Any forbidden RDP_LEVEL transition (*Table 36*).
- SWAP_BANK is locked out by BOOT_LOCK.
- Attempt to modify OB with invalid value not recognized by the specification (enumerated magic numbers only)
- Attempt to modify OB in RDP level where it is not allowed (usually a state open for debug is required for change)

Note: Exceptions and details provided in the OB description (see Section 6.4.9).

OPTCHANGEERR flag is cleared by setting CLR_OPTCHANGEERR bit to 1 in FLASH_CCR.

If OPTCHANGEERRIE bit in FLASH_CR is set to 1, an interrupt is generated when OPTCHANGEERR flag is raised (see *Section 6.9* for details).

It is mandatory to clean the OPTCHANGEERR flag before starting a new operation (option change, erase, or write).

6.8.7 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On main AHB system bus for access targeting code and data with 9-bit ECC.
  - Access to invalid address (including data addresses forbidden for code use)
  - Fetching from HDP area with incorrect HDPL value
  - Fetching from privilege area in unprivileged mode
- On AHB configuration or system bus for accesses targeting OTP/RO (all addresses using 6-bits ECC):
  - Wrong key input to FLASH_KEYR or FLASH_OPTKEYR
  - 8-bit accesses to system AHB interface
  - Wrong unlock sequence on a register

6.9 FLASH interrupts

The FLASH can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see *Section 6.8*)
  - Single ECC error correction during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - Option-byte change operation error
- Security errors (see *Section 6.8*)
  - Write protection error
- Miscellaneous events (described below)
  - End of programming

A NMI is raised on double-ECC-error detection during read operation.

The user can individually enable or disable FLASH interrupt sources by changing the mask bits in FLASH_CR, FLASH_ECCCORR, and FLASH_ECCDETR. Setting the appropriate mask bit to 1 enables the interrupt.

Note: Before writing, FLASH_CR must be unlocked as explained in Section 6.5.4.

The table below summarizes the available FLASH interrupt features. As mentioned, some flags need to be cleared before a new operation is triggered.

Table 42. FLASH interrupt request

Interrupt event flag	Error flag label in register	Enable control bit	Clear flag to resume operation
End of operation event	EOP	EOPIE	N/A
Write protection error	WRPERR	WRPERRIE	Yes
Programming sequence error	PGSERR	PGSERRIE	Yes
Strobe error	STRBERR	STRBERRIE	Yes

Table 42. FLASH interrupt request (continued)

Interrupt event flag	Error flag label in register	Enable control bit	Clear flag to resume operation
Inconsistency error	INCERR	INCERRIE	Yes
Option byte error	OPTCHANGEERR	-	Yes
ECC single error correction event	ECCC	ECCCIE	No
ECC double error detection event(1)	ECCD	Disabled	No

1. NMI.

The status of the individual maskable interrupt sources described in *Table 42* (except for option-byte error and ECC) can be read in FLASH_SR register. They can be cleared by setting to 1 the adequate bit in FLASH_CCR.

Note: No unlocking mechanism is required to clear an interrupt.

End of operation event

Setting the end of operation interrupt enable bit (EOPIE) in FLASH_CR enables the generation of an interrupt at the end of an erase operation, a program operation, or an option-byte change.

Setting CLR_EOP bit to 1 in FLASH_CCR clears EOP flag.

6.10 FLASH registers

Each register is assigned a offset address and a reset value. In case of registers representing option-byte value, the reset value is determined by the OBL process. In case of success, the reset value is loaded from OB. In case of OBL failure, a highly restrictive default value is set.

6.10.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000

Reset value: 0x000X 0027

Bit 16 not reset by system reset

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. For more details, refer to *Section 6.3.4* and *Section 6.3.5*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTEN	Res.	Res.	WRHIGHFREQ[1:0]		LATENCY[3:0]			
							rw			rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **EMPTY**: Main flash memory area empty (not reset by system reset)

Set and reset by software. This bit indicates whether the boot location of the main flash memory area (indicated by BOOT_ADD) is erased or has a programmed value.

0: Boot address in main flash memory area programmed

1: Boot address in main flash memory area empty

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable

When this bit value is modified, the user must read back this register to be sure PRFTEN has been taken into account. This bit is used to control the prefetch.

0: Prefetch disabled

1: Prefetch enabled when latency is at least 1 wait-state

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **WRHIGHFREQ[1:0]**: FLASH signal delay

These bits are used to control the delay between nonvolatile memory signals during programming operations. The application software has to program them to the correct value depending on the FLASH frequency. Refer to *Table 20* for details.

Note: No check is performed to verify that the configuration is correct.

Two WRHIGHFREQ values can be selected for some frequencies.

Bits 3:0 **LATENCY[3:0]**: Read latency

These bits are used to control the number of wait states used during read operations on both nonvolatile memory banks. The application software has to program them to the correct value depending on the FLASH frequency and voltage conditions.

0000: 0 wait state

0001: 1 wait state

0010: 2 wait states

0011-1111: {v} wait states

Note: No check is performed by hardware to verify that the configuration is correct.

6.10.2 FLASH key register (FLASH_KEYR)

Address offset: 0x004

Reset value: 0x0000 0000

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. This register is write-only.

The following values must be programmed consecutively to unlock FLASH_CR and to allow programming/erasing it. A wrong sequence locks FLASH_CR until next system reset.

1st key = 0x4567 0123

2nd key = 0xCDEF 89AB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Nonvolatile memory configuration access unlock key

6.10.3 FLASH option key register (FLASH_OPTKEYR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.This register is write-only.

The following values must be programmed consecutively to unlock FLASH_OPTCR. A wrong sequence locks FLASH_OPTCR until next system reset.

1st key = 0x0819 2A3B

2nd key = 0x4C5D 6E7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OPTKEY[31:0]**: FLASH option-byte control access unlock key

6.10.4 FLASH operation status register (FLASH_OPSR)

Address offset: 0x018

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CODE_OP[2:0]			Res.	Res.	Res.	Res.	OTP_OP	Res.	BK_OP	DATA_OP	Res.	Res.	Res.	Res.	Res.
r	r	r					r		r	r					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_OP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 **CODE_OP[2:0]**: Flash memory operation code
- 000: No FLASH operation ongoing during previous reset
- 001: Single write operation interrupted
- 010: Reserved
- 011: Page erase operation interrupted
- 100: Bank erase operation interrupted
- 101: Mass erase operation interrupted
- 110: Option change operation interrupted
- 111: Reserved

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **OTP_OP**: OTP operation interrupted
This bit indicates that a reset interrupted an ongoing operation in the OTP area.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **BK_OP**: Interrupted operation bank
This bit indicates which bank was concerned by operation. (Refer to *Section Table 24.: Memory map and the swapping option (STM32C53x/542 devices)*)

Bit 21 **DATA_OP**: FLASH data area operation interrupted
This bit indicates if user data FLASH area is concerned by an operation.

Bits 20:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_OP[15:0]**: Interrupted operation address

6.10.5 FLASH option control register (FLASH_OPTCR)

Address offset: 0x01C

Reset value: 0xX000 0001

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Access: No wait state when no flash memory operation is ongoing.

This register is not accessible in write mode when the BSY bit is set. Any attempt to write to it while the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	OPTLOCK
														rw	rs

Bit 31 **SWAP_BANK**: Bank-swapping option configuration bit

SWAP_BANK controls whether bank1 and bank2 are swapped or not. This bit is loaded with the SWAP_BANK bit of FLASH_OPTSR_CUR register only after reset or POR.

0: Bank1 and bank2 not swapped

1: Bank1 and bank2 swapped

Bits 30:2 Reserved, must be kept at reset value.

Bit 1 **OPTSTRT**: Option-byte start change option configuration bit

OPTSTRT triggers an option byte change operation. The user can set OPTSTRT only when the OPTLOCK bit is cleared to 0. It is set only by software and cleared when the option-byte change is completed or an error occurs (PGSERR or OPTCHANGEERR). It is reset at the same time as BSY bit.

The user application cannot modify any FLASH_XXX_PRG register until the option change operation has been completed.

Before setting this bit, the user has to write the required values in FLASH_XXX_PRG. FLASH_XXX_PRG registers are locked until the option-byte change operation has been executed in nonvolatile memory.

Bit 0 **OPTLOCK**: FLASH_OPTCR lock option configuration bit

The OPTLOCK bit locks the FLASH_OPTCR register as well as all _PRG registers. The correct write sequence to FLASH_OPTKEYR register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_OPTKEYR is performed twice, this bit remains locked until next system reset.

It is possible to set OPTLOCK by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When OPTLOCK changes from 0 to 1, the others bits of FLASH_OPTCR register do not change.

0: FLASH_OPTCR register unlocked

1: FLASH_OPTCR register locked.

6.10.6 FLASH status register (FLASH_SR)

Address offset: 0x020

Reset value: 0x0000 000X

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERR	Res.	Res.	INCERR	STRBERR	PGSERR	WRPERR	EOP
								r			r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BSLOCK	OEMLOCK	Res.	Res.	Res.	Res.	DBNE	Res.	WBNE	BSY
						r	r					r		r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERR**: Option-byte change error flag

This flag indicates that an error occurred during an option-byte change operation. When This bit is set to 1, the option-byte change operation did not successfully complete. An interrupt is generated when this flag is raised if OPTCHANGEERRIE in FLASH_NSCR is set to 1.

Writing 1 to CLR_OPTCHANGEERR in FLASH_CCR clears OPTCHANGEERR.

0: No option-byte change errors occurred.

1: One or more errors occurred during an option-byte change operation.

Note: OPTSTRT in FLASH_OPTCR cannot be set while OPTCHANGEERR is set.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **INCERR:** Inconsistency error flag

This flag is raised when an inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1. Writing 1 to CLR_INCERR in FLASH_CCR clears INCERR.

0: No inconsistency error occurred.

1: An inconsistency error occurred.

Bit 19 **STRBERR:** Strobe error flag

This flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if STRBERRIE is set to 1. Writing 1 to CLR_STRBERR in FLASH_CCR clears STRBERR.

0: No strobe error occurred.

1: A strobe error occurred.

Bit 18 **PGSERR:** Programming sequence error flag

This flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Writing 1 to CLR_PGSERR in FLASH_CCR clears PGSERR.

0: No sequence error occurred.

1: A sequence error occurred.

Bit 17 **WRPERR**: Write protection error flag

This flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Writing 1 to CLR_WRPERR bit in FLASH_CCR clears WRPERR.

0: No write-protection error occurred.

1: A write-protection error occurred.

Bit 16 **EOP**: End of operation flag

This flag is set when an operation (program/erase) completes. An interrupt is generated if EOPIE is set to 1. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared by writing 1 to CLR_EOP in FLASH_CCR.

0: No operation completed.

1: An operation completed.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **BSLOCK**: BS lock

This bit indicates that the BS key read during the option byte loading is not default (default is 0xAAAA AAAA).

When set, the BS personalized key mechanism to transition to L2 from L2_wBS is active.

Bit 8 **OEMLOCK**: OEM lock

This bit indicates that the OEM key read during the OBL is not virgin. When set, the OEM RDP lock mechanism is active. The lock mechanism allows the user to regress from RDP L2_wBS or L2 to L0. This bit is set once a 128-bit password is written in OEMKEYRx registers and written in memory. It cannot be cleared.

0: Lock mechanism is not active, progressing to RDP L2_wBS or L2 is irrevocable.

1: Lock mechanism is active, a key exist to regress to RDP L0.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: Data buffer not empty flag

This flag is set when the FLASH is processing 6-bit ECC data (OTP or EDATA area when EDATA_EN option bit set) in a dedicated buffer. This bit cannot be cleared by software. The hardware resets it once the buffer is free.

0: Data buffer not used

1: Data buffer used, wait

Bit 2 Reserved, must be kept at reset value.

Bit 1 **WBNE**: Write buffer not empty flag

This flag is set when the FLASH is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:

- The application software forces the write operation using FW bit in FLASH_CR.
- The FLASH detects an error that involves data loss.

This bit cannot be reset by software writing 0 directly. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: Write buffer empty or full

1: Write buffer waiting data to complete

Bit 0 **BSY**: Busy flag

This flag indicates that a flash memory is busy by an operation (write, erase, option-byte change). It is set at the beginning of a flash memory operation, and cleared when the operation finishes or an error occurs.

0: No programming, erase, or option-byte change operation being executed

1: Programming, erase, or option byte change operation being executed

6.10.7 FLASH control register (FLASH_CR)

Address offset: 0x028

Reset value: 0x0000 0001

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Access: No wait state when no flash memory operation is ongoing.

This register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKSEL	Res.	EDATASEL	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERRIE	Res.	Res.	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE
rw		rw						rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER	Res.	Res.	PNB[6:0]							STRT	FW	BER	PER	PG	LOCK
rw			rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rs

Bit 31 **BKSEL**: Bank selector bit

This bit can only be programmed when LOCK is cleared to 0. This bit selects physical bank, SWAP_BANK setting is ignored.

0: Bank1 is selected for bank erase (BER)/page erase (PER)/interrupt enable.

1: Bank2 is selected for BER/PER.

Bit 30 Reserved, must be kept at reset value.

Bit 29 **EDATASEL**: EDATA erase selector bit

This bit can only be programmed when LOCK is cleared to 0. This bit selects if page erased concern EDATA page or User main flash pages. This bit has no effect if PER bit is reset.

0: Main FLASH page erase

1: FLASH data area EDATA page erase

Note: During a bank erase, the user and the EDATA sectors are always erased, regardless of this bit setting.

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERRIE**: Option-byte change error interrupt enable bit

This bit controls if an interrupt has to be generated when an error occurs during an option-byte change. This bit can be programmed only when LOCK bit is cleared to 0.

0: No interrupt is generated when an error occurs during an option-byte change.

1: An interrupt is generated when and error occurs during an option-byte change.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **INCERRIE**: Inconsistency error interrupt enable bit

When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.

0: No interrupt generated when a inconsistency error occurs.

1: Interrupt generated when a inconsistency error occurs.

Bit 19 **STRBERRIE**: Strobe error interrupt enable bit

When this bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.

0: No interrupt generated when a strobe error occurs.

1: Interrupt generated when strobe error occurs.

Bit 18 **PGSERRIE**: Programming sequence error interrupt enable bit

When this bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.

0: No interrupt generated when a sequence error occurs.

1: Interrupt generated when sequence error occurs.

Bit 17 **WRPERRIE**: Write protection error interrupt enable bit

When this bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.

0: No interrupt generated when a protection error occurs.

1: Interrupt generated when a protection error occurs.

Bit 16 **EOPIE**: End of operation interrupt control bit

Setting this bit to 1 enables the generation of an interrupt at the end of a program or erase operation. EOPIE can be programmed only when LOCK is cleared to 0.

0: No interrupt generated at the end of operation.

1: Interrupt enabled when at the end of operation.

Bit 15 **MER**: Mass erase request

Setting this bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0. If BER or PER are both set, a PGSERR is raised.

0: Mass erase not requested

1: Mass erase requested

Note: An error is triggered when a mass erase is required and some pages are protected.

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:6 **PNB[6:0]**: Page erase selection number

These bits are used to select the target page for an erase operation (they are unused otherwise). PNB can be programmed only when LOCK is cleared to 0.

0x00: Page 0 selected

0x01: Page 1 selected

0x02-0x3F: Page {v} selected

Note: Values corresponding to addresses outside the main memory are not allowed. Refer to Table 24: Memory map and the swapping option (STM32C53x/542 devices), Table 25: Memory map and the swapping option (STM32C55x/562 devices), and Table 26: Memory map and the swapping option (STM32C59x/5A3 devices)

Bit 5 **STRT**: Erase start control bit

This bit is used to start a page erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.

This bit is reset at the end of the operation or when an error occurs. It cannot be reseted by software.

Bit 4 **FW:** Write forcing control bit

This bit forces a write operation even if the write buffer is not full. In this case, all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0. The FLASH resets this bit when the corresponding operation has been acknowledged.

Note: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.

Write forcing is effective only if the write buffer is not empty (in particular, FW does not start several write operations when the force-write operations are performed consecutively). Since there is just one write buffer, FW can force a write in bank1 or bank2.

Bit 3 **BER**: Bank-erase request

Setting this bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0. If MER and PER are also set, a PGSERR is raised.

0: Bank erase not requested

1: Bank erase requested

Note: Write protection error is triggered when a bank erase is required and some pages are protected.

Bit 2 **PER**: Page-erase request

Setting this bit to 1 requests a sector erase. PER can be programmed only when LOCK is cleared to 0. If MER and PER are also set, a PGSERR is raised.

0: Page erase not requested

1: Page erase requested

Bit 1 **PG**: Programming control bit

This bit can be programmed only when LOCK is cleared to 0. It allows programming in bank1 and bank2.

0: Programming disabled

1: Programming enabled

Bit 0 **LOCK**: Configuration lock bit

This bit locks FLASH_CR. The correct write sequence to FLASH_KEYR unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_KEYR is performed twice, this bit remains locked until the next system reset.

LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_CR register do not change.

0: FLASH_CR unlocked

1: FLASH_CR locked

6.10.8 FLASH clear control register (FLASH_CCR)

Address offset: 0x030

Reset value: 0x0000 0000

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_OPTCHANGEERR	Res.	Res.	CLR_INCERR	CLR_STRBERR	CLR_PGSERR	CLR_WRPERR	CLR_EOP
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CLR_OPTCHANGEERR:** OPTCHANGEERR clear bit
Setting this bit to 1 clears the corresponding flag in FLASH_SR.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **CLR_INCERR**: INCERR flag clear bit
Setting this bit to 1 resets to 0 INCERR flag in FLASH_SR.

Bit 19 **CLR_STRBERR**: STRBERR flag clear bit
Setting this bit to 1 resets to 0 STRBERR flag in FLASH_SR.

Bit 18 **CLR_PGSERR**: PGSERR flag clear bit
Setting this bit to 1 resets to 0 PGSERR flag in FLASH_SR.

Bit 17 **CLR_WRPERR**: WRPERR flag clear bit
Setting this bit to 1 resets to 0 WRPERR flag in FLASH_SR.

Bit 16 **CLR_EOP**: EOP flag clear bit
Setting this bit to 1 resets to 0 EOP flag in FLASH_SR.

Bits 15:0 Reserved, must be kept at reset value.

6.10.9 FLASH privilege configuration register (FLASH_PRIVCFGR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register can be read by both privileged and unprivileged access. It can only be written by privileged mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PRIV**: Privilege attribute
0: Access to registers is always granted.
1: Access to registers is denied in case of unprivileged access.

Bit 0 Reserved, must be kept at reset value.

6.10.10 FLASH HDP extension register (FLASH_HDPEXTR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. The register can only be modified in HDPL ≤ 2.

The values in this register cannot be decremented. Attempt to write a lower value that current is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[5:0]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_EXT[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **HDP2_EXT[5:0]:** HDP area extension in 8-Kbyte pages in bank2
Extension is added after the HDP2_END page (included).

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **HDP1_EXT[5:0]**: HDP area extension in 8-Kbyte pages in bank1
Extension is added after the HDP1_END page (included).

6.10.11 FLASH option status register (FLASH_OPTSR_CUR)

Address offset: 0x050

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0x20B0 72D8 (value in case of double ECC issue during OBL).

ST production value: refer to *Table 28: Option-byte organization*

This read-only register reflects the current values of corresponding option bits.

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	SINGLE_BANK	EDATA_EN	Res.	Res.	Res.	Res.	Res.	BOOT0	BOOT_SEL	IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.
r	r	r						r	r	r	r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDP_LEVEL[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r		r	r			

Bit 31 **SWAP_BANK**: Bank-swapping option status bit
SWAP_BANK reflects whether bank1 and bank2 are swapped or not.
SWAP_BANK is loaded to SWAP_BANK in FLASH_OPTCR after a reset.
0: Bank1 and bank2 not swapped
1: Bank1 and bank2 swapped

Bit 30 **SINGLE_BANK**: Dual-bank selection option status bit
For products:
- STM32C53x/542 with 128-Kbyte user memory
- STM32C55x/562 with 256-Kbyte user memory
- STM32C59x/5A3 with 512-Kbyte user memory

SINGLE_BANK reflects single or dual-bank memory selection.
0: User flash is split between bank 1 and bank 2
1: User flash is located in one bank
Note: This bit has no impact on EDATA sectors.

Bit 29 **EDATA_EN:** FLASH data area enable
0: No FLASH data area (EDATA pages are 128-bit writable)
1: FLASH data area is enabled (EDATA pages are 16/32-bit writable)

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **BOOT0**: Boot 0 option bit
0: BOOT0 = 0
1: BOOT0 = 1

Bit 22 **BOOT_SEL**: Boot 0 source selection
0: BOOT0 signal is defined by the BOOT0 option bit.
1: BOOT0 signal is defined by BOOT0 pin value (legacy mode).

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option status bit
When this bit is set, the IWDG is frozen in system Standby mode.
0: IWDG frozen in Standby mode
1: IWDG keeps running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option status bit
When set the independent watchdog IWDG is in system Stop mode.
0: IWDG frozen in system Stop mode
1: IWDG keeps running in system Stop mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:8 **RDP_LEVEL[7:0]**: RDP level code (based on Hamming 8,4)
See *Section 6.5.8*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option status bit
0: A reset is generated when entering Standby mode on core domain.
1: No reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option status bit
0: A reset is generated when entering Stop mode on core domain.
1: No reset generated when entering Stop mode on core domain..

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option status bit
0: WWDG controlled by hardware
1: WWDG controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option status bit
0: IWDG controlled by hardware
1: IWDG controlled by software

Bits 2:0 Reserved, must be kept at reset value.

6.10.12 FLASH option status register (FLASH_OPTSR_PRG)

Address offset: 0x054

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	SINGLE_BANK	EDATA_EN	Res.	Res.	Res.	Res.	Res.	BOOT0	BOOT_SEL	IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.
rw	rw	rw						rw	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDP_LEVEL[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw			

Bit 31 **SWAP_BANK**: Bank-swapping option configuration bit
This bit is used to configure whether the bank1 and bank2 are swapped or not. This bit is loaded with SWAP_BANK in FLASH_OPTSR_CUR after a reset.
0: Bank1 and bank2 not swapped
1: Bank1 and bank2 swapped

Bit 30 **SINGLE_BANK**: Dual-bank option configuration bit
For products:
- STM32C53x/542 with 128-Kbyte user memory
- STM32C55x/562 with 256-Kbyte user memory
- STM32C59x/5A3 with 512-Kbyte user memory

This bit configures memory as single or dual bank. This bit is loaded with SINGLE_BANK in FLASH_OPTSR_CUR after a reset.
0: User flash is split between bank 1 and bank 2
1: User flash is located in one bank

Bit 29 **EDATA_EN:** FLASH data area enable
0: No FLASH data area
1: FLASH data area enabled

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **BOOT0**: Boot 0 option bit
0: BOOT0 = 0
1: BOOT0 = 1

Bit 22 **BOOT_SEL**: Boot 0 source configuration
0: BOOT0 signal is defined by the BOOT0 option bit.
1: BOOT0 signal is defined by BOOT0 pin value (legacy mode).

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option configuration bit
When this bit is set, the IWDG is frozen in system Standby mode.
0: IWDG frozen in Standby mode
1: IWDG keeps running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option configuration bit
When this bit is set, the IWDG is frozen in system Stop mode.
0: IWDG frozen in system Stop mode
1: IWDG keeps running in system Stop mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:8 **RDP_LEVEL[7:0]**: RDP level code (based on Hamming 8,4)
See *Section 6.5.8*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option configuration bit
0: A reset is generated when entering Standby mode on core domain.
1: No reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option configuration bit
0: A reset is generated when entering Stop mode on core domain.
1: No reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option configuration bit
0: WWDG controlled by hardware
1: WWDG controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option configuration bit
0: IWDG controlled by hardware
1: IWDG controlled by software

Bits 2:0 Reserved, must be kept at reset value.

6.10.13 FLASH option status register 2 (FLASH_OPTSR2_CUR)

Address offset: 0x070

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0x0000 0010 (value in case of double ECC issue during OBL).

ST production value: refer to *Table 28: Option-byte organization*

This read-only register reflects the current values of corresponding option bits.

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2_ECC	Res.	Res.	SRAM2_RST	SRAM1_RST
											r			r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SRAM2_RST**: SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs

Bit 0 **SRAM1_RST**: SRAM1 erase upon system reset
0: SRAM1 erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

6.10.14 FLASH option status register 2 (FLASH_OPTSR2_PRG)

Address offset: 0x074

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2_ECC	Res.	Res.	SRAM2_RST	SRAM1_RST
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
- 0: SRAM2 ECC check enabled
- 1: SRAM2 ECC check disabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SRAM2_RST**: SRAM2 erase when system reset
- 0: SRAM2 erased when a system reset occurs
- 1: SRAM2 not erased when a system reset occurs.

Note: SRAM erase is triggered by option-byte change operation, when enabling this feature.

Bit 0 **SRAM1_RST**: SRAM1 erase upon system reset
- 0: SRAM1 erased when a system reset occurs
- 1: SRAM1 not erased when a system reset occurs

Note: SRAM erase is triggered by option-byte change operation, when enabling this feature.

6.10.15 FLASH unique boot entry register (FLASH_BOOTR_CUR)

Address offset: 0x080

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0xFFFF FFB4 (value in case of double ECC issue during OBL).

ST production value: refer to *Table 28: Option-byte organization*

This register reflects the current values of corresponding option bits.

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOOTADD[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOTADD[7:0]								BOOT_LOCK[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **BOOTADD[23:0]**: Unique boot-entry address
These bits reflect the user memory boot address.

Caution: The BOOTADD must be a valid address within user main flash area. Otherwise, the boot stalls, and the flash memory becomes unreadable.

Note: The SRAM or EDATA areas cannot be selected as BOOTADD.

Bits 7:0 **BOOT_LOCK[7:0]**: BOOT0, BOOT_SEL, SWAP_BANK, and BOOTADD option settings lock
0xC3: BOOT0, BOOT_SEL, SWAP_BANK, and BOOTADD can still be modified following their individual rules.
0xB4: BOOT0, BOOT_SEL, SWAP_BANK, and BOOTADD are frozen.

6.10.16 FLASH unique boot entry address (FLASH_BOOTR_PRG)

Address offset: 0x084

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

This register is used to program values in corresponding option bits.

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOOTADD[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOTADD[7:0]								BOOT_LOCK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 **BOOTADD[23:0]**: Unique boot entry address
These bits reflect the user memory boot address.

Bits 7:0 **BOOT_LOCK[7:0]**: BOOT0, BOOT_SEL, SWAP_BANK, and BOOTADD option settings lock
0xC3: BOOT0, BOOT_SEL, SWAP_BANK, and BOOTADD can still be modified following their individual rules.
0xB4: BOOT0, BOOT_SEL, SWAP_BANK, and BOOTADD are frozen.

6.10.17 FLASH OTP block lock (FLASH_OTPBLR_CUR)

Address offset: 0x090

Reset value: 0x00XX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0xFFFF FFFF (value in case of double ECC issue during OBL).

ST production value: refer to *Table 28: Option-byte organization*

This register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKBL[23:16]							
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCKBL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **LOCKBL[23:0]**: OTP block lock
Block *n* corresponds to OTP 16-bit word 96 x *n* to 96 x *n* + 95.
LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP block *n* are locked and attempt to program them results in WRPERR.
LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP block *n* are not locked.
When one block is locked, it is not possible to remove the write protection.
Also if not locked, it is not possible to erase OTP words.

6.10.18 FLASH OTP block lock (FLASH_OTPBLR_PRG)

Address offset: 0x094

Reset value: 0x00XX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKBL[23:16]							
								rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCKBL[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **LOCKBL[23:0]**: OTP block lock

Block *n* corresponds to OTP 16-bit word 96 x *n* to 96 x *n* + 95.
LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP block *n* are locked and attempt to program them results in WRPERR.
LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP block *n* are not locked.
When one block is locked, it is not possible to remove the write protection.
LOCKBL bits can be set if the corresponding bit in FLASH_OTPBLR_CUR is cleared.

6.10.19 FLASH bootloader interface selection (FLASH_BL_COM_CFG_CUR)

Address offset: 0x098

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0xFFFF FFFF (value in case of double ECC issue during OBL).

ST production value: refer to *Table 28: Option-byte organization*

This register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BL_COM_CFG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BL_COM_CFG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BL_COM_CFG[31:0]**: Bootloader interface selection/configuration
This register is used to configure the set of interfaces on which the ST bootloader is active.
BL_COM_CFG[n] = 1 indicates that the corresponding interface is used by the bootloader.
BL_COM_CFG[n] = 0 indicates that the corresponding interface is not used by the bootloader. The GPIO remains in default state.

Note: For assignment of interfaces to register bits, see the bootloader documentation.

6.10.20 FLASH bootloader interface selection (FLASH_BL_COM_CFG_PRG)

Address offset: 0x09C

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

This register is used to program values in the corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BL_COM_CFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BL_COM_CFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **BL_COM_CFG[31:0]**: Bootloader interface selection/configuration
This register is used to configure the set of interfaces on which the ST bootloader is active.
BL_COM_CFG[n] = 1 indicates that the corresponding interface is used by the bootloader.
BL_COM_CFG[n] = 0 indicates that the corresponding interface is not used by the bootloader. The GPIO remains in default state.

Note: For assignment of interfaces to register bits see the bootloader documentation.

6.10.21 FLASH OEM key register 1 (FLASH_OEMKEYR1_PRG)

Address offset: 0xA4

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. This register is read as zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEMKEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEMKEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEMKEY[31:0]:** Least significants bytes of OEMKEY

6.10.22 FLASH OEM key register 2 (FLASH_OEMKEYR2_PRG)

Address offset: 0xAC

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. This register is read as zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEMKEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEMKEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEMKEY[63:32]:** Mid-least significants bytes of OEMKEY

6.10.23 FLASH OEM key register 3 (FLASH_OEMKEYR3_PRG)

Address offset: 0xB4

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. This register is read as zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEMKEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEMKEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEMKEY[95:64]:** Mid-most significants bytes of OEMKEY

6.10.24 FLASH OEM key register 4 (FLASH_OEMKEYR4_PRG)

Address offset: 0xBC

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. This register is read as zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEMKEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEMKEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEMKEY[127:96]:** Most significants bytes of OEMKEY

6.10.25 FLASH boundary scan key register (FLASH_BSKEYR_PRG)

Address offset: 0xC4

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. This register is read as zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BSKEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSKEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **BSKEY[31:0]:** Boundary scan key

6.10.26 FLASH write page protection for bank1 (FLASH_WRP1R_CUR)

Address offset: 0x0E8

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0x0000 0000 (value in case of double ECC issue during OBL).

ST production value: refer to *Table 28: Option-byte organization*

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRPSG1[31:0]**: Bank1 page protection option status byte
Each bit reflects the write protection status of the corresponding group of pages in Bank1.
0: Write protected
1: Not write protected

Note: Refer to Table 29: Write-protection management for group definition.

6.10.27 FLASH write-page protection for bank1 (FLASH_WRP1R_PRG)

Address offset: 0x0EC

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **WRPSG1[31:0]**: Bank1 page protection option status byte
Setting these bits to 0 write-protects the corresponding pages in bank 1
0: Write protected
1: Not write protected
Note: Refer to Table 29: Write-protection management for group definition.

6.10.28 FLASH HDP bank1 register (FLASH_HDP1R_CUR)

Address offset: 0x0F8

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

This register is initially loaded with the values of corresponding option bits.

Default value: 0x000F 0000 (value in case of double ECC issue during OBL) for STM32C53x/542.

Default value: 0x001F 0000 (value in case of double ECC issue during OBL) for STM32C55x/562.

Default value: 0x003F 0000 (value in case of double ECC issue during OBL) for STM32C59x/5A3.

ST production value: refer to *Table 28: Option-byte organization*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[5:0]					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[5:0]					
										r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **HDP1_END[5:0]:** HDPL barrier end set in number of 8-Kbyte pages

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **HDP1_STRT[5:0]:** HDPL barrier start set in number of 8-Kbyte pages

6.10.29 FLASH HDP bank1 register (FLASH_HDP1R_PRG)

Address offset: 0x0FC

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR. Register is accessible only in HDPL1. In HDPL=2 or 3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[5:0]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **HDP1_END[5:0]:** Bank 1 HDPL barrier end set in number of 8-Kbyte pages

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **HDP1_STRT[5:0]:** Bank 1 HDPL barrier start set in number of 8-Kbyte pages

6.10.30 FLASH ECC correction register (FLASH_ECCCORR)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ECCC	Res.	Res.	Res.	Res.	ECCCIE	OTP_ECC	SYSF_ECC	BK_EC C	EDATA _ECC	Res.	Res.	Res.	Res.	Res.
	rw					rw	r	r	r	r					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ECCC:** ECC correction
Set by hardware when single ECC error has been detected and corrected.
Cleared by writing 1.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **ECCCIE**: ECC single-correction error interrupt enable bit
When this bit is set to 1, an interrupt is generated when an ECC single-correction error occurs during a read operation.
0: No interrupt generated when an ECC single-correction error occurs.
1: Interrupt generated when an ECC single-correction error occurs.

Bit 24 **OTP_ECC**: OTP ECC error bit
This bit is set to 1 when one ECC single-correction occurred during the last successful read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bitfield.

Bit 23 **SYSF_ECC**: ECC flag for corrected ECC error in system FLASH
This bit indicates if system flash memory is concerned by an ECC error.

Bit 22 **BK_ECC**: ECC bank flag for corrected ECC error
This bit indicates which bank is concerned by an ECC error .

Bit 21 **EDATA_ECC**: ECC fail for corrected ECC error in FLASH data area
This bit indicates if FLASH data area is concerned by an ECC error.

Bits 20:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address
When an ECC error occurs (for single correction) during a read operation, this bitfield contains the address that generated the error. It is reset when the flag error is reset.
The FLASH programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.
The address in ADDR_ECC is relative to the FLASH area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

6.10.31 FLASH ECC detection register (FLASH_ECCDETR)

Address offset: 0x104

Reset value: 0x0000 0000

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	Res.	Res.	Res.	Res.	Res.	Res.	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	Res.	Res.	Res.	Res.	Res.
rc_w1							r	r	r	r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ECCD:** ECC detection set by hardware when two ECC errors have been detected
When this bit is set, a NMI is generated. This bit is cleared by writing 1. Needs to be cleared in order to detect subsequent double ECC errors.

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **OTP_ECC**: OTP ECC error bit
This bit is set to 1 when a double ECC detection occurred during the last read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bitfield.

Bit 23 **SYSF_ECC**: ECC fail for double ECC error in system flash memory
This bit indicates if system flash memory is concerned by an ECC error.

Bit 22 **BK_ECC**: ECC fail bank for double ECC Error
This bit indicates which bank is concerned by an ECC error.

Bit 21 **EDATA_ECC**: ECC fail for double ECC error in FLASH data area
This bit indicates if FLASH EDATA sector is concerned by an ECC error.

Bits 20:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address
When an ECC error occurs (double detection) during a read operation, this bitfield contains the address that generated the error. It is reset when the flag error is reset.
The FLASH programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an double ECC error is saved. The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

6.10.32 FLASH ECC data (FLASH_ECCDR)

Address offset: 0x108

Reset value: 0x0000 0000

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA_ADDR_ECC[2:0]		
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **DATA_ADDR_ECC[2:0]**: DATA ECC error address
When a double detection ECC error occurs on special areas with 6-bit ECC on 16-bit data (data area, read-only/OTP area), the position of the failing word on the FLASH line can be read to this register.
000: Double error on first 16-bit data or first 32-bit data accessed on the FLASH line
001: Double error on second 16-bit data accessed on the FLASH line
010: Double error on third 16-bit data or second 32-bit data accessed on the FLASH line
011: Double error on fourth 16-bit data accessed on the FLASH line
100: Double error on fifth 16-bit data or third 32-bit data accessed on the FLASH line
101: Double error on sixth 16-bit data
Note: If the data access was 16 bits, the corrupted word can be fully located. If access was 32 bits, use DATA_ECC value to discriminate which one of the two 16-bit word is faulty.

Bits 15:0 **DATA_ECC[15:0]**: ECC error data
When a double detection ECC error occurs on special areas with 6-bit ECC on 16-bit data (data area, read-only/OTP area), the failing data is read to this register.
By checking if it is possible to determine whether the failure was on a real data, or due to access to uninitialized memory.

6.10.33 FLASH write page protection for bank2 (FLASH_WRP2R_CUR)

Address offset: 0x1E8

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0x0000 0000 (value in case of double ECC issue during OBL).

ST production value: refer to *Table 28: Option-byte organization*

This read-only register reflects the current values of corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG2[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRPSG2[31:0]**: Bank2 page protection option status byte

Each bit reflects the write protection status of the corresponding group of pages in Bank2.

0: Write protected

1: Not write protected

Note: Refer to Table 29: Write-protection management group definition.

6.10.34 FLASH write page protection for bank2 (FLASH_WRP2R_PRG)

Address offset: 0x1EC

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **WRPSG2[31:0]**: Bank2 page protection option status byte

Setting WRPSG2 bits to 0 write protects the corresponding pages in bank 2

0: write protected;

1: not write protected

Note: Refer to Table 29: Write-protection management group definition.

6.10.35 FLASH HDP bank2 register (FLASH_HDP2R_CUR)

Address offset: 0x1F8

Reset value: 0xXXXX XXXX

This register is initially loaded with the values of corresponding option bits.

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Default value: 0x000F 0000 (value in case of double ECC issue during OBL) for STM32C53x/542.

Default value: 0x001F 0000 (value in case of double ECC issue during OBL) for STM32C55x/562.

Default value: 0x003F 0000 (value in case of double ECC issue during OBL) for STM32C59x/5A3.

ST production value: refer to *Table 28: Option-byte organization*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[5:0]					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[5:0]					
										r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **HDP2_END[5:0]:** Bank 2 HDPL barrier end set in number of 8-Kbyte pages

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **HDP2_STRT[5:0]:** Bank 2 HDPL barrier start set in number of 8-Kbyte pages

6.10.36 FLASH HDP bank2 register (FLASH_HDP2R_PRG)

Address offset: 0x1FC

Reset value: 0xXXXX XXXX

This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR.

Register is accessible only in HDPL0 and HDPL1. In HDPL=2 or 3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[5:0]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **HDP2_END[5:0]:** Bank 2 HDPL barrier end set in number of 8-Kbyte pages

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **HDP2_STRT[5:0]:** Bank 2 HDPL barrier start set in number of 8-Kbyte pages

6.10.37 FLASH register map

Table 43. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTEN	Res.	Res.	WRHIGHFREQ[1:0]		LATENCY[3:0]			
	Reset value																x								0			1	0	0	1	1	1
0x004	FLASH_KEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	Reserved																																
0x00C	FLASH_OPTKEYR	OPTKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010 - 0x014	Reserved																																
0x018	FLASH_OPSR	CODE_OP[2:0]			Res.	Res.	Res.	Res.	OTP_OP	Res.	BK_OP	DATA_OP	Res.	Res.	Res.	Res.	Res.	ADDR_OP[15:0]															
	Reset value	x	x	x					x		x	x						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x01C	FLASH_OPTCR	SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	OPTLOCK
	Reset value	x																														0	1
0x020	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERR	Res.	Res.	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	BSLOCK	OEMLOCK	Res.	Res.	Res.	Res.	DBNE	Res.	WBNE	BSY
	Reset value									0			0	0	0	0	0							0	0					x		x	x
0x024	Reserved																																

Table 43. FLASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	FLASH_CR	BKSEL	Res.	EDATASEL	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERRIE	Res.	Res.	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	MER	Res.	Res.	PNB[6:0]							STRT	FW	BER	PER	PG	LOCK
	Reset value	0		0						0			0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	1
0x02C	Reserved																																
0x030	FLASH_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_OPTCHANGEERR	Res.	Res.	CLR_INCERR	CLR_STRBERR	CLR_PGSERR	CLR_WRPERR	CLR_EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0			0	0	0	0	0																
0x034 - 0x038	Reserved																																
0x03C	FLASH_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.
	Reset value																															0	
0x040 - 0x044	Reserved																																
0x048	FLASH_HDPEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_EXT[5:0]					
	Reset value											0	0	0	0	0	0											0	0	0	0	0	0
0x04C	Reserved																																
0x050	FLASH_OPTSR_CUR	SWAP_BANK	SINGLE_BANK	EDATA_EN	Res.	Res.	Res.	Res.	Res.	BOOT0	BOOT_SEL	IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.	RDP_LEVEL[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
	Reset value	x	x	x						x	x	x	x					x	x	x	x	x	x	x	x	x	x		x	x			
0x054	FLASH_OPTSR_PRG	SWAP_BANK	SINGLE_BANK	EDATA_EN	Res.	Res.	Res.	Res.	Res.	BOOT0	BOOT_SEL	IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.	RDP_LEVEL[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
	Reset value	x	x	x						x	x	x	x					x	x	x	x	x	x	x	x	x	x		x	x			
0x058 - 0x06C	Reserved																																
0x070	FLASH_OPTSR2_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2_ECC	Res.	Res.	SRAM2_RST	SRAM1_RST
	Reset value																												x			x	x

Table 43. FLASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x074	FLASH_OPTSR2_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2_ECC	Res.	Res.	SRAM2_RST	SRAM1_RST
	Reset value																												x			x	x
0x078 - 0x07C	Reserved																																
0x080	FLASH_BOOTR_CUR	BOOTADD[23:0]																								BOOT_LOCK[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x084	FLASH_BOOTR_PRG	BOOTADD[23:0]																								BOOT_LOCK[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x088 - 0x08C	Reserved																																
0x090	FLASH_OTPBLR_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKBL[23:0]																							
	Reset value									x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x094	FLASH_OTPBLR_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKBL[23:0]																							
	Reset value									x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x098	FLASH_BL_COM_CFG_CUR	BL_COM_CFG[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x09C	FLASH_BL_COM_CFG_PRG	BL_COM_CFG[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0A0	Reserved																																
0x0A4	FLASH_OEMKEYR1_PRG	OEMKEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0A8	Reserved																																
0x0AC	FLASH_OEMKEYR2_PRG	OEMKEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0B0	Reserved																																
0x0B4	FLASH_OEMKEYR3_PRG	OEMKEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0B8	Reserved																																
0x0BC	FLASH_OEMKEYR4_PRG	OEMKEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0C0	Reserved																																

Table 43. FLASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C4	FLASH_BSKEYR_PRG	BSKEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0C8 - 0x0E4	Reserved																																
0x0E8	FLASH_WRP1R_CUR	WRPSG1[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0EC	FLASH_WRP1R_PRG	WRPSG1[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x0F0 - 0x0F4	Reserved																																
0x0F8	FLASH_HDP1R_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[5:0]					
	Reset value											x	x	x	x	x	x											x	x	x	x	x	x
0x0FC	FLASH_HDP1R_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[5:0]					
	Reset value											x	x	x	x	x	x											x	x	x	x	x	x
0x100	FLASH_ECCCORR	Res.	ECCC	Res.	Res.	Res.	Res.	ECCCIE	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	Res.	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	Reset value		0					0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	FLASH_ECCDETR	ECCD	Res.	Res.	Res.	Res.	Res.	Res.	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	Res.	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	Reset value	0							0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	FLASH_ECCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA_ADDR_ECC[2:0]			DATA_ECC[15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C - 0x1E4	Reserved																																
0x1E8	FLASH_WRP2R_CUR	WRPSG2[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1EC	FLASH_WRP2R_PRG	WRPSG2[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1F0 - 0x1F4	Reserved																																

Table 43. FLASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1F8	FLASH_HDP2R_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[5:0]					
	Reset value											x	x	x	x	x	x											x	x	x	x	x	x
0x1FC	FLASH_HDP2R_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[5:0]					
	Reset value											x	x	x	x	x	x											x	x	x	x	x	x

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

7 Instruction cache (ICACHE)

7.1 ICACHE introduction

The instruction cache (ICACHE) is implemented on the C-AHB code bus of the Cortex®-M33 processor to improve performance when fetching instructions and data from internal and external memories (the latter only on STM32C59x/5A3 devices).

Some specific features, like dual master ports (second master port only on STM32C59x/5A3 devices), hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

7.2 ICACHE main features

The main features of ICACHE are listed below:

- Bus interface
  - One 32-bit AHB slave port, the execution port (input from Cortex®-M33 C-AHB code interface)
  - Two AHB master ports: master1 and master2 ports, with outputs to Fast and Slow buses of main AHB bus matrix, respectively (master2 port only on STM32C59x/5A3STM32C5P1xx/5Q1xx devices)
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master (only on STM32C59x/5A3 devices) access: feature used to decouple the traffic according to targeted memory. For example, the ICACHE assigns fast traffic (addressing flash memories and SRAMs) to the AHB master1 port, and slow traffic (addressing external memories) to the AHB master2 port, thus preventing processor stalls on line refills from external memories. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from external memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, **w**, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Memory address remap
  - Possibility to remap input address falling into up to four memory regions (used only on STM32C59x/5A3 devices to remap aliased code in external memories to the code region, for execution from the C-AHB code interface)
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, algorithm with best complexity/performance balance

  - Critical-word-first refill policy, minimizing processor stalls
  - Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error, and optionally, to raise an interrupt
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

7.3 ICACHE implementation

Table 44. ICACHE features for STM32C5 devices

Feature	ICACHE1
Number of ways	2
Cache size	8 Kbytes
Cache line width	16 bytes
Range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	128 bits
Data size of AHB slow master2 interface(1)	32 bits

1. Only on STM32C59x/5A3 devices.

7.4 ICACHE functional description

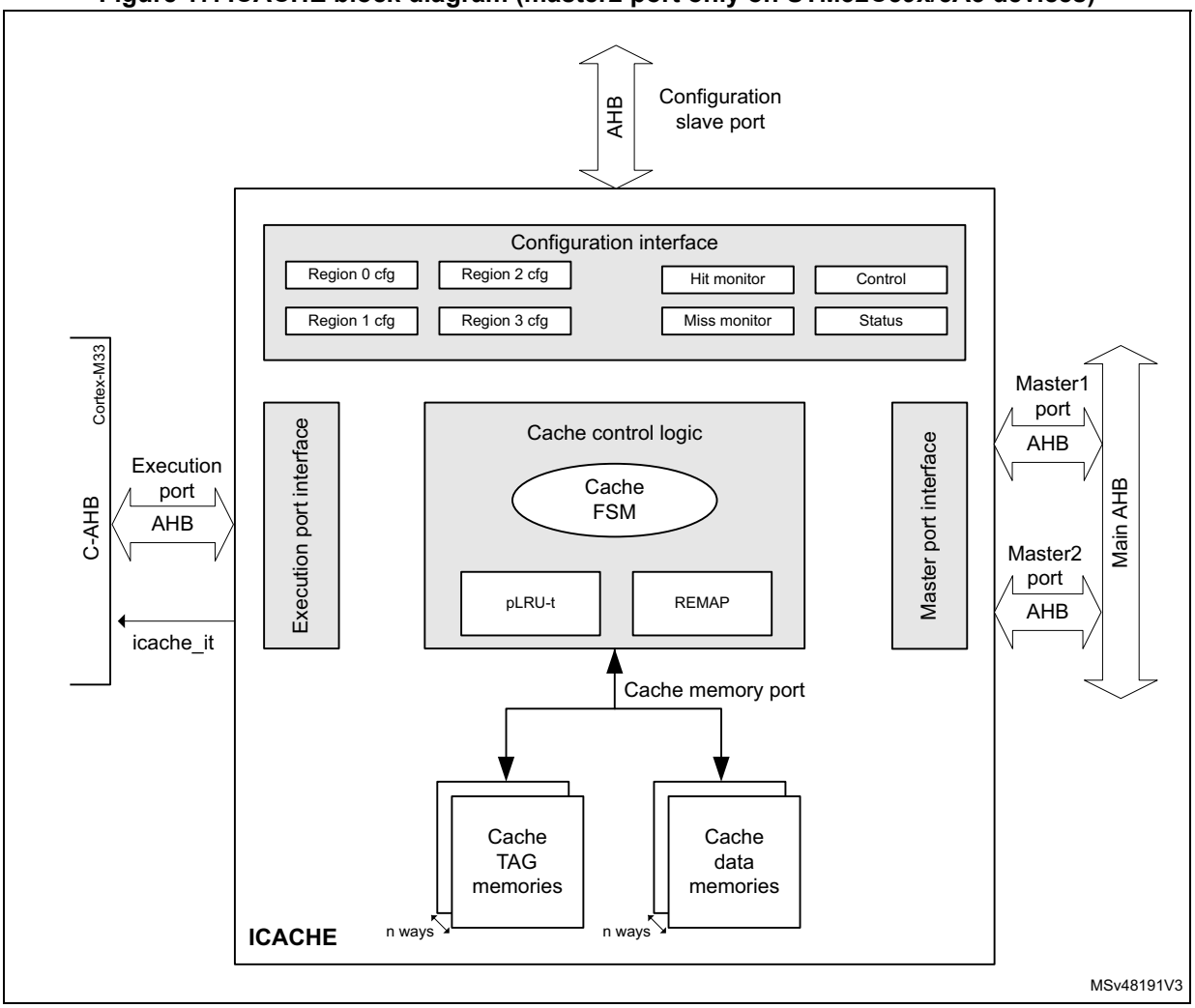
The purpose of this block is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

7.4.1 ICACHE block diagram

Figure 17. ICACHE block diagram (master2 port only on STM32C59x/5A3 devices)



7.4.2 ICACHE reset and clocks

The ICACHE is clocked on the Cortex®-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are forwarded to the master port(s).

7.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:

- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 45* for B and W definitions):

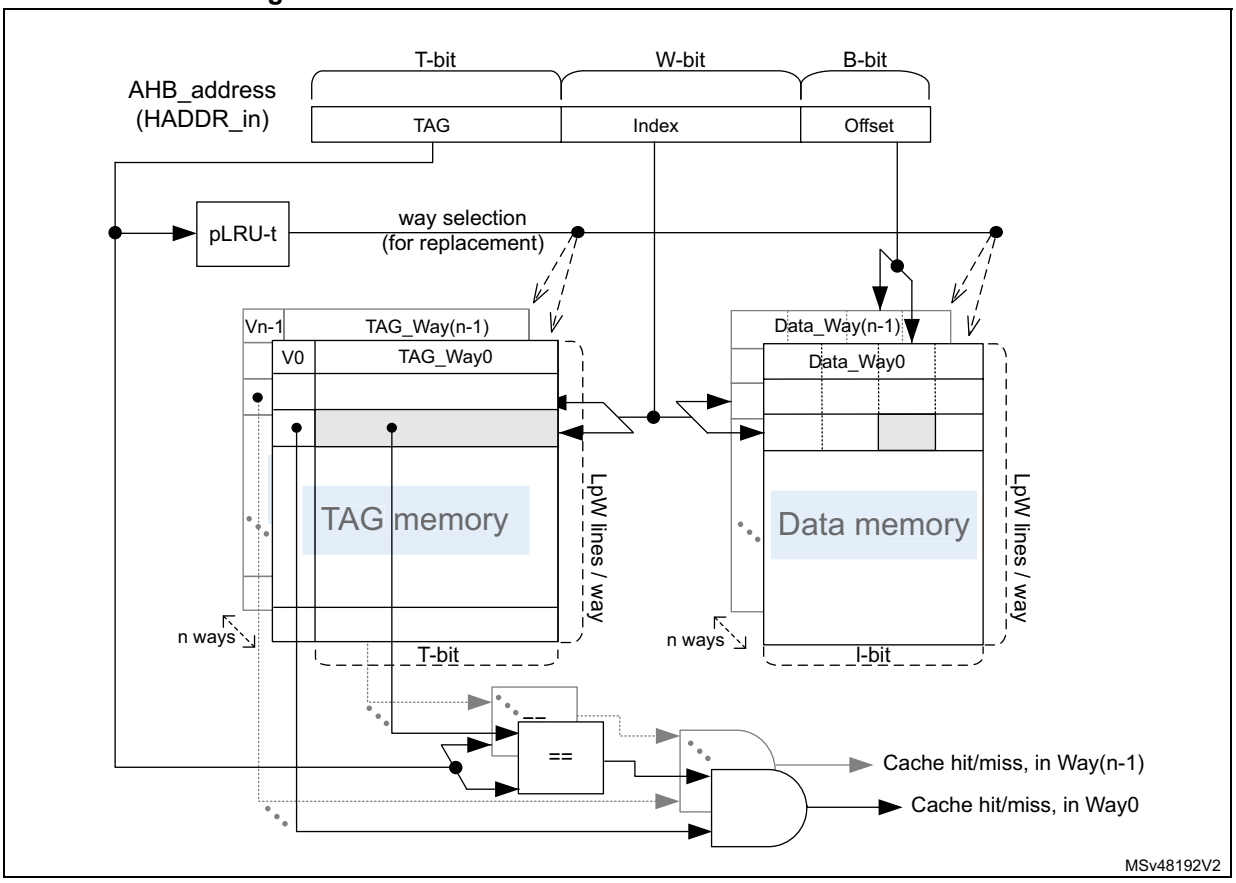
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 18* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 45. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 18. ICACHE TAG and data memories functional view



7.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 46. TAG memory dimensioning parameters for direct-mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, remapping, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 7.4.8*): only one way (the unique one) is possible for any data refill.

7.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master port(s) in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into the memory regions defined and enabled in ICACHE_CRRx (see *Figure 19*).

The ICACHE is, by default, disabled at boot.

7.4.6 Cacheable and noncacheable traffic

The ICACHE is developed for the Cortex®-M33 core. It is placed on the C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

To make other memory regions cacheable, the ICACHE supports a memory region remapping feature, used (only for STM32C59x/5A3 devices) to define up to four external memory regions, whose addresses have an alias in the code region. Addressing these memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any external memory space physically mapped at an address in the range [0x6000 0000:0xAFFF FFFF] can be aliased with an address in the range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the code region, the ICACHE implements the address remapping functionality first. If aliased, it is the remapped address, which is then cached, and, if needed, provided to the master port to address the main AHB bus matrix. The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for noncacheable traffic, and when the cache is disabled.

Additional details on address remapping are provided in *Section 7.4.7*.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 47*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 47. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. The AHB transaction is propagated unchanged to the master output port, except the transaction address, which may be modified due to the address remapping feature (see *Section 7.4.7*).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in *Section 7.4.8*.

Cacheable memory regions are defined and programmed by the user in the MPU responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

Table 48 summarizes programmable configurations of various memories.

Table 48. Memory configurations

Memory	Cacheable (MPU programming)	Remapped in the ICACHE (ICACHE_CRRx programming)
Flash	Yes or no	Not required
SRAM	Not recommended	Not required
External memories	Yes	Required
	No	Required if the user wants code in external memories fetched on C-AHB bus (else on S-AHB bus)

7.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four external memory regions (only for STM32C59x/5A3 devices).

The address remapping is applied on the code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is noncacheable.

Remap regions can have different size: each can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum of 2 Mbytes, and a maximum of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) physical address, as follows:

- HADDR_in belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = $\log_2$ (region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region).
- If the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs

Figure 19 describes the matching and the output address generation.

Figure 19. ICACHE remapping address mechanism

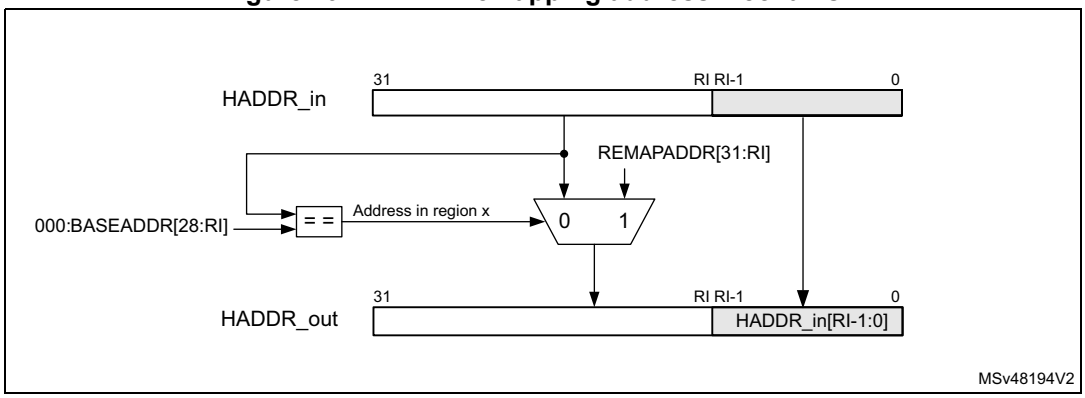


Table 49 summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending upon RSIZE.

Table 49. ICACHE remap region size, base address, and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see *Table 49*), the unneeded extra LSBs are ignored.

Typical remapping example: a 128-Mbyte external RAM region physically located in the external address range [0x9000 0000:0x97FF FFFF], remapped in the code section range [0x1800 0000:0x1FFF FFFF]:

- REMAPADDR[31:21] = 0x480
- BASEADDR[28:21] = 0xC0
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], while HADDR_in and BASEADDR[26:21] are ignored.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through interfaces that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through the interfaces that do not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming the MPU (memory protection unit) attributes for both the external regions and their aliased code subregions.

7.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) if no address remap occurs. If an address remap occurs, the kind of burst depends upon the HBURST programmed in corresponding ICACHE_CRRx.

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill (master1 port, or master2 port if available, as for STM32C59x/5A3 devices).

- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

7.4.9 Dual-master cache

The ICACHE implements a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports (master2 port on STM32C59x/5A3 devices only). This is used to split the traffic going to different destination memories.

The nonremapped traffic goes systematically to master1 port. The remapped traffic can be routed on the master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:

- internal flash memory and internal SRAM on master1 port (Fast bus)
- external flash memory/RAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual-master feature allows the processor to have an alternative path in case of fetching from different memories.

7.4.10 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

7.4.11 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 7.4.8*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

7.4.12 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for a Cortex®-M33 as it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

7.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 18*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves fetch performance, even if for most code execution, it is less efficient than with an n-way set associative cache mode.

7.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 50. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as address decoding issues) from the master1 or master2 port back to the execution port.

7.7 ICACHE registers

7.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISS MRST	HITM RST	MISS MEN	HITM EN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAY SEL	CACHE INV	EN
													rw	w	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

7.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

7.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRIE**: interrupt enable on cache error

Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end

Set by software to enable an interrupt generation at the end of a cache invalidate operation.

0: interrupt disabled on busy end

1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

7.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
													w	w	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CERRF**: clear cache error flag

Set by software.

0: no effect

1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag

Set by software.

0: no effect

1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

7.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

7.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

7.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in the code region for other regions, making them cacheable.

BASEDADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]										
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	BASEADDR[28:21]							
rw				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 7.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 7.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

7.7.8 ICACHE register map

Table 51. ICACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ICACHE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0														1	0	0
0x004	ICACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
	Reset value																														0	0	1

Table 51. ICACHE register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	ICACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																														0	0	
0x00C	ICACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																														0	0	
0x010	ICACHE_HMONR	HITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ICACHE_MMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	ICACHE_CRR0	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x024	ICACHE_CRR1	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x028	ICACHE_CRR2	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x02C	ICACHE_CRR3	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

8 Power control (PWR)

8.1 PWR introduction

The power controller manages the device power supplies and power modes transitions.

8.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - RTC domain (V_{RTC})
  - Analog domain (V_{DDA})
  - V_{DDUSB} for USB transceiver
- System supply voltage regulation
  - Linear voltage regulator (LDO)
- Power supply supervision
  - POR/PDR monitor
  - PVD monitor
- Power management
  - Low-power modes
- Privileged protection

8.3 PWR pins and internal signals

Table 52. PWR input/output pins

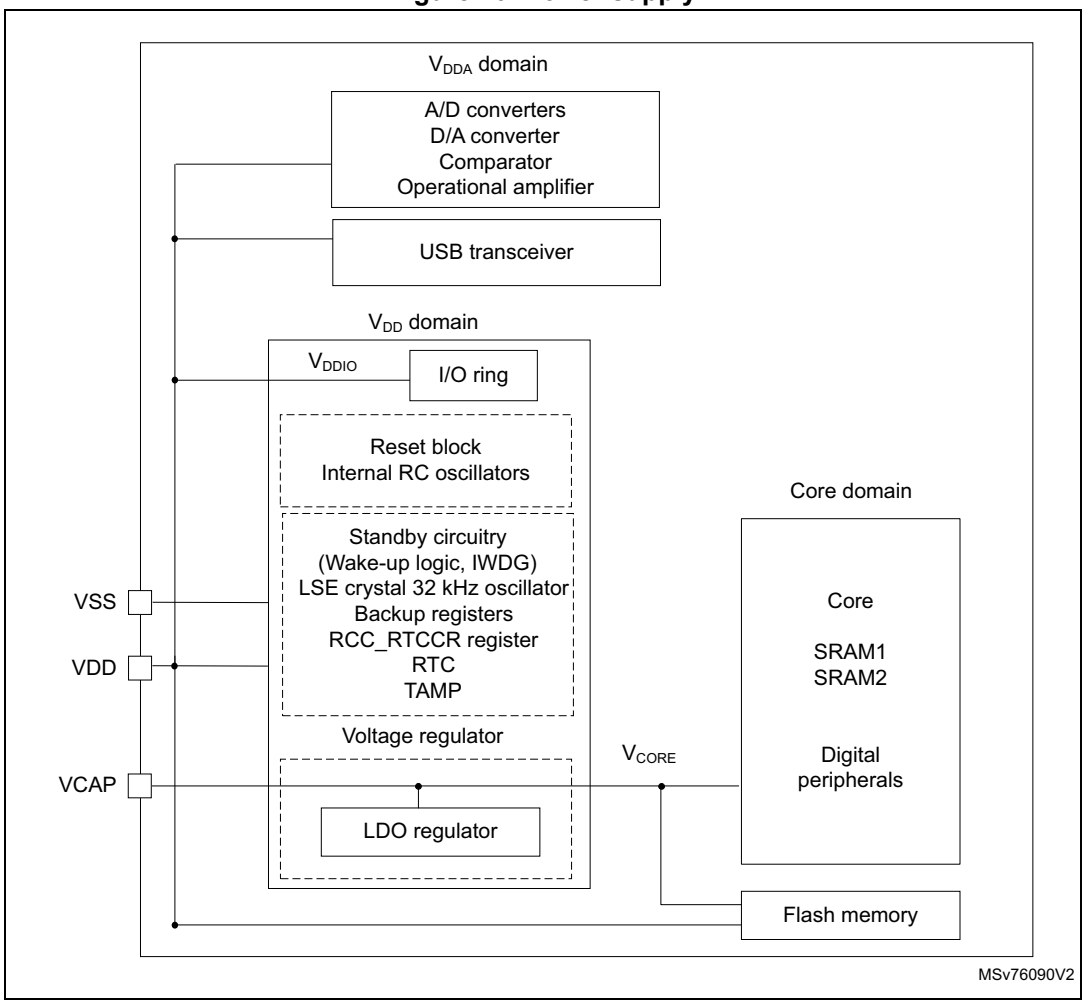
Pin name	Signal type	Description
VDD	Supply	Main supply
GND	Supply	Main ground
VDDA	Supply	Analog peripherals supply
VSSA	Supply	Analog peripherals ground
VCAP	Supply	Logic supply (V_{CORE})
VREF+	Supply	ADC/DAC high reference voltage
VREF-	Supply	ADC/DAC low reference voltage
WKUPx (x = 1 to 7)	Input	Wake-up pins
CSLEEP	Output	MCU in Sleep mode
CSTOP	Output	CPU in Stop modes

Table 53. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx (x = 1 to 7)	Input	Wake-up event source

8.4 PWR power supplies and supply domains

Figure 20. Power supply



8.4.1 External power supplies

The devices require a 2.7 to 3.6 V operating voltage supply V_{DD}:

- V_{DD} = 2.7 to 3.6 V

  V_{DD} is the external power supply for the I/Os, the internal regulator, and the system analog (such as reset, power management, and internal clocks). It is provided externally through the VDD pins.

- V_{DDA} = 2.7 V (ADCs, DAC, COMP, and OPAMP) to 3.6 V

  V_{DDA} is the external analog power supply for A/D converters, D/A converters, operational amplifier, and the analog comparator. The V_{DDA} voltage domain is

physically connected to the V_{DD} voltage.

- V_{CAP} = 0.9 to 1.25 V: digital core domain supply
  This power supply is independent from all the other power supplies:
- V_{RTC} = 2.7 to 3.6 V
  V_{RTC} is the power supply for RTC, external clock 32 kHz oscillator, and backup registers. The V_{RTC} voltage level is physically connected to the V_{DD} voltage.
- V_{REF-}, V_{REF+}
  V_{REF+} is the input reference voltage for ADCs and DAC.
  V_{REF+} can be grounded when ADC and DAC are not active.
  VREF- and VREF+ pins are not available on all packages. When not available, they are bonded to VSS and VDD, respectively.
  V_{REF-} must always be equal to V_{SSA}.

8.4.2 Internal regulator

The devices embed an LDO regulator to provide the V_{CORE} supply for digital peripherals, SRAMs, and embedded flash memory. The LDO generates this voltage on VCAP, with a total external capacitance of 2.2 µF (typical). The LDO regulator is always enabled in Run mode and Stop mode, generating Vcore at 1.2 V in Run and Stop 0 mode and at 0.95 V in Stop 1 mode.

8.4.3 Independent analog peripherals supply

To improve ADC and DAC conversion and OPAMP accuracy, the analog peripherals have an independent power supply that can be filtered and shielded from noise on the PCB:

- The voltage supply input of the analog peripherals is available on VDDA pin.
- An isolated supply ground connection is provided on the VSSA pin.

ADC and DAC reference voltage

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to VREF+, a separate reference voltage lower than V_{DDA}. V_{REF+} is the highest voltage, represented by the full-scale value, for an analog input (ADC) or output (DAC) signal.

Note: The VREF+ and VREF- pins are not available on all packages. When not available, they are internally connected, respectively to VDD and VSS).

8.4.4 RTC domain

To retain the content of the backup registers and to supply the RTC function when the device enters Standby mode.

The RTC domain powers the RTC unit and the LSE oscillator, allowing the RTC to operate even when the core supply domain is switched off.

RTC domain access

After a system reset, the RTC domain (RCC RTC domain control register RCC_RTCCR, RTC registers, TAMP registers, and backup registers) is protected against possible unwanted write accesses. To enable access to the RTC domain, set the DRTCP bit in PWR_RTCCR.

8.5 PWR system supply voltage regulation

8.5.1 Embedded voltage regulator operating modes

There are three different power modes: Run, Stop mode (Stop 0 and Stop 1), and Standby modes.

Run mode

The voltage regulator provides full power to the V_{CORE} domain (core, memories, and digital peripherals). The regulator output voltage is 1.2 V and is stabilized thanks the VCAP capacitance.

Stop mode

The voltage regulator supplies the V_{CORE} domain to retain the content of registers and internal memories. The regulator under Stop 1 mode is configured to output 0.95 V to V_{CORE} domain.

Standby mode

The regulator is OFF and the V_{CORE} domain is powered down. The content of the registers and memories is lost except for the Standby circuitry in the RTC domain.

8.6 PWR power supply and temperature supervision

Power supply level monitoring is available on the following supplies:

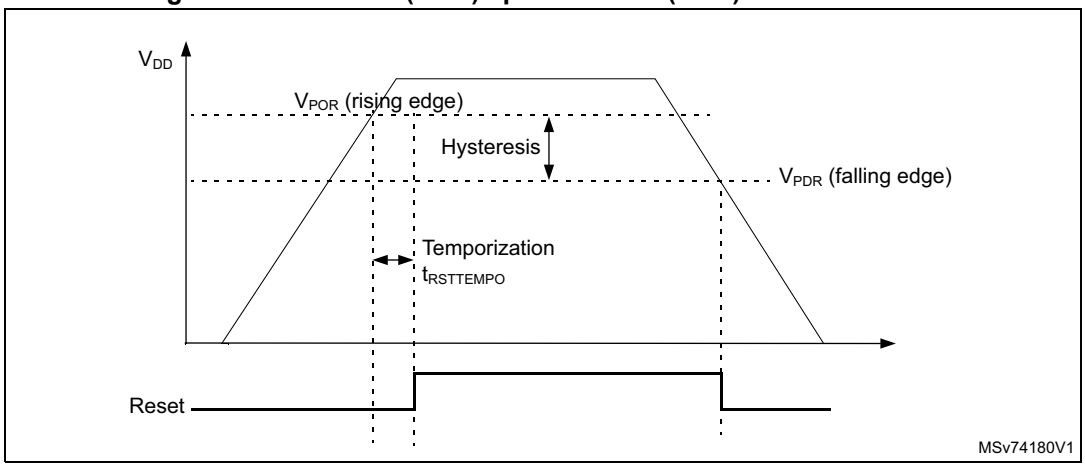
- V_{DD} via POR/PDR (see *Section 8.6.1*) and PVD monitor (see *Section 8.6.2*)
- Temperature monitoring (see *Section 21.4.30*)

8.6.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in reset mode when V_{DD} is below a specified VPOR threshold, without the need for an external reset circuit. Once the supply level is above the VPOR threshold, the system is taken out of reset (see figure below). For more details concerning the reset thresholds refer to the electrical characteristics section of the datasheets.

Figure 21. Power-on (POR) / power-down (PDR) reset waveform



1. For thresholds and hysteresis values refer to the datasheets.

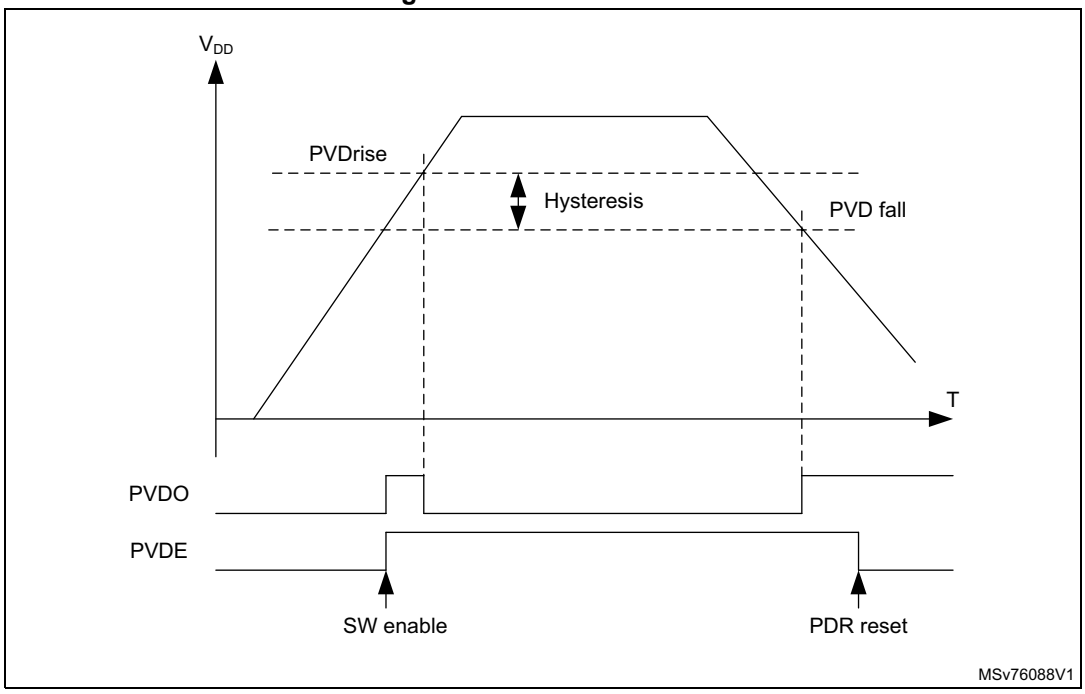
8.6.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a defined threshold.

The PVD is enabled by setting the PVDE bit in PWR_VMCR.

A PVDO flag is available in PWR_VMSR to indicate if V_{DD} voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, provided it has been enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI line must be configured according to PVD output behavior. As an example, if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when V_{DD} voltage drops below the PVD threshold. The service routine can then start an emergency shutdown.

Figure 22. PVD thresholds



1. For thresholds and hysteresis values, refer to the product datasheet.

8.7 Power modes

By default, the microcontroller is in Run mode after a system or a power reset. Three low-power modes are available to reduce consumption when there is no need to keep the CPU running, for example when waiting for an external event. The user can select the mode that gives the best compromise between low-power consumption, short startup time, and wake-up sources.

The device features the following low-power modes:

- Sleep mode

  CPU clock off, peripherals such as NVIC and SysTick can run and wake up the CPU when an interrupt or an event occurs. Refer to *Section 8.7.4*.

- Stop 0 and Stop 1 modes

  Achieve the lowest power consumption, while retaining the content of SRAM and registers. All clocks in the core domain are stopped. The HSE crystal oscillators, HSI (except if HSIKERON is set), PSI (except if PSIKERON is set) are disabled. The LSE or LSI is still running. Under Stop 0, the LDO regulator supplies VCORE at 1.2 V while under Stop 1 VCORE voltage is decreased down to 0.9 V. Consequently wake up sources are reduced and wake up time increased, refer to *Section Table 54.*

  The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

  When exiting from Stop mode, the system clock is HSI clock at 144 MHz or 48 MHz based STOPWUCK value.

  Refer to *Section 8.7.5*.

- Standby mode

  This mode achieves the lowest power consumption. The internal regulator is switched

off so that the core domain is powered off. The HSI, PSI, and the HSE crystal oscillators are also switched off.

The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

The power-on reset and power-down reset (POR/PDR) remains active.

The state of the I/O (except I/Os used by Standby mode) during Standby mode can be retained.

After entering Standby mode, SRAMs and register contents are lost except for registers in the RTC domain and Standby circuitry.

The device exits Standby mode when an external reset (NRST pin), an IWDG reset, WKUP pin event (configurable rising or falling edge), an RTC event occurs (alarm, periodic wake-up, timestamp), or a tamper detection. The tamper detection can be raised either due to external pins or due to an internal failure detection.

The system clock after wake-up is HSI at 48 MHz.

The table below shows the power modes overview.

Table 54. Low-power mode summary

Mode name	Entry	Wake-up source(1)	Wake-up system clock	Effect on clocks
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	– CPU clock OFF – No effect on other clocks or analog clock sources
	WFE	Wake-up event		
Stop 0	LPMS = 00 + SLEEPDEEP bit + WFI or Return from ISR or WFE	– Any EXTI line (configured in the EXTI registers) – Specific peripherals events(2)	– HSI clock at 144 MHz or 48 MHz based STOPWUCK value	– All clocks OFF except LSI and LSE – HSI or PSI can be enabled temporarily when requested by software or kept ON thanks to HSIKERON/PSIKERON bits
Stop 1	LPMS = 01 + SLEEPDEEP bit + WFI or Return from ISR or WFE	– Any GPIO EXTI line and EXTI from PVD, COMP, RTC, LPUART or LPTIM (configured in the EXTI registers)		– All clocks OFF except LSI and LSE
Standby	LPMS = 10 + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, IWDG reset, external reset in NRST pin	HSI clock at 48 MHz	All clocks OFF except LSI and LSE

1. Refer to *Table 55*.
2. Peripherals able to wake up the system from Stop mode.

Table 55. Functionalities depending on the working mode(1)

Peripheral	Run	Sleep	Stop			Standby	
			Available	Wake-up capability		Available	Wake-up capability
				Stop 0	Stop 1		
CPU	Y	-	-	-	-	-	-
Flash memory	O	O	-(2)	-	-	-	-
SRAM1	Y(3)	Y(3)	Y	-	-	-	-
SRAM2	Y(3)	Y(3)	Y	-	-	-	-
Backup registers	Y	Y	Y	Y	Y	Y	-
XSPI1	O	O	-	-	-	-	-
Programmable voltage detector (PVD)	O	O	O	O	O	-	-
LPDMA	O	O	-	-	-	-	-
High-speed internal (HSI)	O	O	-	-	-	-	-
Programmable-speed internal (PSI)	O	O	-	-	-	-	-
High-speed external (HSE)	O	O	-	-	-	-	-
Low-speed internal (LSI)	O	O	O	O	-	O	-
Low-speed external (LSE)	O	O	O	-	-	O	-
Clock security system (CSS)	O	O	-	-	-	-	-
Clock security system on LSE	O	O	O	O	O(4)	O	O
RTC domain voltage	O	O	O	O	O(5)	O	O
RTC/TAMP	O	O	O	O	O	O	O
Number of TAMP tamper pins	3	3	3	-	-	3	-
USB FS	O	O	O	O	-	-	-
USARTx / UARTx	O	O	O	O	-	-	-
Low-power UART (LPUART)	O	O	O	O	O	-	-
I2Cx (x = 1, 2)	O	O	O	O	-	-	-
I3C1	O	O	O	O	-	-	-
SPIx (x = 1 to 3)	O	O	O	O	-	-	-
FDCANx (x = 1, 2)	O	O	-	-	-	-	-
ETH1	O	O	O	O	-	-	-
ADCx (x = 1 to 3)	O	O	-	-	-	-	-
DAC1 (OUT1, OUT2)	O	O	O	-	-	-	-
COMPx (x = 1, 2)	O	O	O	O	O	-	-

Table 55. Functionalities depending on the working mode(1) (continued)

Peripheral	Run	Sleep	Stop			Standby	
			Available	Wake-up capability		Available	Wake-up capability
				Stop 0	Stop 1		
OPAMP1	O	O	-	-	-	-	-
Timers (TIMx)	O	O	-	-	-	-	-
Low-power timer LPTIM1	O	O	O	O	O	-	-
Independent watchdog (IWDG)	O	O	O	O	O	O	O
Window watchdog (WWDG)	O	O	-	-	-	-	-
SysTick timer (SYSTICK)	O	O	O	-	-	-	-
CORDIC coprocessor (CORDIC)	O	O	-	-	-	-	-
Random number generator (RNG)	O	O	-	-	-	-	-
AES/SAES	O	O	-	-	-	-	-
HASH accelerator	O	O	-	-	-	-	-
Public key accelerator (PKA)	O	O	-	-	-	-	-
CCB	O	O	-	-	-	-	-
CRC calculation unit	O	O	-	-	-	-	-
GPIOs	O	O	-	-	-	O(5)	O(6)
EXTI	O	O	O	O	O	-	-

1. Y = yes (enabled). O = optional (disabled by default, can be enabled by software). - = not available. HSI or PSI are available as kernel clock for peripherals only in STOP 0.
2. The memory can be configured in low-power mode. By default, it is not in Low-power mode during Stop.
3. The SRAM clock can be gated on or off independently. By default clock is enabled in Run and Sleep modes.
4. Wake-up with internal tamper.
5. GPIOs state can be retained during Standby mode. By default GPIOs states are not retained.
6. 7 pins are capable of wake-up from Standby mode: PA0, PA2, PB7, PC1, PC13, PD2, and PE6.

In addition, the power consumption in Run mode can be reduced by slowing down the system clocks and by gating the clocks to the APB and AHB peripherals when they are not used.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex-M33 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 49.2.5: Debug and low-power modes*.

8.7.1 Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering the Sleep mode.

For more details, refer to *Section 9: Reset and clock control (RCC)*.

8.7.2 Peripheral clock gating

In Run mode, the HCLK and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

To further reduce the power consumption in Sleep mode, or Stop 0 mode the peripheral clocks can be disabled before executing the WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Disabling the peripheral clocks in Sleep mode or Stop 0 mode can be performed automatically by resetting the corresponding bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers.

8.7.3 Low-power modes

Entering into a low-power mode

The MCU enters in low-power modes by executing the WFI (wait for interrupt), or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex-M33 system control register is set on return from ISR.

Entering into a low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Exiting a low-power mode

The MCU exits the Sleep or Stop mode according to how the low-power mode was entered:

- If the WFI instruction or return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the low-power mode, the MCU exits the low-power mode as soon as an event occurs. The wake-up event can be generated either by:
  - An NVIC IRQ interrupt:
    - When SEVONPEND = 0 in the Cortex-M33 system control register
      By enabling an interrupt in the peripheral control register and in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear

pending register) must be cleared. Only NVIC interrupts with high enough priority can wake up and interrupt the MCU.

> When SEVONPEND = 1 in the Cortex-M33 system control register
By enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with high enough priority can wake up and interrupt the MCU.

– An event:

> Configuring an EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit, as the pending bits corresponding to the event line are not set. It may be necessary to clear the interrupt flag in the peripheral.

The MCU exits Standby mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC/TAMP event (see *Figure 561: RTC block diagram*).

After waking up from Standby mode, the program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

Caution: When the device is in Stop 0 mode, a peripheral interrupt powers on an internal oscillator. The corresponding NVIC interrupt channel must be enabled to allow the interrupt to exit the device from Stop mode. It is not allowed to disable a peripheral interrupt by disabling only the NVIC channel while keeping the peripheral interrupt enable, as the device could remain in Stop mode with clock ON.

8.7.4 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering the Sleep mode

The MCU enters the Sleep mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is clear (see the table below for details on how to enter the Sleep mode).

Exiting the Sleep mode

The MCU exits the Sleep mode as described in *Exiting a low-power mode* (see the table below for details on how to exit the Sleep mode).

Table 56. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) pending Refer to the Cortex-M33 system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt pending Refer to the Cortex-M33 system control register.
Mode exit	If WFI or Return from ISR was used for entry Interrupt (see *Table 100: STM32C5 vector table*) If WFE was used for entry and SEVONPEND = 0: Wake-up event (see *Section 16.3: EXTI functional description* If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (see *Table 100: STM32C5 vector table*) or wake-up event (see *Section 16.3: EXTI functional description*)
Wake-up latency	None

8.7.5 Stop 0 mode

The Stop 0 mode is based on the Cortex-M33 Deepsleep mode combined with the peripheral clock gating. In Stop mode, all clocks in the core domain are stopped. The HSI, PSI, and HSE oscillators are disabled.

It is possible to keep the HSI or PSI clock enabled during Stop mode, to be quickly available as kernel clock for peripherals.

All register contents are preserved. SRAMs can be preserved or powered down, depending on the SRAMxPDSx bits in PWR_PMCR.

Table 57. Memory power down block selection

Selection bit	Power down block in Stop mode		
	STM32C53x/542	STM32C55x/562	STM32C59x/5A3
SRAM1PDS	SRAM1 block from 0 to 32 Kbytes	SRAM1 block from 0 to 64 Kbytes	SRAM1 block from 0 to 128 Kbytes
SRAM2PDS1	SRAM2 block from 0 to 16 Kbytes		
SRAM2PDS2	SRAM2 block from 16 to 32 Kbytes	SRAM2 block from 16 to 64 Kbytes	SRAM2 block from 16 to 64 Kbytes
SRAM2PDS3	-	-	SRAM2 block from 64 to 128 Kbytes

The power-down reset is always available in Stop 0 mode.

I/O states in Stop 0 mode

In the Stop 0 mode, all I/O pins keep the same state as in the Run mode.

Entering the Stop 0 mode

The MCU enters the Stop mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 54* for details on how to enter the Stop mode).

If the flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop mode entry is delayed until the APB access is finished.

In Stop 0 mode, the following features can be selected by programming the individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started, it can be stopped only by a reset (see *Section 37.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in RCC_RTCCR.
- The internal RC oscillator LSI clock is configured by the LSION bit in RCC_RTCCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_RTCCR.

The PVD can be used in Stop mode. If not needed, it must be disabled by software to save power consumption.

The ADCx (x = 1 to 3) and the DAC1 can consume power during the Stop mode, unless they are disabled before entering this mode.

Exiting the Stop 0 mode

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered. Some peripherals are able to wake up the system (refer to *Table 103: EXTI line connections*) from Stop mode.

When exiting Stop 0 mode by issuing an interrupt or a wake-up event, HSI is selected as system clock. The MCU exits Stop 0 mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered.

Table 58. Stop 0 mode

Stop mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 00 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 00 in PWR_PMCR
	Note: To enter Stop 0 mode, all EXTI line pending bits (in Section 16.6.4: EXTI rising edge pending register (EXTI_RPR1) and Section 16.6.10: EXTI rising edge pending register 2 (EXTI_RPR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and the program execution continues.
Mode exit	If WFI or Return from ISR was used for entry: – any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 100: STM32C5 vector table*). – any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: – any EXTI line configured in event mode (see *Section 16.3: EXTI functional description*). If WFE was used for entry and SEVONPEND = 1: – any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 100: STM32C5 vector table*). – any EXTI line is configured in event mode (see *Section 16.3: EXTI functional description*)
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt ([PERIPH]EN and [PERIPH]LPEN bits must be set in the RCC, and a functional independent clock must be selected).
Wake-up latency	Longest wake-up time between: HSI wake-up time and flash memory wake-up time from Stop mode.

8.7.6 Stop 1 mode

The Stop 1 mode is based on the Cortex-M33 Deep sleep mode combined with the peripheral clock gating. In comparison to Stop 0, the voltage regulator is configured to Low voltage retention at 0.9 V. All clocks in the core domain are stopped. The HSI, PSI, and HSE oscillators are disabled.

All register contents are preserved. SRAMs can be preserved or powered down, depending on SRAMxPDSx bits in PWR_PMCR. Refer to *Table 57: Memory power down block selection* for details.

The power-down reset is always available in Stop mode.

I/O states in Stop 1 mode

In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering the Stop 1 mode

The MCU enters the Stop 1 mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 54* for details on how to enter the Stop mode).

If the flash memory programming is ongoing, the Stop 1 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop 1 mode entry is delayed until the APB access is finished.

In Stop 1 mode, the following features can be selected by programming the individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started, it can be stopped only by a reset (see *Section 37.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in RCC_RTCCR.
- The internal RC oscillator LSI clock is configured by the LSION bit in RCC_RTCCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_RTCCR.

The PVD can be used in Stop mode. If not needed, it must be disabled by software to save power consumption.

The ADCx (x = 1 to 3) and the DAC1 can consume power during the Stop 1 mode, unless they are disabled before entering this mode.

Exiting the Stop 1 mode

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered.

When exiting Stop mode by issuing an interrupt or a wake-up event, HSI is selected as system clock. The MCU exits Stop 1 mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered.

Table 59. Stop 1 mode

Stop mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 01 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 01 in PWR_PMCR
	Note: To enter Stop mode, all EXTI line pending bits (in Section 16.6.4: EXTI rising edge pending register (EXTI_RPR1) and Section 16.6.10: EXTI rising edge pending register 2 (EXTI_RPR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and the program execution continues.
Mode exit	If WFI or Return from ISR was used for entry: – any GPIO EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source must be external interrupts with wake-up capability (see *Table 100: STM32C5 vector table*). If WFE was used for entry and SEVONPEND = 0: – any GPIO EXTI line configured in event mode (see *Section 16.3: EXTI functional description*). If WFE was used for entry and SEVONPEND = 1: – any GPIO EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source must be external interrupts with wake-up capability (see *Table 100: STM32C5 vector table*). – any GPIO EXTI line is configured in event mode (see *Section 16.3: EXTI functional description*)
Wake-up latency	Longest wake-up time between: HSI wake-up time, regulator stabilization from 0.9 V to 1.2 V and flash memory wake-up time from Stop mode.

8.7.7 Standby mode

The lowest power mode in which the POR/PDR is active is the Standby mode. It is based on the Cortex-M33 Deepsleep mode, with the voltage regulators disabled. The HSI, PSI, and HSE oscillators are also switched off.

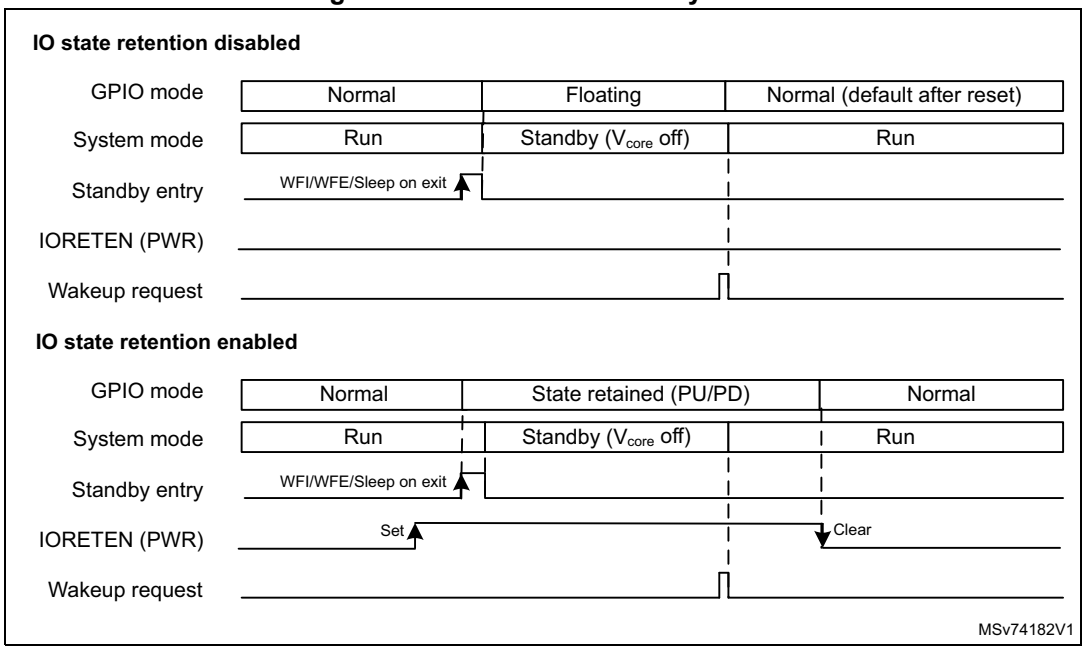
The SRAMs and register contents are lost except for registers in the backup domain and Standby circuitry (see *Figure 23*).

I/O states in Standby mode

In the Standby mode, the I/Os are by default in floating state. If the IORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. I/O retention mode is enabled for all I/Os except those supporting the standby functionality and JTAG I/Os (PA13, PA14, PA15, and PB4). When entering into Standby mode, the state of the output is sampled, and pull-up or pull-down resistors are set to maintain the I/O output during Standby mode.

If the JTAGIORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. I/O retention mode is enabled for PA13, PA14, PA15, and PB4 (default JTAG pull-up/pull-down after wake-up is not enabled).

Figure 23. I/O states in Standby mode



After wake-up from Standby mode, as long as IORETEN (or JTAGIORETEN for JTAG I/Os) is set, the retained state (pull-up/pull-down) remains applied.

The GPIO pin state before standby can be identified with GPIO_IDR register (when both GPIO port clock and input buffer are enabled).

The application can release the I/O state (clear the retained Pull-up/Pull-down) by clearing the IORETEN (or JTAGIORETEN for JTAG I/Os) bit, before or after reconfiguring the GPIOs and related peripherals. The GPIOs can then be configured in a known state before releasing the retained state.

The RTC output on PC13 is functional in Standby mode. PC14 and PC15 used for LSE are also functional. Up to seven wake-up (WKUPx, x = 1 to 7) and three RTC tamper pins are available.

Entering Standby mode

The MCU enters the Standby mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 60* for details on how to enter Standby mode).

In Standby mode, the following features can be selected by programming individual control bits:

- The independent watchdog (IWDG) is started by writing to its Key register or by hardware option. Once started, it can be stopped only by a reset (see *Section 37.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in RCC_RTCCR.
- The internal RC oscillator LSI clock is configured by the LSION bit in RCC_RTCCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_RTCCR.
- The I/Os retention is configured by the IORETEN bit in the PWR_IORETR register.

Exiting Standby mode

The MCU exits the Standby mode as described in *Exiting a low-power mode*. The SBF status flag in PWR_PMSR) indicates that the MCU was in Standby mode. All registers are reset after wake-up from Standby except for PWR_RTCCR and PWR_IORETR *(see Table 60* for more details on how to exit Standby mode).

When exiting Standby mode, I/Os output state that were retained during Standby through IORETEN bit, keep this configuration upon exiting Standby mode until the IORETEN bit in PWR_IORETR register is cleared. Once IORETEN is cleared, the I/Os are configured to their reset values, or to the pull-up/pull-down state according to the GPIOx_PUPDR registers.

For I/Os, with a pull-up or pull-down predefined after reset (some JTAG/SWD I/Os), in case those pull-up or pull-downs are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Also in case the GPIOx_PUPDR values programed after exiting from Standby are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Table 60. Standby mode

Standby mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 10 in PWR_PMCR – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 10 in PWR_PMCR – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wake-up source, cleared
Mode exit	WKUPx pin edge, RTC event, external reset in NRST pin, IWDG reset
Wake-up latency	Reset phase

8.7.8 Power mode output pins

In order to help the debug, two signals are available as device pins alternate functions:

- **CSLEEP**

  When set, CSLEEP indicates that the system is in Sleep mode: WFI or WFE has been executed.

  When cleared, CSLEEP indicates that the system is in Run mode.

- **CSTOP**

  When set, CSTOP indicates that the system is in Stop mode, meaning that the following conditions are fulfilled:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  – No AHB/APB clock is running.

  When cleared, CSTOP indicates that the system is not in Stop mode: AHB/APB clocks are running.

The table below explains the MCU power mode depending on these signal states.

Table 61. Power mode output states versus MCU power modes

CSLEEP	CSTOP	MCU power modes(1)
0	0	Run mode
1	0	Sleep mode
1	1	Stop mode

1. CSLEEP and CSTOP are generated in core domain, consequently they are not driven in Standby mode.

8.8 PWR privileged protection

By default, after a reset, all registers can be read or written with both privileged and unprivileged accesses, except PWR_PRIVCFGR, which can be written only with privileged access. PWR_PRIVCFGR can be read by privileged and unprivileged accesses.

The PRIV bit of PWR_PRIVCFGR can be written only with privileged access. It configures the privileged access of all PWR functions (defined by RCC, or GPIO).
When the PRIV bit is set in PWR_PRIVCFGR:

- The PWR bits can be written only with privileged access.
- The PWR bits can be read only with privileged access except PWR_PRIVCFGR, which can be read by privileged or unprivileged accesses.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as 0, and the write to these bits is ignored (RAZ/WI).

8.9 PWR interrupts

The table below gives a summary of the interrupt sources and the way to control them.

Table 62. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop mode	Exit Standby mode
PVD output	Programmable voltage detector through EXTI line 16	PVDO	EXTI line 16 enabled	Write EXTI PIF16 = 1	Yes	No

8.10 PWR registers

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

For the actual availability of some peripherals/features and their related bits, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them as reserved, and keep them at reset value

8.10.1 PWR power mode control register (PWR_PMCR)

Address offset: 0x000

Reset value: 0x0000 0000

This register is protected against unprivileged access when PRIV = 1 in PWR_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SRAM1PDS	SRAM2PDS2	SRAM2PDS1	SRAM2PDS3	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	Res.	Res.	LPMS[1:0]	
						rw		rw						rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SRAM1PDS**: AHB SRAM1 block power down in Stop mode
0: AHB RAM1 content is kept in Stop mode.
1: AHB RAM1 content is lost in Stop mode.

Bit 25 **SRAM2PDS2**: AHB SRAM2 block 2 power down in Stop mode
0: AHB RAM2 block 2 content is kept in Stop mode.
1: AHB RAM2 block 2 content is lost in Stop mode.
Note: For block definition, refer to *Table 57: Memory power down block selection*.

Bit 24 **SRAM2PDS1**: AHB SRAM2 block 1 power down in Stop mode
0: AHB RAM2 block 1 content is kept in Stop mode.
1: AHB RAM2 block 1 content is lost in Stop mode.
Note: For block definition, refer to *Table 57: Memory power down block selection*.

Bit 23 **SRAM2PDS3**: AHB SRAM2 block 3 power down in Stop mode
0: AHB RAM2 block 3 content is kept in Stop mode.
1: AHB RAM2 block 3 content is lost in Stop mode.
Note: For block definition, refer to *Table 57: Memory power down block selection*.

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FLPS**: Flash memory low-power mode in Stop mode
This bit is used to obtain the best trade-off between low-power consumption and restart time when exiting from Stop mode.
When it is set, the flash memory enters low-power mode when the system is in Stop mode.
0: Flash memory remains in normal mode when the system enters Stop mode (quick restart time).
1: Flash memory enters low-power mode when the system enters Stop mode (low-power consumption).

Bit 8 Reserved, must be kept at reset value.

Bit 7 **CSSF**: Clear Standby and Stop flags (always read as 0)
This bit is cleared to 0 by hardware.
0: No effect
1: STOPF and SBF flags cleared

Bits 6:2 Reserved, must be kept at reset value.

Bits 1:0 **LPMS[1:0]**: low-power mode selection
This bits define the Deepsleep mode.
00: Stop 0 mode when entering DeepSleep.
01: Stop 1 mode when entering DeepSleep.
10: Standby mode when entering DeepSleep.
11: Reserved

8.10.2 PWR status register (PWR_PMSR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.
									r	r					

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SBF**: System standby flag
This bit is set by hardware and cleared only by a POR or by setting the CSSF bit.
0: System has not been in Standby mode.
1: System has been in Standby mode.

Bit 5 **STOPF**: Stop flag
This bit is set by hardware and cleared only by any reset or by setting the CSSF bit.
0: System has not been in Stop mode.
1: System has been in Stop mode.

Bits 4:0 Reserved, must be kept at reset value.

8.10.3 PWR RTC domain control register (PWR_RTCCR)

Address offset: 0x24

Reset value: 0x0000 0000

This register is protected against unprivileged access when PRIV = 1 in PWR_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRTCP
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DRTCP**: Disable RTC domain write protection

In reset state, all registers in RTC domain are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Write access to RTC domain disabled

1: Write access to RTC domain enabled

8.10.4 PWR voltage monitor control register (PWR_VMCR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is protected against unprivileged access when PRIV = 1 in PWR_PRIVCFGR register.

The PVDE bit is protected by lock mechanism. The lock control is in the SBS module controlled by PVDL bit in SBS_CFGR2 register. By default, the PVDE is unlocked.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PVDE**: PVD enable

0: PVD disabled

1: PVD enabled

8.10.5 PWR voltage monitor status register (PWR_VMSR)

Address offset: 0x03C

Reset value: 0x00X0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.	Res.	Res.
									r						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PVDO**: programmable voltage detect output

This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.

0: V_{DD} is equal or higher than the PVD threshold.

1: V_{DD} is lower than the PVD threshold.

Note: Since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bits 21:0 Reserved, must be kept at reset value.

8.10.6 PWR wake-up status clear register (PWR_WUSCR)

Address offset: 0x040

Reset value: 0x0000 0000

Each register bit CWUFx (x = 1 to 7) is protected against unprivileged access when PRIV = 1 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF 7	CWUF 6	CWUF 5	CWUF 4	CWUF 3	CWUF 2	CWUF 1
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **CWUFx**: clear wake-up pin flag for WUFx (x = 7 to 1)

These bits are always read as 0.

0: No effect

1: Writing 1 clears the WUFx wake-up pin flag (bit is cleared to 0 by hardware).

8.10.7 PWR wake-up status register (PWR_WUSR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **WUFx**: wake-up pin WUFx flag (x = 7 to 1)

This bit is set by hardware and cleared only by a RESET pin or by setting the CWUFx bit in PWR_WUSCR register or by hardware when WUPENx = 0.

0: No wake-up event occurred.

1: Wake-up event received from WUFx pin.

8.10.8 PWR wake-up configuration register (PWR_WUCR)

Address offset: 0x048

Reset value: 0x0000 0000

Each WUPPUPDx (x = 1 to 7), WUPPx (x = 1 to 7), and WUPENx (x = 1 to 7) is protected against unprivileged access when PRIV = 1 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	WUPPUPD7[1:0]		WUPPUPD6[1:0]		WUPPUPD5[1:0]		WUPPUPD4[1:0]		WUPPUPD3[1:0]		WUPPUPD2[1:0]		WUPPUPD1[1:0]	
		rw	rw	rw	rw	rw	w	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WUPP 7	WUPP 6	WUPP 5	WUPP 4	WUPP 3	WUPP 2	WUPP 1	Res.	WUPE N7	WUPE N6	WUPE N5	WUPE N4	WUPE N3	WUPE N2	WUPE N1
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **WUPPUPDx[1:0]**: Wake-up pin pull configuration for WKUPx (x = 7 to 1)

These bits define the I/O pad pull configuration used when WUPENx = 1. The associated GPIO port pull configuration must be set to the same value or to 00. The wake-up pin pull configuration is kept in Standby mode.

00: No pull-up

01: Pull-up

10: Pull-down

11: Reserved

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **WUPPx**: Wake-up pin polarity bit for WKUPx (x = 7 to 1)

These bits define the polarity used for event detection on WUPx external wake-up pin.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **WUPENx**: Enable wake-up pin WKUPx (x = 7 to 1)

These bits are set and cleared by software.

0: An event on WUPx pin does not wake-up the system from Standby mode.

1: A rising or falling edge on WUPx pin wakes up the system from Standby mode.

Note: An additional wake-up event is detected if WUPx pin is enabled (by setting the WUPENx bit) when WUPx pin level is already high when WUPPx selects rising edge, or low when WUPPx selects falling edge.

8.10.9 PWR I/O retention register (PWR_IORETR)

Address offset: 0x050

Reset value: 0x0000 0000

This register is protected against unprivileged access PRIV = 1 in the PWR_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JTAGIORETEN
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IORETEN
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **JTAGIORETEN**: IO retention enable for JTAG I/Os

0: I/O retention mode is disabled.

1: I/O retention mode is enabling for PA13, PA14, PA15, and PB4.

When entering Standby mode, the output is sampled, and apply to the output I/O during the Standby power mode.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **IORETEN**: IO retention enable

0: I/O retention mode is disabled.

1: I/O retention mode is enabling for all I/Os except the I/O support the standby functionality and PA13, PA14, PA15, and PB4.

When entering Standby mode, the output is sampled, and apply to the output I/O during the Standby power mode.

Note: The I/O state is not retained if the DBG_STANDBY bit is set in DBGMCU_CR register.

8.10.10 PWR privilege configuration register (PWR_PRIVCFGR)

Address offset: 0x104

Reset value: 0x0000 0000

This register can be written only when the access is privileged. It can be read by privileged or unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PRIV**: PWR functions privilege configuration

Set and reset by software. This bit can be written only by privileged access.

0: Read and write to PWR functions can be done by privileged or unprivileged access.

1: Read and write to PWR functions can be done by privileged access only.

Bit 0 Reserved, must be kept at reset value.

8.10.11 PWR register map

Table 63. PWR register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_PMCR**	Res.	Res.	Res.	Res.	Res.	SRAM1PDS	SRAM2PDS2	SRAM2PDS1	SRAM2PDS3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	Res.	Res.	LPMS[1:0]	
	Reset value						0	0	0	0														0		0						0	0
0x004	**PWR_PMSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0					
0x008 - 0x020	Reserved	Res.																															
0x024	**PWR_RTCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRTCP
	Reset value																																0
0x028 - 0x030	Reserved	Res.																															
0x034	**PWR_VMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDE
	Reset value																																0
0x038	Reserved	Res.																															

Table 63. PWR register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03C	**PWR_VMSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										X																						
0x040	**PWR_WUSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF7	CWUF6	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
	Reset value																										0	0	0	0	0	0	0
0x044	**PWR_WUSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																										0	0	0	0	0	0	0
0x048	**PWR_WUCR**	Res.	Res.	WUPPUPD7[1:0]		WUPPUPD6[1:0]		WUPPUPD5[1:0]		WUPPUPD4[1:0]		WUPPUPD3[1:0]		WUPPUPD2[1:0]		WUPPUPD1[1:0]		Res.	WUPP7	WUPP6	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1	Res.	WUPEN7	WUPEN6	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04C	Reserved	Res.																															
0x050	**PWR_IORETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JTAGIORETEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IORETEN
	Reset value																0																0
0x054 - 0x100	Reserved	Res.																															
0x104	**PWR_PRIVCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.
	Reset value																															0	

Refer to *Section 2.2* for the register boundary addresses.

9 Reset and clock control (RCC)

9.1 RCC introduction

The reset and clock control (RCC) manages the different resets, and generates the clocks for the bus and peripheral.

9.2 RCC pins and internal signals

The table below lists the RCC inputs and output signals connected to package pins or balls.

Table 64. RCC input/output signals connected to package pins or balls

Signal name	Signal type	Description
NRST	I/O	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO	O	Output clock for external devices
LSCO	O	Low-speed output clock for external devices
AUDIOCLK	I	External kernel clock input for I2S1, I2S2, and I2S3

9.3 RCC reset functional description

There are three types of reset:

- A system reset
- A power reset
- A RTC domain reset

9.3.1 Power reset

A power reset is generated when one of the following events occurs:

- A power-on or power-down reset (POR/PDR)
- When exiting Standby mode

A power-on or power-down reset (POR/PDR) sets all registers to their reset values.

When exiting Standby mode, all registers in the core domain are set to their reset value. Registers outside the core domain (RTC, WKUP, IWDG, and GPIO pull-up/pull-down configuration during Standby and Standby mode exit) are not impacted.

9.3.2 System reset

A system reset sets all registers to their reset values, except the reset flags in RCC_RSR, and the registers in the RTC domain.

A system reset is generated when one of the following events occurs:

- A low level on the NRST pin (external reset)
- A window watchdog event (WWDG reset)
- An independent watchdog event (IWDG reset)
- A software reset (SW reset) (see *Software reset*)
- A low-power mode security reset (see *Low-power mode security reset*)
- A POR reset

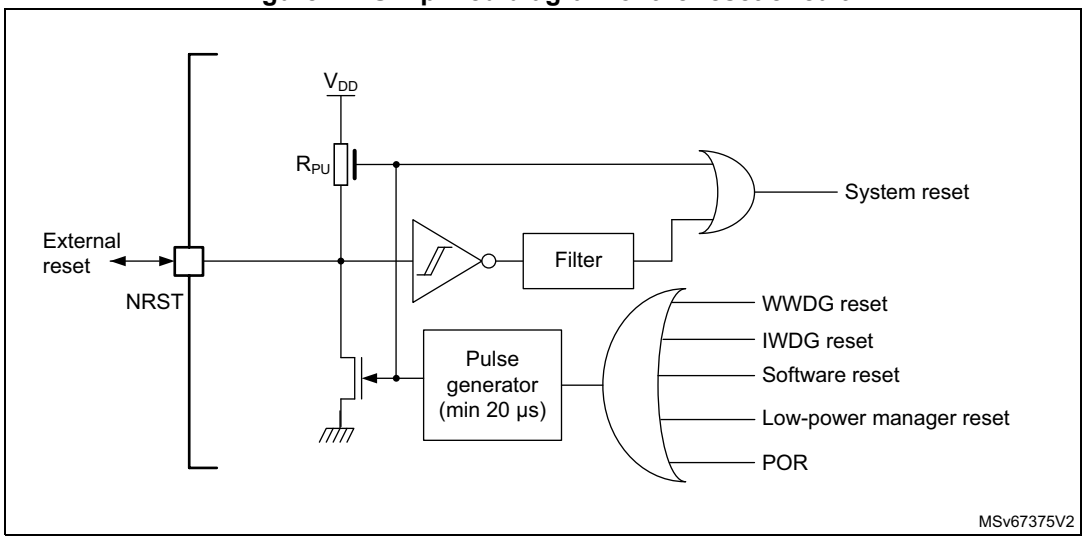
The reset source can be identified by checking the reset flags in RCC_RSR.

These sources act on the NRST pin and this pin is always kept low during the delay phase. The reset service routine vector is selected depending on RDP level, on Boot option bytes or on both.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case of an internal reset, the internal pull-up R_{PU} is deactivated to save the power consumption through the pull-up resistor.

Figure 24. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in Cortex-M33 application interrupt and reset control register must be set to force a software reset on the device.

Low-power mode security reset

To avoid that critical applications mistakenly enter a low-power mode, the following low-power mode security resets are available. If enabled in option bytes, the resets are generated in any of the following conditions:

- Entering Standby mode: this type of reset is enabled by resetting NRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
- Entering Stop mode: this type of reset is enabled by resetting NRST_STOP bit in user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

For further information on the user option bytes, refer to *Section 6.4: FLASH option bytes*.

9.3.3 RTC domain reset

The RTC domain has two specific resets, generated when one of the following events occurs:

- A software reset, is triggered by setting the RTCDRST bit in RCC_RTCCR. Write access to this domain must be enabled before setting RTCDRST bit to perform the reset.
- A V_{DD} power on

An RTC domain reset affects the LSE oscillator, the RTC, the backup registers, and the RCC_RTCCR register.

9.3.4 Reset source identification

The application can identify the reset source by checking the reset flags in the RCC_RSR register.

The software can reset the flags by setting the RMVF bit.

The table below shows how the status bits of the RCC_RSR register behave according to the situation that generated the reset. For example, when an IWDG timeout occurs, if the CPU is reading the RCC_RSR register during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

Table 65. Reset source identification (RCC_RSR)(1)

	Reset	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF
1	Power-on reset	0	0	0	0	1	1
2	Pin/pad reset	0	0	0	0	0	1
3	System reset generated by CPU	0	0	0	1	0	1
4	WWDG reset	0	1	0	0	0	1
5	IWDG reset	0	0	1	0	0	1
6	Illegal stop entry reset	1	0	0	0	0	1

1. Gray cells highlight the register bits that are set.

9.4 RCC clocks functional description

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSIS: high-speed internal clock at 144 MHz
  - and HSIDIV3: high-speed internal clock is divided by 3; at 48 MHz
- PSIS: programmable speed internal clock up to 160MHz (max SYSCLK is 144 MHz)
- HSE: high-speed external crystal or clock, from 4 to 50 MHz

The HSIDIV3 is used as a the system clock source after startup from reset, at 48 MHz.

Note: If HSE or PSIS with HSE as reference are used as SYSCLK and the HSE CSS detect a failure, HSIDIV3 is automatically set as SYSCLK. Adequate wait state supporting both frequencies must be configured in this case.

The device has the following additional clock sources:

- HSIK: high-speed internal clock divided by 1;1.5; 2;...; 7.5; 8
- PSIDIV3: programmable speed internal clock divided by 3
- PSIK: programmable speed internal clock divided by 1;1.5; 2;...; 7.5; 8
- LSI: 32 kHz low-speed internal RC that drives the independent watchdog and optionally the RTC used for auto-wake-up from Stop and Standby modes
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB and APB frequencies. The maximum frequency of the AHB and APB domains is 144 MHz.

The peripheral clocks are derived from their bus clock (HCLK, PCLK1, PCLK2 or PCLK3), except those that receive an independent kernel clock. This kernel clock can be selected by software between several sources thanks to RCC_CCIPRx registers (x = 1 to 3).

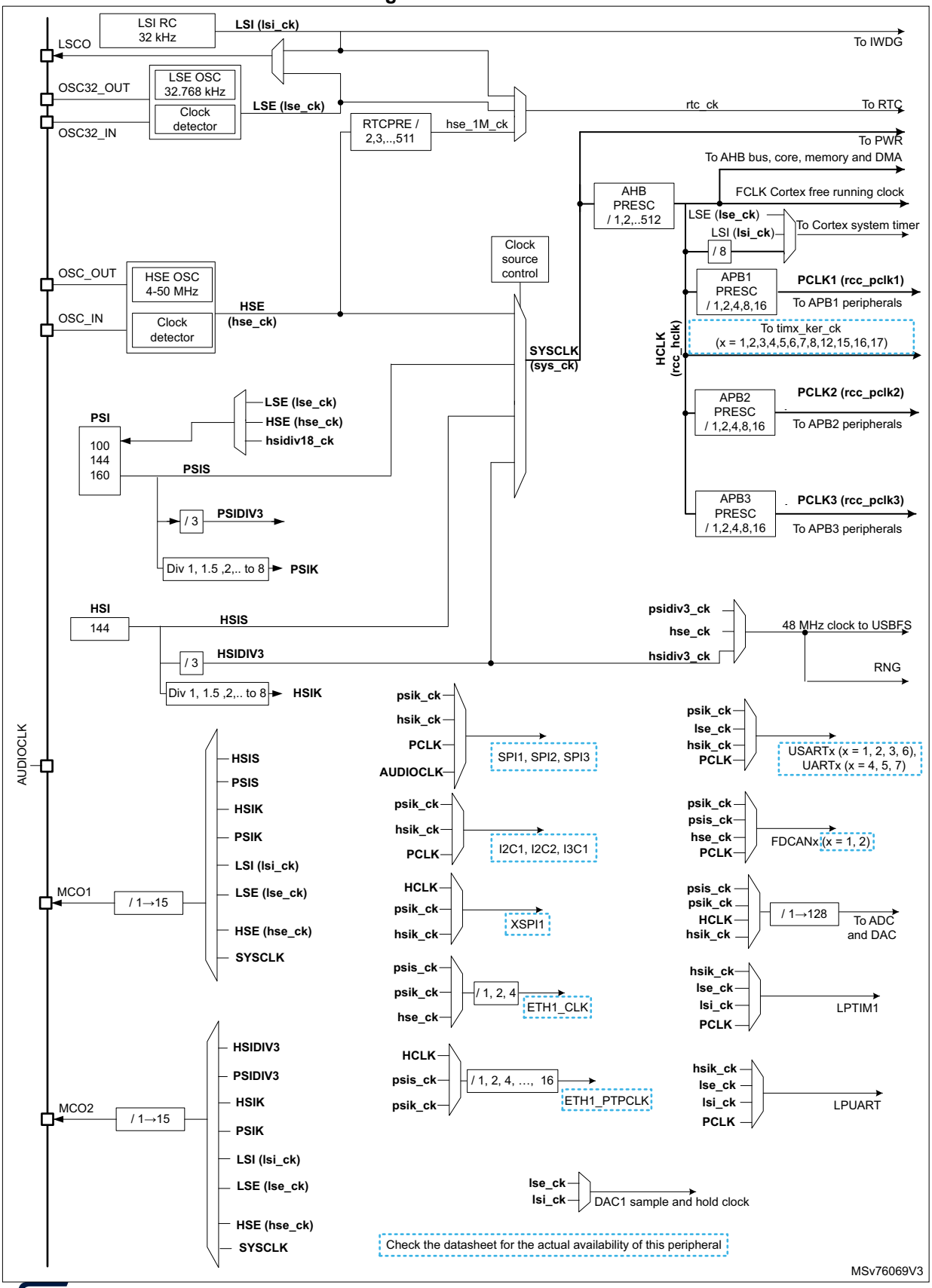
Note: All timers (except LPTIM) are clocked by HCLK, independently of their respective PCLK.

In addition, the RTC kernel clock is selected by software in RCC_RTCCR. The IWDG clock is always the LSI 32 kHz clock.

The RCC feeds the Cortex system timer (SysTick) external clock with the AHB clock (HCLK) divided by eight, or LSE or LSI. The SysTick can work either with this clock or directly with the Cortex clock (HCLK), configurable in the SysTick control and status register.

FCLK acts as a a Cortex-M33 free-running clock.

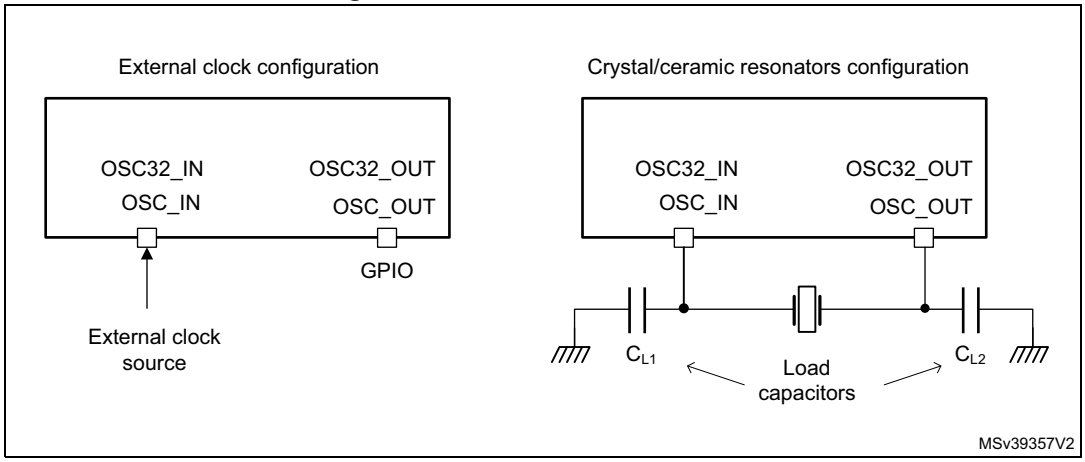
Figure 25. Clock tree



9.4.1 HSE clock

The HSE block can generate a clock from an external crystal/ceramic resonator, or from an external clock source.

Figure 26. HSE/ LSE clock sources



External clock source (HSE bypass)

In this mode, an external clock source must be provided to OSC_IN pin. The external clock can be low swing (analog) or digital. If this clock is directly used by a peripheral, the duty cycle requirement is defined by the peripheral and the application (refer to datasheet for more details).

In case of an analog clock (low swing) the HSEBYP and HSEON bits must be set to 1 in RCC_CR1.

In case of a digital clock, the HSEBYP and the HSEEXT bits must be set to 1 followed by setting the HSEON bit to 1 in RCC_CR1.

External crystal/ceramic resonator

The oscillator is enabled by setting the HSEBYP bit to 0 and HSEON bit to 1.

The HSE can be used when the product requires a very accurate high-speed clock.

The associated hardware configuration is shown in *Figure 26*: the resonator and the load capacitors must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag in RCC_CR1 indicates whether the HSE oscillator is stable or not. At startup, the hse_ck clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in RCC_CIER.

The HSE can be switched on and off through the HSEON bit.

Note: The HSE cannot be switched off if one of the following two conditions is met:

- the HSE is used directly (through software multiplexer) as a system clock
- the HSE is selected as the reference clock for PSI, with PSI enabled and selected to provide the system clock (through software multiplexer).

In that case the hardware does not allow programming the HSEON bit to 0.

The HSE is automatically disabled by hardware when the system enters Stop or Standby mode. It can be kept on under Stop 0 mode if PSIKERON is set and PSI needs HSE as a reference clock.

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as a clock source for other application components.

9.4.2 HSI oscillator and HSIS clock

The HSI block provides the default clocks to the product.

The HSI is a high-speed internal oscillator that can be used as a system clock or peripheral clock offering a 144MHz clock available at startup and upon wake-up from low power modes. The HSI oscillator can produce 3 output clocks; HSIS, HSIDIV3, and HSIK.

Two output dividers let you extent the available frequencies for the device. One configurable divider by 1.5, 2, 2.5, ... 7.5 or 8 is configured thanks to HSIKDIV[3:0] generating HSIK clock and a second divided by 3 intended for system clock and USB generating HSIDIV3.

The HSI advantages are the following:

- Low-cost clock source, as no external crystal is required
- Faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

HSI oscillator can be switched on and off using the HSISON bit, the HSIDIV3ON bit, and the HSIKON bit. Only the enabled clock output is generated.

Note: The HSI cannot be switched off if the HSI is used directly (via SW mux) as a system clock. In that case the hardware does not allow clearing the HSISON bit or HSIDIV3ON bit to 0.

The HSISRDY, HSIKRDY, and HSIDIV3RDY flags indicate if the HSI is stable or not. At startup, the HSIS output clock is not released until HSISRDY is set by hardware.

The HSIDIV3 clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to *Section 9.4.10: Clock security system (CSS)*).

In addition, the HSI clocks can be driven to the MCO1 or MCO2 outputs and used as clock source for other application components.

Care must be taken when the HSIK is used as a kernel clock for communication peripherals, the application must take in account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available
- the frequency accuracy.

Note: The HSI can remain enabled when the system is in Stop 0 mode thanks to HSIKERON bit. In this case, the HSI and its clocks outputs HSIK, HSIDIV3, or HSIS, are kept active if their respective HSIKON, HSIDIV3ON, and HSISON are set. The states of these bits are consequently maintained exiting Stop 0 if HSIKERON bit is set.

HSI calibration

HSI oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACCHSI (refer to the product datasheet for more information).

After a power-on reset, the factory calibration value is loaded. If the application is subject to voltage or temperature variations, this may affect the oscillator frequency. The user application can trim the HSI frequency using the TRIM[6:0] bits in the CRS_CR register.

The clock recovery system can be used to perform this calibration. It requires an accurate reference that can be HSE (through HSE_1MHz connection), LSE, USB Start of Frame, or an external signal.

9.4.3 PSI oscillator and PSIS clock

The PSI oscillator provides a 100 MHz, 144 MHz, or 160 MHz frequency, selected thanks to PSIFREQ[1:0] bitfield, to the product on condition that an adequate reference clock is available. This programmable speed internal oscillator can be used as a system clock or peripheral clock. The PSI oscillator can produce three output clocks; PSIS, PSIDIV3, and PSIK.

The PSI is only operational when receiving a reference clock of 32.768 kHz, 8 MHz, or 8.33 MHz (25 MHz divided by 3) as input. The PSIREF[2:0] bitfield helps you get the required input frequency depending on your chosen external clock. This external clock is selected thanks to PSIREFSRC[1:0] multiplexer between LSE, HSE, or HSI divided by 18. Details of possible settings are listed in the table below.

Table 66. PSI frequency settings

PSIFREQ[1:0]	PSIREFSRC[1:0]	PSIREF[2:0]
160	HSE	48
		32
		24
		16
		8
	HSI/18	8
	LSE	0.032768
144	HSE	48
		32
		24
		16
		8
	HSI/18	8
	LSE	0.032768

Table 66. PSI frequency settings (continued)

PSIFREQ[1:0]	PSIREFSRC[1:0]	PSIREF[2:0]
100	HSE	50
		48
		32
		25
		24
		16
		8
	HSI/18	8
	LSE	0.032768

Note: If LSE is used as reference, an intrinsic error must be anticipated on the output frequency (100.016 MHz, 144.015 MHz, and 160.006 MHz)

Two output dividers can be used to extent the available frequencies for the device. One configurable divider by 1.5, 2, 2.5, 3, 3.5, ... , 7.5, or 8, configured thanks to PSIKDIV[3:0] generating PSIK clock and a second dividing by 3 generating PSIDIV3 clock, intended for USB.

The PSI advantages are the following:

- Programmable high-speed clock source based on a reference clock
- Configurable output frequencies (100 MHz, 144 MHz, 160 MHz)

The PSI frequency accuracy is directly related to the accuracy of its reference clock (for more details regarding the specification of this oscillator, refer to product datasheet).

PSI oscillator can be switched on and off using the PSISON bit, the PSIDIV3ON bit, or the PSIKON bit. Only the enabled clock output is generated.

Note: The PSI cannot be switched off if used (via SW mux) as a system clock. In that case, the hardware does not allow programming the PSISON bit to 0.

The PSISRDY, PSIKRDY, and PSIDIV3RDY flags indicate if the oscillator is stable or not.

In addition, the PSI clocks can be driven to the MCO1 or MCO2 outputs, and used as clock source for other application components.

Care must be taken when the PSI is used as a kernel clock for communication peripherals; the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available

Note: The PSI can remain enabled when the system is in Stop 0 mode thanks to PSIKERON bit. In this case the PSI and its clock outputs PSIK, PSIKDIV3, or PSIS are kept enabled if their respective PSIKON, PSIDIV3ON, and PSISON are set. The states of those bits are maintained exiting Stop 0 if the HSIKERON bit is set.

9.4.4 HSIDIV3 and PSIDIV3 clock

The HSIDIV3 clock signal is generated from the internal 144 MHz RC oscillator, divided by 3 to be used directly for USB and for random number generator (RNG).

This internal 48 MHz clock is mainly dedicated to provide a high-precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE, or an external signal to automatically and quickly adjust the oscillator frequency on-the-fly. It is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI oscillator runs on its default frequency, subject to manufacturing process variations.

For more details on how to configure and use the CRS peripheral, refer to *Section 10: Clock recovery system (CRS)*.

The HSIDIV3RDY flag in the RCC_CR register indicates whether the HSIDIV3 clock is stable or not. At startup, the HSIDIV3 output clock is not released until this bit is set by the hardware.

The HSIDIV3 clock signal can be switched on and off using the HSIDIV3ON bit in the RCC_CR register.

The PSIDIV3 clock signal is generated from the internal 144 MHz RC oscillator, divided by 3 to be used directly for USB and for random number generator (RNG). It benefits from the accuracy of the HSE or LSE thanks to its operation in PLL mode. Similarly to HSIDIV3, it is disabled in Stop or Standby mode, ready once PSIDIV3RDY is set and enabled thanks to PSIDIV3ON bit is set.

9.4.5 HSIK and PSIK clock

THE HSIK and PSIK are two clock outputs offering frequency flexibility to the user. These dividers can be configured thanks to HSIKDIV[3:0] and PSIKDIV[3:0] bitfields respectively with division factors of 1, 1.5, 2, 2.5 ... 7.5, and 8. This allow to create a 96 MHz clock source derived from a 144 MHz for example if divided by 1.5, useful for FDCAN operation, or 64 MHz out of the 160 MHz divided by 2.5 to tune your DAC output sample frequency.

The HSIKRDY and PSIKRDY flags in the RCC_CR register indicate whether the HSIK and PSIK clocks are stable or not. At startup, the HSIK, and PSIK output clocks are not released until this bit is set by hardware.

The HSIK and PSIK clock signals can be switched on and off using the HSIKON and PSIKON bits in the RCC_CR register.

Table 67. PSIK and HSIK output frequency

PSIKDIV[3:0]/ HSIKDIV[3:0] division factor	PSIK frequency if PSI 160MHz	PSIK frequency if PSI 144MHz and hsik frequency	PSIK frequency if PSI 100MHz
1	160 MHz	144 MHz	100 MHz
1.5	106. 67 MHz	96 MHz	66.67 MHz
2	80 MHz	72 MHz	50 MHz
2.5	64 MHz	57.6 MHz	40 MHz
3	53.33 MHz	48 MHz	33.33 MHz
3.5	45.71 MHz	41.14 MHz	28.57 MHz

Table 67. PSIK and HSIK output frequency (continued)

PSIKDIV[3:0]/ HSIKDIV[3:0] division factor	PSIK frequency if PSI 160MHz	PSIK frequency if PSI 144MHz and hsik frequency	PSIK frequency if PSI 100MHz
4	40 MHz	36 MHz	25 MHz
4.5	35.56 MHz	32 MHz	22.22 MHz
5	32 MHz	28.8 MHz	20 MHz
5.5	29.09 MHz	26.18 MHz	18.18 MHz
6	26.67 MHz	24 MHZ	16.67 MHz
6.5	24.62 MHz	22.15 MHz	15.38 MHz
7	22.86 MHz	20.57 MHz	14.28 MHz
7.5	21.33 MHz	19.2 MHz	13.33 MHz
8	20 MHz	18 MHz	12.5 MHz

Caution: 160 MHz is above the maximum frequency supported by the product. It must not be propagated to the kernel clock of peripherals.

9.4.6 LSE clock

The LSE block can generate a clock from an external crystal/ceramic resonator, or from an external user clock.

External clock source (LSE bypass)

In this mode, an external clock source must be provided to OSC32_IN pin. The input clock can have a frequency up to 1 MHz, and be low swing (analog) or digital. A duty cycle close to 50% is recommended.

This external clock is provided to the OSC32_IN pin while the OSC32_OUT pin must be left high-Z (see *Figure 25*).

In case of an analog clock (low swing), the LSEBYP, and LSEON bits must be set to 1 in RCC_RTCCR.

In case of a digital clock, the LSEBYP and the LSEEXT bits must be set to 1 followed by setting the LSEON bit to 1 in RCC_RTCCR.

External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power, highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions.

The LSERDY flag in RCC_RTCCR indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by the hardware. An interrupt can be generated if enabled in RCC_CIER.

The LSE oscillator is switched on and off using the LSEON bit. The LSE remains enabled when the system enters Stop or Standby mode.

In addition, the LSE clock can be driven to the MCO1 and MCO2 output and used as a clock source for other application components.

The LSE also offers a programmable driving capability (LSEDRV[1:0]), which can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to ensure a stable oscillation.

The driving capability must be set before enabling the LSE oscillator.

9.4.7 LSI clock

The LSI acts as a low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG) and auto-wake-up unit (AWU).

The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched on and off using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced on and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode.

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in RCC_CIER.

In addition, the LSI clock can be driven to the MCO1 and MCO2 output, and used as a clock source for other application components.

Note: LSION and LSIRDY are located in RCC_RTCCR.

9.4.8 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI oscillator, through HSIS clock
- HSI oscillator divided by 3, through HSIDIV3 clock source
- PSI oscillator, through PSIS clock
- HSE oscillator

The system clock maximum frequency is 144 MHz. After a system reset (or after leaving Standby mode), the HSI oscillator divided by 3, at 48 MHz, is selected as the system clock. When a clock source is used directly or through PSI as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay). If a clock source not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in RCC_CR1 indicate which clocks are ready, and which clock is currently used as a system clock.

9.4.9 Handling clock generators in Stop and Standby modes

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped, as well as the following clock sources:

- HSI oscillator
- PSI oscillator
- HSE

Note: In Stop 0 mode, if HSIKERON or PSIKERON are set, HSI, PSI, or HSE may be kept running depending on clock output enable bits (HSISON, HSIDIV3ON, HSIKON, PSISON, PSIDIV3ON, PSIKON) and PSI reference clock source setting. in Stop 1 mode, HSIKERON, and PSIKERON have no effect.

The content of the RCC registers is not altered except for HSEON set to 0.

Exiting Stop mode

When the system exits this mode via a wake-up event, the application restarts on HSI oscillator and HSIS or HSIDIV3 clock depending on STOPWUCK bit value.

During Stop mode

There are two specific cases where the HSI can be enabled during this mode.

- When a dedicated peripheral requests the kernel clock the peripheral receives the HSIS, HSIK or HSIDIV3 according to the kernel clock source selected for this peripheral (using the RCC kernel clock configuration register RCC_CCIPRx)).
- When the HSIKERON bit in RCC_CR1 is set, the HSI is kept running under Stop 0 mode. If a clock is enabled thanks to HSISON, HSIDIV3ON or HSIKON it is available immediately when the system exits Stop 0 mode, or when a peripheral requests the kernel clock (see *Table 68* for details). The same applies to the PSI thanks to PSIKERON bit. In this case, its reference clock is also kept on.

Table 68. HSIKERON and PSIKERON behavior

HSIKERON (PSIKERON)	HSI (PSI) state during Stop mode	HSI (PSI) state setting time
0	Off	$t_{su(HSI)}$ $t_{su(PSI)}$ (1)
1	Running and gated if enabled	Immediate

1. $t_{su(HSI)}$ and $t_{su(PSI)}$ are the startup times of, respectively, the HSI and PSI oscillators (refer to the product datasheet for the values of these parameters).

When the microcontroller exits Standby mode, HSIDIV3 is selected as the system clock. The RCC registers are reset to their initial values, except for RCC_RSR and RCC_RTCCR.

Note: The HSI and PSI oscillators provide three clock paths respectively:

- One path for the system clock (hsis_ck or psis_ck)

- One path for the peripheral kernel clock (hsik_ck or psik_ck)

- One path for the peripheral kernel clock divided by 3 (hsidiv3_ck or psidiv3_ck)

When a peripheral requests the kernel clock in system Stop mode, only the path providing the hsik_ck, psik_ck or hsidiv3_ck, psidiv3_ck is activated.

9.4.10 Clock security system (CSS)

Clock security system on HSE

The clock security system can be enabled by software via the HSECSSON bit, which can be set even when the HSEON is cleared.

The CSS on HSE is enabled by the hardware when the HSE is enabled and ready, and HSECSSON set to 1.

The CSS on HSE is disabled when the HSE is disabled. It is not possible to clear directly the HSECSSON bit by software.

The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode.

If a failure is detected on the HSE clock while it is used as a system clock, the system automatically switches to the HSIDIV3, to provide a safe clock. The HSE is then automatically disabled, a clock failure event is sent to the break inputs of the advanced-control timer (TIM1, TIM8, TIM15, TIM16, and TIM17), and an NMI is automatically generated to inform the application about the failure, allowing the MCU to perform rescue operations. If the HSE output was used as clock source for the PSI reference clock when the failure occurred, the oscillator is also disabled.

If an HSE clock failure occurs when the CSS is enabled, the CSS generates an interrupt that causes the automatic generation of an NMI. The HSECSSF flag in RCC_CIFR is set to 1 to allow the application to identify the failure source. The NMI routine is executed indefinitely until the HSECSSF bit is cleared. As a consequence, the application must clear the HSECSSF flag in the NMI ISR by setting the HSECSSC bit in RCC_CICR.

Clock security system on LSE

A clock security system on LSE can be activated by software writing the LSECSSON bit in RCC_RTCCR. This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes. It also works under system reset (excluding power-on reset).

The clock security system on the LSE detects when the LSE disappears or in case of over frequency.

If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC, but no hardware action is made to the registers.

Note: If the LSECSS is enabled and the LSE clock fails, the LSECSSI occurs and an NMI is automatically generated. The NMI is executed infinitely unless the LSECSSF interrupt pending bit is cleared. It is therefore necessary that the NMI ISR clears the LSECSSI by setting the LSECSSC bit in the clock interrupt clear register (RCC_CICR).

In case of CSS on LSE detection event (LSECSSD = 1 in the RCC_RTCCR), the software must change the RTC clock source (no clock or LSI or HSE, with RTCSEL bitfield modification only possible while LSECSSD = 1) then disable the LSECSSON bit that clears the LSECSSD, stop the defective 32 kHz oscillator (disabling LSEON), or act to secure the application.

Refer to the datasheet for CSS on LSE electrical characteristics.

9.4.11 Clock output generation (MCO1/MCO2/LSCO)

Two microcontroller clock output pins (MCO1 and MCO2) are available. A clock source can be selected for each output. The selected clock can be divided thanks to a configurable prescaler (refer to *Figure 25* for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1[2:0], MCO2PRE[3:0], and MCO2[2:0], located in RCC_CFGR1.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The clock provided to the MCOs outputs must not exceed the maximum pin speed (refer to the product datasheet for information on the supported speed). Under Stop 0, only PSIK, HSIK, and PSIDIV3 can be output on MCOx if they are kept enabled thanks to HSIKERON or PSIKERON.

Another output (LSCO) allows one of the low-speed clocks (LSI, LSE) to be output onto the external LSCO pin. This output is available in Stop mode, not available in Standby mode. The selection is controlled by the LSCOSEL bit, and enabled by the LSCOEN bit in RCC_RTCCR.

The MCO clock output requires the corresponding alternate function selected on the MCO pin. The LSCO pin must be left in the default POR state.

9.4.12 Kernel clock selection

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- a clock domain synchronous with the register and bus interface (ckg_bus_perx clock)
- a clock domain generally synchronous with the peripheral (kernel clock)

The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose an optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral. The user application can thus change the bus frequency without reprogramming the peripherals. As an example, an ongoing transfer with UART is not disturbed if its APB clock is changed on-the-fly.

The table below shows the kernel clock that the RCC can deliver to the peripherals. Each row represents a multiplexer and the peripherals connected to its output.

Table 69. Kernel clock distribution overview

Peripherals	Clock multiplexer control bits	Bus clocks(1)	psis_ck	psidiv3_ck	psik_ck	hsis_ck	hsidiv3_ck	hsik_ck	hse_ck	lse_ck	lsi_ck	AUDIOCLK	hse_1M_ck	Disabled
XSPI1	XSPI1SEL	0(2)	-	-	1	-	-	2	-	-	-	-	-	-
FDCANx	FDCANSEL	0	1	-	2	-	-	-	3	-	-	-	-	-
I2Cx	I2CxSEL	0	-	-	1	-	-	2	-	-	-	-	-	-
I3C1	I3C1SEL	0	-	-	1	-	-	2	-	-	-	-	-	-
LPTIM1	LPTIM1SEL	0	-	-	-	-	-	1	-	2	3	-	-	-
SPI(I2S)x	SPIxSEL	0	-	-	1	-	-	2	-	-	-	3	-	-
USARTx	USARTxSEL	0	-	-	1	-	-	2	-	3	-	-	-	-
UARTx	UARTxSEL	0	-	-	1	-	-	2	-	3	-	-	-	-
LPUART1	LPUART1SEL	0	-	-	-	-	-	1	-	2	3	-	-	-
USB(CK48)	CK48SEL	-	-	1	-	-	2	-	3	-	-	-	-	0
ADCDAC	ADCDACSEL	0(2)	1	-	2	-	-	3	-	-	-	-	-	-

Table 69. Kernel clock distribution overview (continued)

Peripherals	Clock multiplexer control bits	Bus clocks(1)	psis_ck	psidiv3_ck	psik_ck	hsis_ck	hsidiv3_ck	hsik_ck	hse_ck	lse_ck	lsi_ck	AUDIOCLK	hse_1M_ck	Disabled
DAC	DACSEL	-	-	-	-	-	-	-	-	0	1	-	-	-
RTC/AWU	RTCSEL	-	-	-	-	-	-	-	-	1	2	-	3	0
ETH1_CLK	ETH1CLKSEL	-	1	-	2	-	-	-	3	-	-	-	-	0
ETH1_PTP	ETH1PTPCLKSEL	1	2	-	3	-	-	-	-	-	-	-	-	0

1. The bus (APB or AHB) clocks are the bus interface clocks to whom the peripherals are connected.
2. When SYSCLK is divided by 2 or more, the AHB prescaler does not maintain the 50% duty cycle required by the ADC and XSPI in DDR mode. To achieve a 50% duty cycle, the ADCDACPRE or XSPI peripheral prescalers must divide HCLK by a factor of 2 or higher.

Peripherals dedicated to control and data transfer

Peripherals such as SPIs, I2Cs, UARTs do not need a specific kernel clock frequency but a clock fast enough to generate the correct baud rate, or the required bit clock on the serial interface. For that purpose the source can be selected among the following ones:

- HSIS or HSIK if the clock must quickly be available upon Stop mode exit.
- PSIS or PSIK if an HSE clock is available and accuracy must be controlled over temperature or if a ratio of 100 MHz or 160 MHz is mandatory for some protocols.

Note: UARTs also can use the LSE clock when high baud rates are not required.

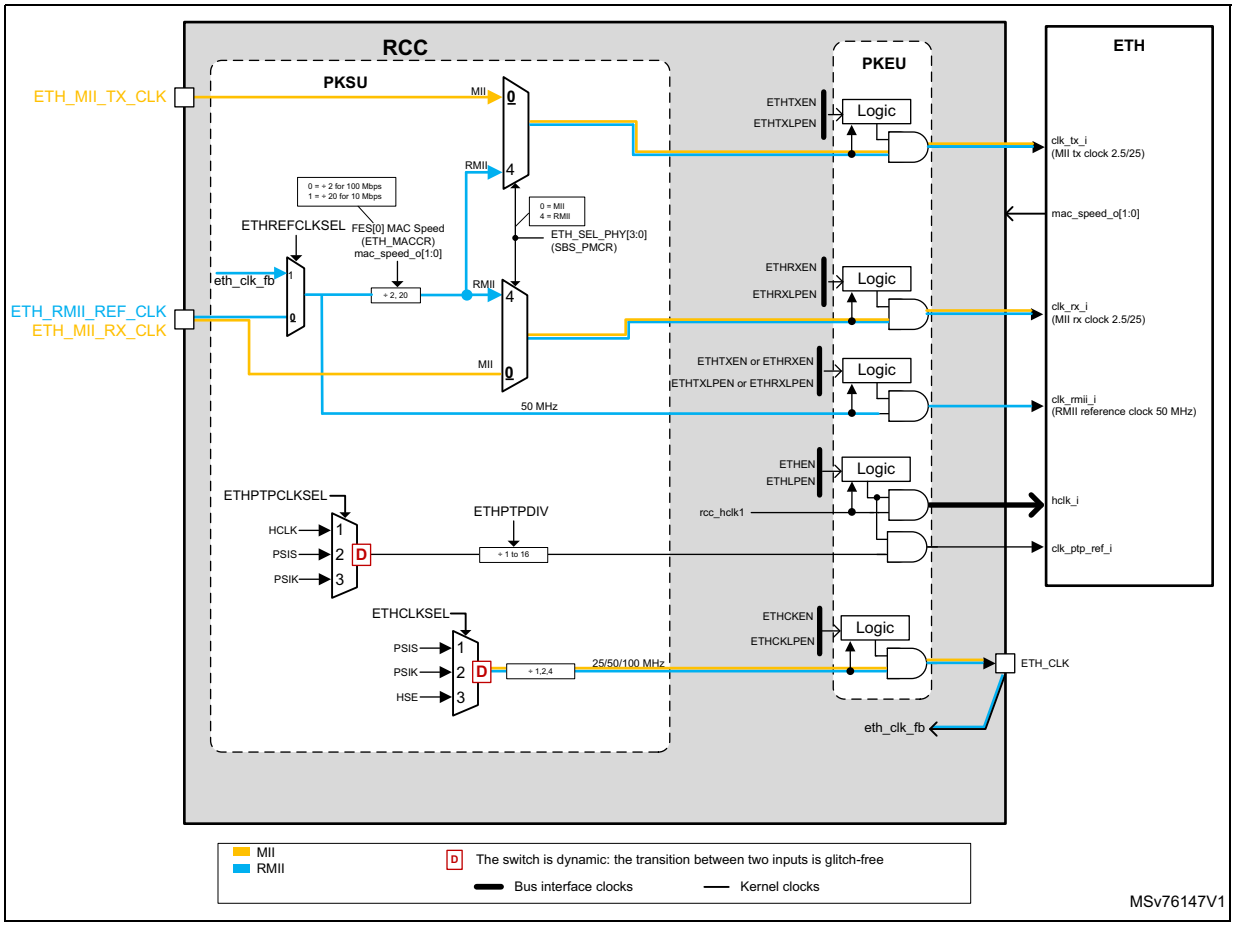
Clock distribution for ETH1

The Ethernet clocks provided by the RCC are available on ETH1_CLK pad. The application can select if the ETH1_CLK is generated from the PSIS, PSIK or from the HSE.

The RCC also provides the bus clock and the reference clock for the PTP function. The bus and PTP clocks generation are controlled via ETH1EN and ETH1LPEN bits.

The Ethernet transmit and receive clocks must be provided from an external Ethernet PHY.

Figure 27. Clock distribution for ETH



The signal ETH1_SEL_PHY is provided by the SBS block and defines the interface type:

- In MII mode (yellow path):
  - The transmit clock is received from the external PHY via the pad ETH_MII_TX_CLK.
  - The receive clock is received from the external PHY via the pad ETH_MII_RX_CLK.
  - The clock frequency is 2.5 (or 25) MHz for Ethernet at 10 (or 100) Mbps.
- In RMII mode (blue path):
  - The reference clock (50MHz) can be selected between two sources, this selection is controlled through ETH1REFCLKSEL.
  - The ETH_RMII_REF_CLK pad, if the reference clock is provided by the PHY.
  - The ETH1_CLK feedback clock (eth_clk_fb) if the RCC is providing the reference clock to the PHY.
- In RMII mode:
  - A clock must be provided to clk_rx_i and clk_tx_i inputs.
  - The clock frequency at clk_rx_i and clk_tx_i inputs is 2.5 MHz (division by 20) for Ethernet at 10 Mbps, and 25 MHz (division 2) for Ethernet at 100 Mbps.

- The transmit and receive clock frequency can be adjusted dynamically to 2.5 to 25 MHz according to the signal mac_speed_o[0] (controlled by FES bit in ETH_MACCR register) provided by the ETH block.
- The clock can be provided by RCC to be used by the external ETH PHY through ETH1_CLK output. This clock can be selected between three sources, this selection is controlled via ETH1CLKSEL.

Clock distribution for cryptographic subsystem

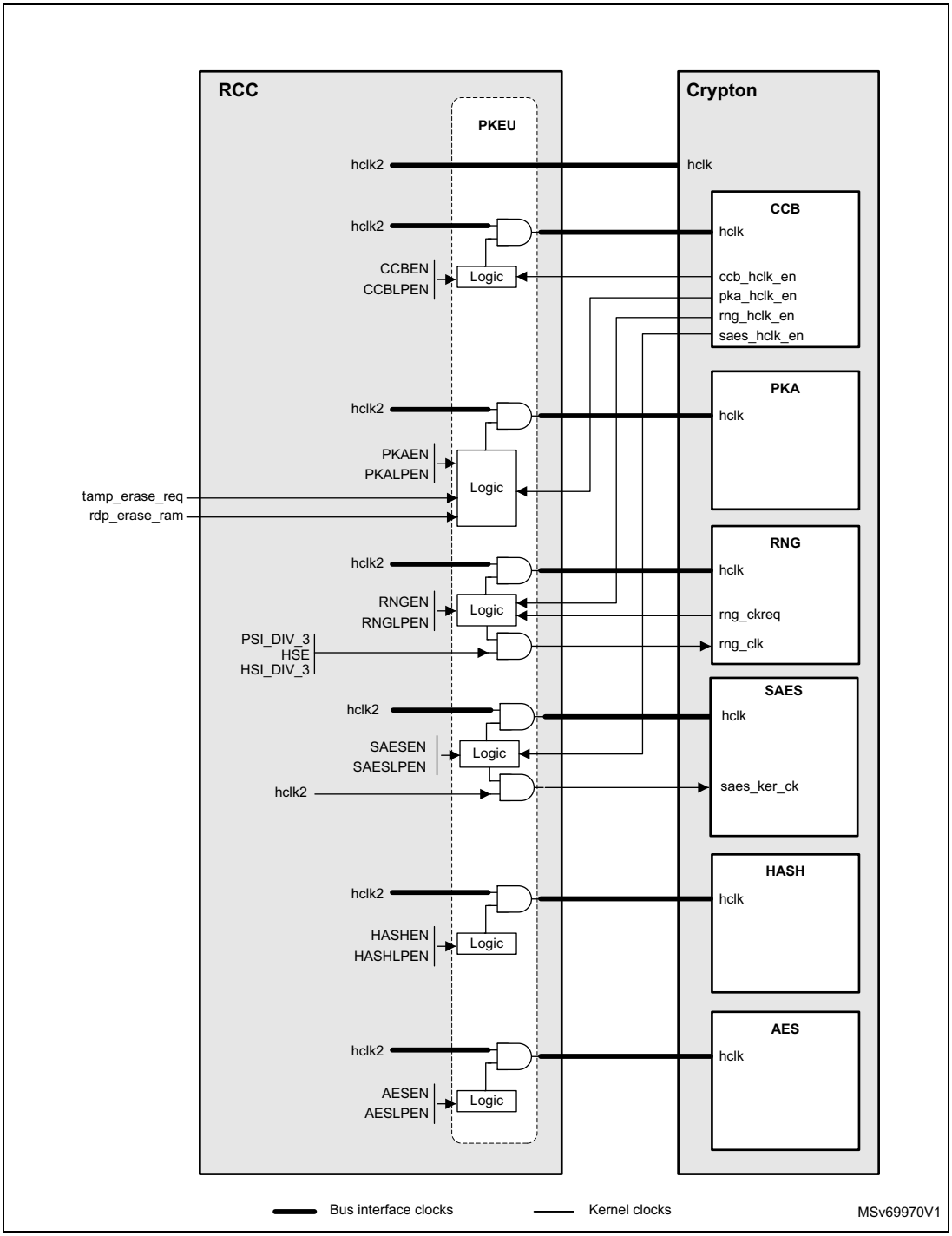
This subsection is only available for STM32C59x/5A3.

Figure 28 shows the clock distribution for the cryptographic subsystem.

Note: All the peripheral clocks of the cryptographic subsystem must be disabled before issuing a software reset to any CCB/PKA/RNG/SAES/HASH/AES peripheral of this subsystem.

For any CCB/PKA/RNG/SAES/HASH/AES peripheral of this subsystem, the software must ensure that the peripheral is not busy (by polling on the BUSY bit flag of the related CCB/PKA/RNG/SAES/HASH/AES peripheral) before disabling the peripheral clock.

Figure 28. Clock distribution for Crypton subsystem



RTC/AWU clock

The rtc_ck clock source can be one of the following:

- The hse_1M_ck (hse_ck divided by a programmable prescaler)
- The lse_ck
- The lsi_ck clock

The source clock is selected by programming the RTCSEL[1:0] bits in RCC_RTCCR and the RTCPRE[8:0] bits in RCC_CFGR1.

This selection cannot be modified without resetting the RTC domain.

The rtc_ck clock is enabled through the RTCEN bit in RCC_RTCCR.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLPEN bits located in RCC_APB3ENR/LPENR registers.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency ($F_{APB} < 7 \times F_{RTCLCK}$), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result than the first one. Otherwise, a third read access must be performed.

Watchdog clocks

The RCC provides the clock for the two watchdog blocks available on the circuit. The independent watchdog (IWDG) is connected to the LSI. The window watchdog (WWDG) is connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced on and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

Clock frequency measurement using TIMx

The HSI is not operating on a reference clock as the PSI. If the PSI cannot be the source to blocks requiring accurate frequency, it can be interesting to calibrate HSI thanks to timers or CRS. Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

- Calibrating the HSI with the LSE:

  The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI. This requires to use the HSI as the system clock source directly. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

  The basic concept consists in providing a relative measurement (such as HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

  The HSI oscillator have dedicated user-accessible calibration bits for this purpose (see RCC_CFGR1).

- Calibrating the LSI with the HSI:

  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process

deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

Clock frequency measurement using CRS

The HSI is as well directly connected to CRS block that can handle its TRIM automatically on HSE, LSE, USB SOF signal or external reference clock.

9.4.13 RTC and TAMP clock

The RTCCLK clock source is used by RTC and TAMP, and can be either the HSE_1MHz, LSE, or LSI clock. It is selected by programming the RTCSEL[1:0] bits in RCC_RTCCR. This selection cannot be modified without resetting the RTC domain. The system must always be configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC. The TAMP does not require any kernel clock if only the backup registers are used, with tampers in edge detection mode. All other tamper detection modes require a kernel clock (refer to *Section 40: Tamper and backup registers (TAMP)* for more details).

The LSE is in the RTC domain, whereas the HSE and LSI clocks are not.

Consequently, if the HSE clock divided by a prescaler is used as the RTC or TAMP clock, the RTC state is not guaranteed if the internal voltage regulator is powered off (removing power from the core domain) during Standby mode for example. Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC remains clocked and functional under system reset.

9.4.14 Timer clock

The timer clock frequencies (excluding LPTIM) are fixed and equal to HCLK.

9.4.15 Peripherals clock gating

Peripherals clock gating in Run mode

Each peripheral clock can be enabled by the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENR registers.

When the peripheral clock is not active, read or write accesses to the peripheral registers are not supported.

The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral. After the enable bit is set, there the clock is active after 2 cycles of the peripheral bus clock.

Caution: Just after enabling the clock for a peripheral, the software must wait for these 2 clock cycles before accessing the peripheral registers.

Peripherals clock gating in Sleep mode

When a peripheral is enabled, its clock can be automatically gated off when the device is in Sleep mode, by clearing the peripheral LPEN bit in the RCC_AHBxLPENR and

RCC_APBxLPENR registers. Both EN and LPEN bit of the peripheral must be set to keep the clock on in Sleep mode.

9.5 RCC privilege functional description

By default, after reset, all RCC registers can be read or written with both privileged and unprivileged access except RCC_PRIVCFGR that can be written with privileged access only. RCC_PRIVCFGR can be read by privileged and unprivileged access.

The PRIV bit in RCC_PRIVCFGR can be written with privileged access only. This bit configures the privileged access of all RCC functions.

When the PRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC bits is possible only with privileged access.
- The RCC bits can be read only with privileged access except RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

9.6 RCC low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU clock. The memory interface clocks (flash memory, cache, and all SRAM interfaces) can be stopped by software during Sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop 0 mode stops all the clocks in the core domain and disable the HSI, PSI, and HSE oscillators. However, HSI and PSI can be kept on (if HSIKERON or PSIKERON are set) or HSI can be started by a peripheral to generate a wake-up interrupt. LSI and LSE remain active in Stop 0 mode.
- Stop 1 mode stops all the clocks in the core domain and disable the HSI, PSI, and HSE oscillators. LSI and LSE remain active in Stop 1 mode.
- Standby mode stops all the clocks in the core domain and disable the HSI, the PSI, and HSE oscillators.

The CPU DeepSleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bit in the DBGMCU_CR register.

When exiting Stop mode, the system clock can be HSIS or HSIDIV3 based on the STOPWUCK value. The frequency (user trim) and the setting of the HSIK and PSIK prescaler are the one configured before entering Stop mode.

When leaving Standby mode, the system clock is HSIDIV3 (48 MHz). The user trim is lost.

If a flash memory programming operation is ongoing, a Stop or Standby mode entry is delayed until the flash memory interface access is finished. If an access to the APB domain is ongoing, a Stop or Standby mode entry is delayed until the APB access is finished.

9.7 RCC interrupts

The table below summarizes the interrupt sources and the way to control them.

Table 70. Interrupt sources and control

Interrupt vector	Interrupt event flag	Description	Enable control bits	Interrupt clear method	Exit Sleep mode	Exit Stop and Standby modes
RCC	LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYC to 1	Yes	No
	LSERDYF	LSE ready	LSERDYIE	Set LSERDYC to 1	Yes	No
	PSISRDYF	PSIS ready	PSISRDYIE	Set PSISRDYC to 1	Yes	No
	PSIDIV3RDYF	PSIDIV3 ready	PSIDIV3RDYIE	Set PSIDIV3RDYC to 1	Yes	No
	HSIKRDYF	HSIK ready	HSIKRDYIE	Set HSIKRDYC to 1	Yes	No
	HSISRDYF	HSIS ready	HSISRDYIE	Set HSISRDYC to 1	Yes	No
	HSIDIV3RDYF	HSIDIV3 ready	HSIDIV3RDYIE	Set HSIDIV3RDYC to 1	Yes	No
	HSIKRDYF	HSIK ready	HSIKRDYIE	Set HSIKRDYC to 1	Yes	No
	HSERDYF	HSE ready	HSERDYIE	Set HSERDYC to 1	Yes	No
TAMP	ITAMP3F(1)	LSE CSS failure	LSECSSON and ITAMP3E(1) and ITAMP3IE(1)	Set CITAMP3F(1) to 1	Yes	Yes
NMI	LSECSSF	LSE CSS failure	LSECSSON(2)	Set LSECSSC to 1	Yes	Yes
	HSECSSF	HSE CSS failure	HSECSSON(3)	Set HSECSSC to 1	Yes	No

1. The LSE CSS failure event (LSECSSD) is connected to TAMP internal tamper 3. To get the interrupt associated with this event, the internal tamper 3 must be enabled, and the internal tamper 3 interrupt must be enabled. The ITAMP3F, ITAMP3E, ITAMP3IE, and CITAMP3F bits are in the TAMP peripheral.
2. It is not possible to mask this interrupt when the security system feature is enabled (LSECSSON = 1).
3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

9.8 RCC registers

For the actual availability of some peripherals and their related bits, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them reserved, and keep them at the reset value.

9.8.1 RCC clock control register (RCC_CR1)

Address offset: 0x000

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON
											rw	rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PSIKRDY	PSIDIV3RDY	PSISRDY	PSIKERON	PSIKON	PSIDIV3ON	PSISON	Res.	HSIKRDY	HSIDIV3RDY	HSISRDY	HSIKERON	HSIKON	HSIDIV3ON	HSISON
	r	r	r	rw	rw	rw	rw		r	r	r	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HSEEXT**: External high speed clock type in Bypass mode

This bit is set and reset by software to select the external clock type (analog or digital). The external clock must be enabled with the HSEON bit to be used by the device. The HSEEXT bit can be written only if the HSE oscillator is disabled.

0: HSE in analog mode (default after reset)

1: HSE in digital mode

Bit 19 **HSECSSON**: HSE clock security system enable

This bit is set by software to enable clock security system on HSE. This bit is “set only” (disabled by a system reset or when the system enters in Standby mode). When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.

0: CSS on HSE off (clock detector off) (default after reset)

1: CSS on HSE on (clock detector on if the HSE oscillator is stable, off if not).

Bit 18 **HSEBYP**: HSE clock bypass

This bit is set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE oscillator not bypassed (default after reset)

1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

This bit is set by hardware to indicate that the HSE oscillator is stable.

0: HSE clock not ready (default after reset)

1: HSE clock ready

Bit 16 **HSEON**: HSE clock enable

This bit is set and cleared by software. It is cleared by hardware to stop the HSE when entering Stop or Standby mode. This bit cannot be cleared if the HSE is used directly or indirectly (via SW mux) as system clock, or if the HSE is selected as reference clock for PSI with PSI enabled (PSISON bit or PSIDIV3ON bit or PSIKON bit set to 1).

0: HSE off (default after reset)

1: HSE on

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PSIKRDY**: PSIK clock ready flag

This bit is set by hardware to indicate that the PSI oscillator divided clock is stable.

0: PSIK clock not ready (default after reset)

1: PSIK clock ready

Bit 13 **PSIDIV3RDY**: PSIDIV3 clock ready flag

This bit is set by hardware to indicate that the PSI oscillator divided by 3 clock is stable.

0: PSIDIV3 clock not ready (default after reset)

1: PSIDIV3 clock ready

Bit 12 **PSISRDY**: PSIS clock ready flag

This bit is set by hardware to indicate that the PSIS clock is stable.

0: PSIS clock not ready (default after reset)

1: PSIS clock ready

Bit 11 **PSIKERON**: PSI clock enable in Stop mode

This bit is set and cleared by software to force maintain PSI enabled in Stop 0 mode, to be quickly available as kernel clock for some peripherals. This bit only impacts PSI if PSISON, PSIDIV3ON, or PSIKON is set.

0: No effect on PSI (default after reset)

1: PSI keep ON even in Stop 0 mode

Bit 10 **PSIKON**: PSIK clock enable

This bit is set and reset by software to enable/disable PSIK clock for peripheral. This bit value is kept under Stop 0 if PSIKERON is set, otherwise it is cleared.

0: PSIK off (default after reset)

1: PSIK on

Bit 9 **PSIDIV3ON**: PSIDIV3 clock enable

This bit is set and reset by software to enable/disable PSIDIV3 clock for peripheral.
This bit value is kept under Stop 0 if PSIKERON is set, otherwise it is cleared.

0: PSIDIV3 off (default after reset)

1: PSIDIV3 on

Bit 8 **PSISON**: PSIS clock enable

This bit is set and reset by software to enable/disable PSIS clock for peripheral and system. This bit value is kept under Stop 0 if PSIKERON is set, otherwise it is cleared. This bit cannot be cleared if the PSIS is used (via SW mux) as system clock.

0: PSIS off (default after reset)

1: PSIS on

Bit 7 Reserved, must be kept at reset value.

Bit 6 **HSIKRDY**: HSIK clock ready flag

This bit is set by hardware to indicate that the HSIK divided clock is stable.

0: HSIK clock not ready (default after reset)

1: HSIK clock ready

Bit 5 **HSIDIV3RDY**: HSIDIV3 clock ready flag

This bit is set by hardware to indicate that the HSI oscillator divided by 3 clock is stable.

0: HSIDIV3 clock not ready (default after reset)

1: HSIDIV3 clock ready

Bit 4 **HSISRDY**: HSIS clock ready flag

This bit is set by hardware to indicate that the HSIS clock is stable.

0: HSIS clock not ready (default after reset)

1: HSIS clock ready

Bit 3 **HSIKERON**: HSI clock enable in Stop mode

This bit is set and cleared by software to maintain HSI enabled in Stop 0 mode, to be quickly available as kernel clock for some peripherals. This bit only impact HSI if HSISON, HSIDIV3ON or HSIKON are set.

0: No effect on HSI (default after reset)

1: HSI keep ON in Stop 0 mode

Bit 2 **HSIKON**: HSIK clock enable

This bit is set and reset by software to enable/disable HSIK clock for peripheral. This bit value is kept under Stop 0 mode if HSIKERON is set, otherwise it is cleared.

0: HSIK off (default after reset)

1: HSIK on

Bit 1 **HSIDIV3ON**: HSIDIV3 clock enable

This bit is set and reset by software to enable/disable HSIDIV3 clock for system or peripheral. It is set by hardware to force the HSIDIV3 to on when product exit Standby mode, or in case of a failure of the HSE which is used directly or indirectly as the system clock source. This bit cannot be cleared if the HSIDIV3 is used directly (via SW mux) as system clock. This bit value is kept under Stop 0 mode if HSIKERON is set, otherwise it is cleared. This bit is set by hardware to force the HSIDIV3 to on when the system leaves Stop mode, if STOPWUCK = 0.

0: HSIDIV3 off

1: HSIDIV3 on (default after reset)

Bit 0 **HSISON**: HSIS clock enable

This bit is set and cleared by software to enable/disable HSIS clock for system and peripheral. This bit is set by hardware to force the HSI to on when the product leaves Stop mode, if STOPWUCK = 1. This bit value is kept under Stop 0 mode if HSIKERON is set, otherwise it is cleared. This bit cannot be cleared if the HSIS is used directly or indirectly (via SW mux) as system clock.

0: HSIS off (default after reset)

1: HSIS on

9.8.2 RCC clock control register (RCC_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PSIFREQ[1:0]		Res.	Res.	Res.	Res.	Res.	PSIREF[2:0]			Res.	Res.	PSIREFSRC[1:0]	
		rw	rw						rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PSIKDIV[3:0]				Res.	Res.	Res.	Res.	HSIKDIV[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **PSIFREQ[1:0]**: PSI target frequency configuration

This bitfield is set and reset by software to configure the PSI oscillator target frequency. Each bit can be written only if the PSI oscillator is disabled.

00: PSI nominal frequency = 100 MHz

01: PSI nominal frequency = 144 MHz

1x: PSI nominal frequency = 160 MHz

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:20 **PSIREF[2:0]**: PSI reference clock frequency selection

This bitfield is set and cleared by software to indicate the adequate reference frequency connected to the PSI oscillator. Each bit can be written only if the PSI oscillator is disabled.

000: 32.768 kHz

001: 8 MHz

010: 16 MHz

011: 24 MHz

100: 25 MHz

101: 32 MHz

110: 48 MHz

111: 50 MHz

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **PSIREFSRC[1:0]**: PSI reference clock source selection

This bitfield is set and cleared by software to indicate the adequate reference clock source connected to the PSI oscillator. Each bit can be written only if the PSI oscillator is disabled.

00: HSE clock used as reference for PSI

01: LSE clock used as reference for PSI

10: HSI clock divided by 18 used as reference for PSI

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **PSIKDIV[3:0]**: PSI clock out divider factor

This bitfield is set and cleared by software to configure the PSIK clock based on PSI and its division factor.

0000: PSI clock divided by 1
0001: PSI clock divided by 1
0010: PSI clock divided by 1.5
0011: PSI clock divided by 2
0100: PSI clock divided by 2.5
....
1110: PSI clock divided by 7.5
1111: PSI clock divided by 8

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **HSIKDIV[3:0]**: HSI clock out divider factor

This bitfield is set and cleared by software to configure the HSIK clock based on HSI and its division factor.

0000: HSI clock divided by 1
0001: HSI clock divided by 1
0010: HSI clock divided by 1.5
0011: HSI clock divided by 2
0100: HSI clock divided by 2.5
....
1110: HSI clock divided by 7.5
1111: HSI clock divided by 8

9.8.3 RCC clock configuration register1 (RCC_CFGR1)

Address offset: 0x01C

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2; word and half-word access

One or two wait states are inserted only if the access occurs during the clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]				Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCPRE[8:0]									STOP WUCK	Res.	SWS[1:0]		Res.	SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		r	r		rw	rw

Bits 31:29 **MCO2SEL[2:0]**: Microcontroller clock output 2

This bitfield is set and cleared by software. Clock source selection may generate glitches on MCO2. It is highly recommended to configure these bits only after reset, before enabling the external oscillators.

000: system clock selected (sys_ck) (default after reset)

001: HSE clock selected (hse_ck)

010: LSE clock selected (lse_ck)

011: LSI clock selected (lsi_ck)

100: PSIK clock selected (psik_ck)

101: HSIK clock selected (hsik_ck)

110: PSIDIV3 clock selected (psidiv3_ck)

111: HSIDIV3 clock selected (hsidiv3_ck): if this clock is retained by HSIKERON, it is not visible on MCO output under the Stop 0 mode.

Other: Reserved

Bits 28:25 **MCO2PRE[3:0]**: MCO2 prescaler

This bitfield is set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

0000: Prescaler disabled (default after reset)

0001: Division by 1 (bypass)

0010: Division by 2

0011: Division by 3

0100: Division by 4

...

1111: Division by 15

Bits 24:22 **MCO1SEL[2:0]**: Microcontroller clock output 1

This bitfield is set and cleared by software. Clock source selection may generate glitches on MCO1. It is highly recommended to configure these bits only after reset, before enabling the external oscillators.

000: System clock selected (sys_ck) (default after reset)

001: HSE clock selected (hse_ck)

010: LSE clock selected (lse_ck)

011: LSI clock selected (lsi_ck)

100: PSIK clock selected (psik_ck)

101: HSIK clock selected (hsik_ck)

110: PSIS clock selected (psis_ck), if this clock is retained by PSIKERON, it is not visible on MCO output under the Stop 0 mode.

111: HSIS clock selected (hsis_ck), if this clock is retained by HSIKERON, it is not visible on MCO output under the Stop 0 mode.

Other: Reserved

Bits 21:18 **MCO1PRE[3:0]**: MCO1 prescaler

This bitfield is set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

0000: Prescaler disabled (default after reset)

0001: Division by 1 (bypass)

0010: Division by 2

0011: Division by 3

0100: Division by 4

...

1111: Division by 15

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:7 **RTCPRE[8:0]**: HSE division factor for RTC clock (source of HSE_1MHz clock)

This bitfield is set and cleared by software to divide the HSE to generate a clock for RTC.

Caution: Software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.

000000000: No clock (default after reset)

000000001: No clock

000000010: HSE/2

000000011: HSE/3

000000100: HSE/4

...

111111110: HSE/510

111111111: HSE/511

Bit 6 **STOPWUCK**: System clock selection after a wake-up from system Stop mode

This bit is set and reset by software to select the system wake-up clock from system Stop.

0: HSIDIV3 selected as wake-up clock from system Stop mode (default after reset)

1: HSIS selected as wake-up clock from system Stop mode

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **SWS[1:0]**: System clock switch status

This bitfield is set and reset by hardware to indicate which clock source is used as system clock.

00: HSIDIV3 used as system clock (hsidiv3_ck) (default after reset).

01: HSIS used as system clock (hsis_ck)

10: HSE used as system clock (hse_ck)

11: PSIS used as system clock (psis_ck)

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SW[1:0]**: System clock and trace clock switch

This bitfield is set and reset by software to select system clock and trace clock sources (sys_ck). It is set by hardware to force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system Stop mode, and to force the selection of the HSI in case of failure of the HSE when used directly or indirectly as system clock

00: HSIDIV3 selected as system clock (hsidiv3_ck) (default after reset)

01: HSIS selected as system clock (hsis_ck)

10: HSE selected as system clock (hse_ck)

11: PSIS selected as system clock (psis_ck)

9.8.4 RCC CPU domain clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

Reset value: 0x0000 0000

1 or 2 wait states are inserted only if the access occurs during the clock source switch.

From 0 to 15 wait states are inserted if the access occurs when the APB or AHB prescalers values update is ongoing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	APB2DIS	APB1DIS	AHB4DIS	Res.	AHB2DIS	AHB1DIS
									rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PPRE3[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **APB3DIS**: APB3 clock disable value.Set and cleared by software

This bit can be set to further reduce power consumption, when none of the APB3 peripherals are used, and when their clocks are disabled in RCC_APB3ENR. When this bit is set, all the APB3 peripheral clocks are off.

0: APB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB3 clock disabled

Bit 21 **APB2DIS**: APB2 clock disable value

This bit can be set to further reduce power consumption, when none of the APB2 peripherals are used, and when their clocks are disabled in RCC_APB2ENR. When this bit is set, all the APB2 peripheral clocks are off.

0: APB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB2 clock disabled

Bit 20 **APB1DIS**: APB1 clock disable value

This bit can be set to further reduce power consumption, when none of the APB1 peripherals (except IWDG) are used, and when their clocks are disabled in RCC_APB1ENR. When this bit is set, all the APB1 peripheral clocks are off, except for IWDG.

0: APB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB1 clock disabled

Bit 19 **AHB4DIS**: AHB4 clock disable value

This bit can be set to further reduce power consumption, when none of the AHB4 peripherals from RCC_AHB4ENR are used, and when their clocks are disabled in RCC_AHB4ENR. When this bit is set, all the AHB4 peripheral clocks are off.

0: AHB4 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB4 clock disabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **AHB2DIS**: AHB2 clock disable

This bit can be set to further reduce power consumption, when none of the AHB2 peripherals from RCC_AHB2ENR are used, and when their clocks are disabled in RCC_AHB2ENR. When this bit is set, all the AHB2 peripheral clocks are off.

0: AHB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB2 clock disabled

Bit 16 **AHB1DIS**: AHB1 clock disable

This bit can be set to further reduce power consumption, when none of the AHB1 peripherals from RCC_AHB1ENR are used, and when their clocks are disabled in RCC_AHB1ENR. When this bit is set, all the AHB1 peripheral clocks are off, except for FLASH, ICACHE, SRAM1, and SRAM2.

0: AHB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB1 clock disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PPRE3[2:0]**: APB low-speed prescaler (APB3)

This bitfield is set and reset by software to control APB low-speed clocks division factor. The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE3 write.

0xx: rcc_pclk3 = rcc_hclk1

100: rcc_pclk3 = rcc_hclk1 / 2

101: rcc_pclk3 = rcc_hclk1 / 4

110: rcc_pclk3 = rcc_hclk1 / 8

111: rcc_pclk3 = rcc_hclk1 / 16

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE2[2:0]**: APB high-speed prescaler (APB2)

This bitfield is set and reset by software to control APB high-speed clocks division factor. The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE2 write.

0xx: rcc_pclk2 = rcc_hclk1

100: rcc_pclk2 = rcc_hclk1 / 2

101: rcc_pclk2 = rcc_hclk1 / 4

110: rcc_pclk2 = rcc_hclk1 / 8

111: rcc_pclk2 = rcc_hclk1 / 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE1[2:0]**: APB low-speed prescaler (APB1)

This bitfield is set and reset by software to control the division factor of rcc_pclk1. The clock is divided by the new prescaler factor from 1 to 16 cycles of rcc_hclk after PPRE write.

0xx: rcc_pclk1 = rcc_hclk1 (default after reset)

100: rcc_pclk1 = rcc_hclk1 / 2

101: rcc_pclk1 = rcc_hclk1 / 4

110: rcc_pclk1 = rcc_hclk1 / 8

111: rcc_pclk1 = rcc_hclk1 / 16

Bits 3:0 **HPRE[3:0]**: AHB prescaler

This bitfield is set and reset by software to control the division factor of rcc_hclk. Changing this division ratio has an impact on the frequency of all bus matrix clocks

0xxx: rcc_hclk = sys_ck (default after reset)

1000: rcc_hclk = sys_ck / 2

1001: rcc_hclk = sys_ck / 4

1010: rcc_hclk = sys_ck / 8

1011: rcc_hclk = sys_ck / 16

1100: rcc_hclk = sys_ck / 64

1101: rcc_hclk = sys_ck / 128

1110: rcc_hclk = sys_ck / 256

1111: rcc_hclk = sys_ck / 512

9.8.5 RCC clock source interrupt enable register (RCC_CIER)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDYIE	PSIKRDYIE	PSIDIV3RDYIE	PSISRDYIE	HSIKRDYIE	HSIDIV3RDYIE	HSISRDYIE	LSERDYIE	LSIRDYIE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **HSERDYIE**: HSE ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.

0: HSE ready interrupt disabled (default after reset)

1: HSE ready interrupt enabled

Bit 7 **PSIKRDYIE**: PSIK ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the PSIK clock stabilization.

0: PSIK ready interrupt disabled (default after reset)

1: PSIK ready interrupt enabled

Bit 6 **PSIDIV3RDYIE**: PSIDIV3 ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the PSIDIV3 clock stabilization.

0: PSIDIV3 ready interrupt disabled (default after reset)

1: PSIDIV3 ready interrupt enabled

Bit 5 **PSISRDYIE**: PSIS ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the PSIS clock stabilization.

0: PSI ready interrupt disabled (default after reset)

1: PSI ready interrupt enabled

Bit 4 **HSIKRDYIE**: HSIK ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the HSIK clock stabilization.

0: HSIK ready interrupt disabled (default after reset)

1: HSIK ready interrupt enabled

Bit 3 **HSIDIV3RDYIE**: HSIDIV3 ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the HSIDIV3 clock stabilization.

0: HSIDIV3 ready interrupt disabled (default after reset)

1: HSIDIV3 ready interrupt enabled

Bit 2 **HSISRDYIE**: HSIS ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the HSIS clock stabilization.

0: HSI ready interrupt disabled (default after reset)

1: HSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.

0: LSE ready interrupt disabled (default after reset)

1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.

0: LSI ready interrupt disabled (default after reset)

1: LSI ready interrupt enabled

9.8.6 RCC clock source interrupt flag register (RCC_CIFR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LSECSSF	HSECSSF	Res.	HSERDYF	PSIKRDYF	PSIDIV3RDYF	PSISRDYF	HSIKRDYF	HSIDIV3RDYF	HSISRDYF	LSERDYF	LSIRDYF
				r	r		r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LSECSSF**: LSE clock security system interrupt flag
Reset by software by writing LSECSSC bit. Set by hardware in case of LSE clock failure.
0: No clock security interrupt caused by LSE clock failure (default after reset)
1: Clock security interrupt caused by LSE clock failure

Bit 10 **HSECSSF**: HSE clock security system interrupt flag
Reset by software by writing HSECSSC bit. Set by hardware in case of HSE clock failure.
0: No clock security interrupt caused by HSE clock failure (default after reset)
1: Clock security interrupt caused by HSE clock failure

Bit 9 Reserved, must be kept at reset value.

Bit 8 **HSERDYF**: HSE ready interrupt flag
Reset by software by writing HSERDYC bit. Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE

Bit 7 **PSIKRDYF**: PSIK ready interrupt flag
Reset by software by writing PSIKRDYC bit.
Set by hardware when the PSIK clock becomes stable and PSIKDYIE is set.
0: No clock ready interrupt caused by PSIK (default after reset)
1: Clock ready interrupt caused by PSIK

Bit 6 **PSIDIV3RDYF**: PSIDIV3 ready interrupt flag
Reset by software by writing PSIDIV3RDYC bit.
Set by hardware when the PSIDIV3 clock becomes stable and PSIDIV3DYIE is set.
0: No clock ready interrupt caused by PSIDIV3 (default after reset)
1: Clock ready interrupt caused by PSIDIV3

Bit 5 **PSISRDYF**: PSIS ready interrupt flag
Reset by software by writing PSISRDYC bit.
Set by hardware when the PSIS clock becomes stable and PSISDYIE is set.
0: No clock ready interrupt caused by PSIS (default after reset)
1: Clock ready interrupt caused by PSIS

Bit 4 **HSIKRDYF**: HSIK ready interrupt flag
Reset by software by writing HSIKRDYC bit.
Set by hardware when the HSIK clock becomes stable and HSIKDYIE is set.
0: No clock ready interrupt caused by HSIK (default after reset)
1: Clock ready interrupt caused by HSIK

Bit 3 **HSIDIV3RDYF**: HSIDIV3 ready interrupt flag
Reset by software by writing HSIDIV3RDYC bit.
Set by hardware when the HSIDIV3 clock becomes stable and HSIDIV3DYIE is set.
0: No clock ready interrupt caused by HSIDIV3 (default after reset)
1: Clock ready interrupt caused by HSIDIV3

Bit 2 **HSISRDYF**: HSIS ready interrupt flag
Reset by software by writing HSISRDYC bit.
Set by hardware when the HSIS clock becomes stable and HSISDYIE is set.
0: No clock ready interrupt caused by HSIS (default after reset)
1: Clock ready interrupt caused by HSIS

Bit 1 **LSERDYF**: LSE ready interrupt flag
Reset by software by writing LSERDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
0: No clock ready interrupt caused by the LSI (default after reset)
1: Clock ready interrupt caused by the LSI

9.8.7 RCC clock source interrupt clear register (RCC_CICR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LSECSSC	HSECSSC	Res.	HSERDYC	PSIKRDYC	PSIDIV3RDYC	PSISRDYC	HSIKRDYC	HSIDIV3RDYC	HSISRDYC	LSERDYC	LSIRDYC
				rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LSECSSC**: LSE clock security system interrupt clear
Set by software to clear LSECSSF. Reset by hardware when clear done.
0: LSECSSF no effect (default after reset)
1: LSECSSF cleared

Bit 10 **HSECSSC**: HSE clock security system interrupt clear
Set by software to clear HSECSSF. Reset by hardware when clear done.
0: HSECSSF no effect (default after reset)
1: HSECSSF cleared

Bit 9 Reserved, must be kept at reset value.

Bit 8 **HSERDYC**: HSE ready interrupt clear
Set by software to clear HSERDYF. Reset by hardware when clear done.
0: HSERDYF no effect (default after reset)
1: HSERDYF cleared

Bit 7 **PSIKRDYC**: PSIK ready interrupt clear
Set by software to clear PSIKRDYF. Reset by hardware when clear done.
0: PSIKRDYF no effect (default after reset)
1: PSIKRDYF cleared

Bit 6 **PSIDIV3RDYC**: PSIDIV3 ready interrupt clear
Set by software to clear PSIDIV3RDYF. Reset by hardware when clear done.
0: PSIDIV3RDYF no effect (default after reset)
1: PSIDIV3RDYF cleared

Bit 5 **PSISRDYC**: PSIS ready interrupt clear
Set by software to clear PSISRDYF. Reset by hardware when clear done.
0: PSISRDYF no effect (default after reset)
1: PSISRDYF cleared

Bit 4 **HSIKRDYC**: HSIK ready interrupt clear
Set by software to clear HSIKRDYF. Reset by hardware when clear done.
0: HSIKRDYF no effect (default after reset)
1: HSIKRDYF cleared

Bit 3 **HSIDIV3RDYC**: HSIDIV3 ready interrupt clear
Set by software to clear HSIDIV3RDYF. Reset by hardware when clear done.
0: HSIDIV3RDYF no effect (default after reset)
1: HSIDIV3RDYF cleared

Bit 2 **HSISRDYC**: HSIS ready interrupt clear
Set by software to clear HSIRDYF. Reset by hardware when clear done.
0: HSISRDYF no effect (default after reset)
1: HSISRDYF cleared

Bit 1 **LSERDYC**: LSE ready interrupt clear
Set by software to clear LSERDYF. Reset by hardware when clear done.
0: LSERDYF no effect (default after reset)
1: LSERDYF cleared

Bit 0 **LSIRDYC**: LSI ready interrupt clear
Set by software to clear LSIRDYF. Reset by hardware when clear done.
0: LSIRDYF no effect (default after reset)
1: LSIRDYF cleared

9.8.8 RCC AHB1 reset register (RCC_AHB1RSTR)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RST	Res.	RAMCFGRST	Res.
												rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CORDICRST	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA2RST	LPDMA1RST
	rw		rw											rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **ETH1RST**: ETHERNET reset
Set and reset by software.
0: ETHERNET not reset (default after reset)
1: ETHERNET reset

Bit 18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGRST**: RAMCFG reset
Set and reset by software.
0: RAMCFG not reset (default after reset)
1: RAMCFG reset

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **CORDICRST**: CORDIC reset
Set and reset by software.
0: CORDIC not reset (default after reset)
1: CORDIC reset

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset
Set and reset by software.
0: CRC not reset (default after reset)
1: CRC reset

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **LPDMA2RST**: LPDMA2 reset
Set and reset by software.
0: LPDMA2 not reset (default after reset)
1: LPDMA2 reset

Bit 0 **LPDMA1RST**: LPDMA1 reset
Set and reset by software.
0: LPDMA1 not reset (default after reset)
1: LPDMA1 reset

9.8.9 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3RST	Res.	Res.	CCBRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST
							rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1RST	ADC12RST	Res.	Res.	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
				rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC3RST**: ADC3 reset
Set and reset by software.
0: ADC3 not reset (default after reset)
1: ADC3 reset

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCBRST**: CCB reset
Set and reset by software.
0: CCB not reset (default after reset)
1: CCB reset

Bit 20 **SAESRST**: SAES reset
Set and reset by software.
0: SAES not reset (default after reset)
1: SAES reset

Bit 19 **PKARST**: PKA reset
Set and reset by software.
0: PKA not reset (default after reset)
1: PKA reset

Bit 18 **RNGRST**: RNG reset
Set and reset by software.
0: RNG not reset (default after reset)
1: RNG reset

Bit 17 **HASHRST**: HASH reset
Set and reset by software.
0: HASH not reset (default after reset)
1: HASH reset

Bit 16 **AESRST**: AES reset
Set and reset by software.
0: AES not reset (default after reset)
1: AES reset

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DAC1RST**: DAC reset
Set and reset by software.
0: DAC1 not reset (default after reset)
1: DAC1 reset

Bit 10 **ADC12RST**: ADC1 and ADC2 reset
Set and reset by software.
0: ADC1 and ADC2 not reset (default after reset)
1: ADC1 and ADC2 reset

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST**: GPIOH reset
Set and reset by software.
0: GPIOH not reset (default after reset)
1: GPIOH reset

Bit 6 **GPIOGRST**: GPIOG reset
Set and reset by software.
0: GPIOG not reset (default after reset)
1: GPIOG reset

Bit 5 **GPIOFRST**: GPIOF reset
Set and reset by software.
0: GPIOF not reset (default after reset)
1: GPIOF reset

Bit 4 **GPIOERST**: GPIOE reset
Set and reset by software.
0: GPIOE not reset (default after reset)
1: GPIOE reset

Bit 3 **GPIODRST**: GPIOD reset
Set and reset by software.
0: e GPIOD not reset (default after reset)
1: GPIOD reset

Bit 2 **GPIOCRST**: GPIOC reset
Set and reset by software.
0: GPIOC not reset (default after reset)
1: GPIOC reset

Bit 1 **GPIOBRST**: GPIOB reset
Set and reset by software.
0: GPIOB not reset (default after reset)
1: GPIOB reset

Bit 0 **GPIOARST**: GPIOA reset
Set and reset by software.
0: GPIOA not reset (default after reset)
1: GPIOA reset

9.8.10 RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1RST	Res.	Res.	Res.	Res.
											rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **XSPI1RST**: XSPI1 reset
Set and reset by software.
0: XSPI1 not reset (default after reset)
1: XSPI1 reset

Bits 19:0 Reserved, must be kept at reset value.

9.8.11 RCC APB1 peripheral low reset register (RCC_APB1LRSTR)

Address offset: 0x074

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	UART7RST	Res.	Res.	Res.	Res.	USART6RST	CRSRST	I3C1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.
	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	OPAMP1RST	Res.	Res.	Res.	Res.	Res.	Res.	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **UART7RST**: UART7 reset
Set and reset by software.
0: UART7 not reset (default after reset)
1: UART7 reset

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **USART6RST**: USART6 reset
Set and reset by software.
0: USART6 not reset (default after reset)
1: USART6 reset

Bit 24 **CRSRST**: CRS reset
Set and reset by software.
0: CRS not reset (default after reset)
1: CRS reset

Bit 23 **I3C1RST**: I3C1 block reset
Set and reset by software.
0: I3C1 not reset (default after reset)
1: I3C1 reset

Bit 22 **I2C2RST**: I2C2 reset
Set and reset by software.
0: I2C2 not reset (default after reset)
1: I2C2 reset

Bit 21 **I2C1RST**: I2C1 reset
Set and reset by software.
0: I2C1 not reset (default after reset)
1: I2C1 reset

Bit 20 **UART5RST**: UART5 reset
Set and reset by software.
0: UART5 not reset (default after reset)
1: UART5 reset

Bit 19 **UART4RST**: UART4 reset
Set and reset by software.
0: UART4 not reset (default after reset)
1: UART4 reset

Bit 18 **USART3RST**: USART3 reset
Set and reset by software.
0: USART3 not reset (default after reset)
1: USART3 reset

Bit 17 **USART2RST**: USART2 reset
Set and reset by software.
0: USART2 not reset (default after reset)
1: USART2 reset

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST**: SPI3 reset
Set and reset by software.
0: SPI3 not reset (default after reset)
1: SPI3 reset

Bit 14 **SPI2RST**: SPI2 reset
Set and reset by software.
0: SPI2 not reset (default after reset)
1: SPI2 reset

Bit 13 **OPAMP1RST**: OPAMP1 reset
Set and reset by software.
0: OPAMP1 not reset (default after reset)
1: OPAMP1 reset

Bits 12:7 Reserved, must be kept at reset value.

Bit 6 **TIM12RST**: TIM12 reset
Set and reset by software.
0: TIM12 not reset (default after reset)
1: TIM12 reset

Bit 5 **TIM7RST**: TIM7 reset
Set and reset by software.
0: TIM7 not reset (default after reset)
1: TIM7 reset

Bit 4 **TIM6RST**: TIM6 reset
Set and reset by software.
0: TIM6 not reset (default after reset)
1: TIM6 reset

Bit 3 **TIM5RST**: TIM5 reset
Set and reset by software.
0: TIM5 not reset (default after reset)
1: TIM5 reset

Bit 2 **TIM4RST**: TIM4 reset
Set and reset by software.
0: TIM4 not reset (default after reset)
1: TIM4 reset

Bit 1 **TIM3RST**: TIM3 reset
Set and reset by software.
0: TIM3 not reset (default after reset)
1: TIM3 reset

Bit 0 **TIM2RST**: TIM2 reset
Set and reset by software.
0: TIM2 not reset (default after reset)
1: TIM2 reset

9.8.12 RCC APB1 peripheral high reset register (RCC_APB1HRSTR)

Address offset: 0x078

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN RST	Res.	Res.	Res.	Res.	Res.	COMP RST	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FDCANRST**: FDCAN1 and FDCAN2 reset
Set and reset by software.
0: FDCAN1 and FDCAN2 not reset (default after reset)
1: FDCAN1 and FDCAN2 reset

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **COMPRST**: COMP reset
Set and reset by software.
0: COMP not reset (default after reset)
1: COMP reset

Bits 2:0 Reserved, must be kept at reset value.

9.8.13 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBRST	Res.	Res.	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST
							rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBRST**: USB reset
Set and reset by software.
0: USB not reset (default after reset)
1: USB reset

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST**: TIM17 reset
Set and reset by software.
0: TIM17 not reset (default after reset)
1: TIM17 reset

Bit 17 **TIM16RST**: TIM16 reset
Set and reset by software.
0: TIM16 not reset (default after reset)
1: TIM16 reset

Bit 16 **TIM15RST**: TIM15 reset
Set and reset by software.
0: TIM15 not reset (default after reset)
1: TIM15 reset

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 reset
Set and reset by software.
0: USART1 not reset (default after reset)
1: USART1 reset

Bit 13 **TIM8RST**: TIM8 reset
Set and reset by software.
0: TIM8 not reset (default after reset)
1: TIM8 reset

Bit 12 **SPI1RST**: SPI1 reset
Set and reset by software.
0: SPI1 not reset (default after reset)
1: SPI1 reset

Bit 11 **TIM1RST**: TIM1 reset
Set and reset by software.
0: TIM1 not reset (default after reset)
1: TIM1 reset

Bits 10:0 Reserved, must be kept at reset value.

9.8.14 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPTIM1RST	Res.	Res.	Res.	Res.	LPUART1RST	Res.	Res.	Res.	Res.	SBSRST	Res.
				rw					rw					rw	

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1RST**: LPTIM1 reset
Set and reset by software.
0: LPTIM1 not reset (default after reset)
1: LPTIM1 reset

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **LPUART1RST**: LPUART1 reset
Set and reset by software.
0: LPUART1 not reset (default after reset)
1: LPUART1 reset

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SBSRST**: SBS reset
Set and reset by software.
0: SBS not reset (default after reset)
1: SBS reset

Bit 0 Reserved, must be kept at reset value.

9.8.15 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x088

Reset value: 0xC000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1EN	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXEN	ETH1TXEN	ETH1EN	ETH1CKEN	RAMCFGEN	Res.
rw	rw									rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CORDICEN	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA2EN	LPDMA1EN
	rw		rw				rw							rw	rw

Bit 31 **SRAM1EN**:SRAM1 clock enable
Set and reset by software.
0: SRAM1 clock disabled
1: SRAM1 clock enabled (default after reset)

Bit 30 **SRAM2EN**:SRAM2 clock enable
Set and reset by software.
0: SRAM2 clock disabled
1: SRAM2 clock enabled (default after reset)

Bits 29:22 Reserved, must be kept at reset value.

Bit 21 **ETH1RXEN**: ETH1RX clock enable
Set and reset by software.
0: ETH1RX clock disabled (default after reset)
1: ETH1RX clock enabled

Bit 20 **ETH1TXEN**: ETH1TX clock enable
Set and reset by software.
0: ETH1TX clock disabled (default after reset)
1: ETH1TX clock enabled

Bit 19 **ETH1EN**: ETH1 clock enable
Set and reset by software.
0: ETH1 clock disabled (default after reset)
1: ETH1 clock enabled

Bit 18 **ETH1CKEN**: ETH1 internal clock enable
Set and reset by software.
0: ETH1 internal clock disabled (default after reset)
1: ETH1 internal clock enabled

Bit 17 **RAMCFGEN**: RAMCFG clock enable
Set and reset by software.
0: RAMCFG clock disabled (default after reset)
1: RAMCFG clock enabled

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **CORDICEN**: CORDIC clock enable
Set and reset by software.
0: CORDIC clock disabled (default after reset)
1: CORDIC clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable
Set and reset by software.
0: CRC clock disabled (default after reset)
1: CRC clock enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHEN**: Flash interface clock enable
Set and reset by software.
0: FLASH clock disabled
1: FLASH clock enabled (default after reset)

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **LPDMA2EN**: LPDMA2 clock enable
Set and reset by software.
0: LPDMA2 clock disabled (default after reset)
1: LPDMA2 clock enabled

Bit 0 **LPDMA1EN**: LPDMA1 clock enable
Set and reset by software.
0: LPDMA1 clock disabled (default after reset)
1: LPDMA1 clock enabled

9.8.16 RCC AHB2 peripheral clock register (RCC_AHB2ENR)

Address offset: 0x08C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3EN	Res.	Res.	CCBEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN
							rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1EN	ADC12EN	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
				rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC3EN**: ADC3 clock enable
Set and reset by software.
0: ADC3 clock disabled (default after reset)
1: ADC3 clock enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCBEN**: CCB clock enable
Set and reset by software.
0: CCB clock disabled (default after reset)
1: CCB clock enabled

Bit 20 **SAESEN**: SAES clock enable
Set and reset by software.
0: SAES clock disabled (default after reset)
1: SAES clock enabled

Bit 19 **PKAEN**: PKA clock enable
Set and reset by software.
0: PKA clock disabled (default after reset)
1: PKA clock enabled

Bit 18 **RNGEN**: RNG clock enable
Set and reset by software.
0: RNG clock disabled (default after reset)
1: RNG clock enabled

Bit 17 **HASHEN**: HASH clock enable
Set and reset by software.
0: HASH clock disabled (default after reset)
1: HASH clock enabled

Bit 16 **AESEN**: AES clock enable
Set and reset by software.
0: AES clock disabled (default after reset)
1: AES clock enabled

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DAC1EN**: DAC clock enable
Set and reset by software.
0: DAC1 clock disabled (default after reset)
1: DAC1 clock enabled

Bit 10 **ADC12EN**: ADC1 and ADC2 clock enable
Set and reset by software.
0: ADC1 and ADC2 clock disabled (default after reset)
1: ADC1 and ADC2 clock enabled

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN**: GPIOH clock enable
Set and reset by software.
0: GPIOH clock disabled (default after reset)
1: GPIOH clock enabled

Bit 6 **GPIOGEN**: GPIOG clock enable
Set and reset by software.
0: GPIOG clock disabled (default after reset)
1: GPIOG clock enabled

Bit 5 **GPIOFEN**: GPIOF clock enable
Set and reset by software.
0: GPIOF clock disabled (default after reset)
1: GPIOF clock enabled

Bit 4 **GPIOEEN**: GPIOE clock enable
Set and reset by software.
0: GPIOE clock disabled (default after reset)
1: GPIOE clock enabled

Bit 3 **GPIODEN**: GPIOD clock enable
Set and reset by software.
0: GPIOD clock disabled (default after reset)
1: GPIOD clock enabled

Bit 2 **GPIOCEN**: GPIOC clock enable
Set and reset by software.
0: GPIOC clock disabled (default after reset)
1: GPIOC clock enabled

Bit 1 **GPIOBEN**: GPIOB clock enable
Set and reset by software.
0: GPIOB clock disabled (default after reset)
1: GPIOB clock enabled

Bit 0 **GPIOAEN**: GPIOA clock enable
Set and reset by software.
0: GPIOA clock disabled (default after reset)
1: GPIOA clock enabled

9.8.17 RCC AHB4 peripheral clock register (RCC_AHB4ENR)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1EN	Res.	Res.	Res.	Res.
											rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **XSPI1EN**: XSPI1 clock enable
Set and reset by software.
0: XSPI1 peripheral clock disabled
1: XSPI1 peripheral clock enabled

Bits 19:0 Reserved, must be kept at reset value.

9.8.18 RCC APB1 peripheral clock register (RCC_APB1LENR)

Address offset: 0x09C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	UART7EN	Res.	Res.	Res.	Res.	USART6EN	CRSEN	I3C1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.
	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	OPAMP1EN	Res.	WWDGEN	Res.	Res.	Res.	Res.	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw	rw	rw		rs					rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **UART7EN**: UART7 clock enable
Set and reset by software.
0: UART7 clock disabled (default after reset)
1: UART7 clock enabled

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **USART6EN**: USART6 clock enable
Set and reset by software.
0: USART6 clock disabled (default after reset)
1: USART6 clock enabled

Bit 24 **CRSEN**: CRS clock enable
Set and reset by software.
0: CRS clock disabled (default after reset)
1: CRS clock enabled

Bit 23 **I3C1EN**: I3C1 clock enable
Set and reset by software.
0: I3C1 clock disabled (default after reset)
1: I3C1 clock enabled

Bit 22 **I2C2EN**: I2C2 clock enable
Set and reset by software.
0: I2C2 clock disabled (default after reset)
1: I2C2 clock enabled

Bit 21 **I2C1EN**: I2C1 clock enable
Set and reset by software.
0: I2C1 clock disabled (default after reset)
1: I2C1 clock enabled

Bit 20 **UART5EN**: UART5 clock enable
Set and reset by software.
0: UART5 clock disabled (default after reset)
1: UART5 clock enabled

Bit 19 **UART4EN**: UART4 clock enable
Set and reset by software.
0: UART4 clock disabled (default after reset)
1: UART4 clock enabled

Bit 18 **USART3EN**: USART3 clock enable
Set and reset by software.
0: USART3 clock disabled (default after reset)
1: USART3 clock enabled

Bit 17 **USART2EN**: USART2 clock enable
Set and reset by software.
0: USART2 clock disabled (default after reset)
1: USART2 clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3EN**: SPI3 clock enable
Set and reset by software.
0: SPI3 clock disabled (default after reset)
1: SPI3 clock enabled

Bit 14 **SPI2EN**: SPI2 clock enable
Set and reset by software.
0: SPI2 clock disabled (default after reset)
1: SPI2 clock enabled

Bit 13 **OPAMP1EN**: OPAMP1 clock enable
Set and reset by software.
0: OPAMP1 clock disabled (default after reset)
1: OPAMP1 clock enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG clock enable
Set by software and reset by hardware system reset.
0: WWDG clock disabled (default after reset)
1: WWDG clock enabled

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **TIM12EN**: TIM12 clock enable
Set and reset by software.
0: TIM12 clock disabled (default after reset)
1: TIM12 clock enabled

Bit 5 **TIM7EN**: TIM7 clock enable
Set and reset by software.
0: TIM7 clock disabled (default after reset)
1: TIM7 clock enabled

Bit 4 **TIM6EN**: TIM6 clock enable
Set and reset by software.
0: TIM6 clock disabled (default after reset)
1: TIM6 clock enabled

Bit 3 **TIM5EN**: TIM5 clock enable
Set and reset by software.
0: TIM5 clock disabled (default after reset)
1: TIM5 clock enabled

Bit 2 **TIM4EN**: TIM4 clock enable
Set and reset by software.
0: TIM4 clock disabled (default after reset)
1: TIM4 clock enabled

Bit 1 **TIM3EN**: TIM3 clock enable
Set and reset by software.
0: TIM3 clock disabled (default after reset)
1: TIM3 clock enabled

Bit 0 **TIM2EN**: TIM2 clock enable
Set and reset by software.
0: TIM2 clock disabled (default after reset)
1: TIM2 clock enabled

9.8.19 RCC APB1 peripheral clock register (RCC_APB1HENR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN EN	Res.	Res.	Res.	Res.	Res.	COMP EN	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FDCANEN**: FDCAN1 and FDCAN2 clock enable
Set and reset by software.
0: FDCAN1 and FDCAN2 clock disabled (default after reset)
1: FDCAN1 and FDCAN2 clock enabled

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **COMPEN**: COMP clock enable
Set and reset by software.
0: COMP clock disabled (default after reset)
1: COMP clock enabled

Bits 2:0 Reserved, must be kept at reset value.

9.8.20 RCC APB2 peripheral clock register (RCC_APB2ENR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBEN	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
							rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBEN**: USB clock enable
Set and reset by software.
0: USB clock disabled (default after reset)
1: USB clock enabled

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM17 clock enable
Set and reset by software.
0: TIM17 clock disabled (default after reset)
1: TIM17 clock enabled

Bit 17 **TIM16EN**: TIM16 clock enable
Set and reset by software.
0: TIM16 clock disabled (default after reset)
1: TIM16 clock enabled

Bit 16 **TIM15EN**: TIM15 clock enable
Set and reset by software.
0: TIM15 clock disabled (default after reset)
1: TIM15 clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1 clock enable
Set and reset by software.
0: USART1 clock disabled (default after reset)
1: USART1 clock enabled

Bit 13 **TIM8EN**: TIM8 clock enable
Set and reset by software.
0: TIM8 clock disabled (default after reset)
1: TIM8 clock enabled

Bit 12 **SPI1EN**: SPI1 clock enable
Set and reset by software.
0: SPI1 clock disabled (default after reset)
1: SPI1 clock enabled

Bit 11 **TIM1EN**: TIM1 clock enable
Set and reset by software.
0: TIM1 clock disabled (default after reset)
1: TIM1 clock enabled

Bits 10:0 Reserved, must be kept at reset value.

9.8.21 RCC APB3 peripheral clock register (RCC_APB3ENR)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPTIM1EN	Res.	Res.	Res.	Res.	LPUART1EN	Res.	Res.	Res.	Res.	SBSEN	Res.
				rw					rw					rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBEN**: RTC APB interface clock enable
Set and reset by software.
0: RTC APB interface clock disabled (default after reset)
1: RTC APB interface clock enabled

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1EN**: LPTIM1 clock enable
Set and reset by software.
0: LPTIM1 clock disabled (default after reset)
1: LPTIM1 clock enabled

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **LPUART1EN**: LPUART1 clock enable
Set and reset by software.
0: LPUART1 clock disabled (default after reset)
1: LPUART1 clock enabled

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SBSEN**: SBS clock enable
Set and reset by software.
0: SBS clock disabled (default after reset)
1: SBS clock enabled

Bit 0 Reserved, must be kept at reset value.

9.8.22 RCC AHB1 sleep clock register (RCC_AHB1LPENR)

Address offset: 0x0B0

Reset value: 0xC402 5103 (for STM32C53x/542/55x/562)

Reset value: 0xC43E 5103 (for STM32C59x/5A3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1LPEN	SRAM2LPEN	Res.	Res.	Res.	ICACHELPEN	Res.	Res.	Res.	Res.	ETH1RXLPEN	ETH1TXLPEN	ETH1LPEN	ETH1CLKLPEN	RAMCFGLPEN	Res.
rw	rw				rw					rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CORDICLPEN	Res.	CRCLPEN	Res.	Res.	Res.	FLASHLPEN	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA2LPEN	LPDMA1LPEN
	rw		rw				rw							rw	rw

Bit 31 **SRAM1LPEN**: SRAM1 clock enable during Sleep mode
Set and reset by software
0: SRAM1 clock disabled during Sleep mode
1: SRAM1 clock enabled during Sleep mode (default after reset)

Bit 30 **SRAM2LPEN**: SRAM2 clock enable during Sleep mode
Set and reset by software
0: SRAM2 clock disabled during Sleep mode
1: SRAM2 clock enabled during Sleep mode (default after reset)

Bits 29:27 Reserved, must be kept at reset value.

Bit 26 **ICACHELPEN**: ICACHE clock enable during Sleep mode
Set and reset by software.
0: ICACHE clock disabled during Sleep mode
1: ICACHE clock enabled during Sleep mode (default after reset)

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **ETH1RXLPEN**: ETH1RX clock enable during Sleep mode
Set and reset by software.
0: ETH1RX clock disabled during Sleep mode
1: ETH1RX clock enabled during Sleep mode (default after reset)

Bit 20 **ETH1TXLPEN**: ETH1TX clock enable during Sleep mode
Set and reset by software.
0: ETH1TX clock disabled during Sleep mode
1: ETH1TX clock enabled during Sleep mode (default after reset)

Bit 19 **ETH1LPEN**: ETH1 clock enable during Sleep mode
Set and reset by software.
0: ETH1 clock disabled during Sleep mode
1: ETH1 clock enabled during Sleep mode (default after reset)

Bit 18 **ETH1CLKLPEN**: ETH1 internal clock enable during Sleep mode
Set and reset by software.
0: ETH1 internal clock disabled during Sleep mode
1: ETH1 internal clock enabled during Sleep mode (default after reset)

Bit 17 **RAMCFGLPEN**: RAMCFG clock enable during Sleep mode
Set and reset by software.
0: RAMCFG clock disabled during Sleep mode
1: RAMCFG clock enabled during Sleep mode (default after reset)

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **CORDICLPEN**: CORDIC clock enable during Sleep mode
Set and reset by software.
0: CORDIC clock disabled during Sleep mode
1: CORDIC clock enabled during Sleep mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCLPEN**: CRC clock enable during Sleep mode
Set and reset by software.
0: CRC clock disabled during Sleep mode
1: CRC clock enabled during Sleep mode (default after reset)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHLPEN**: Flash interface clock enable during Sleep mode
Set and reset by software.
0: FLASH clock disabled during Sleep mode
1: FLASH clock enabled during Sleep mode (default after reset)

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **LPDMA2LPEN**: LPDMA2 clock enable during Sleep mode
Set and reset by software.
0: LPDMA2 clock disabled during Sleep mode
1: LPDMA2 clock enabled during Sleep mode (default after reset)

Bit 0 **LPDMA1LPEN**: LPDMA1 clock enable during Sleep mode
Set and reset by software.
0: LPDMA1 clock disabled during Sleep mode
1: LPDMA1 clock enabled during Sleep mode (default after reset)

9.8.23 RCC AHB2 sleep clock register (RCC_AHB2LPENR)

Address offset: 0x0B4

Reset value: 0x0007 0C9F (for STM32C53x/C542/C55x/C562)

Reset value: 0x013F 0CFF (for STM32C59x/C5A3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3LPEN	Res.	Res.	CCBLPEN	SAESLPEN	PKALPEN	RNGLPEN	HASHLPEN	AESLPEN
							rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1LPEN	ADC12LPEN	Res.	Res.	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
				rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC3LPEN**: ADC3 clock enable during Sleep mode
Set and reset by software.
0: ADC3 clock disabled during Sleep mode
1: ADC3 clock enabled during Sleep mode (default after reset)

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCBLPEN**: CCB clock enable during Sleep mode
Set and reset by software.
0: CCB clock disabled during Sleep mode
1: CCB clock enabled during Sleep mode (default after reset)

Bit 20 **SAESLPEN**: SAES clock enable during Sleep mode
Set and reset by software.
0: SAES clock disabled during Sleep mode
1: SAES clock enabled during Sleep mode (default after reset)

Bit 19 **PKALPEN**: PKA clock enable during Sleep mode
Set and reset by software.
0: PKAclock disabled during Sleep mode
1: PKA clock enabled during Sleep mode (default after reset)

Bit 18 **RNGLPEN**: RNG clock enable during Sleep mode
Set and reset by software.
0: RNG clock disabled during Sleep mode
1: RNG clock enabled during Sleep mode (default after reset)

Bit 17 **HASHLPEN**: HASH clock enable during Sleep mode
Set and reset by software.
0: HASH clock disabled during Sleep mode
1: HASH clock enabled during Sleep mode (default after reset)

Bit 16 **AESLPEN**: AES clock enable during Sleep mode
Set and reset by software.
0: AES clock disabled during Sleep mode
1: AES clock enabled during Sleep mode (default after reset)

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DAC1LPEN**: DAC clock enable during Sleep mode
Set and reset by software.
0: DAC1 clock disabled during Sleep mode
1: DAC1 clock enabled during Sleep mode (default after reset)

Bit 10 **ADC12LPEN**: ADC1 and ADC2 clock enable during Sleep mode
Set and reset by software.
0: ADC1 and ADC2 clock disabled during Sleep mode
1: ADC1 and ADC2 clock enabled during Sleep mode (default after reset)

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHLPEN**: GPIOH clock enable during Sleep mode
Set and reset by software.
0: GPIOH clock disabled during Sleep mode
1: GPIOH clock enabled during Sleep mode (default after reset)

Bit 6 **GPIOGLPEN**: GPIOG clock enable during Sleep mode
Set and reset by software.
0: GPIOG clock disabled during Sleep mode
1: GPIOG clock enabled during Sleep mode (default after reset)

Bit 5 **GPIOFLPEN**: GPIOF clock enable during Sleep mode
Set and reset by software.
0: GPIOF clock disabled during Sleep mode
1: GPIOF clock enabled during Sleep mode (default after reset)

Bit 4 **GPIOELPEN**: GPIOE clock enable during Sleep mode
Set and reset by software.
0: GPIOE clock disabled during Sleep mode
1: GPIOE clock enabled during Sleep mode (default after reset)

Bit 3 **GPIODLPEN**: GPIOD clock enable during Sleep mode
Set and reset by software.
0: GPIOD clock disabled during Sleep mode
1: GPIOD clock enabled during Sleep mode (default after reset)

Bit 2 **GPIOCLPEN**: GPIOC clock enable during Sleep mode
Set and reset by software.
0: GPIOC clock disabled during Sleep mode
1: GPIOC clock enabled during Sleep mode (default after reset)

Bit 1 **GPIOBLPEN**: GPIOB clock enable during Sleep mode
Set and reset by software.
0: GPIOB clock disabled during Sleep mode
1: GPIOB clock enabled during Sleep mode (default after reset)

Bit 0 **GPIOALPEN**: GPIOA clock enable during Sleep mode
Set and reset by software.
0: GPIOA clock disabled during Sleep mode
1: GPIOA clock enabled during Sleep mode (default after reset)

9.8.24 RCC AHB4 peripheral sleep clock register (RCC_AHB4LPENR)

Address offset: 0x0BC

Reset value: 0x0010 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1LPEN	Res.	Res.	Res.	Res.
											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **XSPI1LPEN**: XSPI1 clock enable during sleep mode
Set and reset by software.
0: XSPI1 clock disabled during sleep mode
1: XSPI1 clock enabled during sleep mode (default after reset)

Bits 19:0 Reserved, must be kept at reset value.

9.8.25 RCC APB1 sleep clock register (RCC_APB1LLPENR)

Address offset: 0x0C4

Reset value: 0x01BA 6871 (for STM32C53x/C542)

Reset value: 0x01FE C879 (for STM32C55x/C562)

Reset value: 0x43FE C87F (for STM32C59x/C5A3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	UART7 LPEN	Res.	Res.	Res.	Res.	USART 6LPEN	CRSLP EN	I3C1LP EN	I2C2LP EN	I2C1LP EN	UART5 LPEN	UART4 LPEN	USART 3LPEN	USART 2LPEN	Res.
	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LP EN	SPI2LP EN	OPAMP 1LPEN	Res.	WWDG LPEN	Res.	Res.	Res.	Res.	TIM12L PEN	TIM7LP EN	TIM6LP EN	TIM5LP EN	TIM4LP EN	TIM3LP EN	TIM2LP EN
rw	rw	rw		rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **UART7LPEN**: UART7 clock enable during Sleep mode
Set and reset by software.
0: UART7 clock disabled during Sleep mode
1: UART7 clock enabled during Sleep mode (default after reset)

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **USART6LPEN**: USART6 clock enable during Sleep mode
Set and reset by software.
0: USART6 clock disabled during Sleep mode
1: USART6 clock enabled during Sleep mode (default after reset)

Bit 24 **CRSLPEN**: CRS clock enable during Sleep mode
Set and reset by software.
0: CRS clock disabled during Sleep mode
1: CRS clock enabled during Sleep mode (default after reset)

Bit 23 **I3C1LPEN**: I3C1 clock enable during Sleep mode
Set and reset by software.
0: I3C1 clock disabled during Sleep mode
1: I3C1 clock enabled during Sleep mode (default after reset)

Bit 22 **I2C2LPEN**: I2C2 clock enable during Sleep mode
Set and reset by software.
0: I2C2 clock disabled during Sleep mode
1: I2C2 clock enabled during Sleep mode (default after reset)

Bit 21 **I2C1LPEN**: I2C1 clock enable during Sleep mode
Set and reset by software.
0: I2C1 clock disabled during Sleep mode
1: I2C1 clock enabled during Sleep mode (default after reset)

Bit 20 **UART5LPEN**: UART5 clock enable during Sleep mode
Set and reset by software.
0: UART5 clock disabled during Sleep mode
1: UART5 clock enabled during Sleep mode (default after reset)

Bit 19 **UART4LPEN**: UART4 clock enable during Sleep mode
Set and reset by software.
0: UART4 clock disabled during Sleep mode
1: UART4 clock enabled during Sleep mode (default after reset)

Bit 18 **USART3LPEN**: USART3 clock enable during Sleep mode
Set and reset by software.
0: USART3 clock disabled during Sleep mode
1: USART3 clock enabled during Sleep mode (default after reset)

Bit 17 **USART2LPEN**: USART2 clock enable during Sleep mode
Set and reset by software.
0: USART2 clock disabled during Sleep mode
1: USART2 clock enabled during Sleep mode (default after reset)

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN**: SPI3 clock enable during Sleep mode
Set and reset by software.
0: SPI3 clock disabled during Sleep mode
1: SPI3 clock enabled during Sleep mode (default after reset)

Bit 14 **SPI2LPEN**: SPI2 clock enable during Sleep mode
Set and reset by software.
0: SPI2 clock disabled during Sleep mode
1: SPI2 clock enabled during Sleep mode (default after reset)

Bit 13 **OPAMP1LPEN**: OPAMP1 clock enable during Sleep mode
Set and reset by software.
0: OPAMP1 clock disabled during Sleep mode
1: OPAMP1 clock enabled during Sleep mode (default after reset)

Bit 12 Reserved, must be kept at reset value.

Bit 11 **WWDGLPEN**: WWDG clock enable during Sleep mode
Set and reset by software.
0: WWDG clock disabled during Sleep mode
1: WWDG clock enabled during Sleep mode (default after reset)

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **TIM12LPEN**: TIM12 clock enable during Sleep mode
Set and reset by software.
0: TIM12 clock disabled during Sleep mode
1: TIM12 clock enabled during Sleep mode (default after reset)

Bit 5 **TIM7LPEN**: TIM7 clock enable during Sleep mode
Set and reset by software.
0: TIM7 clock disabled during Sleep mode
1: TIM7 clock enabled during Sleep mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 clock enable during Sleep mode
Set and reset by software.
0: TIM6 clock disabled during Sleep mode
1: TIM6 clock enabled during Sleep mode (default after reset)

Bit 3 **TIM5LPEN**: TIM5 clock enable during Sleep mode
Set and reset by software.
0: TIM5 clock disabled during Sleep mode
1: TIM5 clock enabled during Sleep mode (default after reset)

Bit 2 **TIM4LPEN**: TIM4 clock enable during Sleep mode
Set and reset by software.
0: TIM4 clock disabled during Sleep mode
1: TIM4 clock enabled during Sleep mode (default after reset)

Bit 1 **TIM3LPEN**: TIM3 clock enable during Sleep mode
Set and reset by software.
0: TIM3 clock disabled during Sleep mode
1: TIM3 clock enabled during Sleep mode (default after reset)

Bit 0 **TIM2LPEN**: TIM2 clock enable during Sleep mode
Set and reset by software.
0: TIM2 clock disabled during Sleep mode
1: TIM2 clock enabled during Sleep mode (default after reset)

9.8.26 RCC APB1 sleep clock register (RCC_APB1HLPENR)

Address offset: 0x0C8

Reset value: 0x4000 0208

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN LPEN	Res.	Res.	Res.	Res.	Res.	COMP LPEN	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FDCANLPEN**: FDCAN1 and FDCAN2 clock enable during Sleep mode
Set and reset by software.
0: FDCAN1 and FDCAN2 clock disabled during Sleep mode
1: FDCAN1 and FDCAN2 clock enabled during Sleep mode (default after reset)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **COMPLPEN**: COMP clock enable during Sleep mode
Set and reset by software.
0: COMP clock disabled during Sleep mode
1: COMP clock enabled during Sleep mode (default after reset)

Bits 2:0 Reserved, must be kept at reset value.

9.8.27 RCC APB2 sleep clock register (RCC_APB2LPENR)

Address offset: 0x0CC

Reset value: 0x0107 7800 (for STM32C55x/562/59x/5A3)

Reset value: 0x0101 7800 (for STM32C53x/542)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBLPEN	Res.	Res.	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN
							rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1LPEN	TIM8LPEN	SPI1LPEN	TIM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBLPEN**: USB clock enable during Sleep mode
Set and reset by software.
0: USB clock disabled during Sleep mode
1: USB clock enabled during Sleep mode (default after reset)

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TIM17LPEN**: TIM17 clock enable during Sleep mode
Set and reset by software.
0: TIM17 clock disabled during Sleep mode
1: TIM17 clock enabled during Sleep mode (default after reset)

Bit 17 **TIM16LPEN**: TIM16 clock enable during Sleep mode
Set and reset by software.
0: TIM16 clock disabled during Sleep mode
1: TIM16 clock enabled during Sleep mode (default after reset)

Bit 16 **TIM15LPEN**: TIM15 clock enable during Sleep mode
Set and reset by software.
0: TIM15 clock disabled during Sleep mode
1: TIM15 clock enabled during Sleep mode (default after reset)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1LPEN**: USART1 clock enable during Sleep mode
Set and reset by software.
0: USART1 clock disabled during Sleep mode
1: USART1 clock enabled during Sleep mode (default after reset)

Bit 13 **TIM8LPEN**: TIM8 clock enable during Sleep mode
Set and reset by software.
0: TIM8 clock disabled during Sleep mode
1: TIM8 clock enabled during Sleep mode (default after reset)

Bit 12 **SPI1LPEN**: SPI1 clock enable during Sleep mode
Set and reset by software.
0: SPI1 clock disabled during Sleep mode
1: SPI1 clock enabled during Sleep mode (default after reset)

Bit 11 **TIM1LPEN**: TIM1 clock enable during Sleep mode
Set and reset by software.
0: TIM1 clock disabled during Sleep mode
1: TIM1 clock enabled during Sleep mode (default after reset)

Bits 10:0 Reserved, must be kept at reset value.

9.8.28 RCC APB3 sleep clock register (RCC_APB3LPENR)

Address offset: 0x0D0

Reset value: 0x0020 0842

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	Res.	Res.
										rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPTIM1LPEN	Res.	Res.	Res.	Res.	LPUART1LPEN	Res.	Res.	Res.	Res.	SBSLPEN	Res.
				rw					rw					rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBLPEN**: RTC APB interface clock enable during Sleep mode
Set and reset by software.
0: RTC APB interface clock disabled during Sleep mode
1: RTC APB interface clock enabled during Sleep mode (default after reset)

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1LPEN**: LPTIM1 clock enable during Sleep mode
Set and reset by software.
0: LPTIM1 clock disabled during Sleep mode
1: LPTIM1 clock enabled during Sleep mode (default after reset)

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **LPUART1LPEN**: LPUART1 clock enable during Sleep mode
Set and reset by software.
0: LPUART1 clock disabled during Sleep mode
1: LPUART1 clock enabled during Sleep mode (default after reset)

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SBSLPEN**: SBS clock enable during Sleep mode
Set and reset by software.
0: SBS clock disabled during Sleep mode
1: SBS clock enabled during Sleep mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

9.8.29 RCC kernel clock configuration register (RCC_CCIPR1)

Address offset: 0x0D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	Res.	SPI3SEL[1:0]		SPI2SEL[1:0]		SPI1SEL[1:0]	
				rw	rw					rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPUART1SEL[1:0]		UART7SEL[1:0]		USART6SEL[1:0]		UART5SEL[1:0]		UART4SEL[1:0]		USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:26 **FDCANSEL[1:0]**: FDCAN1 and FDCAN2 kernel clock source selection
- 00: rcc_pclk1 selected as kernel clock (default after reset)
- 01: psis_ck selected as kernel clock
- 10: psik_ck selected as kernel clock
- 11: hse_ck selected as kernel clock

Bits 25:22 Reserved, must be kept at reset value.

Bits 21:20 **SPI3SEL[1:0]**: SPI3 kernel clock source selection
- 00: rcc_pclk1 selected as kernel clock (default after reset)
- 01: psik_ck selected as kernel clock
- 10: hsik_ck selected as kernel clock
- 11: AUDIOCLK selected as kernel clock

Bits 19:18 **SPI2SEL[1:0]**: SPI2 kernel clock source selection
- 00: rcc_pclk1 selected as kernel clock (default after reset)
- 01: psik_ck selected as kernel clock
- 10: hsik_ck selected as kernel clock
- 11: AUDIOCLK selected as kernel clock

Bits 17:16 **SPI1SEL[1:0]**: SPI1 kernel clock source selection
- 00: rcc_pclk2 selected as kernel clock (default after reset)
- 01: psik_ck selected as kernel clock
- 10: hsik_ck selected as kernel clock
- 11: AUDIOCLK selected as kernel clock

Bits 15:14 **LPUART1SEL[1:0]**: LPUART1 kernel clock source selection
- 00: rcc_pclk3 selected as kernel clock (default after reset)
- 01: hsik_ck selected as kernel clock
- 10: lse_ck selected as kernel clock
- 11: lsi_ck selected as kernel clock

Bits 13:12 **UART7SEL[1:0]**: UART7 kernel clock source selection
- 00: rcc_pclk1 selected as kernel clock (default after reset)
- 01: psik_ck selected as kernel clock
- 10: hsik_ck selected as kernel clock
- 11: lse_ck selected as kernel clock

Bits 11:10 **USART6SEL[1:0]**: UART6 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
11: lse_ck selected as kernel clock

Bits 9:8 **UART5SEL[1:0]**: UART5 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
11: lse_ck selected as kernel clock

Bits 7:6 **UART4SEL[1:0]**: UART4 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
11: lse_ck selected as kernel clock

Bits 5:4 **USART3SEL[1:0]**: UART3 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
11: lse_ck selected as kernel clock

Bits 3:2 **USART2SEL[1:0]**: USART2 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
11: lse_ck selected as kernel clock

Bits 1:0 **USART1SEL[1:0]**: USART1 kernel clock source selection
00: rcc_pclk2 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
11: lse_ck selected as kernel clock

9.8.30 RCC kernel clock configuration register (RCC_CCIPR2)

Address offset: 0x0DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYSTICKSEL[1:0]		Res.	Res.	Res.	Res.	CK48SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1SEL[1:0]	
rw	rw					rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACSEL	ADCDACPRE[2:0]			ADCDACSEL[1:0]		Res.	Res.	I3C1SEL[1:0]		Res.	Res.	I2C2SEL[1:0]		I2C1SEL[1:0]	
rw	rw	rw	rw	rw	rw			rw	rw			rw	rw	rw	rw

Bits 31:30 **SYSTICKSEL[1:0]**: SYSTICK clock source selection
00: rcc_hclk/8 selected as clock source (default after reset)
01: lsi_ker_ck[1] selected as clock source
10: lse_ck[1] selected as clock source
11: reserved, the kernel clock is disabled
*Note: rcc_hclk frequency must be four times higher than lsi_ker_ck/lse_ck, that is, period (LSI/LSE) ≥ 4 * period (HCLK).*

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **CK48SEL[1:0]**: CK48 clock source selection
01: psi_div_3_ck selected as kernel clock
10: hsi_div_3_ck selected as kernel clock
11: hse_ck selected as kernel clock
Other: Reserved, kernel clock disabled (default after reset)

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **LPTIM1SEL[1:0]**: LPTIM1 kernel clock source selection
00: rcc_pclk3 selected as kernel clock (default after reset)
01: hsik_ck selected as kernel clock
10: lse_ck selected as kernel clock
11: lsi_ck selected as kernel clock

Bit 15 **DACSEL**: DAC sample and hold clock
0: lse_ck selected as sample and hold clock (default after reset)
1: lsi_ck selected as sample and hold clock

Bits 14:12 **ADCDACPRE[2:0]**: ADC and DAC prescaler for kernel clock source selection
ADCDACPRE bitfield must not be changed when ADC or DAC are enabled.
000: ADC and DAC kernel clock are not divided (default after reset)
001: ADC and DAC kernel clock divided by 2
010: ADC and DAC kernel clock divided by 4
011: ADC and DAC kernel clock divided by 8
100: ADC and DAC kernel clock divided by 16
101: ADC and DAC kernel clock divided by 32
110: ADC and DAC kernel clock divided by 64
111: ADC and DAC kernel clock divided by 128

Bits 11:10 **ADCDACSEL[1:0]**: ADC and DAC kernel clock source selection
This bitfield must not be changed when ADC or DAC are enabled.
00: rcc_hclk selected as kernel clock
01: psis_ck selected as kernel clock (default after reset)
10: psik_ck selected as kernel clock
11: hsik_ck selected as kernel clock

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:6 **I3C1SEL[1:0]**: I3C1 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:2 **I2C2SEL[1:0]**: I2C2 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
Other: Reserved, kernel clock disabled

Bits 1:0 **I2C1SEL[1:0]**: I2C1 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
Other: Reserved, kernel clock disabled

9.8.31 RCC kernel clock configuration register (RCC_CCIPR3)

Address offset: 0x0E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETH1PTPDIV[3:0]				ETH1CLKDIV[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ETH1CLKSEL[1:0]		Res.	ETH1PTPCLKSEL[1:0]		Res.	ETH1REFCLKSEL	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1SEL[1:0]	
	rw	rw		rw	rw		rw							rw	rw

Bits 31:28 **ETH1PTPDIV[3:0]**: Ethernet PTP clock division
This field selects the division factor of the PTP clock.
0000: divided by 1(default after reset)
0001: divided by 2
0010-1111: divided by {v+1}

Bits 27:26 **ETH1CLKDIV[1:0]**: Ethernet clock division
This field selects the division factor of the ETH1 clock.
00: divided by 1(default after reset)
01: divided by 2
10: divided by 4
11: Reserved

Bits 25:15 Reserved, must be kept at reset value.

Bits 14:13 **ETH1CLKSEL[1:0]**: ETH1 clock source selection
This field selects the reference clock of the PAD ETH1_CLK.
Set and reset by software.
00: kernel clock disabled (default after reset)
01: psis_ck selected as kernel clock
10: psik_ck selected as kernel clock
11: hse_ck selected as kernel clock

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **ETH1PTPCLKSEL[1:0]**: ETH1 PTP clock source selection
Set and reset by software.
00: kernel clock disabled (default after reset)
01: rcc_hclk1 selected as kernel clock
10: psis_ck selected as kernel clock
11: psik_ck selected as kernel clock

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ETH1REFCLKSEL**: ETH1 RMII reference clock source selection
This bit selects the reference clock of the Ethernet for RMII mode. It must be set to 0 in any other mode than RMII.
0: ETH1_RMII_REF_CLK (default after reset)
1: eth_clk_fb

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **XSPI1SEL[1:0]**: XSPI1 kernel clock source selection
00: rcc_hclk4 selected as kernel clock (default after reset)
01: psik_ck selected as kernel clock
10: hsik_ck selected as kernel clock
Other: Reserved, kernel clock disabled

9.8.32 RCC RTC domain control register (RCC_RTCCR)

Address offset: 0x0F0

Reset value: 0x0000 0000

Reset by RTC domain reset.

Access: 0 ≤ wait state ≤ 3, word, half-word, and byte access.

Wait states are inserted in case of successive accesses to this register.

Note: After a system reset, this register is write-protected (except bits 27:26 and bit 16). To modify the RTC domain bits, the DRTCP bit in PWR_RTCCR must be set to 1. All bits (except bits 27:26 and bit 16) are reset only after an RTC domain reset (see Section 9.3.3). Other resets do not affect these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSIRDY	LSION	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDRST
				r	rw	rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		LSEEXT	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw						rw	rw	rw	r	rw	rw	rw	rw	r	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **LSIRDY**: LSI oscillator ready
Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after three internal low-speed oscillator clock cycles.
This bit is set when the LSI is used by IWDG or RTC, even if LSION = 0.
0: LSI oscillator not ready
1: LSI oscillator ready

Bit 26 **LSION**: LSI oscillator enable
Set and cleared by software.
0: LSI oscillator off
1: LSI oscillator on

Bit 25 **LSCOSEL**: Low-speed clock output selection
Set and cleared by software.
0: LSI clock selected
1: LSE clock selected

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable
Set and cleared by software.
0: LSCO output disabled
1: LSCO output enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **RTCDRST**: RTC domain software reset
Set and reset by software.
0: reset not activated (default after RTC domain reset)
1: resets the entire VSW domain

Bit 15 **RTCEN**: RTC clock enable
Set and reset by software.
0: rtc_ck disabled (default after RTC domain reset)
1: rtc_ck enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**:RTC clock source selection
Set by software to select the clock source for the RTC. These bits can be written only one time (except in case of failure detection on LSE). These bits must be written before LSECSSON is enabled. The RTCDRST bit can be used to reset them, then it can be written one time again. If HSE is selected as RTC clock, this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).
00: No clock (default after RTC domain reset)
01: LSE selected as RTC clock
10: LSI selected as RTC clock
11: HSE divided by RTCPRE value selected as RTC clock

Bit 7 **LSEEXT**: Low-speed external clock type in bypass mode
Set and reset by software to select the external clock type (analog or digital). The external clock must be enabled with the LSEON bit, to be used by the device. The LSEEXT bit can be written only if the LSE oscillator is disabled.
0: LSE in analog mode (default after RTC domain reset)
1: LSE in digital mode

Bit 6 **LSECSSD**: LSE clock security system failure detection
Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator. Clearing LSECSSON (only possible after failure detection) also clears this bit.
0: No failure detected on 32 kHz oscillator (default after RTC domain reset)
1: Failure detected on 32 kHz oscillator

Bit 5 **LSECSSON**: LSE clock security system enable

Set by software to enable the clock security system on 32 kHz oscillator. This bit must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware), and after RTCSEL is selected.

Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case, software must disable LSECSSON.

0: CSS on 32 kHz oscillator off (default after RTC domain reset)

1: CSS on 32 kHz oscillator on

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator driving capability

Set by software to select the driving capability of the LSE oscillator.

These bits can be written only if LSE oscillator is disabled (LSEON = 0 and LSERDY = 0).

00: Lowest drive (default after RTC domain reset)

01: Medium-low drive

10: Medium-high drive

11: Highest drive

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and reset by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (by LSEON) or ready (LSERDY = 1)

0: LSE oscillator not bypassed (default after RTC domain reset)

1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready

Set and reset by hardware to indicate when the LSE is stable. This bit needs 6 cycles of lse_ck clock to fall down after LSEON has been set to 0.

0: LSE oscillator not ready (default after RTC domain reset)

1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enabled

Set and reset by software. This bit cannot be cleared if the LSE is selected as reference clock for PSI with PSI enabled (PSISON bit or PSIDIV3ON bit or PSIKON bit set to 1).

0: LSE oscillator off (default after RTC domain reset)

1: LSE oscillator on

9.8.33 RCC reset status register (RCC_RSR)

Address offset: 0x0F4

Reset value: 0xXX00 0000

Reset by power-on reset only.

Access: 0 ≤ wait state ≤ 3, word, half-word, and byte access.

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r			w							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **LPWRRSTF**: Low-power reset flag

Set by hardware when a reset occurs due to Stop or Standby mode entry, whereas the corresponding NRST_STOP, NRST_STBY option bit is cleared.
Cleared by writing to the RMVF bit.
0: No illegal low-power mode reset occurred
1: An illegal low-power mode reset occurred

Bit 30 **WWDGRSTF**: Window watchdog reset flag

Reset by software by writing the RMVF bit.
Set by hardware when a window watchdog reset occurs.
0: No WWDG reset occurred (default after power-on reset)
1: A WWDG reset occurred

Bit 29 **IWDGRSTF**: Independent watchdog reset flag

Reset by software by writing the RMVF bit. Set by hardware when an IWDG reset occurs.
0: No IWDG reset occurred (default after power-on reset)
1: An IWDG reset occurred

Bit 28 **SFTRSTF**: System reset from CPU reset flag

Reset by software by writing the RMVF bit. Set by hardware when the system reset is due to CPU.The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the Cortex-M33.
0: No CPU software reset occurred (default after power-on reset)
1: A system reset has been generated by the CPU.

Bit 27 **BORRSTF**: POR reset flag

Reset by software by writing the RMVF bit. Set by hardware when a POR reset occurs (pwr_por_rst).
0: No POR reset occurred
1: A POR reset occurred (default after power-on reset)

Bit 26 **PINRSTF**: Pin reset flag (NRST)

Reset by software by writing the RMVF bit. Set by hardware when a reset from pin occurs.
0: No reset from pin occurred
1: A reset from pin occurred (default after power-on reset)

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove reset flag

Set and reset by software to reset the value of the reset flags.
0: Reset of the reset flags not activated (default after power-on reset)
1: Reset the value of the reset flags

Bits 22:0 Reserved, must be kept at reset value.

9.8.34 RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x114

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register can be written only by a privileged access. It can be read by privileged or unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PRIV**: RCC function privileged configuration
Set and reset by software. This bit can be written only by privileged access.
0: Read and write to RCC functions can be done by privileged or unprivileged access.
1: Read and write to RCC functions can be done by privileged access only

Bit 0 Reserved, must be kept at reset value.

9.8.35 RCC register map

Table 71. RCC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON	Res.	PSIKRDY	PSIDIV3RDY	PSISRDY	PSIKERON	PSIKON	PSIDIV3ON	PSISON	Res.	HSIKRDY	HSIDIV3RDY	HSISRDY	HSIKERON	HSIKON	HSIDIV3ON	HSISON
	Reset value												0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	1
0x004	RCC_CR2	Res.	Res.	PSIFREQ[1:0]		Res.	Res.	Res.	Res.	Res.	PSIREF[2:0]			Res.	Res.	PSIREFSRC[1:0]		Res.	Res.	Res.	Res.	PSIKDIV[3:0]				Res.	Res.	Res.	Res.	HSIKDIV[3:0]			
	Reset value			0	0						0	0	0			0	0					0	0	0	0					0	0	0	0
0x008 - 0x018	Reserved																																
0x01C	RCC_CFGR1	MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]				Res.	Res.	RTCPRE[8:0]									STOPWUCK	Res.	SWS [1:0]		Res.	SW[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0		0	0

Table 71. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	APB2DIS	APB1DIS	AHB4DIS	Res.	AHB2DIS	AHB1DIS	Res.	PPRE3[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
	Reset value										0	0	0	0		0	0		0	0	0		0	0	0		0	0	0	0	0	0	0
0x024 - 0x04C	Reserved																																
0x050	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDYIE	PSIKRDYIE	PSIDIV3RDYIE	PSISRDYIE	HSIKRDYIE	HSIDIV3RDYIE	HSISRDYIE	LSERDYIE	LSIRDYIE
	Reset value																								0	0	0	0	0	0	0	0	0
0x054	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSF	HSECSSF	Res.	HSERDYF	PSIKRDYF	PSIDIV3RDYF	PSISRDYF	HSIKRDYF	HSIDIV3RDYF	HSISRDYF	LSERDYF	LSIRDYF
	Reset value																					0	0		0	0	0	0	0	0	0	0	0
0x058	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSC	HSECSSC	Res.	HSERDYC	PSIKRDYC	PSIDIV3RDYC	PSISRDYC	HSIKRDYC	HSIDIV3RDYC	HSISRDYC	LSERDYC	LSIRDYC
	Reset value																					0	0		0	0	0	0	0	0	0	0	0
0x05C	Reserved																																
0x060	RCC_AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RST	Res.	RAMCFGRST	Res.	Res.	CORDICRST	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA2RST	LPDMA1RST
	Reset value													0		0			0		0										0	0
0x064	RCC_AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3RST	Res.	Res.	CCBRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST	Res.	Res.	Res.	Res.	DAC1RST	ADC12RST	Res.	Res.	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value								0			0	0	0	0	0	0					0	0			0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	RCC_AHB4RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0																				
0x074	RCC_APB1LRSTR	Res.	UART7RST	Res.	Res.	Res.	Res.	USART6RST	CRSRST	I3C1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	OPAMP1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value		0					0	0	0	0	0	0	0	0	0		0	0	0							0	0	0	0	0	0	0

Table 71. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x078	RCC_APB1HRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	Res.	Res.	COMPRST	Res.	Res.	Res.
	Reset value																							0						0			
0x07C	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBRST	Res.	Res.	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0						0	0	0		0	0	0	0											
0x080	RCC_APB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1RST	Res.	Res.	Res.	Res.	LPUART1RST	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0					0						
0x084	Reserved																																
0x088	RCC_AHB1ENR	SRAM1EN	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1RXEN	ETH1TXEN	ETH1EN	ETH1CKEN	RAMCFGEN	Res.	Res.	CORDICEN	Res.	CRCEN	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA2EN	LPDMA1EN
	Reset value	1	1									0	0	0	0	0			0		0			1							0	0
0x08C	RCC_AHB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3EN	Res.	Res.	CCBEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN	Res.	Res.	Res.	Res.	DAC1EN	ADC12EN	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value								0			0	0	0	0	0	0					0	0			0	0	0	0	0	0	0	0
0x090	Reserved																																
0x094	RCC_AHB4ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0																				
0x09C	RCC_APB1LENR	Res.	UART7EN	Res.	Res.	Res.	Res.	USART6EN	CRSEN	I3C1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	OPAMP1EN	Res.	WWDGEN	Res.	Res.	Res.	Res.	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value		0					0	0	0	0	0	0	0	0	0		0	0	0		0					0	0	0	0	0	0	0
0x0A0	RCC_APB1HENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	Res.	Res.	COMPEN	Res.	Res.	Res.
	Reset value																							0						0			
0x0A4	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBEN	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0						0	0	0		0	0	0	0											

Table 71. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A8	RCC_APB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1EN	Res.	Res.	Res.	Res.	LPUART1EN	Res.	Res.	Res.	Res.	SBSEN	Res.
	Reset value											0										0					0					0	
0x0AC	Reserved																																
0x0B0	RCC_AHB1LPENR	SRAM1LPEN	SRAM2LPEN	Res.	Res.	Res.	ICACHELPEN	Res.	Res.	Res.	Res.	ETH1RXLPEN	ETH1TXLPEN	ETH1LPEN	ETH1CKLPEN	RAMCFGLPEN	Res.	Res.	CORDICLPEN	Res.	CRCLPEN	Res.	Res.	Res.	FLASHLPEN	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA2LPEN	LPDMA1LPEN
	Reset value	1	1				1					1	1	1	1	1			1		1				1							1	1
0x0B4	RCC_AHB2LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3LPEN	Res.	Res.	CCBLPEN	SAESLPEN	PKALPEN	RNGLPEN	HASHLPEN	AESLPEN	Res.	Res.	Res.	Res.	DAC1LPEN	ADC12LPEN	Res.	Res.	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value								1			1	1	1	1	1	1					1	1			1	1	1	1	1	1	1	1
0x0B8	Reserved																																
0x0BC	RCC_AHB4LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												1																				
0x0C4	RCC_APB1LLPENR	Res.	UART7LPEN	Res.	Res.	Res.	Res.	USART6LPEN	CRSLPEN	I3C1LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	OPAMP1LPEN	Res.	WWDGLPEN	Res.	Res.	Res.	Res.	Res.	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value		1					1	1	1	1	1	1	1	1	1		1	1	1		1					1	1	1	1	1	1	1
0x0C8	RCC_APB1HLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	Res.	Res.	Res.	COMPLPEN	Res.	Res.	Res.
	Reset value																							1						1			
0x0CC	RCC_APB2LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBLPEN	Res.	Res.	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	Res.	USART1LPEN	TIM8LPEN	SPI1LPEN	TIM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								1						1	1	1		1	1	1	1											
0x0D0	RCC_APB3LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1LPEN	Res.	Res.	Res.	Res.	LPUART1LPEN	Res.	Res.	Res.	Res.	SBSLPEN	Res.
	Reset value											1										1					1					1	
0x0D4	Reserved																																

Table 71. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D8	RCC_CCIPR1	Res.	Res.	Res.	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	Res.	SPI3SEL[1:0]		SPI2SEL[1:0]		SPI1SEL[1:0]		LPUART1SEL[1:0]		UART7SEL[1:0]		USART6SEL[1:0]		UART5SEL[1:0]		UART4SEL[1:0]		USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[1:0]	
	Reset value					0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DC	RCC_CCIPR2	SYSTICKSEL[1:0]		Res.	Res.	Res.	Res.	CK48SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1SEL[1:0]		DACSEL		ADCDACPRE[2:0]		ADCDACSEL[1:0]		Res.	Res.	I3C1SEL[1:0]		Res.	Res.	I2C2SEL[1:0]		I2C1SEL[1:0]	
	Reset value	0	0					0	0							0	0	0	0	0	0	0	0			0	0			0	0	0	0
0x0E0	RCC_CCIPR3	ETH1PTPDIV[3:0]				ETH1CLKDIV[3:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1CLKSEL[1:0]		Res.	ETH1PTPCLKSEL[1:0]		ETH1REFCLKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1SEL[1:0]	
	Reset value	0	0	0	0	0	0												0	0		0	0	0								0	0
0x0E4 - 0x0EC	Reserved																																
0x0F0	RCC_RTCCR	Res.	Res.	Res.	Res.	LSIRDY	LSION	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		LSEEXT	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	Reset value					0	0	0	0								0	0						0	0	0	0	0	0	0	0	0	0
0x0F4	RCC_RSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X			0																							
0x0F8 - 0x110	Reserved																																
0x114	RCC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.
	Reset value																															0	

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

10 Clock recovery system (CRS)

10.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI144. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from alternative synchronization or generated by the user software.

10.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity (see *Section 10.4.2: CRS internal signals*)
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

10.3 CRS implementation

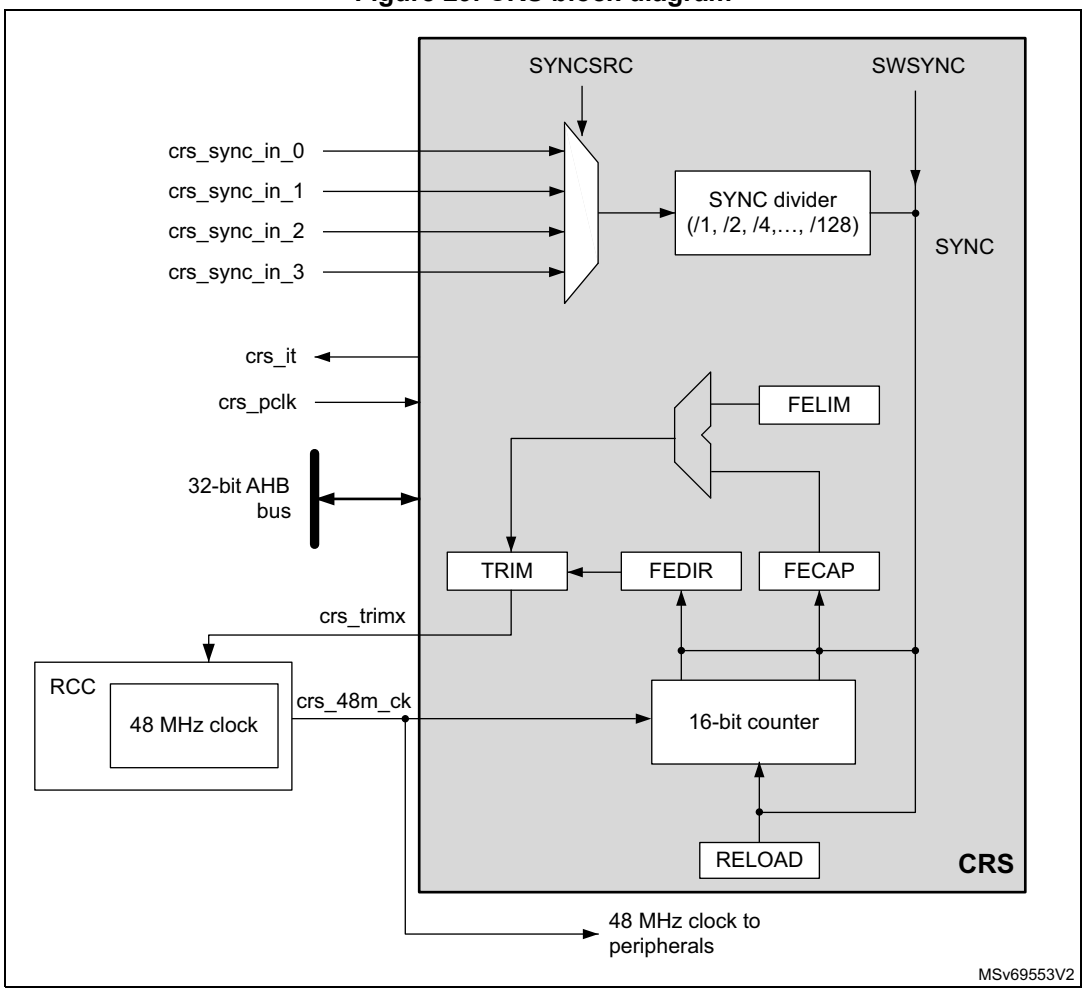
Table 72. CRS features

Feature	CRS
TRIM width	7 bits

10.4 CRS functional description

10.4.1 CRS block diagram

Figure 29. CRS block diagram



10.4.2 CRS internal signals

Below the list of CRS internal signals.

Table 73. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
crs_48m_ck	Digital input	HSIDIV3 48 MHz clock
crs_trim[0:6]	Digital output	HSI144 oscillator smooth trimming value

Table 73. CRS internal input/output signals (continued)

Internal signal name	Signal type	Description
crs_sync_in_0	Digital input	SYNC signal source
crs_sync_in_1		
crs_sync_in_2		
crs_sync_in_3		

Table 74. CRS interconnection

Internal signal name	Description
crs_sync_in_0	GPIO AF selected as SYNC signal source
crs_sync_in_1	LSE selected as SYNC signal source
crs_sync_in_2	USB SOF selected as SYNC signal source (default)
crs_sync_in_3	HSE_1MHz clock (output of RTCPRE) selected as SYNC signal source

10.4.3 Synchronization input

The CRS synchronization source (crs_sync_in_x) can be selected through SYNCSRC[1:0] bitfield of CRS_CFGR register (refer to *Section 10.4.2: CRS internal signals* for the possible sources). For a better robustness of the crs_sync_in_x input, a simple digital filter (2 out of 3 majority votes, sampled by the 48 MHz clock) is implemented to filter out glitches. In addition, the source signal has a configurable polarity (selected through SYNCPOL bit of CRS_CFGR). The signal can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to CRS_CFGR register.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

For more information on the CRS synchronization source configuration, refer to *CRS configuration register (CRS_CFGR)*.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

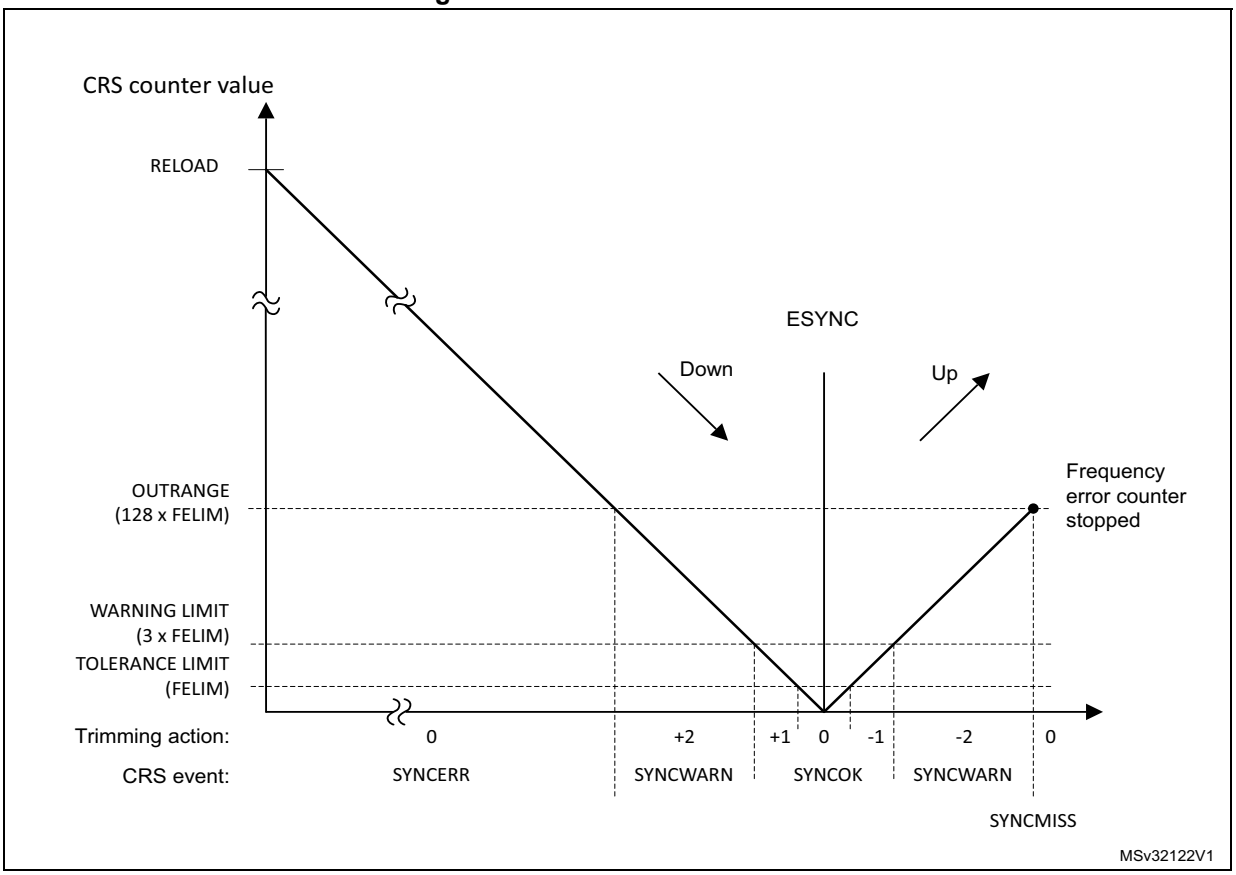
10.4.4 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual

frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 30. CRS counter behavior



10.4.5 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated

  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

10.4.6 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI144 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to $(f_{TARGET} / f_{SYNC}) = 48000$, and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

10.5 CRS in low-power modes

Table 75. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI144 oscillator is restarted.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.

10.6 CRS interrupts

Table 76. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

10.7 CRS registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

10.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 3000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIM[6:0]							SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERR IE	SYNC WARNIE	SYNC OKIE
	rw	rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TRIM[6:0]**: HSI144 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI144 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.

The default value of the HSI144 oscillator smooth trimming is 48, for a complete TRIM range of 128 steps.

The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 10.4.5* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled

1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled

1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled

1: SYNC event OK (SYNCOKF) interrupt enabled

10.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]							
rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.

0: SYNC active on rising edge (default)

1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see *Section 10.4.2: CRS internal signals*).

00: crs_sync_in_0 selected as SYNC signal source

01: crs_sync_in_1 selected as SYNC signal source

10: crs_sync_in_2 selected as SYNC signal source

11: crs_sync_in_3 selected as SYNC signal source

Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the 48 MHz clock on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

000: SYNC not divided (default)

001: SYNC divided by 2

010: SYNC divided by 4

011: SYNC divided by 8

100: SYNC divided by 16

101: SYNC divided by 32

110: SYNC divided by 64

111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 10.4.5* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 10.4.4* for more details about counter behavior.

10.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 10.4.5* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

0: Up-counting direction, the actual frequency is above the target

1: Down-counting direction, the actual frequency is below the target

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No trimming error signaled

1: Trimming error signaled

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signaled

1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signaled

1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signaled

1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled

1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signaled

1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signaled

1: SYNC event OK signaled

10.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNC WARNC	SYNC OKC
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

10.7.5 CRS register map

Table 77. CRS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[6:0]							SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value	0x0000 3000																															
0x04	CRS_CFGR	SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	CRS_ISR	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	CRS_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

11 General-purpose I/Os (GPIO)

11.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit reset register (GPIOx_BRR), and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

11.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bit-wise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions
- I/Os state retention during Standby mode

11.3 GPIO functional description

Based on the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

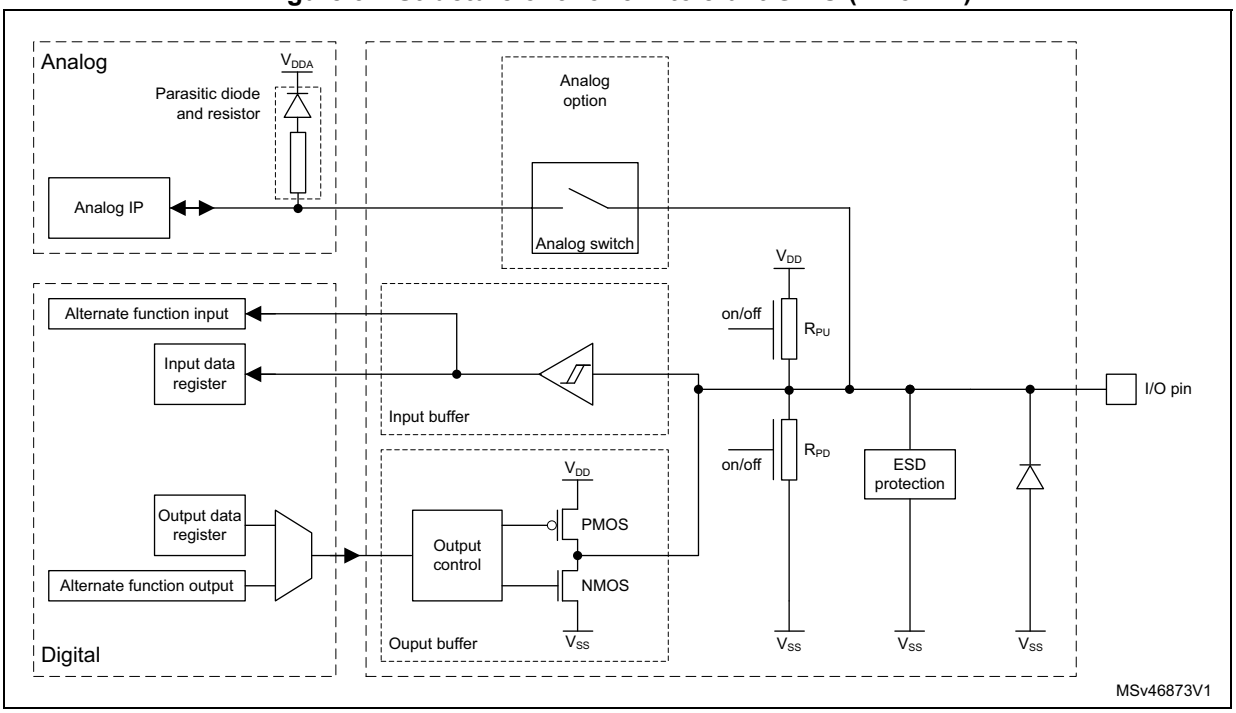
- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words, or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow

atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

The figure below shows the basic structure of a 3- or 5-V tolerant GPIO (TT or FT). *Table 78* gives the possible port bit configurations.

Figure 31. Structure of 3- or 5-V tolerant GPIO (TT or FT)



Note: On a TT GPIO, the analog switch is not present, it is replaced by a direct connection. The analog bloc parasitic circuitry does not allow 5-V tolerance.

Table 78. Port bit configuration(1)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 78. Port bit configuration[1] (continued)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		I/O configuration	
10	0	SPEED[1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

11.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state no pull-up/pull-down

BOOT0 is in input mode during the reset until at least the end of the option byte loading phase (see *Section 11.3.15*).

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, the high level is high-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not, depending upon the value in the GPIOx_PUPDR register.

11.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through the GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- Debug function: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- GPIO: configure the desired I/O as output, input, or analog in the GPIOx_MODER register.
- Alternate function:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down, and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
  - The Cortex-M33 output EVENTOUT signal can be output as alternate function on several I/Os. An event can be indicated through the configured pin after executing SEV instruction.
  - The EVENTOUT signal can be used internally as a trigger for some peripherals (see *Section 13: Peripherals interconnect matrix*)
- Additional functions:
  - For the ADC, DAC, COMP, and OPAMP configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC, COMP, and OPAMP registers.
  - For the additional functions like RTC, WKUPx, LSCO, and oscillators, configure the required function in the related RTC, PWR, and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

11.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type

(push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

11.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR (x = A to H)).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data entered through the I/Os are stored into the input data register (GPIOx_IDR), a read-only register.

11.3.5 I/O data bitwise handling

GPIOx_BSRR is a 32-bit register that allows the application to set and reset each individual bit in GPIOx_ODR. The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the corresponding ODR(i) bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

11.3.6 GPIO locking mechanism

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence can be performed only using a word (32-bit long) access to the GPIOx_LCKR register, because GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

11.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that some peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

11.3.8 External interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output, or alternate function mode (the port must not be configured in analog mode). Refer to *Section 16: Extended interrupts and event controller (EXTI)*.

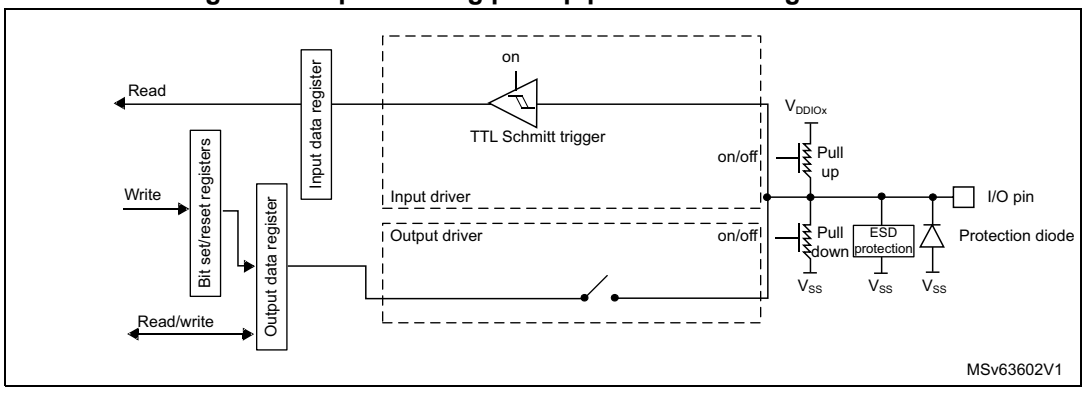
11.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

Figure 32. Input floating/pull-up/pull-down configurations



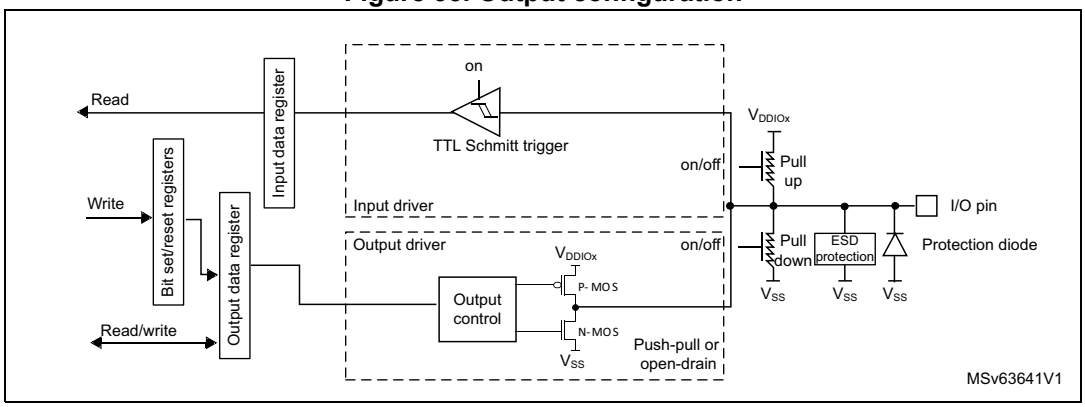
11.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in high-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

Figure 33. Output configuration



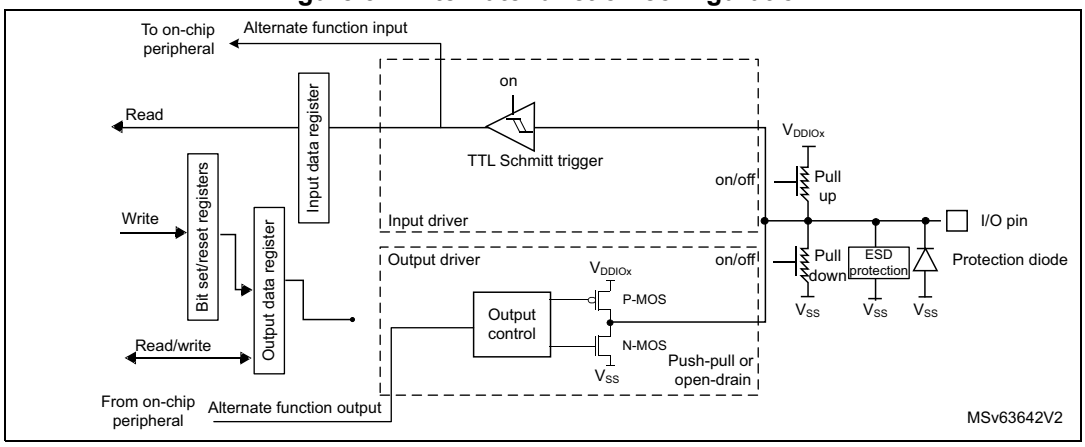
11.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

The figure below shows the alternate function configuration of the I/O port bit.

Figure 34. Alternate function configuration



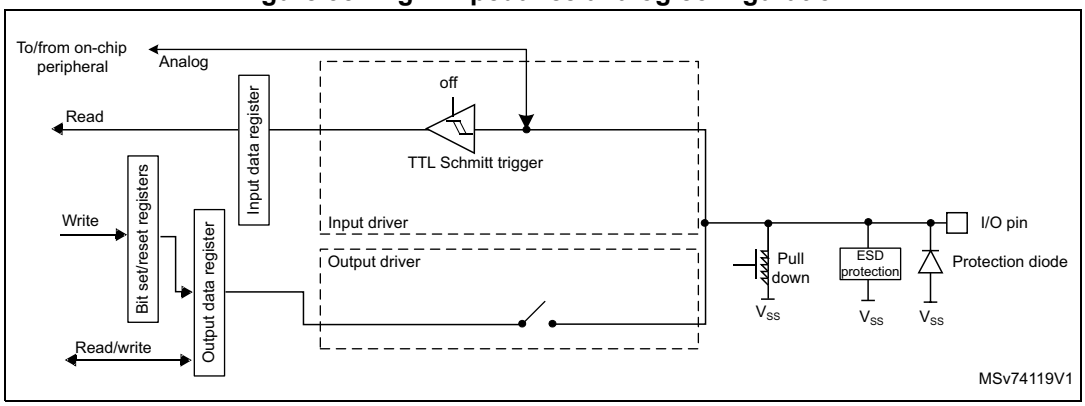
11.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up is disabled by hardware. The weak pull-down is configurable.
- Read access to the input data register gets the value 0.

The figure below shows the high-Z, analog-input configuration of the I/O port bits.

Figure 35. High-impedance analog configuration



11.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in the RCC_CR1 register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

11.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 39.3: RTC functional description*.

11.3.15 I/Os state retention during standby mode

In the Standby mode, the I/Os are by default in floating state.

If the IORETEN bit in the PWR_IORETR register is set, the I/Os state is sampled during standby entry. The state of I/Os is applied to the pin via pull-up and pull-down resistors. The pull-up and pull-down resistors remains applied after Standby wake-up until the IORETEN bit in the PWR_IORETR register is cleared by software.

11.3.16 Privileged and unprivileged modes

All GPIO registers can be read and written by privileged and unprivileged accesses.

11.3.17 I/O compensation cell

The I/O commutation slew rate (tfall / trise) can be adapted by software depending on process, voltage and temperatures conditions, to reduce the I/O noise on power supply. Refer to*Section 12: System configuration, boot, and security (SBS)* for more details.

11.4 GPIO registers

This section gives a detailed description of the GPIO registers. The peripheral registers can be written in word, half word, or byte mode.

For the actual availability of GPIO ports and their related bits, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them as reserved, and keep them at reset value.

11.4.1 GPIO port mode register (GPIOx_MODER) (x = A to H)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFF3F FEBF (for port B)

Reset value: 0xFFFF FFFF (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to H)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to H)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very-high speed

Note: Refer to the device datasheet for the frequency specifications, the power supply, and the load conditions for each speed.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to H)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.5 GPIO port input data register (GPIOx_IDR) (x = A to H)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.6 GPIO port output data register (GPIOx_ODR) (x = A to H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A to H).

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 **BSy:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to H)

Address offset: 0x1C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

- LOCK key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

- LOCK key read

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the LOCK.

After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0

0: Port configuration not locked

1: Port configuration locked

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to H)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15
Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to H)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit
Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

11.4.12 GPIO register map

Table 79. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOx_MODER (x = A to H)	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value for port A	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
	Reset value for ports C...H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	GPIOx_OTYPER (x = A to H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_OSPEEDR (x = A to H)	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value for port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	Reset value for ports C...H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (x = A to H)	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value for port A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value for ports C...H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (x = A to H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	GPIOx_ODR (x = A to H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (x = A to H)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (x = A to H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (x = A to H)	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (x = A to H)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (x = A to H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

12 System configuration, boot, and security (SBS)

12.1 SBS introduction

The STM32C5 devices feature a set of configuration registers located in the SBS. On top of various device configurations, this SBS peripheral controls key boot and security features, including debug control.

12.2 SBS main features

- System configuration
  - Manage robustness feature.
  - Manage ADC input channel remap.
  - Manage the I/O compensation cell.
  - Configure register security access.
- Boot control
  - Upon system reset, configure the Cortex-M33 boot address and the temporal isolation level depending on the current configuration (such as RDP_LEVEL (sbs_rdp_level).
  - Manage the temporal isolation feature implemented thanks to the hide protect level (HDPL) monotonic counter.

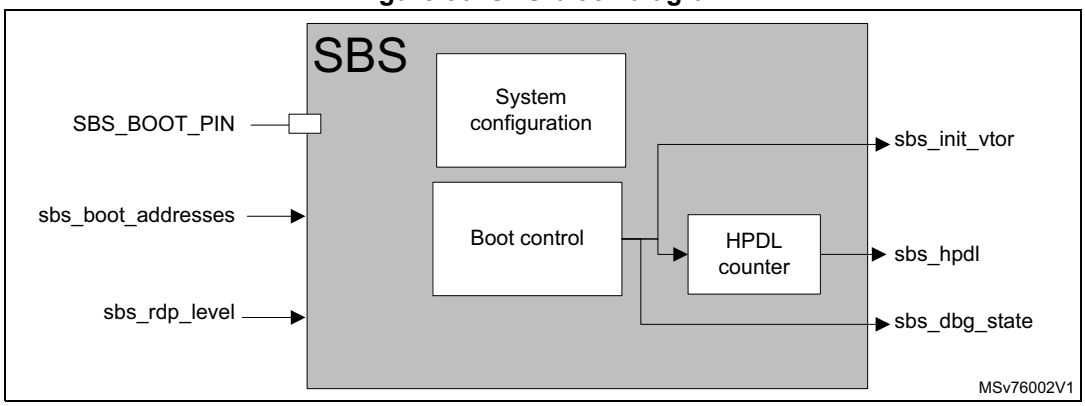
12.3 SBS functional description

12.3.1 SBS block diagram

The figure below shows the SBS block diagram, including the following main functions:

- System configuration
- Boot control

Figure 36. SBS block diagram



12.3.2 SBS signals

The following table details the user-relevant internal signals that interface the SBS.

Table 80. SBS internal input/output signals

Signal name	Type	Description
SBS_BOOT_PIN	Input	Select booting on user flash memory or bootloader
sbs_boot_addresses		List of addresses defined by the flash memory: – BOOT_ADDR: boot address
sbs_rdp_level		Signal based on the RDP_LEVEL option byte to activate the different security mechanisms depending on the product use. Expected values are described in *Section 6: Embedded flash memory (FLASH)*.
sbs_init_vtor	Output	Vector address for Cortex-M33
sbs_hdpl		HDPL (temporal isolation level, ID of the boot level, used to isolate boot levels) This signal reflects a monotonic counter that can be incremented from 0 to 3, and that is reset to zero only on software reset or POR.
sbs_ap_unlocked		Control the Cortex-M33 access port.

12.3.3 SBS reset and clocks

The SBS configuration port is clocked by the AHB bus clock. There is a general reset and a debug configuration reset controlled in DBGMCU.

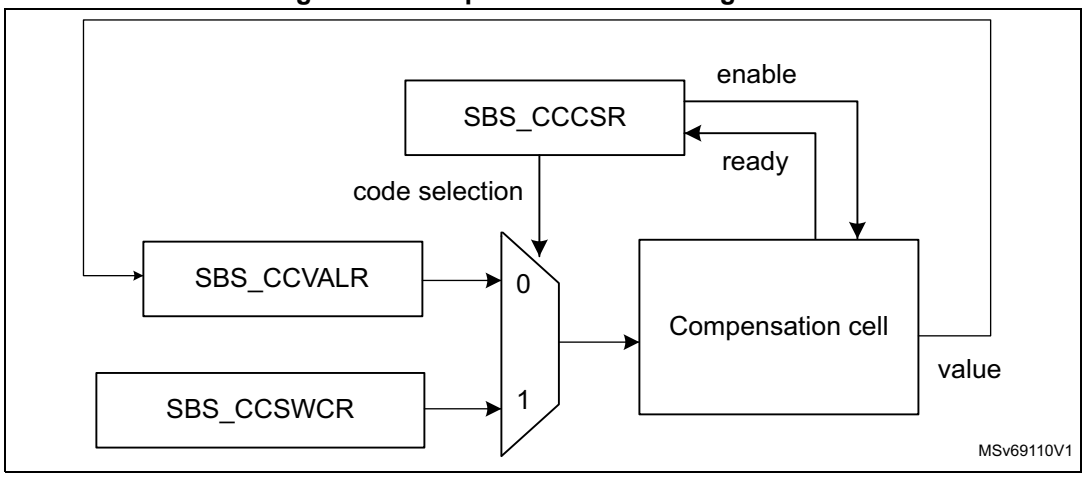
12.3.4 SBS I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer: 4 bits for N-MOS and 4 bits for P-MOS. This value depends on the process, voltage, temperature (PVT) operating conditions. These bits are used to control the output impedance in the I/O buffer, and the slew rate of the I/O commutation (the t_{fall} and t_{rise} time), in order to reduce the I/O noise on the power supply.

As shown in *Figure 37*, the compensation cell is split in two blocks:

- One block to provide an optimal code for the current PVT
- One block to drive the block controlled by software

Figure 37. Compensation cell management

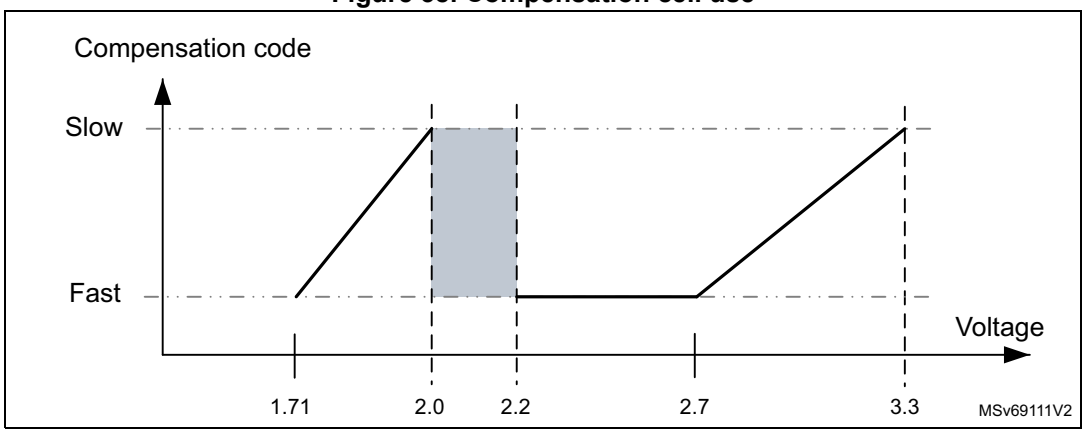


The compensation cell value can be read when the READY flag is set in SBS_CCCSR. With CODESEL in SBS_CCCSR, the application can select the value to apply between two options: the code from the cell, or the code from SBS_CCSWCR.

By default, the compensation cells are disabled, and a fixed code is applied to all I/Os.

The compensation cell can be used only when the HSI oscillator is enabled, see Section 9: Reset and clock control (RCC) for more details on the HSI oscillator.

Figure 38. Compensation cell use



12.3.5 SBS boot control

The SBS can be used to control the boot entry points considering the product settings. The main boot control actions are listed below:

- Select boot between the bootloader or the user flash memory boot.
- Initialize the HDPL boot value.

Note: For more details on STM32C5xx BOOT control, refer to Section 4: Boot modes.

SBS HDPL (temporal isolation level) management

The HDPL is a monotonic counter incremented during the boot stages. The HDPL is reset to its default value only after a power-on or a system reset. This default value is 1.

The devices use HDPL information to automatically isolate code and its associated secrets (like keys) during the boot process. Incrementing HPDL ensures that private code and data for one boot stage cannot be directly accessible from later boot stages.

The HDPL is used by the user flash memory. See *Section 6: Embedded flash memory (FLASH)* for more details.

The HDPL can take the value from 1 to 3. When reaching the value 3, HDPL keeps this value until reset. The current HDPL value is readable in HDPL bitfield of SBS_HDPLSR.

To increment the HDPL by one, the application must write 0x6A to INCR_HDPL in SBS_HDPLCR. After such increment, and before doing any subsequent action, the user must check that the HDPL has effectively been incremented reading SBS_HDPLSR.

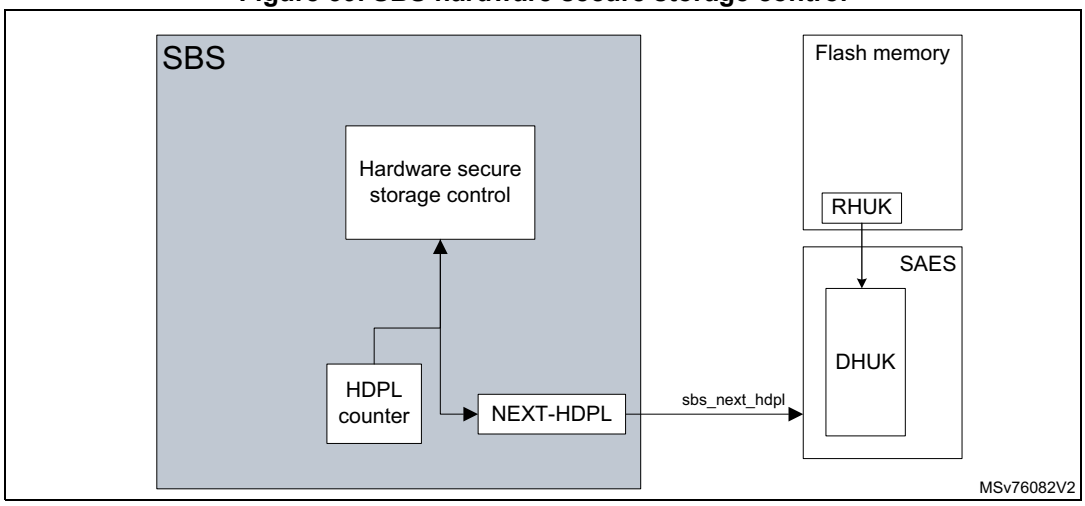
Table 81. HDPL encoded values

HDPL	Code
1	0x51
2	0x8A
3	0x6F
	All other values

12.3.6 SBS hardware secure storage control

This feature ensures the isolation of keys including the key derivation (DHUK) mechanism protecting data using a hardware key different for the identified domains.

Figure 39. SBS hardware secure storage control



- sbs_next_hdpl: select secure storage context for current HDPL, or greater ones. This signal is set thanks to the SBS_NEXTHDPLCR register. This information is used by the SAES when processing the derived key to generate a key derived related to HDPL context.

All keys encrypted using the SAES/DHUK inherit of the RHUK property: unique per device.

The table below gives the value of HDPL function of NEXTHDPL value.

Table 82. NEXT-HDPL logic

HDPL[7:0] in SBS_HDPLSR	NEXTHDPL[1:0]			
	0	1	2	3
1 (0x51)	0x51	0x8A	0x6F	
2 (0x8A)	0x8A	0x6F		
3 (0x6F)	0x6F	0x6F		
Others	0x6F			

12.4 SBS interrupts

SBS does not support interrupts.

12.5 SBS registers

For the actual availability of some peripherals/features and their related bits, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them as reserved, and keep them at reset value.

12.5.1 SBS temporal isolation control register (SBS_HDPLCR)

Address offset: 0x010

Reset value: 0x0000 00B4

Reset: system reset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR_HDPL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INCR_HDPL[7:0]**: HDPL value increment
0xB4: No increment
0x6A: Recommended value to increment HDPL level by one
Other: All other values allow an HDPL level increment.

12.5.2 SBS temporal isolation status register (SBS_HDPLSR)

Address offset: 0x014

Reset value: 0x0000 0051

Reset: system reset

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPL[7:0]**: Temporal isolation level

This bitfield returns the current temporal isolation level.

0x51: HDPL1, hide protection level to be used to execute and protect an immutable root of trust (IROT) stage. First code to be executed at each reset of the platform. To ensure secure boot, this code must be immutable)

0x8A: HDPL2, hide protection level to be used to execute and protect an updatable Root of Trust (UROT) stage. UROT is executed from IROT to complete the immutable root of trust execution during boot stages

0x6F: HDPL3, hide protection level to be used to execute the application. It is a temporal isolation level that can be used to run the application when it has to be isolated from IROT and UROT)

12.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR)

Note: This section applies only to STM32C59xx/C5Axx devices.

Address offset: 0x018

Reset value: 0x0000 0000

Reset: system reset

Register security: no restriction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NEXTHDPL[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **NEXTHDPL[1:0]**: Index to point to a higher HDPL than the current one
This bitfield is used to add the increment to HDPL to access to the next secure storage areas through NEXT-HDPL.
00: NEXT-HDPL = HDPL
01: NEXT-HDPL = HDPL + 1
10: NEXT-HDPL = HDPL + 2
11: NEXT-HDPL = HDPL + 3

12.5.4 SBS product mode and configuration register (SBS_PMCR)

Address offset: 0x100

Reset value: 0x8000 0000

Reset: system reset

Register security: there is no access restriction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ETH1TXLPI	ETH1PDACK	ETH1INTPOL	ETH1_SEL_PHY[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	r	r	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC1_IN7_REMAP	ADC1_IN6_REMAP	ADC1_IN5_REMAP	ADC1_IN2_REMAP
												rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ETH1TXLPI**: Ethernet TxLPI status
This bit is set and reset by hardware.
When set, this bit indicates that the MAC has entered Tx LPI mode.

Bit 29 **ETH1PDACK**: Ethernet power-down acknowledge
This bit is set and reset by hardware.
When set, this bit indicates that the Ethernet controller has complete its power-down sequence and its clock can be stopped.
0: Ethernet functional / power-down sequence not yet completed
1: Ethernet power-down sequence completed

Bit 28 **ETH1INTPOL**: Ethernet external PHY interrupt polarity configuration
0: active high
1: active low

Bits 27:24 **ETH1_SEL_PHY[3:0]**: Ethernet PHY interface selection
0000: GMII or MII
0100: RMII
Other: reserved

Bits 23:4 Reserved, must be kept at reset value.

Bit 3 **ADC1_IN7_REMAP**: ADC1 channel IN7 remap
0: ADC1_IN7 is mapped on PA7
1: ADC1_IN7 is mapped on PB1
Note: This bit is only available for STM32C53x/542 device.

Bit 2 **ADC1_IN6_REMAP**: ADC1 channel IN6 remap
0: ADC1_IN6 is mapped on PA6
1: ADC1_IN6 is mapped on PB0
Note: This bit is only available for STM32C53x/542 device.

Bit 1 **ADC1_IN5_REMAP**: ADC1 channel IN5 remap
0: ADC1_IN5 is mapped on PA5
1: ADC1_IN5 is mapped on PC5
Note: This bit is only available for STM32C53x/542 device.

Bit 0 **ADC1_IN2_REMAP**: ADC1 channel IN2 remap
0: ADC1_IN2 is mapped on PA2
1: ADC1_IN2 is mapped on PC4
Note: This bit is only available for STM32C53x/542 device.

12.5.5 SBS FPU interrupt mask register (SBS_FPUIMR)

Address offset: 0x104

Reset value: 0x0000 001F

Reset: system reset

This register is only accessible through a privilege transaction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **FPU_IE[5:0]**: FPU interrupt enable
This bitfield is set and cleared by software to enable the Cortex-M33 FPU interrupts.
FPU_IE[5]: Inexact interrupt enable (interrupt disabled at reset)
FPU_IE[4]: Input abnormal interrupt enable
FPU_IE[3]: Overflow interrupt enable
FPU_IE[2]: Underflow interrupt enable
FPU_IE[1]: Divide-by-zero interrupt enable
FPU_IE[0]: Invalid operation interrupt enable

12.5.6 SBS memory erase status register (SBS_MESR)

Address offset: 0x108

Reset value: 0x000X 000X

Note: Bits 0 and 16 are not affected by system reset.

Register security: always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
															rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IPMEE**: ICACHE erase status

This bit is set by hardware when an ICACHE erase is completed after potential tamper detection, or a product state regression (refer to *Section 40: Tamper and backup registers (TAMP)* for more details). This bit is cleared by software by writing 1 to it.

0: ICACHE erase ongoing

1: ICACHE erase ended

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MCLR**: Device memories erase status

This bit is set by hardware when a SRAM2 and ICACHE erase is completed after power-on reset or tamper detection, or product state regression (refer to *Section 40: Tamper and backup registers (TAMP)* for more details).

This bit is not reset by system reset and is cleared by software by writing 1 to it.

0: Memory erase ongoing if not yet cleared by software

1: Memory erase done

12.5.7 SBS compensation cell for I/Os control and status register (SBS_CCCSR)

Address offset: 0x110

Reset value: 0x0000 0002

Reset: system reset

Register security: always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
							r							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RDY1**: VDDIO compensation cell ready flag
This bit provides the status of the compensation cell.
0: VDDIO compensation cell not ready
1: VDDIO compensation cell ready (code value provided by the cell can be used)

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CS1**: Code selection for VDDIO power rail (reset value set to 1)
This bit selects the code to be applied for the I/O compensation cell.
0: Code from the cell (available in the SBS_CCVALR)
1: Code from SBS_CCWCR

Bit 0 **EN**1: Enable compensation cell for VDDIO power rail
This bit enables the I/O compensation cell.
0: I/O compensation cell disabled
1: I/O compensation cell enabled

12.5.8 SBS compensation cell for I/Os value register (SBS_CCVALR)

Address offset: 0x114

Reset value: 0x0000 00XX

Reset: system reset

Register security: always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APSRC1[3:0]				ANSRC1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **APSRC1[3:0]**: Compensation value for the PMOS transistor
This value is provided by the cell ,and the processor must interpret it to compensate the slew rate in the functional range.

Bits 3:0 **ANSRC1[3:0]**: Compensation value for the NMOS transistor
This value is provided by the cell, and the processor must interpret it to compensate the slew rate in the functional range.

12.5.9 SBS compensation cell for I/Os software code register (SBS_CCSWCR)

Address offset: 0x118

Reset value: 0x0000 0078

Reset: system reset

Register security: always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_APSRC1[3:0]				SW_ANSRC1[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SW_APSRC1[3:0]**: PMOS compensation code for the VDD power rails
This bitfield is written by software to define an I/O compensation cell code for PMOS transistors of the VDDIO power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.

Bits 3:0 **SW_ANSRC1[3:0]**: NMOS compensation code for VDD power rails
This bitfield is written by software to define an I/O compensation cell code for NMOS transistors of the VDD power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.

12.5.10 SBS Class B register (SBS_CFGR2)

Address offset: 0x120

Reset value: 0x0000 0000

Reset: system reset

Register security: always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SEL	CLL
												rw	rs	rs	rs

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCL**: ECC lock
This bit is set and cleared by software. It can be used to enable and lock the flash memory double ECC error with break input of TIM1/8/15/16/17.
0: Double ECC error flag disconnected to timer break inputs
1: Double ECC error flag connected to timer break inputs

Bit 2 **PVDL**: PVD lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection with TIM1/8/15/16/17 break inputs.

0: PVD interrupt disconnected from timer break inputs. PVD_EN and PVD_SEL[2:0] in PWR registers are read/write.

1: PVD interrupt is connected to timer break inputs. PVD_EN and PVD_SEL[2:0] in PWR registers are read only

Bit 1 **SEL**: SRAM ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM double ECC error signal with break input of TIM1/8/15/16/17.

0: SRAM double ECC error flag disconnected from timer break inputs

1: SRAM double ECC error flag connected to timer break inputs

Bit 0 **CLL**: Core lockup lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the lockup (HardFault) output of the Cortex-M33 with TIM1/8/15/16/17 break inputs.

0: Lockup output disconnected from timer break inputs

1: Lockup output connected to timer break inputs

12.5.11 SBS CPU lock register (SBS_CLCKR)

Address offset: 0x144

Reset value: 0x0000 0000

Reset: system reset

Register security: This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

This register is used to lock the configuration of the MPU and VTOR registers of the Cortex-M33.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKMPU	LOCKVTOR
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LOCKMPU**: MPU register lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to MPU_CTRL, MPU_RNR, and MPU_RBAR registers.

0: MPU registers write enabled

1: MPU registers write disabled

Bit 0 **LOCKVTOR**: VTOR register lock

This bit is set by software and cleared only by a system reset.

0: VTOR register write enabled

1: VTOR register write disabled

12.5.12 SBS ECC NMI mask register (SBS_ECCNMIR)

Address offset: 0x14C

Reset value: 0x0000 0000

Reset: system reset

Register security: This register is only accessible through a privilege transaction.

This register sets up the expected behavior on NMI regarding double ECC errors from the flash memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI_MASK_EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ECCNMI_MASK_EN**: NMI behavior setup when a double ECC error occurs on FLASH data part

0: NMI generated if a double ECC error in the FLASH data part

1: NMI not generated if a double ECC error in the FLASH data part

12.5.13 SBS register map

Table 83. SBS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SBS_HDPLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR_HDPL[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x014	SBS_HDPLSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									0	1	0	1	0	0	0	1
0x018	SBS_NEXTHDPLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NEXTHDPL[1:0]	
	Reset value																															0	0
0x01C-0x0FC	Reserved																																
0x100	SBS_PMCR	Res.	ETHTXLPI	ETHPDACK	ETHINTPOL	ETH_SEL_PHY[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC1_IN7_REMAP	ADC1_IN6_REMAP	ADC1_IN5_REMAP	ADC1_IN2_REMAP
	Reset value		0	0	0		0	0	0																					0	0	0	0
0x104	SBS_FPUIMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
	Reset value																											0	1	1	1	1	1

Table 83. SBS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x108	SBS_MESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
	Reset value																0																0
0x110	SBS_CCCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
	Reset value																								0							1	0
0x114	SBS_CCVALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APSRC1 [3:0]				ANSRC1 [3:0]			
	Reset value																									X	X	X	X	X	X	X	X
0x118	SBS_CCSWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_APSRC1 [3:0]				SW_ANSRC1 [3:0]			
	Reset value																									0	1	1	1	1	0	0	0
0x120	SBS_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SEL	CLL
	Reset value																													0	0	0	0
0x124-0x140	Reserved																																
0x144	SBS_CLCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKMPU	LOCKVTOR
	Reset value																															0	0
0x14C	SBS_ECCNMIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI_MASK_EN
	Reset value																																0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

13 Peripherals interconnect matrix

13.1 Interconnect matrix introduction

Several peripherals have direct connections between them, enabling autonomous communication and/or synchronization: this approach saves CPU resources and power supply consumption. In addition, these hardware connections remove software latency and help the design of a predictable system.

Depending on peripherals, these interconnections can operate in Run, Sleep, and Stop modes.

13.2 Connection summary

There are several types of connections.

- Asynchronous connections (A)

  The source output signal is sampled by the destination clock, leading to introduction of a possible jitter in the latency between the source output event and the destination event detection
- Synchronous connections (S)

  Both source and destination are synchronous (they run on the same clock), and the latency from the source to the destination is deterministic. No jitter is introduced.
- Immediate connections (I)

  Either the source or the destination is an analog signal.
- Break/fault connection for TIM outputs (B)

  The source output signal disables the timer outputs through a pure combinational logic path, without any latency.

Table 84. Peripherals interconnect matrix(1)(2)

Source	Destination																
	TIM1	TIM2	TIM5	TIM6	TIM7	TIM8	TIM12	TIM15	TIM16	TIM17	LPTIM1	CRS	LPDMA1	LPDMA2	ADC	DAC	COMP
TIM1	-	S	S	-	-	S/B	S	S/B	S/B	S/B	-	-	-	-	S/A	S/A	A
TIM2	S	-	S	-	-	S	S	S	S	S	-	-	-	-	S/A	S/A	A
TIM5	S	S	-	-	-	S	S	S	S	S	-	-	-	-	S/A	S/A	A
TIM6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S/A	S/A	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S/A	S/A	-
TIM8	S/B	S	S	-	-	-	S	S/B	S/B	S/B	-	-	-	-	S/A	S/A	I
TIM12	S	S	S	-	-	S	-	S	S	S	-	-	-	-	S/A	S/A	-
TIM15	S/B	S	S	-	-	S/B	S	-	S/B	S/B	-	-	S	S	S/A	S/A	A
TIM16	S/B	S	S	-	-	S/B	S	S/B	-	S/B	-	-	-	-	-	-	-
TIM17	S/B	S	S	-	-	S/B	S	S/B	S/B	-	-	-	-	-	-	-	-
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	-	A	A	A	A	-
LPDMA1	-	-	-	-	-	-	-	-	-	-	-	-	S	S	-	-	-
LPDMA2	-	-	-	-	-	-	-	-	-	-	-	-	S	S	-	-	-
ADC	A	A	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-
COMP	A/B	A	A	-	-	A/B	A	A/B	A/B	A/B	-	-	-	-	-	-	-
Temp Sensor	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I	-	-
Vrefint	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I	-	-
FDCAN	-	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
USB_FS	-	-	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-
I3C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RTC	-	A	-	-	-	-	-	-	A	-	A	-	A	A	-	-	-
TAMP	-	-	-	-	-	-	-	-	-	-	A	-	A	A	-	-	-

Table 84. Peripherals interconnect matrix[1][2] (continued)

Source	Destination																
	TIM1	TIM2	TIM5	TIM6	TIM7	TIM8	TIM12	TIM15	TIM16	TIM17	LPTIM1	CRS	LPDMA1	LPDMA2	ADC	DAC	COMP
HSE	-	A	-	-	-	-	A	-	-	A	A	A	-	-	-	-	-
LSE	-	A	-	-	-	-	-	A	A	-	A	A	-	-	-	S	-
LSI	-	A	-	-	-	-	-	-	A	-	A	-	-	-	-	-	-
MCO1	-	A	-	-	-	-	A	-	-	A	A	-	-	-	-	-	-
MCO2	-	A	-	-	-	-	A	-	-	A	-	-	-	-	-	-	-
SYS Error	B	-	-	-	-	B	-	-	-	-	-	-	-	-	-	-	-

1. "-" means no interconnect.
2. Refer to the product datasheet for the availability of peripherals. If the peripheral is not present, consider the associated interconnection as unavailable.

13.3 Interconnection details

Table 85. peripheral interconnect matrix details

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
TIM2	tim2_trgo	tim_itr2	TIM1
TIM5	tim5_trgo	tim_itr5	
TIM8	tim8_trgo	tim_itr6	
TIM8	tim8_trgo2	tim_itr7	
TIM12	tim12_trgo	tim_itr8	
TIM15	tim15_trgo	tim_itr9	
TIM16	tim16_oc1	tim_itr10	
TIM17	tim17_oc1	tim_itr11	
COMP1	comp1_out	tim_etr1	
ADC1	adc1_awd1	tim_etr3	
ADC1	adc1_awd2	tim_etr4	
ADC1	adc1_awd3	tim_etr5	
COMP1	comp1_out	tim_ti1_in1	
COMP1	comp1_out	tim_ocref_clr0	
GPIO	BKIN	tim_brk_0	
COMP1	comp1_out	tim_brk_1	
TIM8	TIM8_BKIN	tim_brk_6	
TIM15	TIM15_BKIN	tim_brk_8	
TIM16	TIM16_BKIN	tim_brk_9	
TIM17	TIM17_BKIN	tim_brk_10	
GPIO	BKIN2	tim_brk2_0	
COMP1	comp1_out	tim_brk2_1	
TIM8	TIM8_BKIN2	tim_brk2_6	
-	PWM_SAFE LOCKUP PVD RAM ECC	tim_brk	
TIM1	tim1_trgo	tim_itr0	TIM2
TIM1	tim1_trgo2	tim_itr1	
TIM5	tim5_trgo	tim_itr5	
TIM8	tim8_trgo	tim_itr6	
TIM8	tim8_trgo2	tim_itr7	

Table 85. peripheral interconnect matrix details (continued)

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
TIM12	tim12_trgo	tim_itr8	TIM2 (continued)
TIM15	tim15_trgo	tim_itr9	
TIM16	tim16_oc1	tim_itr10	
TIM17	tim17_oc1	tim_itr11	
COMP1	comp1_out	tim_etr1	
ADC1	adc1_awd1	tim_etr3	
ADC1	adc1_awd2	tim_etr4	
ADC1	adc1_awd3	tim_etr5	
RCC	LSE	tim_etr6	
RCC	MCO1	tim_etr7	
TIM5	TIM5_ETR	tim_etr11	
COMP1	comp1_out	tim_ti1_in1	
RCC	LSI	tim_ti1_in3	
RCC	LSE	tim_ti1_in4	
RTC	rtc_wut_trg	tim_ti1_in5	
TIM5	TIM5_CH1	tim_ti1_in6	
FDCAN1	fdcan1_rxeof_evt	tim_ti1_in7	
TIM2	TIM2_CH2	tim_ti2_in0	
RCC	hse_1M_ck	tim_ti2_in3	
RCC	MCO1	tim_ti2_in4	
RCC	MCO2	tim_ti2_in5	
FDCAN1	fdcan1_rxeof_evt	tim_ti2_in7	
COMP1	comp1_out	tim_ti4_in1	
TIM1	tim1_trgo	tim_itr0	TIM5
TIM1	tim1_trgo2	tim_itr1	
TIM2	tim2_trgo	tim_itr2	
TIM8	tim8_trgo	tim_itr6	
TIM8	tim8_trgo2	tim_itr7	
TIM12	tim12_trgo	tim_itr8	
TIM15	tim15_trgo	tim_itr9	
TIM16	tim16_oc1	tim_itr10	
TIM17	tim17_oc1	tim_itr11	
COMP1	comp1_out	tim_etr1	

Table 85. peripheral interconnect matrix details (continued)

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
COMP1	comp1_out	tim_ti1_in1	TIM5 (continued)
COMP1	comp1_out	tim_ocref_clr0	
TIM1	tim1_trgo	tim_itr0	TIM8
TIM1	tim1_trgo2	tim_itr1	
TIM2	tim2_trgo	tim_itr2	
TIM5	tim5_trgo	tim_itr5	
TIM12	tim12_trgo	tim_itr8	
TIM15	tim15_trgo	tim_itr9	
TIM16	tim16_oc1	tim_itr10	
TIM17	tim17_oc1	tim_itr11	
COMP1	comp1_out	tim_etr1	
ADC2	adc2_awd1	tim_etr3	
ADC2	adc2_awd2	tim_etr4	
ADC2	adc2_awd3	tim_etr5	
COMP1	comp1_out	tim_ti1_in1	
GPIO	BKIN	tim_brk_0	
COMP1	comp1_out	tim_brk_1	
TIM1	TIM1_BKIN	tim_brk_5	
TIM15	TIM15_BKIN	tim_brk_8	
TIM16	TIM16_BKIN	tim_brk_9	
TIM17	TIM17_BKIN	tim_brk_10	
GPIO	BKIN2	tim_brk2_0	
COMP1	comp1_out	tim_brk2_1	
TIM1	TIM1_BKIN2	tim_brk2_5	
-	PWM_SAFE LOCKUP PVD RAM ECC	tim_brk	
TIM1	tim1_trgo	tim_itr0	TIM12
TIM1	tim1_trgo2	tim_itr1	
TIM2	tim2_trgo	tim_itr2	
TIM5	tim5_trgo	tim_itr5	
TIM8	tim8_trgo	tim_itr6	

Table 85. peripheral interconnect matrix details (continued)

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
TIM8	tim8_trgo2	tim_itr7	TIM12 (continued)
TIM15	tim15_trgo	tim_itr9	
TIM16	tim16_oc1	tim_itr10	
TIM17	tim17_oc1	tim_itr11	
COMP1	comp1_out	tim_ti1_in1	
RCC	MCO1	tim_ti1_in3	
RCC	MCO2	tim_ti1_in4	
RCC	hse_1m_ck	tim_ti1_in5	
I3C1	i3c1_ibiack	tim_ti1_in7	
TIM1	tim1_trgo	tim_itr0	TIM15
TIM1	tim1_trgo2	tim_itr1	
TIM2	tim2_trgo	tim_itr2	
TIM5	tim5_trgo	tim_itr5	
TIM8	tim8_trgo	tim_itr6	
TIM8	tim8_trgo2	tim_itr7	
TIM12	tim12_trgo	tim_itr8	
TIM16	tim16_oc1	tim_itr10	
TIM17	tim17_oc1	tim_itr11	
COMP1	comp1_out	tim_ti1_in1	
RCC	LSE	tim_ti1_in5	
COMP1	comp1_out	tim_ocref_clr0	
GPIO	BKIN	tim_brk_0	
COMP1	comp1_out	tim_brk_1	
TIM1	TIM1_BKIN	tim_brk_5	
TIM8	TIM8_BKIN	tim_brk_6	
TIM16	TIM16_BKIN	tim_brk_9	
TIM17	TIM17_BKIN	tim_brk_10	
COMP1	comp1_out	tim_ti1_in1	TIM16
RCC	LSI	tim_ti1_in3	
RCC	LSE	tim_ti1_in4	
RTC	rtc_wut_trg	tim_ti1_in5	
RCC	MCO1	tim_ti1_in6	
RCC	MCO2	tim_ti1_in7	

Table 85. peripheral interconnect matrix details (continued)

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
COMP1	comp1_out	tim_ocref_clr0	TIM16 (continued)
GPIO	BKIN	tim_brk_0	
COMP1	comp1_out	tim_brk_1	
TIM1	TIM1_BKIN	tim_brk_5	
TIM8	TIM8_BKIN	tim_brk_6	
TIM15	TIM15_BKIN	tim_brk_8	
TIM17	TIM17_BKIN	tim_brk_10	
COMP1	comp1_out	tim_ti1_in1	TIM17
RCC	hse_1M_ck	tim_ti1_in4	
RCC	MCO1	tim_ti1_in5	
RCC	MCO2	tim_ti1_in6	
I3C1	i3c1_ibiack	tim_ti1_in7	
COMP1	comp1_out	tim_ocref_clr0	
GPIO	BKIN	tim_brk_0	
COMP1	comp1_out	tim_brk_1	
TIM1	TIM1_BKIN	tim_brk_5	
TIM8	TIM8_BKIN	tim_brk_6	
TIM15	TIM15_BKIN	tim_brk_8	
TIM16	TIM16_BKIN	tim_brk_9	
RTC	rtc_alra_trg	lptim_ext_trig1	LPTIM1
RTC	rtc_alrb_trg	lptim_ext_trig2	
TAMP	tamp_trg1	lptim_ext_trig3	
TAMP	tamp_trg2	lptim_ext_trig4	
LPDMA	lpdma_ch1_tcf	lptim_ext_trig5	
COMP1	comp1_out	lptim_ext_trig6	
CPU	EVENTOUT	lptim_ext_trig7	
COMP1	comp1_out	lptim_in1_mux1	
COMP1	comp1_out	lptim_ic1_mux1	
CPU	EVENTOUT	lptim_ic1_mux2	
RCC	MCO1	lptim_ic1_mux3	
RCC	LSI	lptim_ic2_mux1	
RCC	LSE	lptim_ic2_mux2	
RCC	RCC_HSE_1MHZ	lptim_ic2_mux3	

Table 85. peripheral interconnect matrix details (continued)

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
GPIO	CRS_SYNC	crs_sync_in_0	CRS
RCC	LSE	crs_sync_in_1	
USB	USB_SOF	crs_sync_in_2	
RCC	hse_1M_ck	crs_sync_in_3	
-	VREFINT	ADC1 VIN[13]	ADC12
-	VSENSE	ADC1 VIN[12]	
GPIO	exti11	adc_ext_trg0	
TIM1	tim1_oc1	adc_ext_trg1	
TIM1	tim1_oc2	adc_ext_trg2	
TIM1	tim1_oc3	adc_ext_trg3	
TIM1	tim1_trgo	adc_ext_trg4	
TIM1	tim1_trgo2	adc_ext_trg5	
TIM2	tim2_oc2	adc_ext_trg6	
TIM2	tim2_trgo	adc_ext_trg7	
TIM5	tim5_oc4	adc_ext_trg10	
TIM5	tim5_trgo	adc_ext_trg11	
TIM6	tim6_trgo	adc_ext_trg12	
TIM8	tim8_trgo	adc_ext_trg13	
TIM8	tim8_trgo2	adc_ext_trg14	
TIM15	tim15_trgo	adc_ext_trg15	
LPTIM1	lptim1_ch1	adc_ext_trg16	
GPIO	exti15	adc_jext_trg0	
TIM1	tim1_oc4	adc_jext_trg1	
TIM1	tim1_trgo	adc_jext_trg2	
TIM1	tim1_trgo2	adc_jext_trg3	
TIM2	tim2_oc1	adc_jext_trg4	
TIM2	tim2_trgo	adc_jext_trg5	
TIM5	tim5_oc1	adc_jext_trg10	
TIM5	tim5_oc2	adc_jext_trg11	
TIM5	tim5_oc3	adc_jext_trg12	
TIM5	tim5_trgo	adc_jext_trg13	
TIM7	tim7_trgo	adc_jext_trg14	
TIM8	tim8_oc4	adc_jext_trg15	

Table 85. peripheral interconnect matrix details (continued)

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
TIM8	tim8_trgo	adc_jext_trg16	ADC12 (continued)
TIM8	tim8_tgo2	adc_jext_trg17	
TIM12	tim12_tgo	adc_jext_trg18	
TIM15	tim15_trgo	adc_jext_trg19	
LPTIM1	lptim1_ch1	adc_jext_trg20	
RCC	ck_lsi OR ck_lse	dac_hold_ck	DAC
TIM1	tim1_trgo	dac_ch1_trg1	
TIM2	tim2_trgo	dac_ch1_trg2	
TIM5	tim5_trgo	dac_ch1_trg5	
TIM6	tim6_trgo	dac_ch1_trg6	
TIM7	tim7_trgo	dac_ch1_trg7	
TIM8	tim8_trgo	dac_ch1_trg8	
TIM12	tim12_trgo	dac_ch1_trg9	
TIM15	tim15_trgo	dac_ch1_trg10	
LPTIM1	lptim1_ch1	dac_ch1_trg11	
GPIO	exti0	lpdma_trg0	LPDMA12
GPIO	exti1	lpdma_trg1	
GPIO	exti2	lpdma_trg2	
GPIO	exti3	lpdma_trg3	
GPIO	exti4	lpdma_trg4	
GPIO	exti5	lpdma_trg5	
GPIO	exti6	lpdma_trg6	
GPIO	exti7	lpdma_trg7	
TAMP	tamp_trg1	lpdma_trg8	
TAMP	tamp_trg2	lpdma_trg9	
TAMP	tamp_trg3	lpdma_trg10	
LPTIM1	lptim1_ch1	lpdma_trg11	
LPTIM1	lptim1_ch2	lpdma_trg12	
RTC	rtc_alra_trg	lpdma_trg13	
RTC	rtc_alrb_trg	lpdma_trg14	
RTC	rtc_wut_trg	lpdma_trg15	
TIM2	tim2_trgo	lpdma_trg16	
TIM15	tim15_trgo	lpdma_trg17	

Table 85. peripheral interconnect matrix details (continued)

SOURCE (output)		DESTINATION (input)	
PERIPH	**SIGNAL**	**SIGNAL**	**PERIPH**
COMP1	comp1_out	lpdma_trg18	LPDMA12 (continued)
CPU	EVENTOUT	lpdma_trg19	
LPDMA1	lpdma1_ch0_tc	lpdma_trg20	
LPDMA1	lpdma1_ch1_tc	lpdma_trg21	
LPDMA1	lpdma1_ch2_tc	lpdma_trg22	
LPDMA1	lpdma1_ch3_tc	lpdma_trg23	
LPDMA1	lpdma1_ch4_tc	lpdma_trg24	
LPDMA1	lpdma1_ch5_tc	lpdma_trg25	
LPDMA1	lpdma1_ch6_tc	lpdma_trg26	
LPDMA1	lpdma1_ch7_tc	lpdma_trg27	
LPDMA2	lpdma2_ch0_tc	lpdma_trg28	
LPDMA2	lpdma2_ch1_tc	lpdma_trg29	
LPDMA2	lpdma2_ch2_tc	lpdma_trg30	
LPDMA2	lpdma2_ch3_tc	lpdma_trg31	
TIM1	tim1_oc5	comp1_blk1	COMP1
TIM2	tim2_oc3	comp1_blk2	
TIM5	tim5_oc3	comp1_blk3	
TIM5	tim5_oc4	comp1_blk4	
TIM8	tim8_oc5	comp1_blk5	
TIM15	tim15_oc2	comp1_blk6	

14 Low-power direct memory access controller (LPDMA)

14.1 LPDMA introduction

The low-power direct memory access (LPDMA) controller is a bus master and system peripheral.

The LPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

14.2 LPDMA main features

- Single bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Transfers arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- Up to 8 concurrent LPDMA1 channels, up to 8 concurrent LPDMA2 channels:
  - Intra-channel LPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel LPDMA transfers chaining via programmable LPDMA input triggers connection to LPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive single transfers
  - Programmable LPDMA request and trigger selection
  - Programmable LPDMA half-transfer and transfer-complete events generation

  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the LPDMA linked-list control registers
- Debug:
  - Channel suspend and resume support
  - Channel status reporting and event flags
- Privileged/unprivileged support:
  - Support for privileged and unprivileged LPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

14.3 LPDMA implementation

14.3.1 LPDMA instances

The number of available channels per instance are given in *Table 86: LPDMA1/2 channels implementation*. *Section 14.8: LPDMA registers* reflects the LPDMA registers, with x = 0 to 7 for the channel number.

LPDMA instances have the same registers, excepted for those with x = 4 to 7.

Any LPDMA instance is connected to the same requests and triggers.

Table 86.LPDMA1/2 channels implementation

LPDMA instance	STM32C53xxx/542xx	STM32C55xxx/562xx	STM32C59xxx/5A3xx
LPDMA1	x = 0 to 3	x = 0 to 7	x = 0 to 7
LPDMA2	x = 0 to 3	x = 0 to 3	x = 0 to 7

14.3.2 LPDMA channels

A given LPDMA channel x is implemented with the hardware parameters as per the table below.

Table 87.LPDMA1 channels implementation

Channel x	Hardware parameters		Features
	dma_fifo_ size[x]	dma_ addressing[x]	
x = 0 to 7	0	0	Channel x (x = 0 to 7) is implemented with: – no FIFO. Only a single source transfer cell is internally registered. – fixed/contiguously incremented addressing

Table 88.LPDMA2 channels implementation

Channel x	Hardware parameters		Features
	dma_fifo_ size[x]	dma_ addressing[x]	
x = 0 to 7	0	0	Channel x (x = 0 to 7) is implemented with: – no FIFO. Only a single source transfer cell is internally registered. – fixed/contiguously incremented addressing

14.3.3 LPDMA in low-power modes

The LPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 89. LPDMA1/2 autonomous mode and wake-up in low-power modes

Feature	Low-power modes
Wake-up	LPDMA1/2 in Sleep mode

14.3.4 LPDMA requests

A LPDMA request from a peripheral can be assigned to a LPDMA channel x, via the REQSEL[6:0] field in the *LPDMA channel x transfer register 2 (LPDMA_CxTR2)*, provided that SWREQ = 0.

The LPDMA requests mapping is specified in the table below.

Table 90. Programmed LPDMA1/2 request

LPDMA_CxTR2.REQSEL[6:0]	Selected LPDMA request
0	adc1_dma
1	adc2_dma
2	tim6_upd_dma
3	tim7_upd_dma
4	spi1_rx_dma
5	spi1_tx_dma
6	spi2_rx_dma
7	spi2_tx_dma
8	spi3_rx_dma
9	spi3_tx_dma
10	i2c1_rx_dma
11	i2c1_tx_dma
12	usart1_rx_dma
13	usart1_tx_dma

Table 90. Programmed LPDMA1/2 request (continued)

LPDMA_CxTR2.REQSEL[6:0]	Selected LPDMA request
14	usart2_rx_dma
15	usart2_tx_dma
16	usart3_rx_dma
17	usart3_tx_dma
18	uart4_rx_dma
19	uart4_tx_dma
20	uart5_rx_dma
21	uart5_tx_dma
22	lpuart1_rx_dma
23	lpuart1_tx_dma
24	tim1_cc1_dma
25	tim1_cc2_dma
26	tim1_cc3_dma
27	tim1_cc4_dma
28	tim1_upd_dma
29	tim1_trig_dma
30	tim1_com_dma
31	tim2_cc1_dma
32	tim2_cc2_dma
33	tim2_cc3_dma
34	tim2_cc4_dma
35	tim2_upd_dma
36	tim2_trig_dma
37	tim5_cc1_dma
38	tim5_cc2_dma
39	tim5_cc3_dma
40	tim5_cc4_dma
41	tim5_upd_dma
42	tim5_trig_dma
43	tim15_cc1_dma
44	tim15_cc2_dma
45	tim15_upd_dma
46	tim15_trig_dma
47	tim15_com_dma
48	tim16_cc1_dma

Table 90. Programmed LPDMA1/2 request (continued)

LPDMA_CxTR2.REQSEL[6:0]	Selected LPDMA request
49	tim16_upd_dma
50	tim17_cc1_dma
51	tim17_upd_dma
52	lptim1_ic1_dma
53	lptim1_ic2_dma
54	lptim1_ue_dma
55	cordic_rd_dma
56	cordic_wr_dma
57	i3c1_rx_dma
58	i3c1_tx_dma
59	i3c1_tc_dma
60	i3c1_rs_dma
61	aes_out_dma
62	aes_in_dma
63	hash_in_dma
64	i2c2_rx_dma
65	i2c2_tx_dma
66	tim8_cc1_dma
67	tim8_cc2_dma
68	tim8_cc3_dma
69	tim8_cc4_dma
70	tim8_upd_dma
71	tim8_trig_dma
72	tim8_com_dma
73	dac1_ch1_dma
74	dac1_ch2_dma
75	usart6_rx_dma
76	usart6_tx_dma
77	uart7_tx_dma
78	uart7_rx_dma
79	adc3_dma
80	tim3_cc1_dma
81	tim3_cc2_dma
82	tim3_cc3_dma
83	tim3_cc4_dma

Table 90. Programmed LPDMA1/2 request (continued)

LPDMA_CxTR2.REQSEL[6:0]	Selected LPDMA request
84	tim3_upd_dma
85	tim3_trgi_dma
86	tim4_cc1_dma
87	tim4_cc2_dma
88	tim4_cc3_dma
89	tim4_cc4_dma
90	tim4_upd_dma
91	tim4_trgi_dma
92	saes_out_dma
93	saes_in_dma
94	xspi1_dma

14.3.5 LPDMA block requests

Some LPDMA requests must be programmed as a block request, and not as a single request. Then the BREQ bit in *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* must be set for a correct LPDMA execution of the requested peripheral transfer at the hardware level.

The LPDMA block requests are listed in the table below.

Table 91. Programmed LPDMA1/2 request as a block request

LPDMA block requests
lptim1_ue_dma

14.3.6 LPDMA channels with peripheral early termination

A LPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 92. LPDMA1/2 channel with peripheral early termination

LPDMA channel x with peripheral early termination
LPDMA1: x = 0 and x = 7 LPDMA2: x = 0 and x = 7

This LPDMA support is activated when the channel x is programmed with LPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before the LPDMA has transferred the whole block (see *Section 14.4.14* for more details).

Table 93. Programmed LPDMA1/2 request with peripheral early termination

Programmed LPDMA channel x request with peripheral early termination
i3c1_rx_dma

14.3.7 LPDMA triggers

A LPDMA trigger can be assigned to a LPDMA channel x, via the TRIGSEL[5:0] field in the *LPDMA channel x transfer register 2 (LPDMA_CxTR2)*, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

The LPDMA triggers mapping is specified in the table below.

Table 94. Programmed LPDMA1/2 trigger

LPDMA_CxTR2.TRIGSEL[5:0]	Selected LPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trg1
9	tamp_trg2
10	tamp_trg3
11	lptim1_ch1
12	lptim1_ch2
13	rtc_alra_trg
14	rtc_alrb_trg
15	rtc_wut_trg
16	tim2_trgo
17	tim15_trgo
18	comp1_out_dma
19	eventout
20	lpdma1_ch0_tc
21	lpdma1_ch1_tc
22	lpdma1_ch2_tc
23	lpdma1_ch3_tc
24	lpdma1_ch4_tc

Table 94. Programmed LPDMA1/2 trigger (continued)

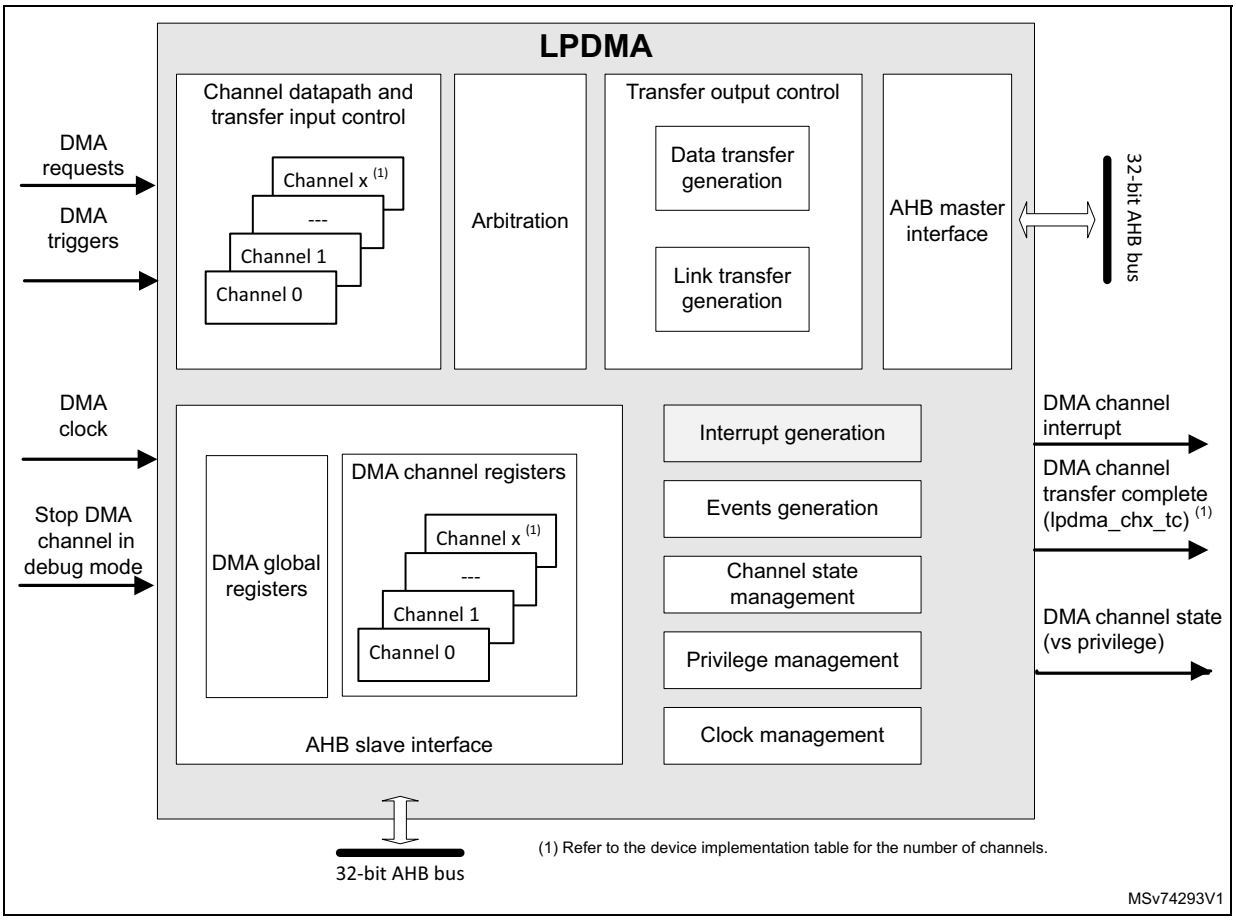
LPDMA_CxTR2.TRIGSEL[5:0]	Selected LPDMA trigger
25	lpdma1_ch5_tc
26	lpdma1_ch6_tc
27	lpdma1_ch7_tc
28	lpdma2_ch0_tc
29	lpdma2_ch1_tc
30	lpdma2_ch2_tc
31	lpdma2_ch3_tc
32	lpdma2_ch4_tc
33	lpdma2_ch5_tc
34	lpdma2_ch6_tc
35	lpdma2_ch7_tc

14.4 LPDMA functional description

14.4.1 LPDMA block diagram

The LPDMA block diagram is illustrated in the figure below.

Figure 40. LPDMA block diagram



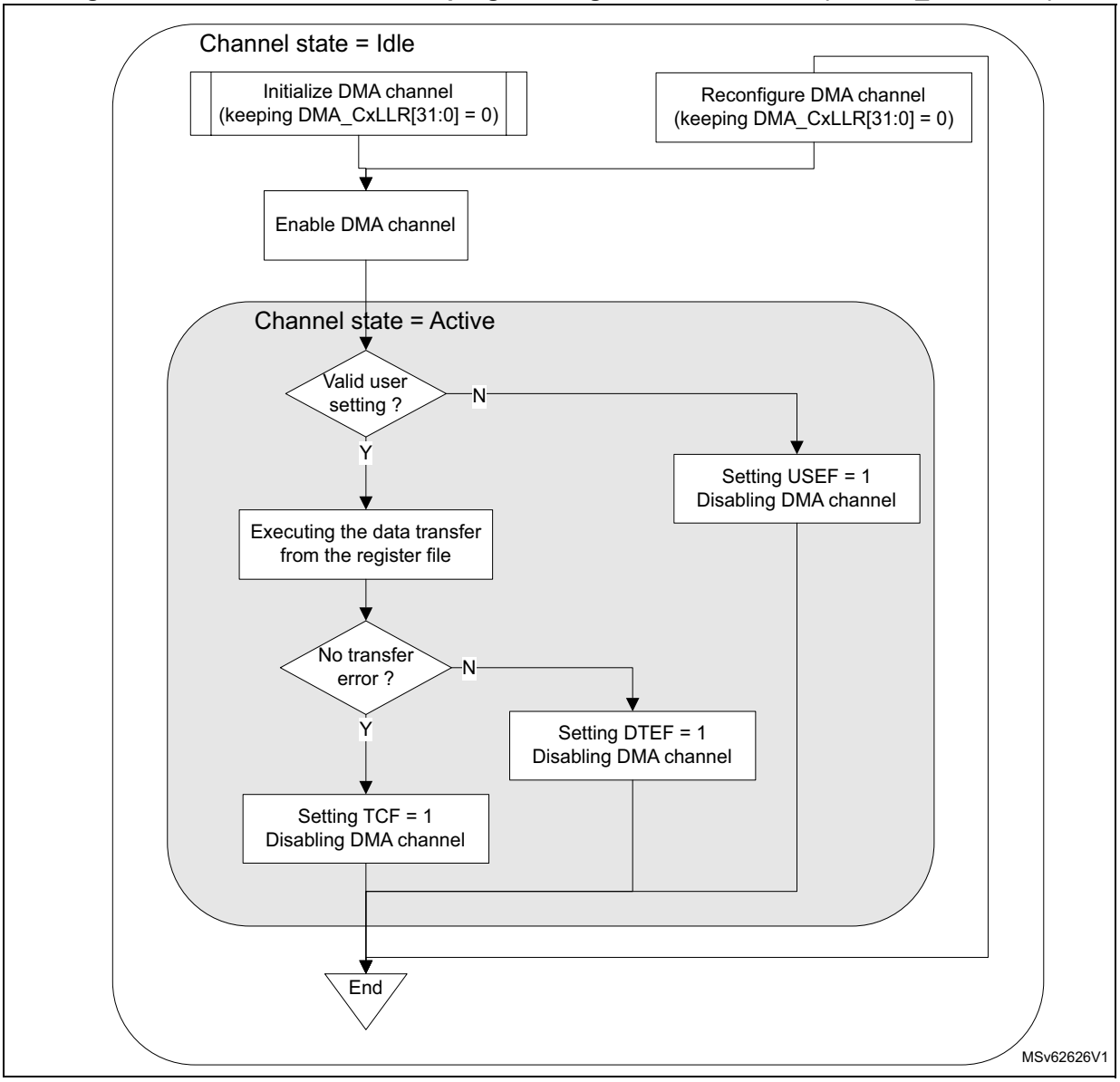
14.4.2 LPDMA channel state and direct programming without any linked-list

After a LPDMA reset, a LPDMA channel x is in idle state. When the software writes 1 into the in LPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (LPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a LPDMA channel transfer by writing 1 into the LPDMA_CxCR.EN bit, a LPDMA channel interrupt on a complete transfer notifies the software that the LPDMA channel is back in idle state (EN is then de-asserted by hardware) and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this LPDMA direct programming without any linked-list (LPDMA_CxLLR = 0).

Figure 41. LPDMA channel direct programming without linked-list (LPDMA_CxLLR = 0)



14.4.3 LPDMA channel suspend and resume

The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the LPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag LPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to LPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of

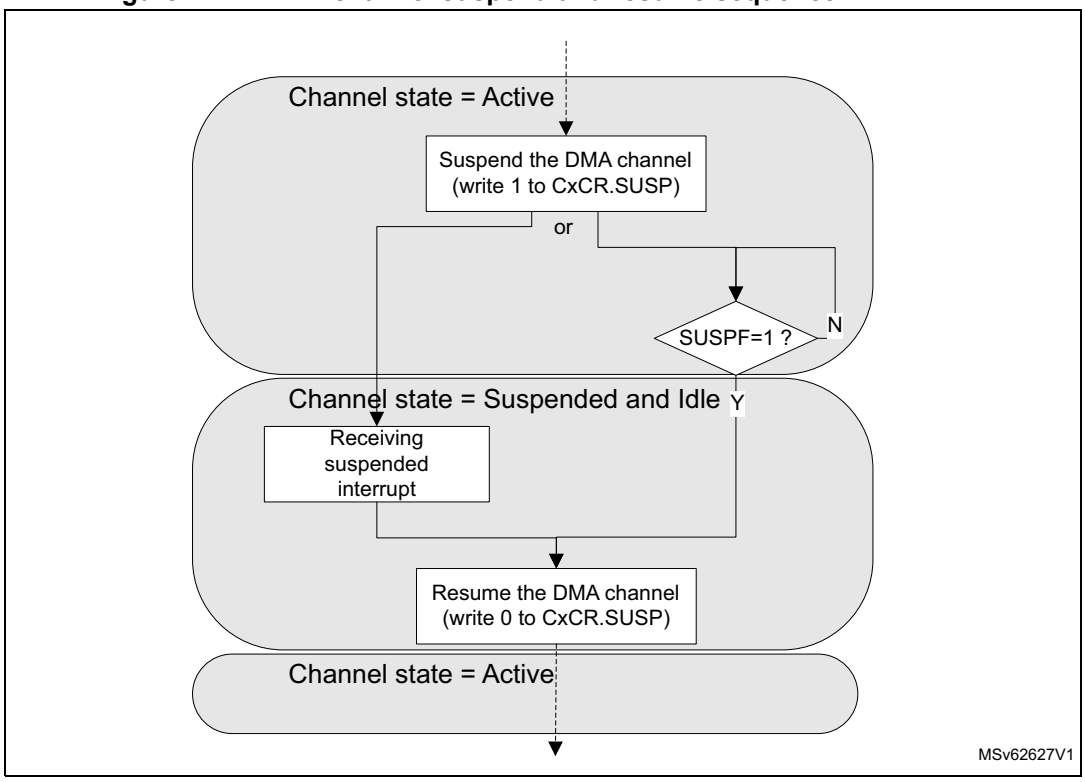
any ongoing LPDMA transfer over its master port. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note that an ongoing LPDMA transfer can be a data transfer (a read-followed-by-write single transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to LPDMA_CxCR.SUSP.

The suspend and resume sequence is illustrated in the figure below.

Figure 42. LPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the LPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

14.4.4 LPDMA channel abort and restart

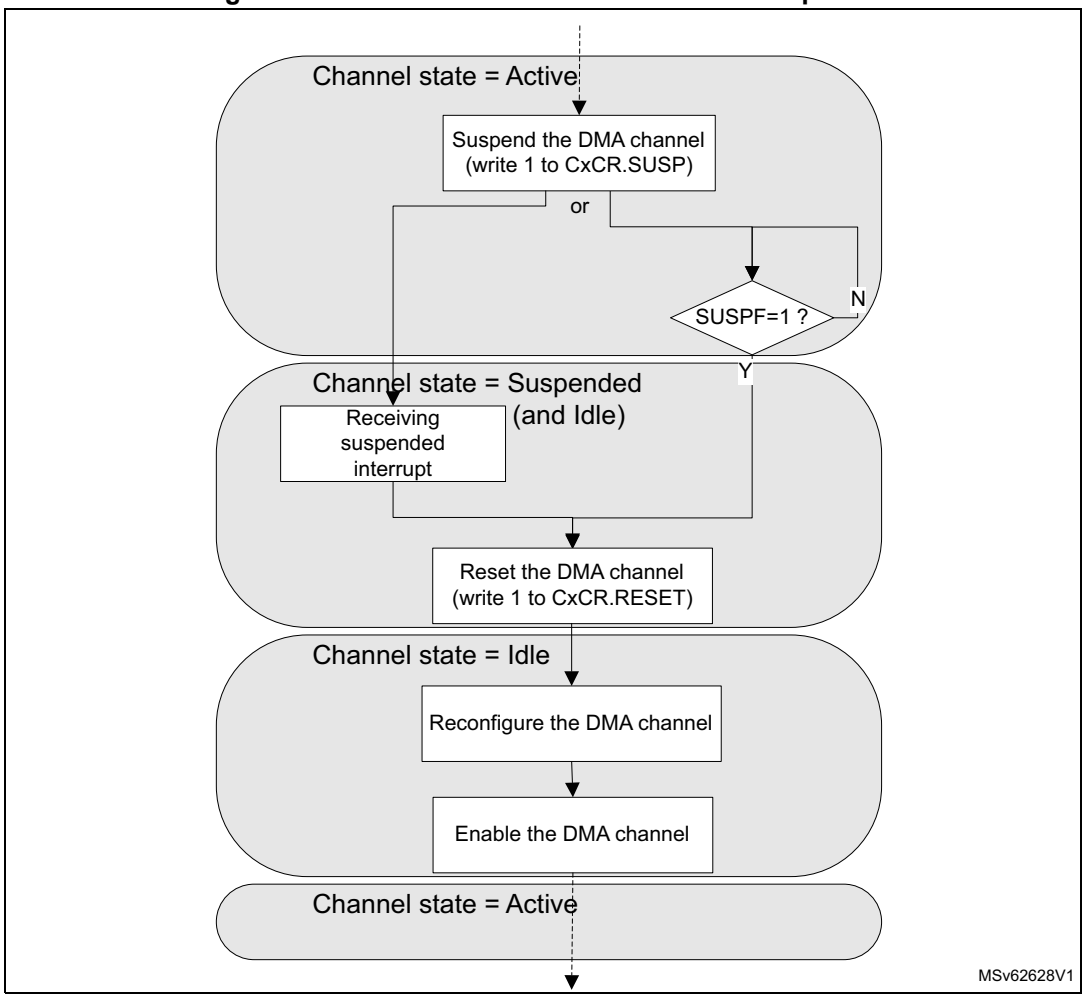
Alternatively, like for aborting a continuous LPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the LPDMA_CxCR.SUSP bit.
2. The software polls suspended flag LPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to LPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing LPDMA transfer over its master port.
3. The software resets the channel by writing 1 to LPDMA_CxCR.RESET. This causes the reset of the channel internal state, the reset of the LPDMA_CxCR.EN bit, and the reset of the LPDMA_CxCR.SUSP bit.

4. The software safely reconfigures the channel. The software must reprogram the hardware-modified LPDMA_CxBR1, LPDMA_CxSAR, and LPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the LPDMA_CxCR.EN bit.

The abort and restart sequence is illustrated in the figure below.

Figure 43. LPDMA channel abort and restart sequence



14.4.5 LPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in LPDMA_CxLBAR)
- the link address offset (LA[15:2] field in LPDMA_CxLLR)

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits of the LPDMA_CxLLR register.

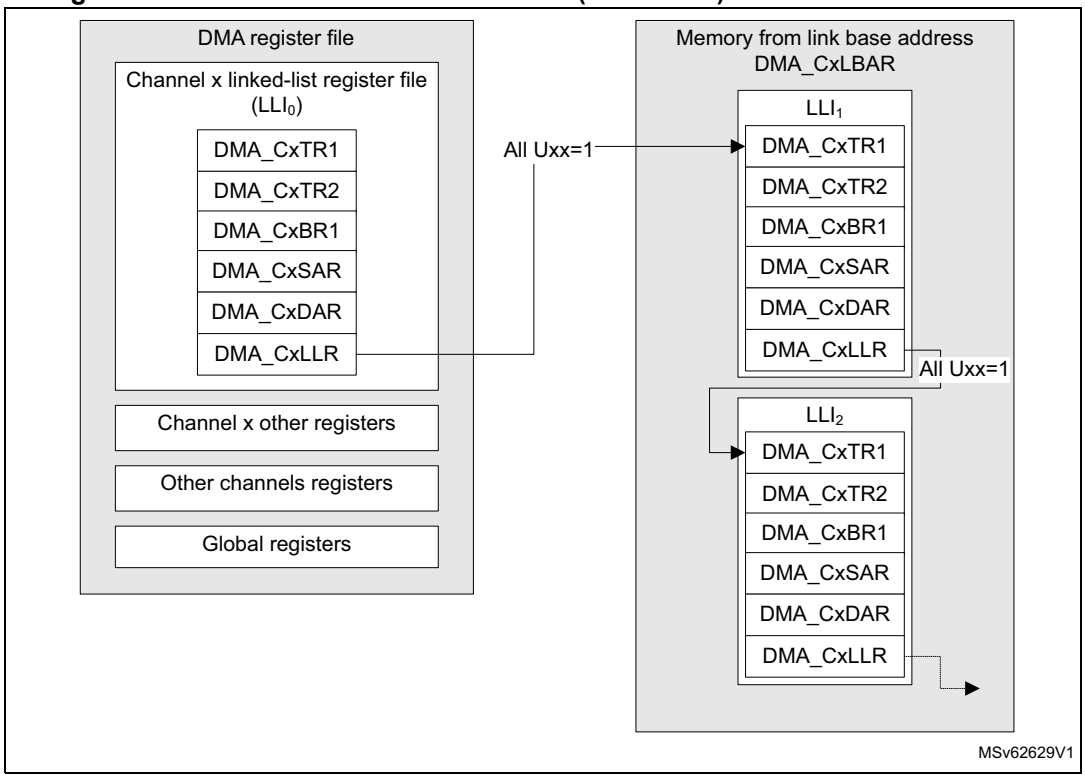
In linked-list mode, each LPDMA linked-list register (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR or LPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the LPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (LPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (LPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL) of the LPDMA_CxLLR register are all asserted, the linked-list data structure in memory is maximal with six contiguous 32-bit locations, including LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR (see the figure below) and including the first linked-list register file (LLI_0) and the next LLIs (LLI_1, LLI_2,...) in the memory.

Figure 44. Static linked-list data structure (all Uxx = 1) of channel x



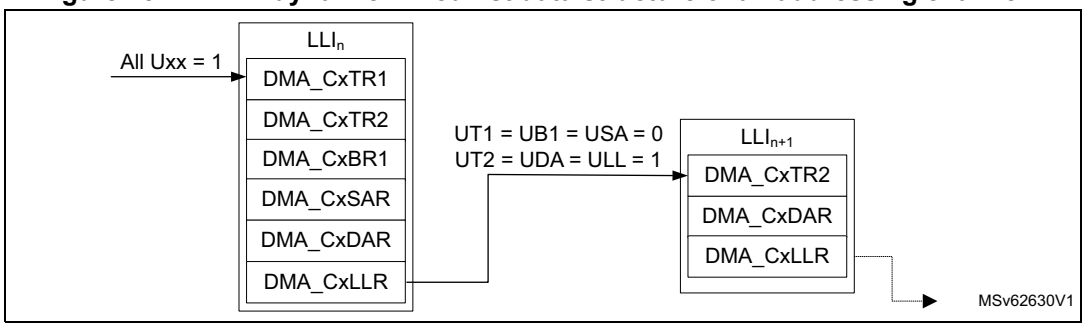
Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register LPDMA_CxLLR is pointing to the updated value of the LPDMA_CxTR2 instead of the LPDMA_CxTR1 which is not to be modified.

Example: if UT1 = UB1 = USA = 0, and if UT2 = UDA = ULL = 1, the next LLI does not contain an updated value for LPDMA_CxTR1, nor LPDMA_CxBR1, nor LPDMA_CxSAR. The next LLI contains an updated value for LPDMA_CxTR2, LPDMA_CxDAR, and LPDMA_CxLLR, as shown in the figure below.

Figure 45. LPDMA dynamic linked-list data structure of an addressing channel x



14.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: LPDMA executes the data transfer as described by the LPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: LPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the LPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (LPDMA_CxBR1.BNDT[15:0] = 0 and LPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in LPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended LPDMA usage, a LPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, LPDMA_CxCR.LSM = 0) or a LPDMA channel x can be programmed for a single execution of a LLI (link step mode, LPDMA_CxCR.LSM = 1), as described in the next paragraphs.

14.4.7 LPDMA channel state and linked-list programming in run-to-completion mode

When LPDMA_CxCR.LSM = 0, a LPDMA channel x is initially programmed, started by writing 1 to LPDMA_CxCR.EN, and after (possibly) completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the LPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when LPDMA_CxLLR[31:0] = 0 and LPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

LPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits LPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits Uxx of LPDMA_CxLLR are null (UT1, UT2, UB1, USA, UDA, ULL).

The channel may never be completed when LPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either LPDMA_CxLLR.ULL = 1 and LPDMA_CxLLR.LA[15:2] is updated with the same value
  - or LPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI.

In the regular data transfer completion at a block level, LPDMA_CxBR1.BNDT[15:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 14.4.14* for more details).

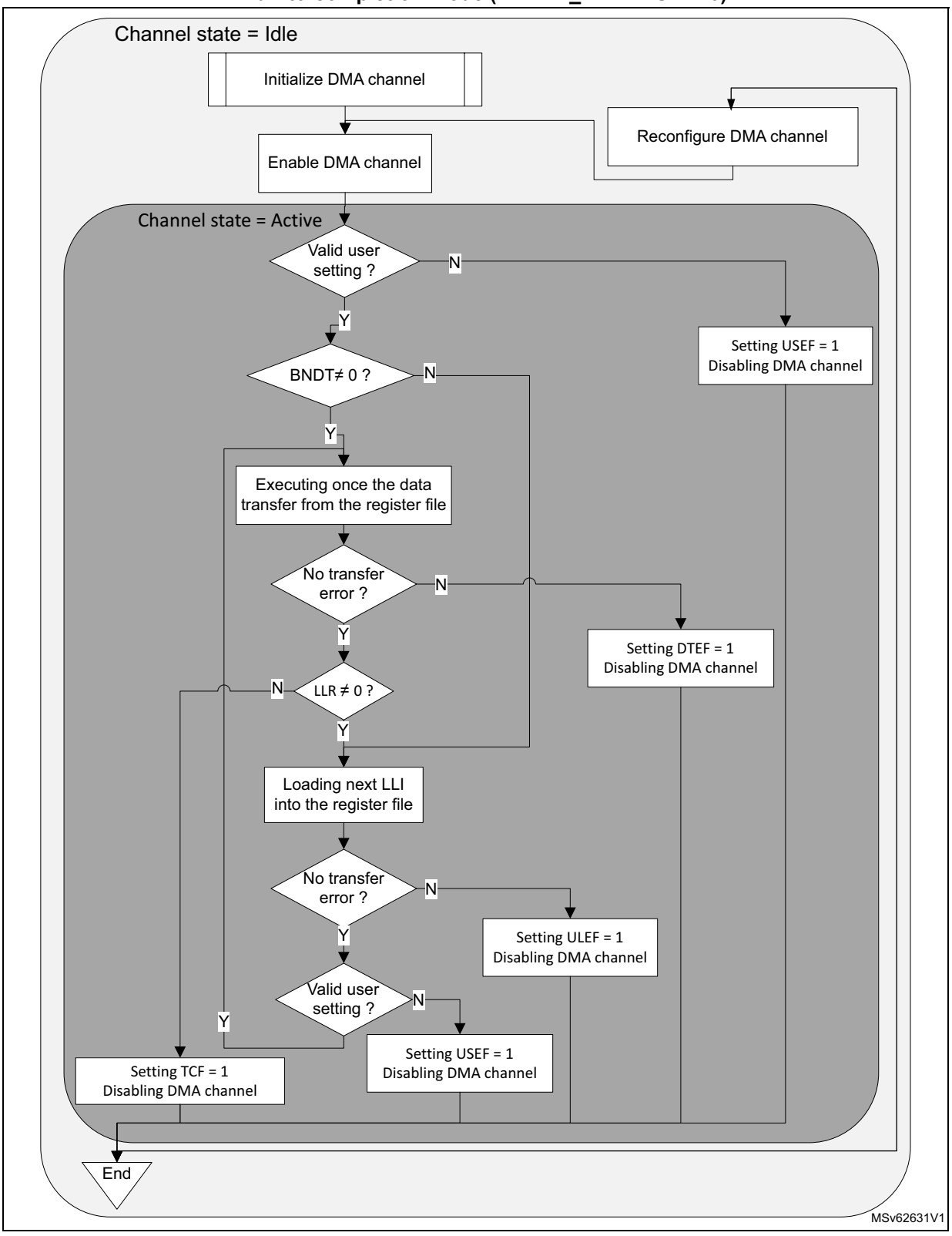
In the typical run-to-completion mode, the allocation of a LPDMA channel, including its fine programming, is done once during the LPDMA initialization. In order to have a reserved data communication link and LPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 46 depicts the LPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 46 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in LPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 46, BNDT ? 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ? 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 14.3.6).

Figure 46. LPDMA channel execution and linked-list programming in run-to-completion mode (LPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the LPDMA_CxTR2 in the data structure of the LLI_{n-1}:

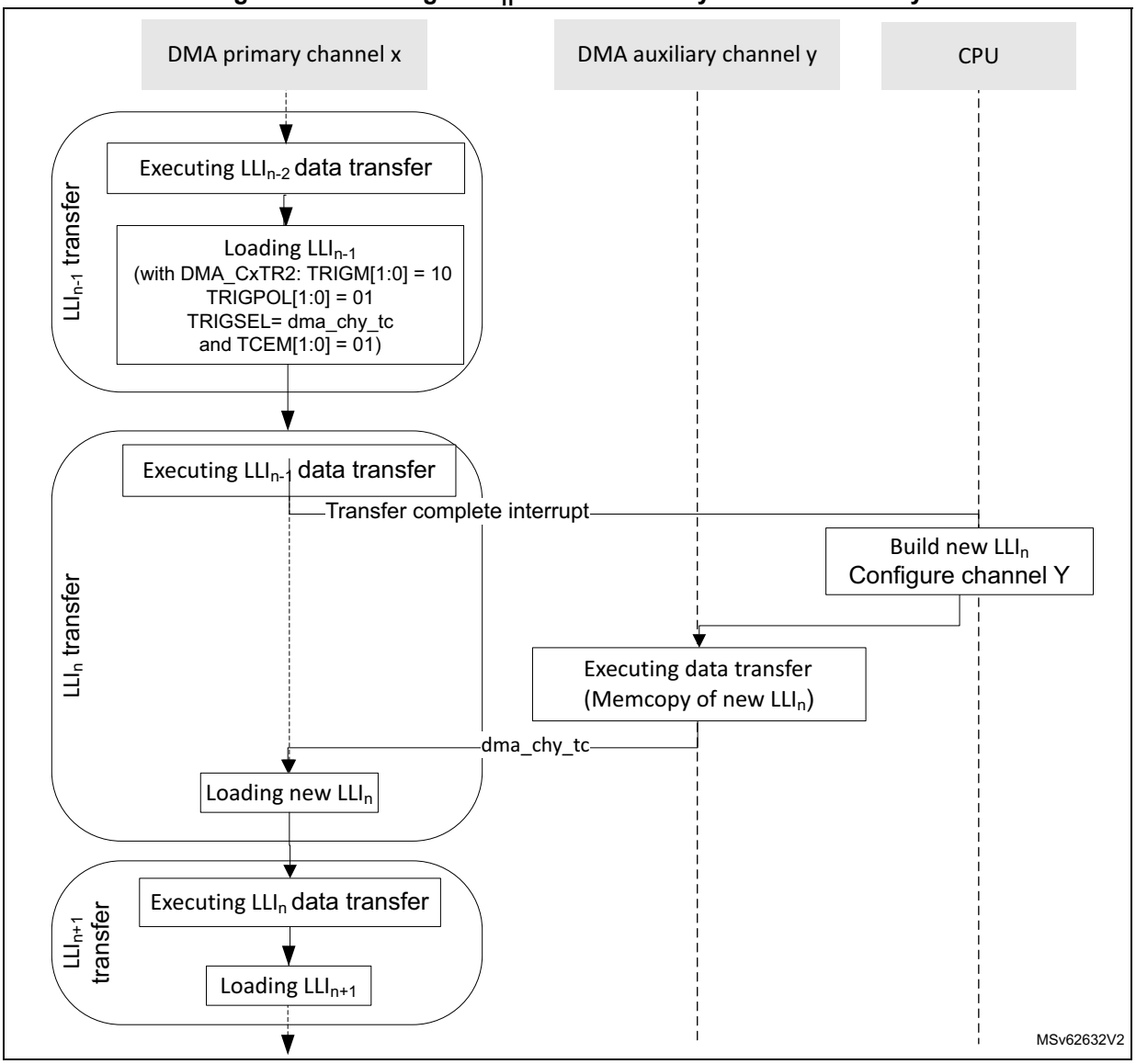
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 14.3.7* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event lpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 47 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 47. Inserting a LLI_n with an auxiliary LPDMA channel y



14.4.8 LPDMA channel state and linked-list programming in link step mode

When LPDMA_CxCR.LSM = 1, a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A LPDMA channel transfer can be programmed at LLI level, started by writing 1 into LPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - LPDMA_CxTR1 defines the source/destination elementary single transfers.
  - LPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]).
  - LPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - LPDMA_CxSAR/LPDMA_CxDAR defines the source/destination transfer start address.
  - LPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when LPDMA_CxBR1.BNDT[15:0] = 0)
  - after the (conditional) update of the LPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either LPDMA_CxLLR.ULL = 1 and LPDMA_CxLLR.LA[15:2] is updated with the same value, or LPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

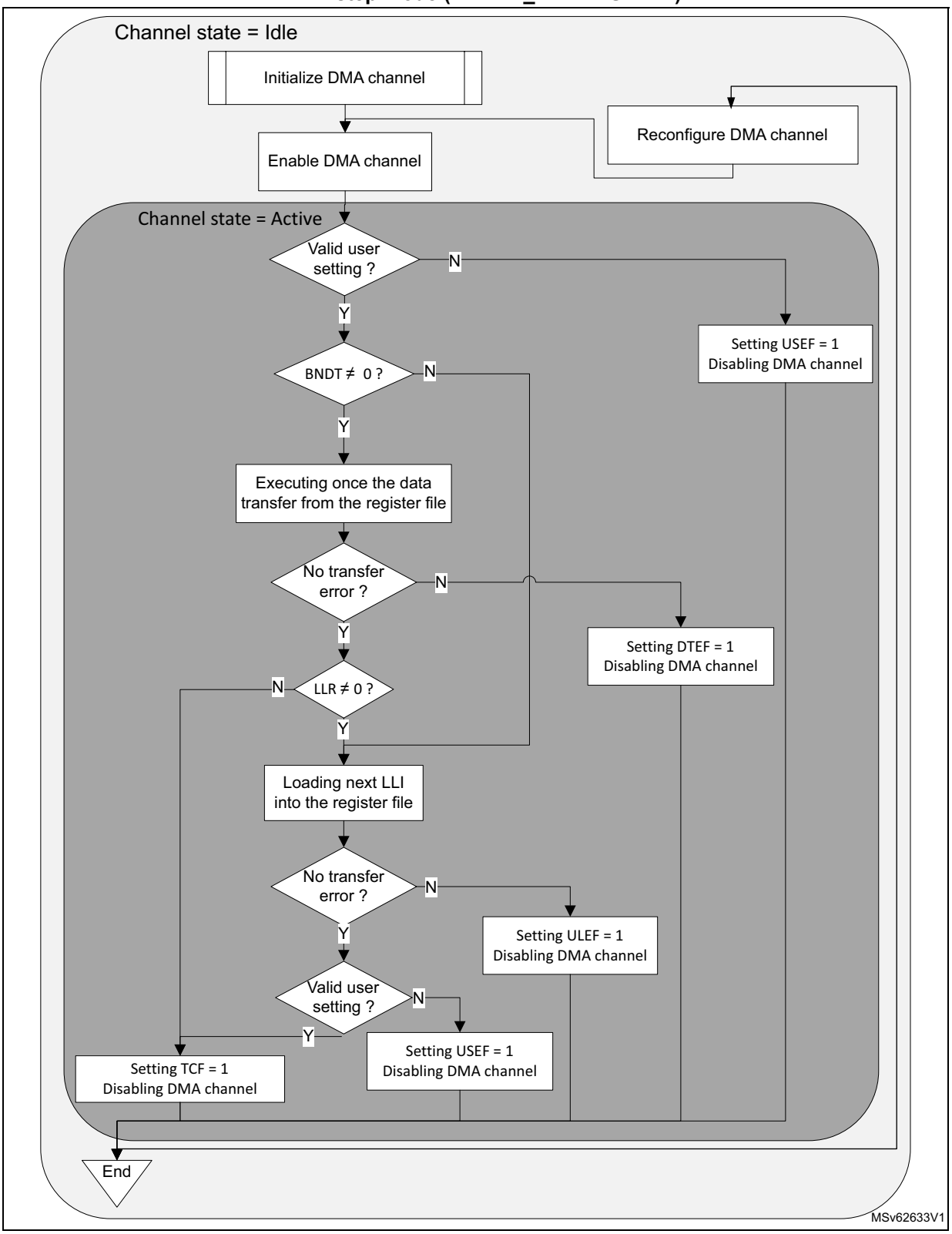
In the regular data transfer completion at a block level, LPDMA_CxBR1.BNDT[15:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 14.4.14* for more details).

Figure 48 depicts the LPDMA channel execution mode, and its programming in link step mode.

Note: Figure 48 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in LPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 48, BNDT ? 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ? 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 14.3.6).

Figure 48. LPDMA channel execution and linked-list programming in link step mode (LPDMA_CxCR.LSM = 1)

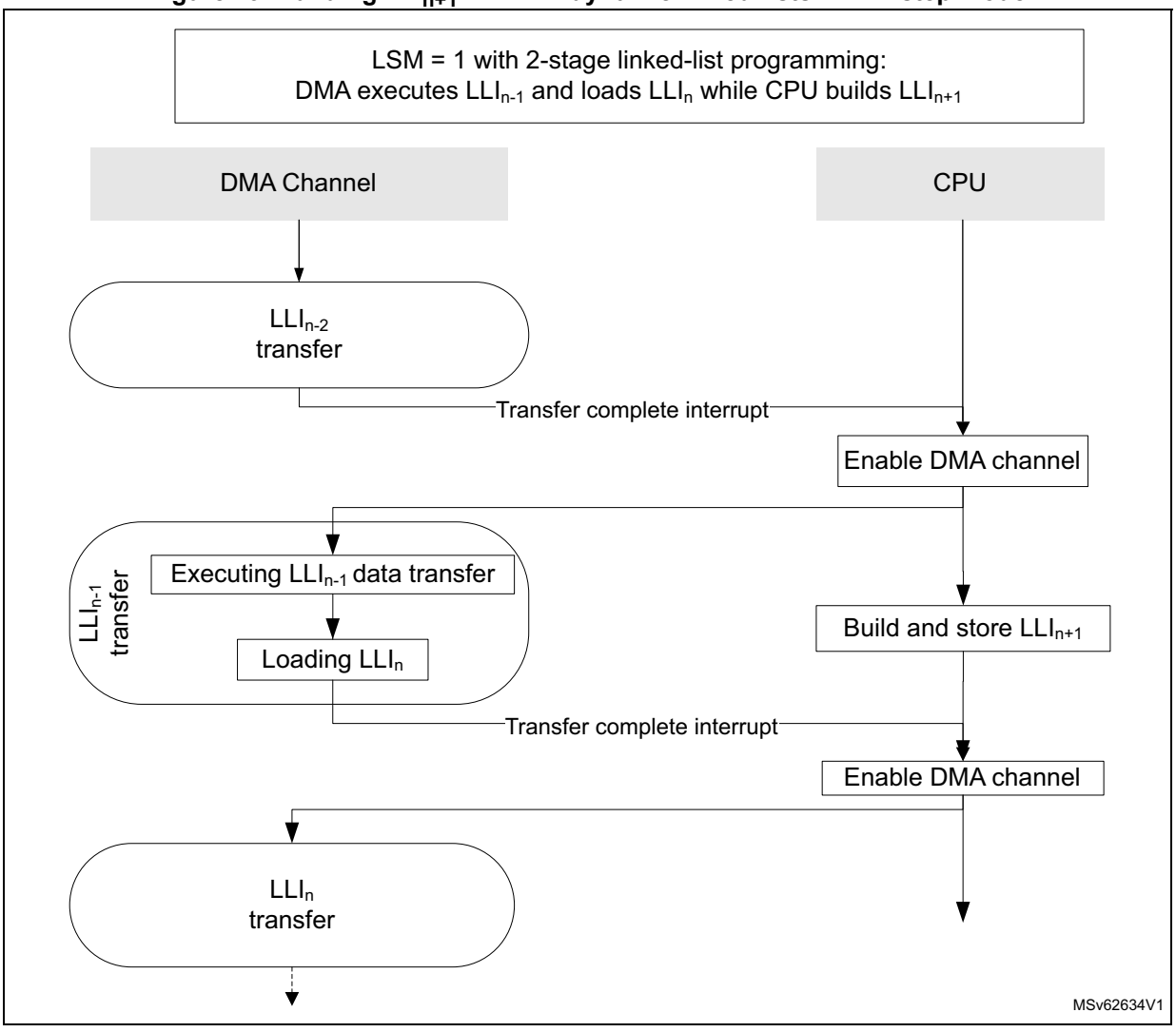


The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of LPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after LPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

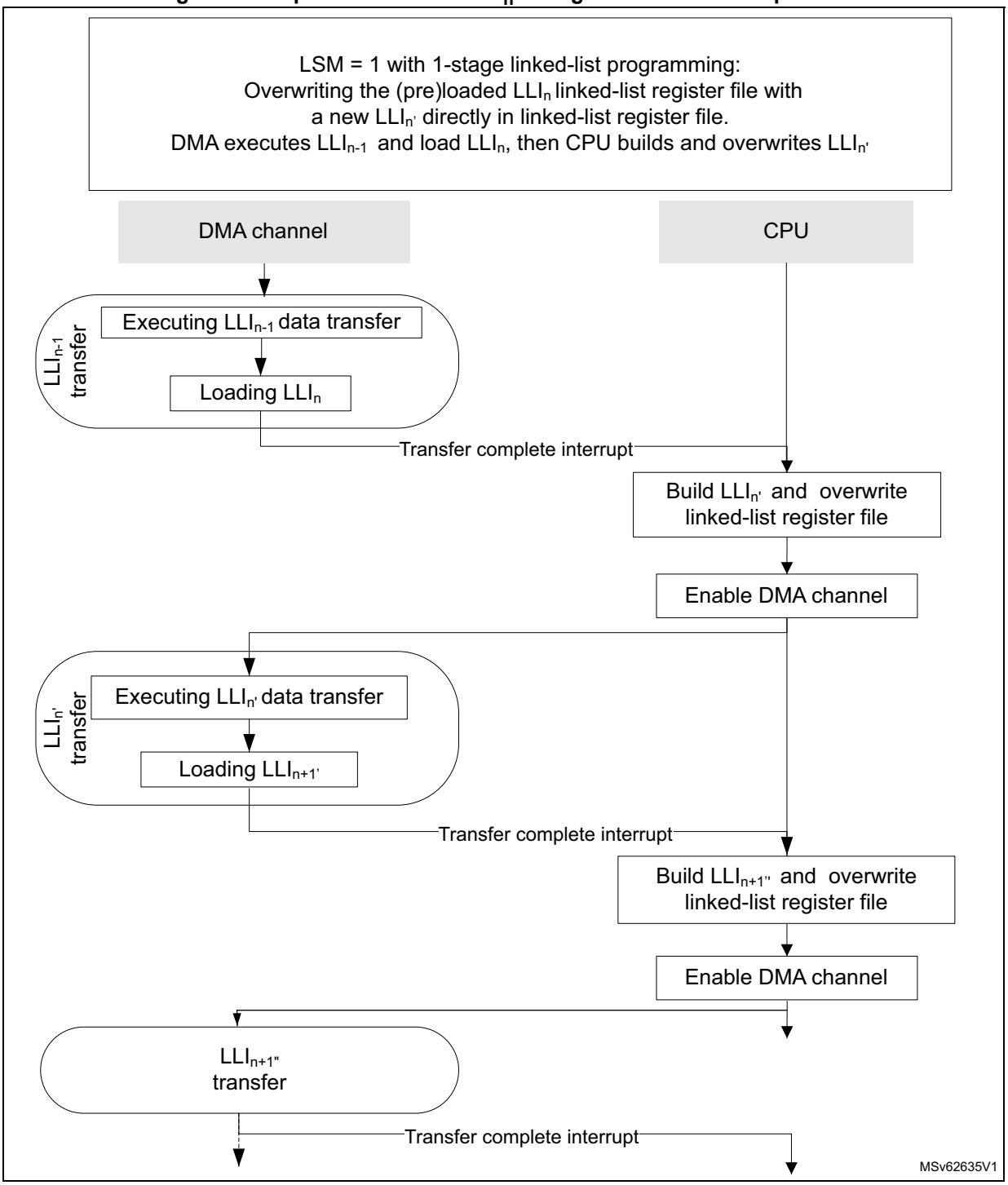
Figure 49. Building LLI_{n+1}: LPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new $LLI_{n'}$, after LPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new $LLI_{n'}$, as shown in *Figure 50*.

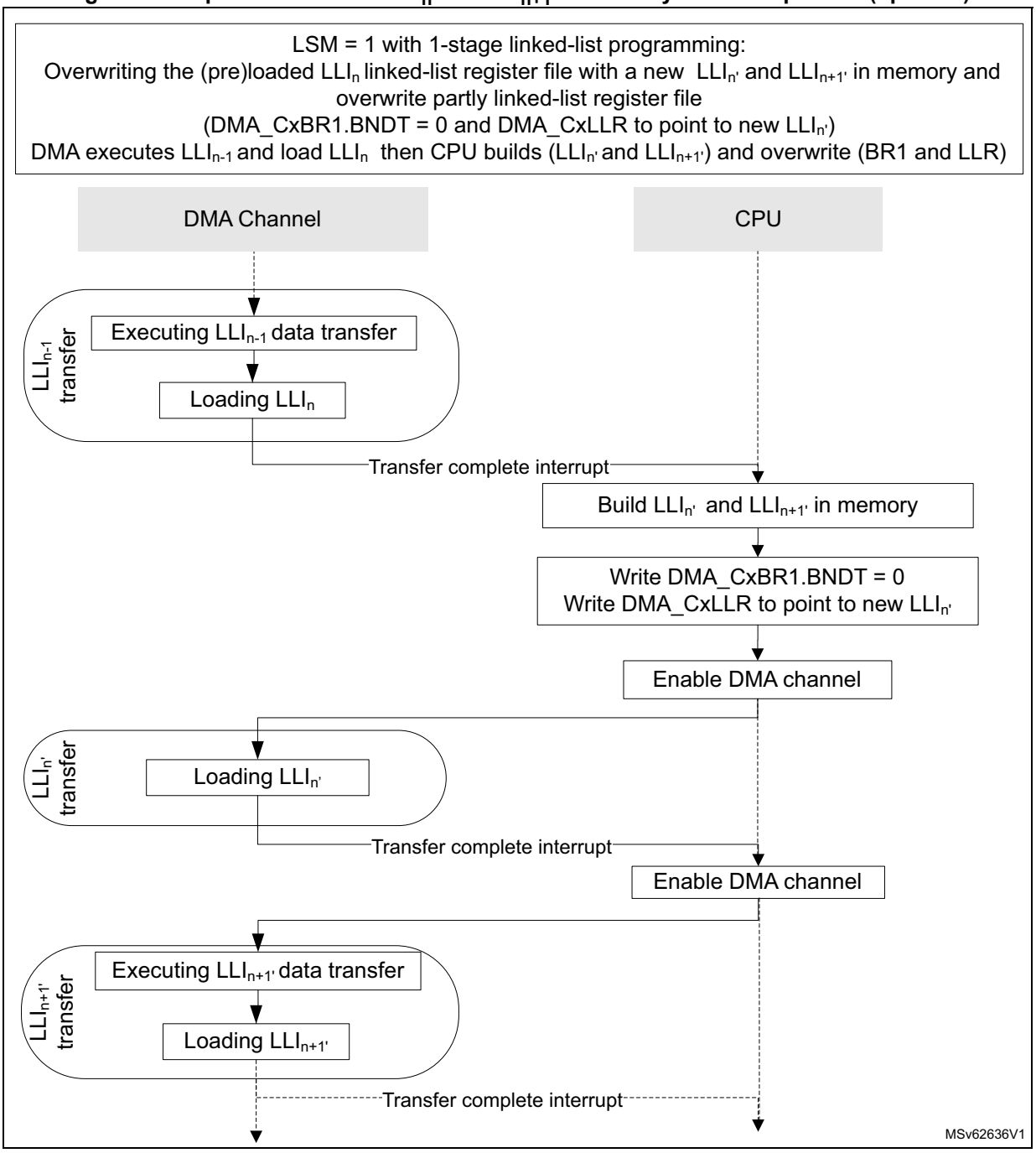
Figure 50. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after LPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (LPDMA_CxBR1.BNDT[15:0] to be null and LPDMA_CxLLR to point to new $LLI_{n'}$) as shown in *Figure 51*.

Figure 51. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



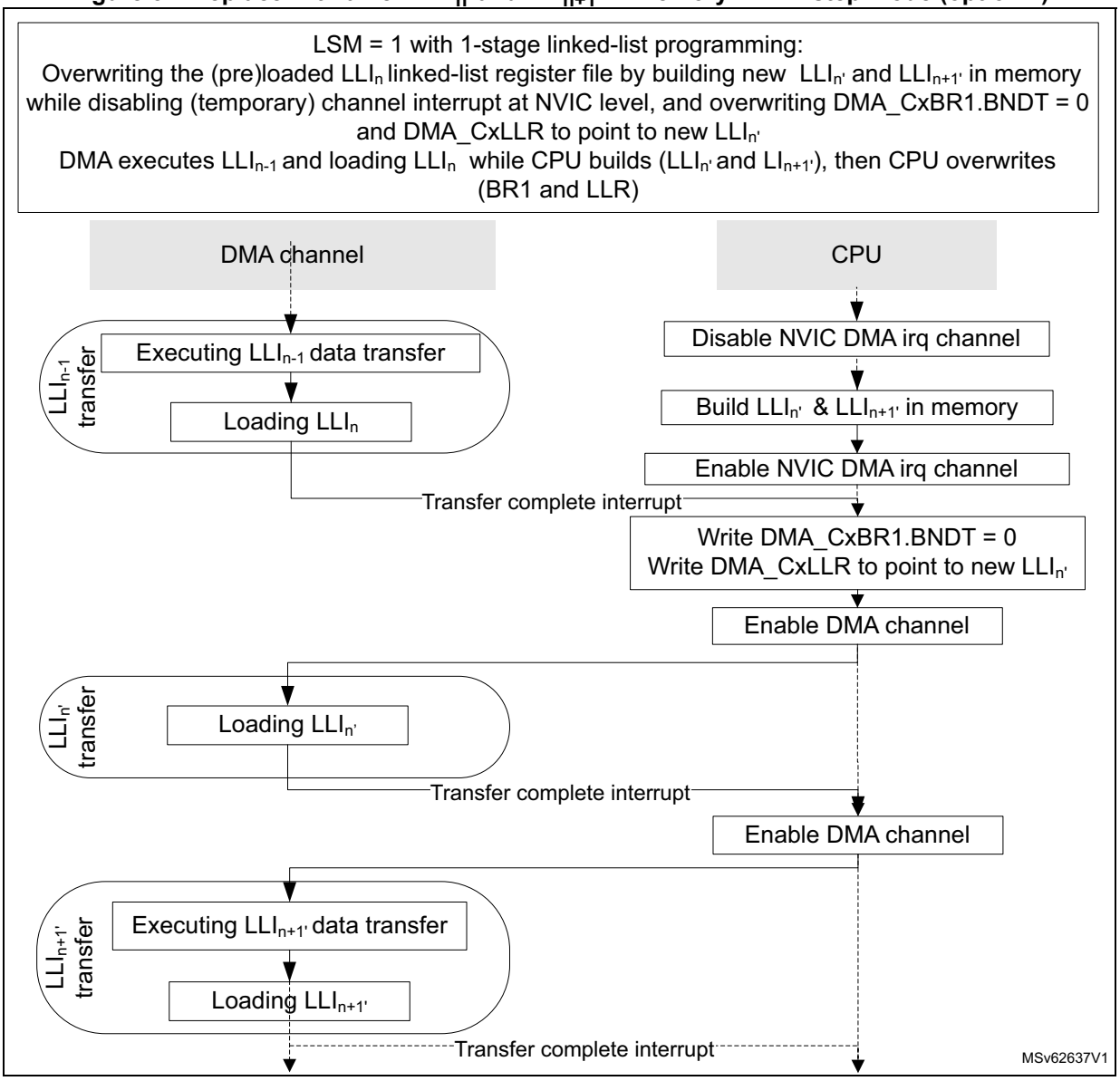
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile LPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite LPDMA_CxBR1.BNDT[15:0] to be null and LPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 52*.

Figure 52. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



14.4.9 LPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (LPDMA_CxCR.EN = 0) and update the execution mode (LPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the LPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the LPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 14.4.4*)

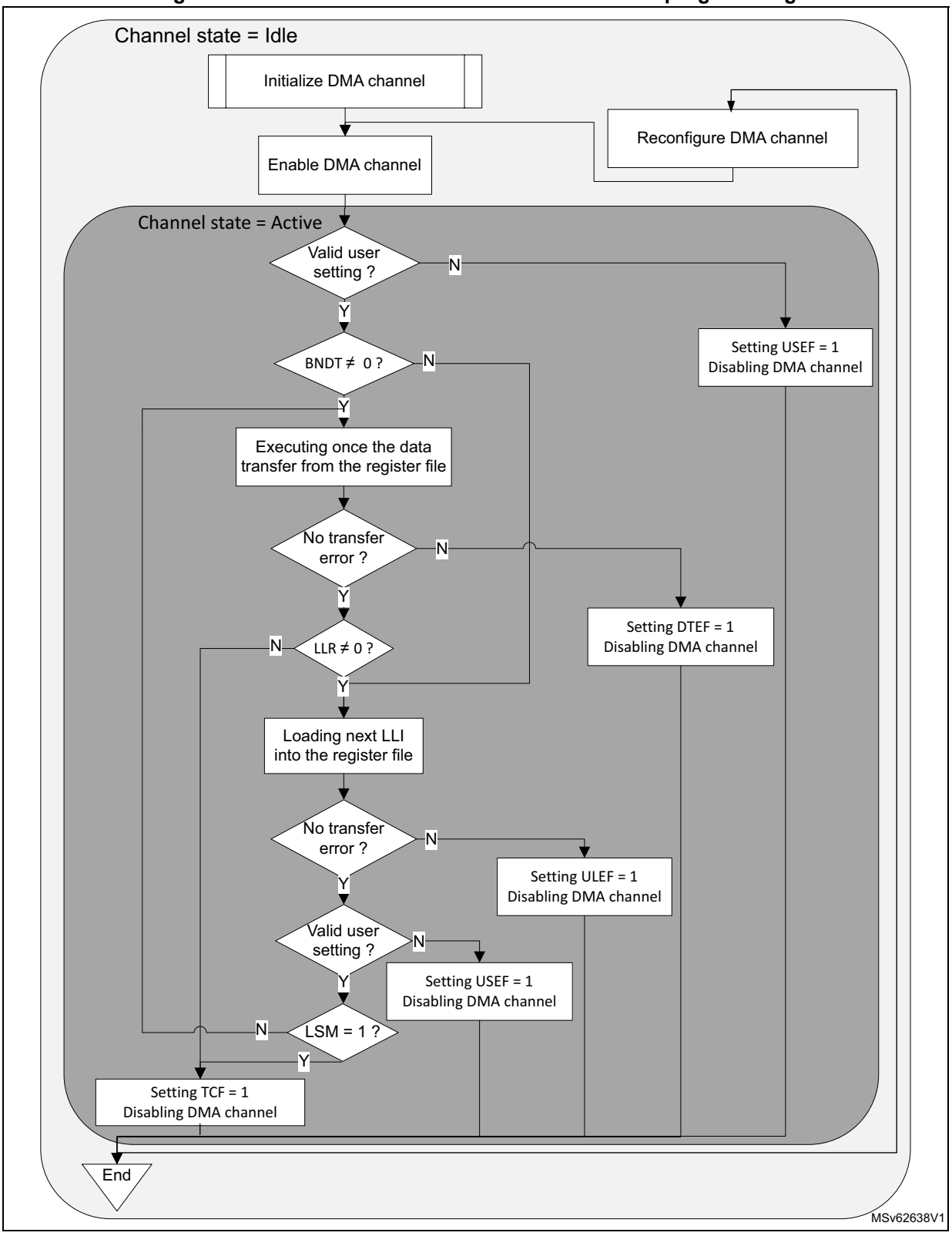
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 53 shows the overall and unified LPDMA linked-list programming, whatever is the execution mode.

Note: Figure 53 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in LPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 53, BNDT ? 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ? 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 14.3.6).

Figure 53. LPDMA channel execution and linked-list programming



14.4.10 LPDMA direct transfers

There is a single transfer operation mode called the direct mode. Any LPDMA channel is used in direct mode. Any channel is implemented without any FIFO (for every channel x, dma_fifo_size[x] = 0).

LPDMA single

A programmed transfer at the lowest level is called a LPDMA single.

A LPDMA single data width is 1, 2 or 4 bytes, as defined by the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields of LPDMA_CxTR1(respectively for source and destination).

Note: The user must not assign a 8-byte data width (SDW_LOG2[1:0] = 0b11 or DDW_LOG2[1:0] = 0b11) else a user setting is reported and no transfer is issued.

The addressing mode after each data of a LPDMA single is defined by the SINC and DINC bits of LPDMA_CxTR1(respectively for source and destination): either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a LPDMA source/destination single (defined by LPDMA_CxSAR and LPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a LPDMA single.

The next single address in the table is the next source/destination address, pointed by LPDMA_CxSAR and LPDMA_CxDAR, once the programmed source/destination single is completed.

Table 95. Programmed LPDMA source/destination single

Programmed LPDMA source/destination single	SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/ DINC	Addressing mode	SAR/DAR next single address	Address alignment
Fixed byte single	00	1	0	Fixed	+0	1
Fixed half-word single	01	2				2
Fixed word single	10	4				4
Incremented byte single	00	1	1	Incremented	+1	1
Incremented half-word single	01	2			+2	2
Incremented word single	10	4			+4	4
Forbidden	11	Causes USEF generation and none single to be issued.				

In direct mode, a LPDMA single is an AHB single transfer.

LPDMA data handling: byte-based padding/truncation, sign extension and left/right alignment

The user can configure some data manipulation between a transferred data from the source and its transfer to the destination. Data handling is controlled by the LPDMA_CxTR1 register:

- If destination data width = source data width (DDW_LOG2[1:0] = SDW_LOG2[1:0]), the source data is copied as is and transferred to the destination.
- Else, depending on PAM:
  - If destination data width > source data width, the source data can be either right-aligned and padded with 0 s, or sign extended up to the destination data width.
  - If destination data width < source data width, the source data can be either right-aligned and left-truncated down to the destination data width, or left-aligned and right-truncated down to the destination data width.

There is no data manipulation between two distinct transferred data from the source, before the generation of the destination transfer.

The table below lists possible data handling from the source to the destination.

Table 96. Programmed data handling

SDW_ LOG2[1:0]	Source data	Source data stream(1)	DDW_ LOG2[1:0]	Destination data	PAM[0]	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	00	Byte	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
			01	Half-word	0 (RA, 0P)(2)(3)	$0B_3,0B_2,0B_1,0B_0$
					1 (RA, SE)(2)(4)	SB_3,SB_2,SB_1,SB_0
			10	Word	0 (RA, 0P)	$000B_1,000B_0$
					1 (RA, SE)	$SSSB_1,SSSB_0$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	00	Byte	0 (RA, LT)(2)	B_6,B_4,B_2,B_0
					1 (LA, RT)(2)	B_7,B_5,B_3,B_1
			01	Half-word	xx	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
			10	Word	0 (RA, 0P)	$00B_3B_2,00B_1B_0$
					1 (RA, SE)	SSB_3B_2,SSB_1B_0
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	00	Byte	0 (RA, LT)	B_{12},B_8,B_4,B_0
					1 (LA, RT)	B_{15},B_{11},B_7,B_3
			01	Half-word	0 (RA, LT)	B_5B_4,B_1B_0
				Word	1 (LA, RT)	B_7B_6,B_3B_2
			10		xx	$B_7B_6B_5B_4,B_3B_2B_1B_0$

1. Data stream is timely ordered starting from the byte with the lowest index a.k.a B_0.
2. RA = right aligned. LA = left aligned. RT= right truncated. LT = left truncated.
3. 0P = zero bit padding up to the destination data width.
4. SE = sign bit extended up to the destination data width.

14.4.11 LPDMA transfer request and arbitration

LPDMA transfer request

As defined by LPDMA_CxTR2, a programmed LPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the LPDMA hardware peripheral request is driven by the REQSEL[6:0] field (see *Section 14.3.5*). The selected hardware request can be one of the following:
  - a hardware request from a peripheral configured in LPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - a hardware request from a peripheral for its control registers update from the memory
  - a hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral LPDMA request via LPDMA_CxTR.REQSEL[6:0] to two different channel if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

LPDMA transfer request for arbitration

For a given channel, a LPDMA requested data transfer from the source address to the destination address is arbitrated versus simultaneous requested LPDMA transfers from other channels, in order to be scheduled over the AHB master port. A LPDMA data transfer is atomic to the LPDMA arbitration: it consists of an AHB read single, immediately followed by an AHB write single. It is granted by the arbiter once for both AHB transfers, based on the channel priority defined by LPDMA_CxCR.PRIO[1:0].

An arbitrated LPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in the memory to one of the linked-list registers (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR or LPDMA_CxLLR). Each 32-bit read from the memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the master port.

The re arbitration occurs after each granted single transfer:

- whatever how the requested data transfer is programmed:
  - with a software request for a memory-to-memory transfer (LPDMA_CxTR2.SWREQ = 1)
  - with a hardware request (LPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer
- whatever the hardware request type

When the requested data transfer is programmed with a hardware request from a peripheral (LPDMA_CxTR2.SWREQ = 0), the first memory read of a block is gated by the occurrence of the corresponding and selected hardware request, whatever the peripheral is source or destination of the transfer. This first read request to the memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

LPDMA arbitration

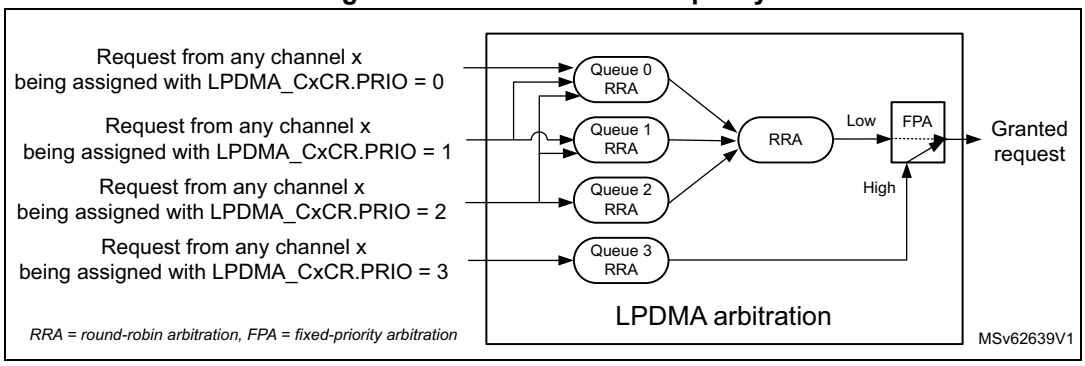
The LPDMA arbitration is directed from the 4-grade assigned channel priority (LPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 54*, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2

  Each requested single transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
  - Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
  - Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
  - Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 54. LPDMA arbitration policy



1. RRA: round-robin arbitration
2. FPA: fixed-priority arbitration

LPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive LPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).

    $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).

    $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$

    $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).

    $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$

    $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$

  The above computed bandwidth calculation is based on a theoretical input request, always active for any LPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one LPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
- **Example 2**: application where the user distributes a same non-null N number of LPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is

    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is

    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is

    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is

    $B_{n, P0} = 1/N * 1/9 * (1 - B_{Q3})$

    $B_{n, P1} = 1/N * 5/18 * (1 - B_{Q3})$

    $B_{n, P2} = 1/N * 11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ? 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ? 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ? 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for J > 0 (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ (K>0 in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 (K > 0)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 (K = 0)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the LPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

14.4.12 LPDMA triggered transfer

A programmed LPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by LPDMA_CxTR2.TRIGPOL[1:0] and LPDMA_CxTR2.TRIGSEL[5:0] (see *Section 14.3.7* for the trigger selection).

The triggered transfer, as defined by the trigger mode in LPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first single read of a block, or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see the TRIGM[1:0] description in *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details).

Trigger hit memorization and trigger overrun flag generation

The LPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

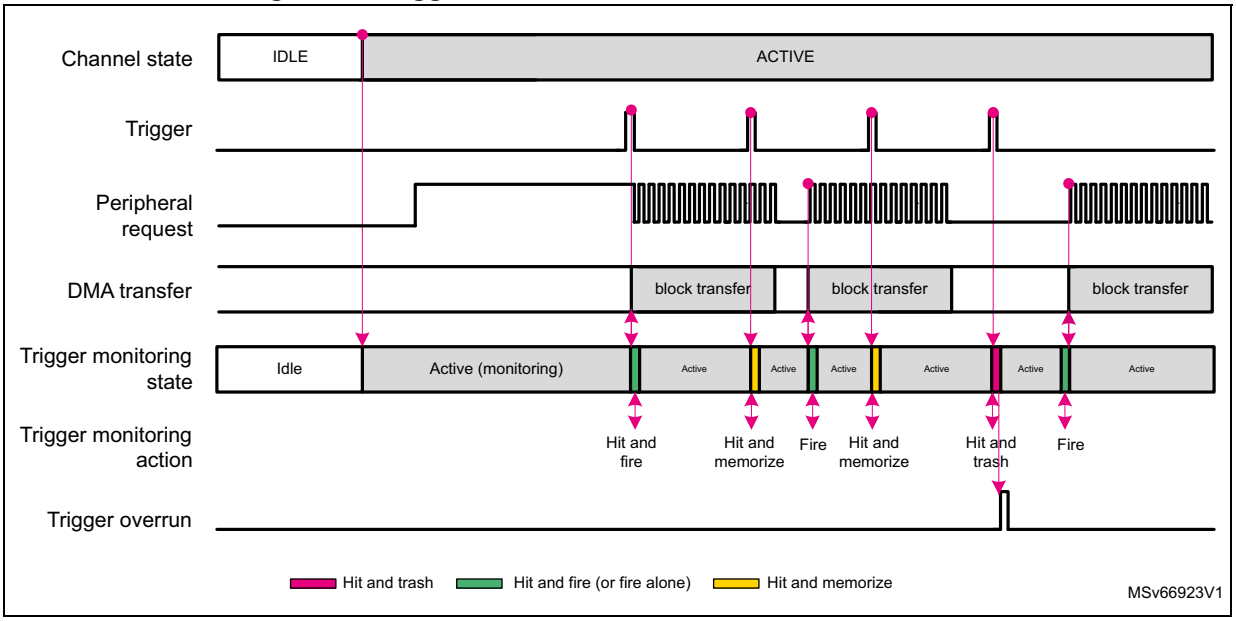
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next LLI_{n+1}, that updates the LPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (LPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if LPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 55. Trigger hit, memorization and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

14.4.13 LPDMA circular buffering with linked-list programming

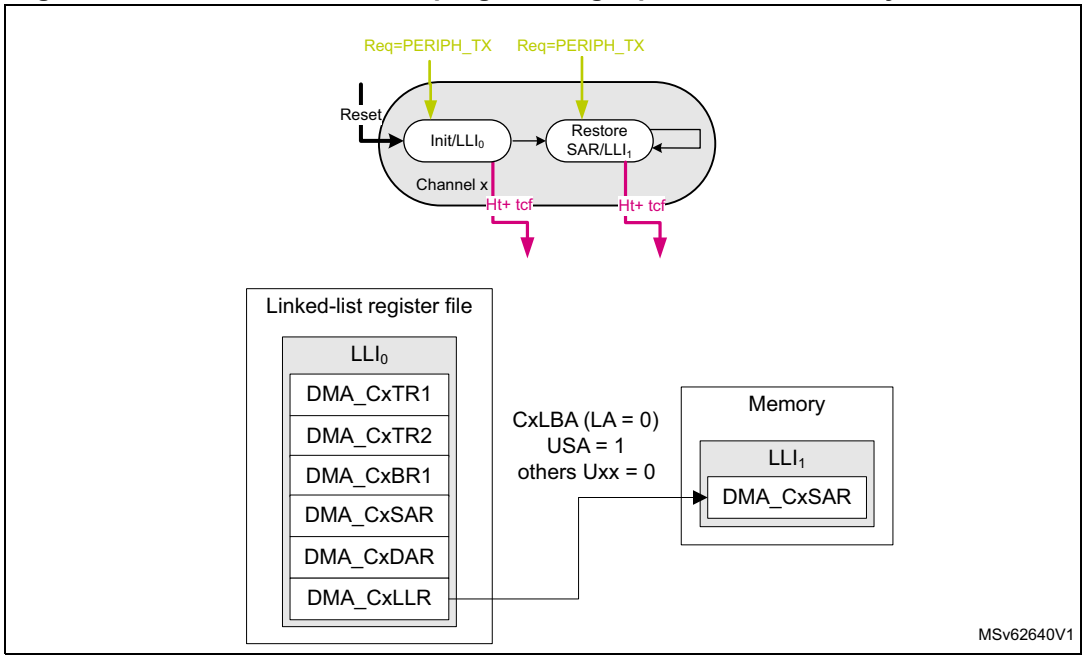
LPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete transfer events/interrupts generation (LPDMA_CxCR.HTIE = 1 and LPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). LPDMA automatically reloads the initially programmed LPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore LPDMA_CxBR1.

The figure below illustrates this programming with a LPDMA channel and a source circular buffer.

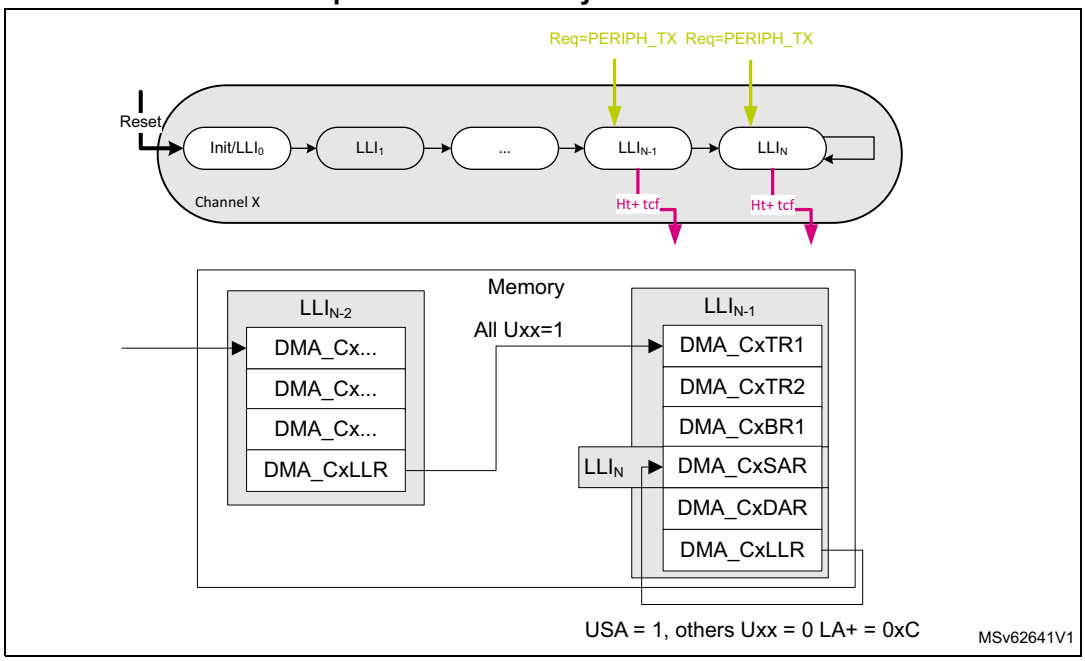
Figure 56. LPDMA circular buffer programming: update of the memory start address



If circular buffering must be executed after some other transfers over the shared LPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

The figure below illustrates this programming with a shared LPDMA channel, and a source circular buffer.

Figure 57. Shared LPDMA channel with circular buffering: update of the memory start address



14.4.14 LPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate an LPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 14.3.6*).

If the related LPDMA channel x is also programmed in peripheral flow-control mode (LPDMA_CxTR2.PFREQ = 1):

- The LPDMA block transfer starts as follows:
  - If LPDMA_CxBR1.BNDT[15:0] ? 0, the programmed value is internally taken into account by the LPDMA hardware.
  - If LPDMA_CxBR1.BNDT[15:0] = 0, the LPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The LPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when LPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on LPDMA_CxTR2 (see *LPDMA channel x transfer register 2 (LPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (LPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (LPDMA_CxSAR[31:0] or LPDMA_CxDAR[31:0]).

14.4.15 LPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via LPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for LPDMA_PRIVCFGR, LPDMA_SECCFGR, that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the LPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the master port.

LPDMA generates a DMA channel state versus privilege, reflecting LPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each DMA channel x.

Additionally, the LPDMA generates the privileged illegal access pulse signal on an illegal unprivileged access to a privileged LPDMA register. This signal may be used or not, depending on the product (see the system security section for more details).

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have LPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the LPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (LPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

14.4.16 LPDMA error management

LPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (LPDMA_CxCR.EN cleared by the LPDMA hardware). The channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and the data error (LPDMA_CxSR.DTEF = 1).
- After a LPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a LPDMA data transfer error, the user must issue a channel reset (set LPDMA_CxCR.RESET) to reset the hardware LPDMA channel data path before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a LPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (LPDMA_CxCR.EN cleared by the LPDMA hardware), the channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and the link error (LPDMA_CxSR.ULEF = 1).
- After a LPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (LPDMA_CxLBAR and LPDMA_CxLLR), including the protection attributes.
- After a LPDMA link error, the user must explicitly write the linked-list register file (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a LPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (LPDMA_CxCR.EN forced and cleared by the LPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and a user setting error (LPDMA_CxSR.USEF = 1).
- After a LPDMA user setting error, the user must perform a debug session, taking care of the LPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (LPDMA_CxBR1.BNDT[15:0] = 0 and LPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source single transfer (LPDMA_CxBR1.BNDT[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source single transfer (LPDMA_CxSAR[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination single transfer (LPDMA_CxDAR[2:0] versus LPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed double-word source data width (LPDMA_CxTR1.SDW_LOG2[1:0] = 0b11)
  - a programmed double-word destination data width (LPDMA_CxTR1.DDW_LOG2[1:0] = 0b11
  - a programmed linked-list item LLI_{n+1} with a null data transfer (LPDMA_CxLLR.UB1 = 1 and LPDMA_CxBR1. BNDT = 0)

14.5 LPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, LPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, LPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

14.6 LPDMA in low-power modes

Table 97. Effect of low-power modes on LPDMA

Mode	Description
Sleep	No effect. LPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPDMA registers is kept when entering Stop mode.
Standby	The LPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 14.3.3* to know which Stop mode is supported.

14.7 LPDMA interrupts

There is one LPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 98. LPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
LPDMA_CHx	Transfer complete	LPDMA_CxCR.TCIE	LPDMA_CxSR.TCF	Write 1 to LPDMA_CxFCR.TCF
	Half transfer	LPDMA_CxCR.HTIE	LPDMA_CxSR.HTF	Write 1 to LPDMA_CxFCR.HTF
	Data transfer error	LPDMA_CxCR.DTEIE	LPDMA_CxSR.DTEF	Write 1 to LPDMA_CxFCR.DTEF
	Update link error	LPDMA_CxCR.ULEIE	LPDMA_CxSR.ULEF	Write 1 to LPDMA_CxFCR.ULEF
	User setting error	LPDMA_CxCR.USEIE	LPDMA_CxSR.USEF	Write 1 to LPDMA_CxFCR.USEF
	Suspended	LPDMA_CxCR.SUSPIE	LPDMA_CxSR.SUSPF	Write 1 to LPDMA_CxFCR.SUSPF
	Trigger overrun	LPDMA_CxCR.TOFIE	LPDMA_CxSR.TOF	Write 1 to LPDMA_CxFCR.TOF

A LPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal lpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (LPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (LPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into LPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by LPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode LPDMA_CxTR2.TCEM[1:0].
- A half-transfer event is a half-block transfer. A half-block transfer occurs when half of the source block size bytes (rounded-up integer of LPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

See *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single data transfer from the source or to the destination (DTEF)
- during an update of a LPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a LPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the LPDMA channel programming versus the USEF root causes list (see *Section 14.4.16*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

14.8 LPDMA registers

The LPDMA registers must be accessed with an aligned 32-bit word data access.

14.8.1 LPDMA privileged configuration register (LPDMA_PRIVCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (LPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a LPDMA channel (LPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVx**: privileged state of channel x (x = 7 to 0)

0: unprivileged

1: privileged

14.8.2 LPDMA configuration lock register (LPDMA_RCFGLOCKR)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be written by a software agent with privileged attribute in order to individually lock, for example at boot time, the privileged attributes of any DMA channel/resource (to lock the setting of LPDMA_PRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
								rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LOCKx**: lock the configuration of LPDMA_PRIVCFGR.PRIVx until a global DMA reset (x = 7 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global DMA reset.

0: privilege configuration of the channel x is writable.

1: privilege configuration of the channel x is not writable.

14.8.3 LPDMA masked interrupt status register (LPDMA_MISR)

Address offset: 0x00C

Reset value: 0x0000 0000

This is a read register.

A masked interrupt status bit MISx is asserted by hardware for each channel x. It is a logical OR of all the LPDMA_CxSR flags, each source flag being enabled by the corresponding LPDMA_CxCR interrupt enable bit.

Every bit is de-asserted by hardware when writing 1 to the corresponding LPDMA_CxFCR flag clear bit.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel LPDMA_PRIVCFGR.PRIVx. A privileged software can read the full interrupt status. An unprivileged software is restricted to read the status of unprivileged channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MISx**: masked interrupt status of channel x (x = 7 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

14.8.4 LPDMA channel x linked-list base address register (LPDMA_CxLBAR)

Address offset: 0x050 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x LPDMA_PRIVCFGR.PRIVx.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the LPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit LPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LBA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **LBA[31:16]**: linked-list base address of LPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

14.8.5 LPDMA channel x flag clear register (LPDMA_CxFCR)

Address offset: 0x05C+ 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This is a write register privileged or unprivileged, depending on the privileged state of the channel x (LPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: clears the corresponding TOF flag

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half-transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

14.8.6 LPDMA channel x status register (LPDMA_CxSR)

Address offset: 0x060 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is privileged or unprivileged, depending on the privileged state of the channel (LPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: clears the corresponding TOF flag

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half-transfer flag
0: no half-transfer event
1: a half-transfer event occurred
A half-transfer event is a half-block transfer that occurs when half of the bytes of the source block size (rounded-up integer of LPDMA_CxBR1.BNDT[15:0] / 2) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is a block transfer complete or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode LPDMA_CxTR2.TCEM[1:0].

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is de-asserted by hardware when the channel is enabled (LPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

14.8.7 LPDMA channel x control register (LPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x (LPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	Res.	LSM
								rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x LPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (LPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0). Then LPDMA_CxBR1.BNDT[15:0] = 0.
1: channel executed **once** for the current LLI
First the block transfer is executed as defined by the current internal register file until LPDMA_CxBR1.BNDT[15:0] = 0). Secondly the next linked-list data structure is conditionally uploaded from memory as defined by LPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half-transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **SUSP:** suspend

Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:

Software must write 1 in order to suspend an active channel (with an ongoing DMA transfer over its master ports).

The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 42*.

0: write: resume channel, read: channel not suspended

1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:

- active channel in suspended state (LPDMA_CxSR.SUSPF = 1 and LPDMA_CxSR.IDLEF = LPDMA_CxCR.EN = 1)
- channel in disabled state (LPDMA_CxSR.IDLEF = 1 and LPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (LPDMA_CxBR1, LPDMA_CxSAR and LPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 43*).

0: no channel reset

1: channel reset

Bit 0 **EN:** enable

Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:

this bit is de-asserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example: if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: write: ignored, read: channel disabled

1: write: enable channel, read: channel enabled

14.8.8 LPDMA channel x transfer register 1 (LPDMA_CxTR1)

Address offset: 0x090 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x in LPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DINC	Res.	DDW_LOG2[1:0]	
												rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PAM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SINC	Res.	SDW_LOG2[1:0]	
				rw								rw		rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **DINC**: destination incrementing single

0: fixed single

1: contiguously incremented single

The destination address, pointed by LPDMA_CxDAR, is kept constant after a single transfer, or is incremented by the offset value corresponding to a contiguous data after a single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a single in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

A destination single transfer must have an aligned address with its data width (start address LPDMA_CxDAR[2:0] versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and none transfer is issued.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **PAM**: padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a single destination transfer is equal to the data width of a single source transfer, this bit is ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher that source data width, and the condition PAM_2 is when destination data width is higher than source data width.

Condition: PAM_1

0: source data is transferred as right aligned, padded with 0s up to the destination data width

1: source data is transferred as right aligned, sign extended up to the destination data width

Condition: PAM_2

0: source data is transferred as right aligned, left-truncated down to the destination data width

1: source data is transferred as left-aligned, right-truncated down to the destination data width

Bits 10:4 Reserved, must be kept at reset value.

Bit 3 **SINC**: source incrementing single

0: fixed single

1: contiguously incremented single

The source address, pointed by LPDMA_CxSAR, is kept constant after a single transfer or is incremented by the offset value corresponding to a contiguous data after a single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a single in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

a source block size must be a multiple of the source data width (LPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source single transfer must have an aligned address with its data width (start address LPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

14.8.9 LPDMA channel x transfer register 2 (LPDMA_CxTR2)

Address offset: 0x094 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory, if LPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
rw	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	Res.	SWREQ	Res.	Res.	REQSEL[6:0]						
rw	rw		rw	rw		rw			rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: at block level (when LPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with LPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: same as 00

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer (the LLI data transfer being a block transfer), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (i.e. directly programmed by the internal register file with LPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address LPDMA_CxLLR.LA[15:2] to zero and clears all the LPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].
00: no trigger (masked trigger event)
01: trigger on the rising edge
10: trigger on the falling edge
11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the LPDMA transfer (as per *Section 14.3.7*), with an active trigger event if TRIGPOL[1:0] = 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.
If the channel x is enabled (LPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 0b00 or 0b11, these TRIGM[1:0] bits are ignored.
Else, a DMA transfer is conditioned by at least one trigger hit:
00: at block level: the first single read of each block transfer is conditioned by one hit trigger.
01: same as 00
10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.
11: at programmed single level: each programmed single read is conditioned by one hit trigger.
The LPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 0b01 or respectively TRIGPOL[1:0] = 0b10).
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.
Transferring a next LLI_{n+1} that updates the LPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.
After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized, and a trigger overrun flag is reported (LPDMA_CxSR.TOF = 1), an interrupt is generated if enabled (LPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 14.3.6 for the list of the implemented channels with this feature.
If the channel x is activated (LPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:
0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in LPDMA control mode. The LPDMA is programmed with LPDMA_CxCBR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.
1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The LPDMA block transfer can be early completed by the peripheral itself (see *Section 14.3.6* for more details).

Bit 11 **BREQ**: block hardware request

If the channel x is activated (LPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a single level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 14.3.5*).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when LPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[6:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[6:0] is ignored.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **REQSEL[6:0]**: DMA hardware request selection

These bits are ignored if channel x is activated (LPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 14.3.4*.

Caution: The user must not assign a same input hardware request (same REQSEL[6:0] value) to different active DMA channels (LPDMA_CxCR.EN = 1 and LPDMA_CxTR2.SWREQ = 0 for these channels). DMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

14.8.10 LPDMA channel x block register 1 (LPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if LPDMA_CxLLR.UB1 = 1, this register is automatically updated by DMA from the next LLI in memory.
- If LPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed values for the field BNDT[15:0].
- If all the update bits LPDMA_CxLLR.Uxx are null and if LPDMA_CxLLR.LA[15:0] # 0, the current LLI is the last one and is continuously executed: this register is

automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI

- If LPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if LPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if LPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all LPDMA_CxLLR.Uxx = 0 and if LPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if LPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

14.8.11 LPDMA channel x source address register (LPDMA_CxSAR)

Address offset: 0x09C + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next single transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.USA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (LPDMA_CxTR1.SINC), this field is either kept fixed or incremented by the data width (LPDMA_CxTR1.SDW_LOG2[1:0]) after each single source data, reflecting the next address from which data is read.

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by LPDMA from the memory, provided the LLI is set with LPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source single (SA[32:0] versus LPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

14.8.12 LPDMA channel x destination address register (LPDMA_CxDAR)

Address offset: 0x0A0 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next single transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.UDA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (LPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (LPDMA_CxTR1.DDW_LOG2[1:0]) after each single destination data, reflecting the next address from which data is written.

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by DMA from the memory, provided the LLI is set with LPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination single (DA[2:0] versus LPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

14.8.13 LPDMA channel x linked-list address register (LPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update LPDMA_CxTR1 from memory

This bit controls the update of the LPDMA_CxTR1 register from the memory during the link transfer.

0: no LPDMA_CxTR1 update

1: LPDMA_CxTR1 update

Bit 30 **UT2**: Update LPDMA_CxTR2 from memory

This bit controls the update of the LPDMA_CxTR2 register from the memory during the link transfer.

0: no LPDMA_CxTR2 update

1: LPDMA_CxTR2 update

Bit 29 **UB1**: Update LPDMA_CxBR1 from memory

This bit controls the update of the LPDMA_CxBR1 register from the memory during the link transfer.

0: no LPDMA_CxBR1 update from memory and internally restored to the previous programmed value

1: LPDMA_CxBR1 update

Bit 28 **USA**: update LPDMA_CxSAR from memory

This bit controls the update of the LPDMA_CxSAR register from the memory during the link transfer.

0: no LPDMA_CxSAR update

1: LPDMA_CxSAR update

Bit 27 **UDA**: Update LPDMA_CxDAR register from memory

This bit is used to control the update of the LPDMA_CxDAR register from the memory during the link transfer.

0: no LPDMA_CxDAR update

1: LPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update LPDMA_CxLLR register from memory

This bit is used to control the update of the LPDMA_CxLLR register from the memory during the link transfer.

0: no LPDMA_CxLLR update

1: LPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure

If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list DMA register file.

Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list DMA internal register file (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register LPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

14.8.14 LPDMA register map

Table 99. LPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	LPDMA_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																									0	0	0	0	0	0	0	0
0x008	LPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																									0	0	0	0	0	0	0	0
0x00C	LPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0
0x014 - 0x04C	Reserved	Res.																															
0x050+ 0x080 * x (x = 0 to 7)	LPDMA_CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x05C+ 0x080 * x (x = 0 to 7)	LPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x060+ 0x080 * x (x = 0 to 7)	LPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value																		0	0	0	0	0	0	0								1
0x064+ 0x080 * x (x = 0 to 7)	LPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	Res.	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0						0		0	0	0	0	0	0	0						0	0	0

Table 99. LPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x090+ 0x080 * x (x = 0 to 7)	LPDMA_CxTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DINC	Res.	DDW_LOG2 [1:0]		Res.	Res.	Res.	Res.	PAM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SINC	Res.	SDW_LOG2[1:0]	
	Reset value													0		0	0					0								0		0	0
0x094+ 0x080 * x (x = 0 to 7)	LPDMA_CxTR2	TCEM [1:0]		Res.	Res.	Res.	Res.	TRIGPOL [1:0]		Res.	TRIGSEL [5:0]						Res.	TRIGM [1:0]		Res.	Res.	PFREQ	BREQ	SWREQ	Res.	Res.	REQSEL [6:0]						
	Reset value	0	0					0	0		0	0	0	0	0	0		0	0			0	0	0			0	0	0	0	0	0	0
0x098+ 0x080 * x (x = 0 to 7)	LPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C+ 0x080 * x (x = 0 to 7)	LPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0+ 0x080 * x (x = 0 to 7)	LPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0CC+ 0x080 * x (x = 0 to 7)	LPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

15 Nested vectored interrupt controller (NVIC)

15.1 NVIC main features

- 98 maskable interrupt channels (not including the 16 Cortex®-M33 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, enabling low-latency interrupt processing and efficient processing of late-arriving interrupts.

The NVIC manages all interrupts including the core exceptions.

15.2 SysTick calibration value register

The Cortex-M33 embeds one SysTick timer. The SysTick timer calibration value (STCALIB) is 0x3E8. It gives a reference time base of 1 ms based on a SysTick clock frequency of 1 MHz. To match the 1 ms time base for an application running at a given frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- When the SysTick clock source is CPU clock HCLK
  the reload value = (HCLK x STCALIB) - 1
- When the SysTick clock source is an external clock (HCLK/8)
  the reload value = ((HCLK/8) x STCALIB) - 1

The HCLK refers to the AHB frequency value in MHz.

Example: The SysTick clock source is a CPU clock HCLK of 100 MHz, to match a time base of 1 ms:

SysTick reload value = (100 x STCALIB) - 1 = 0x1869F

15.3 Interrupt and exception vectors

The gray rows in the table below describe the vectors without specific position.

For the actual availability of some peripherals, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them as reserved.

Table 100. STM32C5 vector table

IRQ number	Priority(1)	Acronym	Description	Address offset
-15	-4	Reset	Reset	0x0000 0004
-14	-2	NMI	Non maskable interrupt. The RCC clock security system (CSS) is linked to the NMI vector.	0x0000 0008
-13	-1	HardFault	Hard fault. All classes of fault	0x0000 000C
-12	Settable	MemManage	Memory management	0x0000 0010

Table 100. STM32C5 vector table (continued)

IRQ number	Priority[1]	Acronym	Description	Address offset
-11	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-10	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	Reserved	0x0000 001C
-	-	-	Reserved	0x0000 0020
-	-	-	Reserved	0x0000 0024
-	-	-	Reserved	0x0000 0028
-5	Settable	SVC	System service call via SWI instruction	0x0000 002C
-4	Settable	Debug Monitor	Monitor	0x0000 0030
-	-	-	Reserved	0x0000 0034
-2	Settable	PendSV	Pendable request for system service	0x0000 0038
-1	Settable	SysTick	System tick timer	0x0000 003C
0	Settable	WWDG	Window watchdog interrupt	0x0000 0040
1	Settable	PVD	Power voltage monitor	0x0000 0044
2	Settable	RTC	RTC global interrupts	0x0000 0048
3	Settable	TAMP	Tamper global interrupts	0x0000 004C
4	Settable	RAMCFG	RAM configuration global interrupt	0x0000 0050
5	Settable	FLASH	Flash global interrupt	0x0000 0054
6	Settable	RCC	RCC global interrupt	0x0000 0058
7	Settable	EXTI0	EXTI Line0 interrupt	0x0000 005C
8	Settable	EXTI1	EXTI Line1 interrupt	0x0000 0060
9	Settable	EXTI2	EXTI Line2 interrupt	0x0000 0064
10	Settable	EXTI3	EXTI Line3 interrupt	0x0000 0068
11	Settable	EXTI4	EXTI Line4 interrupt	0x0000 006C
12	Settable	EXTI5	EXTI Line5 interrupt	0x0000 0070
13	Settable	EXTI6	EXTI Line6 interrupt	0x0000 0074
14	Settable	EXTI7	EXTI Line7 interrupt	0x0000 0078
15	Settable	EXTI8	EXTI Line8 interrupt	0x0000 007C
16	Settable	EXTI9	EXTI Line9 interrupt	0x0000 0080
17	Settable	EXTI10	EXTI Line10 interrupt	0x0000 0084
18	Settable	EXTI11	EXTI Line11 interrupt	0x0000 0088
19	Settable	EXTI12	EXTI Line12 interrupt	0x0000 008C
20	Settable	EXTI13	EXTI Line13 interrupt	0x0000 0090
21	Settable	EXTI14	EXTI Line14 interrupt	0x0000 0094
22	Settable	EXTI15	EXTI Line15 interrupt	0x0000 0098

Table 100. STM32C5 vector table (continued)

IRQ number	Priority(1)	Acronym	Description	Address offset
23	Settable	LPDMA1_CH0	LPDMA1 channel0 global interrupt	0x0000 009C
24	Settable	LPDMA1_CH1	LPDMA1 channel1 global interrupt	0x0000 00A0
25	Settable	LPDMA1_CH2	LPDMA1 channel2 global interrupt	0x0000 00A4
26	Settable	LPDMA1_CH3	LPDMA1 channel3 global interrupt	0x0000 00A8
27	Settable	LPDMA1_CH4	LPDMA1 channel4 global interrupt	0x0000 00AC
28	Settable	LPDMA1_CH5	LPDMA1 channel5 global interrupt	0x0000 00B0
29	Settable	LPDMA1_CH6	LPDMA1 channel6 global interrupt	0x0000 00B4
30	Settable	LPDMA1_CH7	LPDMA1 channel7 global interrupt	0x0000 00B8
31	Settable	IWDG	Independent watchdog interrupt	0x0000 00BC
32	Settable	ADC1	ADC1 global interrupt	0x0000 00C0
33	Settable	ADC2	ADC2 global interrupt	0x0000 00C4
34	Settable	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 00C8
35	Settable	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 00CC
36	Settable	TIM1_BRK TIM1_TERR TIM1_IERR	TIM1 Break/TIM1 transition error/TIM1 Index error	0x0000 00D0
37	Settable	TIM1_UP	TIM1 update	0x0000 00D4
38	Settable	TIM1_TRG_COM TIM1_DIR TIM1_IDX	TIM1 trigger and commutation/TIM1 direction change interrupt/TIM1 Index	0x0000 00D8
39	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00DC
40	Settable	TIM2	TIM2 global interrupt	0x0000 00E0
41	Settable	TIM5	TIM5 global interrupt	0x0000 00E4
42	Settable	TIM6	TIM6 global interrupt	0x0000 00E8
43	Settable	TIM7	TIM7 global interrupt	0x0000 00EC
44	Settable	I2C1_EV	I2C1 event interrupt	0x0000 00F0
45	Settable	I2C1_ER	I2C1 error interrupt	0x0000 00F4
46	Settable	I3C1_EV	I3C1 event interrupt	0x0000 00F8
47	Settable	I3C1_ER	I3C1 error interrupt	0x0000 00FC
48	Settable	SPI1	SPI1 global interrupt	0x0000 0100
49	Settable	SPI2	SPI2 global interrupt	0x0000 0104
50	Settable	SPI3	SPI3 global interrupt	0x0000 0108
51	Settable	USART1	USART1 global interrupt	0x0000 010C
52	Settable	USART2	USART2 global interrupt	0x0000 0110
53	Settable	USART3	USART3 global interrupt	0x0000 0114

Table 100. STM32C5 vector table (continued)

IRQ number	Priority(1)	Acronym	Description	Address offset
54	Settable	UART4	UART4 global interrupt	0x0000 0118
55	Settable	UART5	UART5 global interrupt	0x0000 011C
56	Settable	LPUART1	LPUART1 global interrupt	0x0000 0120
57	Settable	LPTIM1	LPTIM1 global interrupt OR LPTIM1through EXTI line	0x0000 0124
58	Settable	TIM12	TIM12 global interrupt	0x0000 0128
59	Settable	TIM15	TIM15 global interrupt	0x0000 012C
60	Settable	TIM16	TIM16 global interrupt	0x0000 0130
61	Settable	TIM17	TIM17 global interrupt	0x0000 0134
62	Settable	USB_FS	USB FS global interrupt	0x0000 0138
63	Settable	CRS	Clock recovery system global interrupt	0x0000 013C
64	Settable	RNG	RNG global interrupt	0x0000 0140
65	Settable	FPU	Floating point interrupt	0x0000 0144
66	Settable	ICACHE	Instruction cache global interrupt	0x0000 0148
67	Settable	CORDIC	CORDIC interrupt	0x0000 014C
68	Settable	AES	AES global interrupt	0x0000 0150
69	Settable	HASH	HASH interrupt	0x0000 0154
70	Settable	I2C2_EV	I2C2 event interrupt	0x0000 0158
71	Settable	I2C2_ER	I2C2 error interrupt	0x0000 015C
72	Settable	TIM8_BRK TIM8_TERR TIM8_IERR	TIM8 break interrupt/TIM8 transition error/ TIM8 index error	0x0000 0160
73	Settable	TIM8_UP	TIM8 update interrupt	0x0000 0164
74	Settable	TIM8_TRG_COM TIM8_DIR TIM8_IDX	TIM8 trigger and commutation interrupt/ TIM8 direction change interrupt/TIM8 Index	0x0000 0168
75	Settable	TIM8_CC	TIM8 capture compare interrupt	0x0000 016C
76	Settable	COMP1	COMP1 global interrupt OR COMP1 through EXTI line	0x0000 0170
77	Settable	DAC1	DAC1 global interrupt	0x0000 0174
78	Settable	LPDMA2_CH0	LPDMA2 channel0 global interrupt	0x0000 0178
79	Settable	LPDMA2_CH1	LPDMA2 channel1 global interrupt	0x0000 017C
80	Settable	LPDMA2_CH2	LPDMA2 channel2 global interrupt	0x0000 0180
81	Settable	LPDMA2_CH3	LPDMA2 channe3 global interrupt	0x0000 0184
82	Settable	LPDMA2_CH4	LPDMA2 channel4 global interrupt	0x0000 0188
83	Settable	LPDMA2_CH5	LPDMA2 channel5 global interrupt	0x0000 018C
84	Settable	LPDMA2_CH6	LPDMA2 channel6 global interrupt	0x0000 0190

Table 100. STM32C5 vector table (continued)

IRQ number	Priority[1]	Acronym	Description	Address offset
85	Settable	LPDMA2_CH7	LPDMA2 channel7 global interrupt	0x0000 0194
86	Settable	FDCAN2_IT0	FDCAN2 Interrupt 0	0x0000 0198
87	Settable	FDCAN2_IT1	FDCAN2 Interrupt 1	0x0000 019C
88	Settable	COMP2	COMP2 global interrupt OR COMP2 through EXTI line	0x0000 01A0
89	Settable	TIM3	TIM3 global interrupt	0x0000 01A4
90	Settable	TIM4	TIM4 global interrupt	0x0000 01A8
91	Settable	XSPI1	XSPI1 global interrupt	0x0000 01AC
92	Settable	SAES	SAES global interrupt	0x0000 01B0
93	Settable	PKA	PKA global interrupt	0x0000 01B4
94	Settable	ETH1	ETH interrupt	0x0000 01B8
95	Settable	ETH1_WKUP	Ethernet wakeup interrupt through EXTI line	0x0000 01BC
96	Settable	USART6	USART6 global interrupt	0x0000 01C0
97	Settable	UART7	UART7 global interrupt	0x0000 01C4
98	Settable	ADC3	ADC3 global interrupt	0x0000 01C8

1. If multiple pending interrupts have the same programmed priority level, the pending interrupt with the lowest IRQ number takes precedence. For example, if both interrupt[n] and interrupt[n+1] are pending and have the same priority, then interrupt[n] is processed before interrupt[n+1]. For further information, refer to the Arm Cortex-M33 technical documentation.

16 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wake-up through configurable and direct event inputs. It provides wake-up requests to the power control, and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU, an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wake-up requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can be used also in Run modes.

The EXTI also includes the EXTI mux I/O port selection.

16.1 EXTI main features

The EXTI main features are the following:

- 40 input events are supported.
- All event inputs allow the possibility to wake up the system.
- Events that do not have an associated wake-up flag in the peripheral, have a flag in the EXTI, and generate an interrupt to the CPU from the EXTI.
- Events can be used to generate a CPU wake-up event.

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), with the following features
  - Selectable active trigger edge
  - Interrupt pending status register bits independent for the rising and falling edge
  - Individual interrupt and event generation mask, used for conditioning the CPU wake-up, interrupt and event generation
  - Software trigger possibility
  - EXTI I/O port selection
- Direct events (interrupt and wake-up sources from peripherals having an associated flag which requires to be cleared in the peripheral), with the following features:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask, used to condition the CPU wake-up and event generation
  - No software trigger possibility

16.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block and EXTI mux as shown in *Figure 58*.

The register block contains all EXTI registers. The event input trigger block provides event input edge trigger logic. The masking block provides the event input distribution to the different wake-up, interrupt, and event outputs, and their masking.

The EXTI mux provides the I/O port selection on to the EXTI event signal.

Figure 58. EXTI block diagram

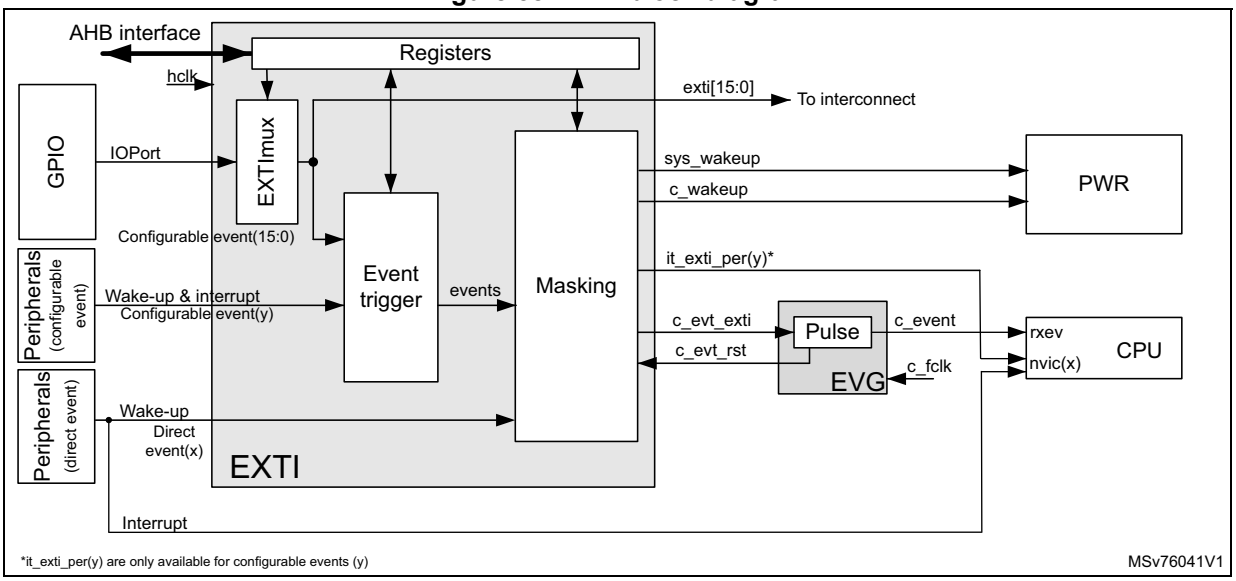


Table 101. EXTI signals

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface.
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wake-up events from peripherals without an associated interrupt and flag
Direct event(x)	I	Synchronous and asynchronous wake-up events from peripherals with an associated interrupt and flag
IOPort(n)	I	GPIO block I/O ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals
it_exti_per (y)	O	Interrupts to the CPU associated with configurable event (y)
c_evt_exti	O	High-level sensitive event output for CPU, synchronous to hclk
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wake-up	O	Asynchronous system wake-up request to PWR for ck_sys and hclk
c_wake-up	O	Wake-up request to PWR for CPU, synchronous to hclk

Table 102. EVG signals

Pin name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

16.2.1 EXTI connections between peripherals and CPU

Some peripherals able to generate wake-up or interrupt events when the system is in Stop mode, are connected to the EXTI.

- Peripheral wake-up signals that generate a pulse or do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit to be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- Peripheral interrupt and wake-up signals with a status bit in the peripheral to be cleared are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.

All GPIO ports input to the EXTI multiplexer allow the selection of a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wake-up signals are connected to the PWR and are used to wake up the system and the CPU subsystem bus clocks.

16.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in the following table.

For the actual availability of some peripherals, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them as reserved.

Table 103. EXTI line connections

EXTI Line	Line source	Line type
0-15	GPIO	Configurable
16	PVD output	Configurable
17	RTC	-
18	TAMP	-
19	LSE CSS	-
20	I2C1 wake-up	-
21	I3C1 wake-up	-
22	SPI1 wake-up	-

Table 103. EXTI line connections (continued)

EXTI Line	Line source	Line type
23	SPI2 wake-up	-
24	SPI3 wake-up	-
25	USART1 wake-up	-
26	USART2 wake-up	-
27	USART3 wake-up	-
28	UART4 wake-up	-
29	UART5 wake-up	-
30	LPUART1 wake-up	-
31	LPTIM1	-
32	USB wake-up	-
33	I2C2 wake-up	-
34	COMP1 output	Configurable
35	IWDG	-
36	COMP2 output	Configurable
37	USART6 wake-up	-
38	UART7 wake-up	-
39	ETH1 wake-up	Configurable

16.3 EXTI functional description

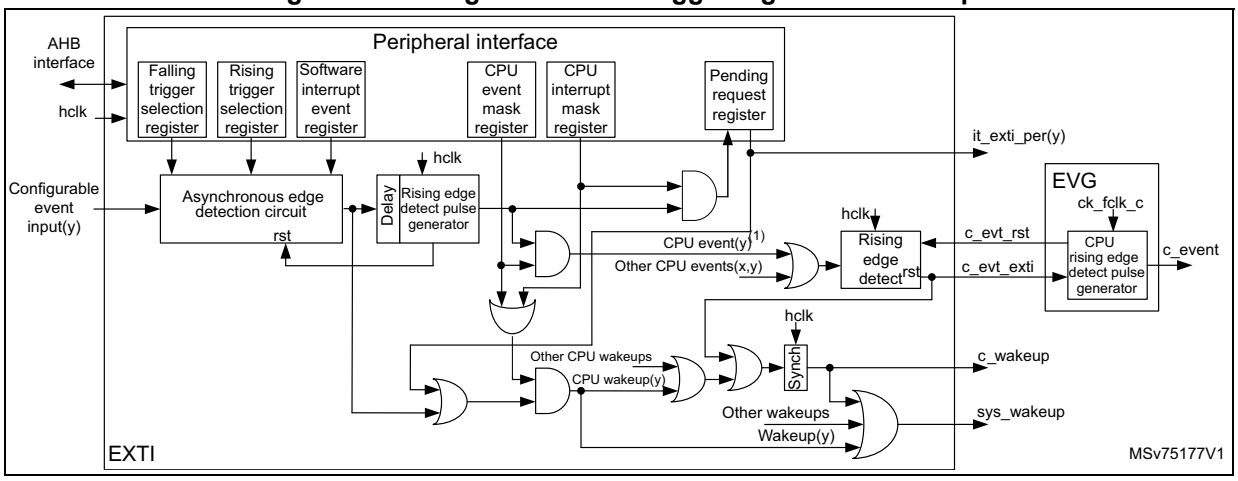
The events features are controlled from register bits as follows:

- Active trigger edge enable
  - by rising edge selection in EXTI_RTSRx
  - by falling edge selection in EXTI_FTSRx
- Software trigger in EXTI_SWIERx
- Interrupt pending flag in:
  - EXTI_RPRx
  - EXTI_FPRx
- CPU wake-up and interrupt enable in EXTI_IMR1 and EXTI_IMR2
- CPU wake-up and event enable in EXTI_EMRx

16.3.1 EXTI configurable event input wake-up

The figure below is a detailed representation of the logic associated with configurable event inputs that wake up the CPU sub-system bus clocks and generate an EXTI pending flag and interrupt to the CPU, and/or a CPU wake-up event.

Figure 59. Configurable event trigger logic CPU wake-up



1. Only for the input events that support CPU rxev generation c_event.

The software interrupt event register allows configurable events to be triggered by software, writing the corresponding register bit, whatever the edge selection setting.

The configurable event active trigger edge (or both edges) is selected and enabled in the rising/falling edge selection registers.

The CPU has its dedicated wake-up (interrupt) mask register and a dedicated event mask registers. When the event is enabled, it is generated to the CPU. All events for the CPU are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts must be acknowledged by software in the EXTI_RPR and/or EXTI_FPR registers.

When a CPU wake-up (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

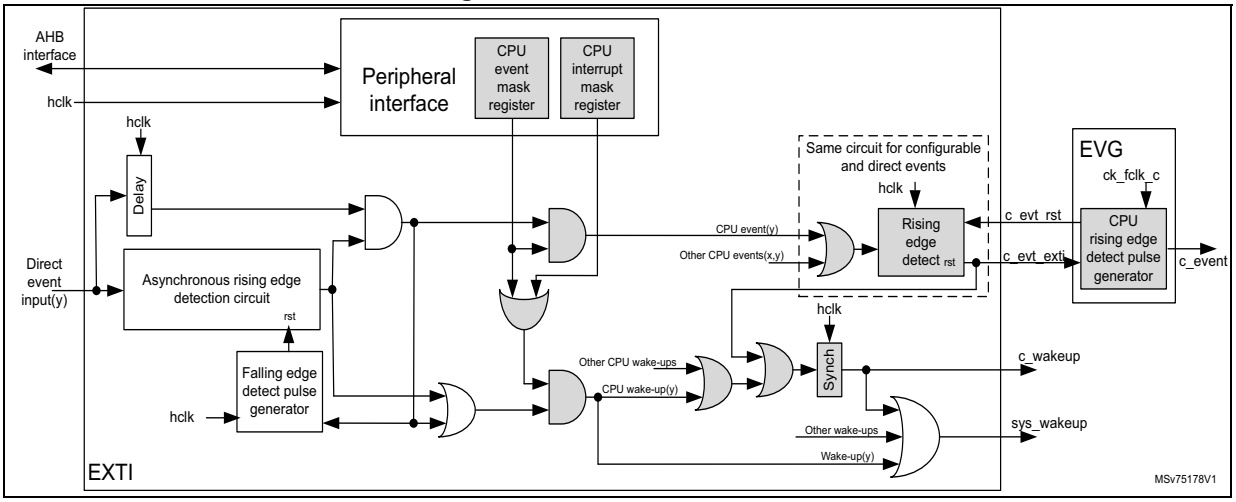
Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

16.3.2 EXTI direct event input wake-up

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU subsystem clocks, and can generate a CPU wake-up event. The peripheral synchronous interrupt, associated with the direct wake-up event, wakes up the CPU.

The EXTI direct event is able to generate a CPU event that wakes up the CPU. The CPU event may occur before the associated peripheral interrupt flag is set.

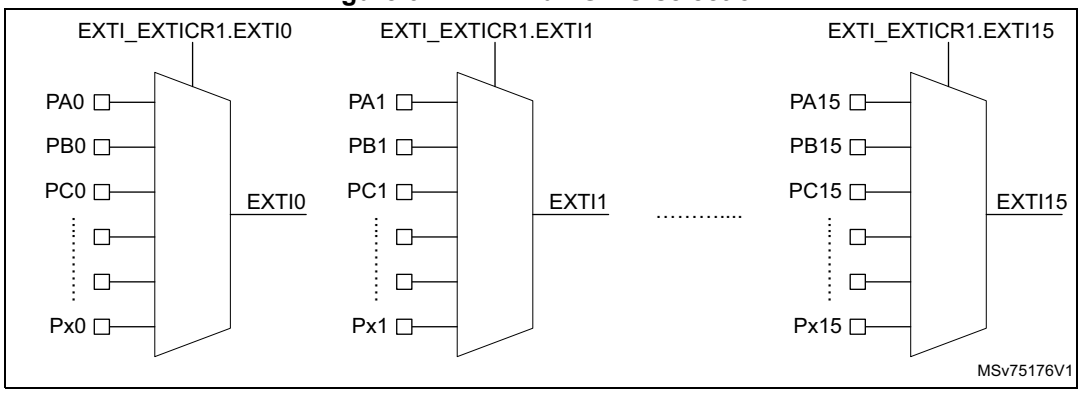
Figure 60. EXTI direct events



16.3.3 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wake-up. GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of a GPIO port as EXTI mux output is controlled in EXTI_EXTICRx (x = 1 to 4).

Figure 61. EXTI mux GPIO selection



The EXTI mux outputs are available as output signals from the EXTI to trigger other peripherals, whatever the masking in EXTI_IMR and EXTI_EMR registers.

16.4 EXTI functional behavior

The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the CPU wake-up generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 104. Masking functionality

CPU interrupt enable (in EXTI_IMR.IMn)	CPU event enable (in EXTI_EMR.EMn)	Configurable event inputs (in EXTI_RPR.RPIFn and EXTI_FPR.FPIFn)	Exti(n) interrupt(1)	CPU event	CPU wake-up
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes(2)
	1	Status latched	Yes	Yes	Yes

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU NVIC.
2. Only if CPU event is enabled in EXTI_IMR.IMn.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending bits EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set: the CPU sub-system is woken up and the CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn pending bits must be cleared by software writing it to 1. This action clears the CPU interrupt.

For the configurable event inputs, an event request can be generated by software when writing a 1 in the software interrupt/event register EXTI_SWIER, allowing the generation of a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

16.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_PRIVCFGR. At EXTI level, the protection consists in preventing the following unauthorized write access:

- Change the settings of the privileged configurable events.
- Change the masking of the privileged input events.
- Clear pending status of the privileged input events.

Table 105. Register protection overview

Register name	Access type	Protection(1)
EXTI_RTSR	RW	Privilege can be bit-wise enabled in EXTI_PRIVCFGR
EXTI_FTSR	RW	
EXTI_SWIER	RW	
EXTI_RPR	RW	
EXTI_FPR	RW	
EXTI_PRIVCFGR	RW	Always privilege.
EXTI_EXTICRn	RW	Privilege can be bit-wise enabled in EXTI_PRIVCFGR.

Table 105. Register protection overview (continued)

Register name	Access type	Protection[1]
EXTI_IM	RW	Privilege can be bit-wise enabled in EXTI_PRIVCFGR.
EXTI_EMR	RW	

1. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR register).

16.5.1 EXTI privilege protection

When privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privileged access. An unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privileged access and unprivileged access.

16.6 EXTI registers

The EXTI register map is divided in the following sections:

Table 106. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [57:32] configuration
0x060 - 0x06C	EXTI IO port mux selection
0x070 - 0x0BC	CPU input event configuration

All registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

For the actual availability of some peripherals, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them as reserved and keep them at reserved value.

16.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RTx:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 16 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FTx:** Falling trigger event configuration bit of configurable event input x [(1)] (x = 16 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and Interrupt) for input line

1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **SWIx:** Software interrupt on event x (x = 16 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

16.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RPIFx:** configurable event inputs x rising edge pending bit (x = 16 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

16.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FPIFx:** configurable event inputs x falling edge pending bit (x = 16 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

16.6.6 EXTI privilege configuration register (EXTI_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for privilege capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVx:** Privilege enable on event input x (x = 31 to 0)

0: Event privilege disabled (unprivileged)

1: Event privilege enabled (privileged)

16.6.7 EXTI rising trigger selection register 2 (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT39	Res.	Res.	RT36	Res.	RT34	Res.	Res.
								rw			rw		rw		

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RT39:** Rising trigger event configuration bit of configurable event input 39[1]
When EXTI_PRIVCFGR.PRIV39 is disabled, RT39 can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIV39 is enabled, RT39 can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RT36:** Rising trigger event configuration bit of configurable event input 36[1]
When EXTI_PRIVCFGR.PRIV36 is disabled, RT36 can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIV36 is enabled, RT36 can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 3 Reserved, must be kept at reset value.

Bit 2 **RT34:** Rising trigger event configuration bit of configurable event input 34[1]
When EXTI_PRIVCFGR.PRIV34 is disabled, RT34 can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIV34 is enabled, RT34 can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bits 1:0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.8 EXTI falling trigger selection register 2 (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT39	Res.	Res.	FT36	Res.	FT34	Res.	Res.
								rw			rw		rw		

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **FT39:** Falling trigger event configuration bit of configurable event input 39(1)
When EXTI_PRIVCFGR.PRIV39 is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIV39 is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **FT36:** Falling trigger event configuration bit of configurable event input 36(1)
When EXTI_PRIVCFGR.PRIV36 is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIV36 is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FT34:** Falling trigger event configuration bit of configurable event input 34(1)
When EXTI_PRIVCFGR.PRIV34 is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIV34 is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bits 1:0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.9 EXTI software interrupt event register 2 (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI39	Res.	Res.	SWI36	Res.	SWI34	Res.	Res.
								rw			rw		rw		

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SWI39:** Software interrupt on event 39

When EXTI_PRIVCFGR.PRIV39 is disabled, SWI39 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV39 is enabled, SWI39 can only be accessed with privileged access. Unprivileged write to this SWI39 is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event 39. This bit is auto cleared by hardware.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **SWI36:** Software interrupt on event 36

When EXTI_PRIVCFGR.PRIV36 is disabled, SWI36 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV36 is enabled, SWI36 can only be accessed with privileged access. Unprivileged write to this SWI36 is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event 36. This bit is auto cleared by hardware.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SWI34:** Software interrupt on event 34

When EXTI_PRIVCFGR.PRIV34 is disabled, SWI34 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV34 is enabled, SWI34 can only be accessed with privileged access. Unprivileged write to this SWI34 is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event 34. This bit is auto cleared by hardware.

Bits 1:0 Reserved, must be kept at reset value.

16.6.10 EXTI rising edge pending register 2 (EXTI_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF39	Res.	Res.	RPIF36	Res.	RPIF34	Res.	Res.
								rw			rw		rw		

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RPIF39:** configurable event inputs 39 rising edge pending bit

When EXTI_PRIVCFGR.PRIV39 is disabled, RPIF39 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV39 is enabled, RPIF39 can only be accessed with privileged access. Unprivileged write to this RPIF39 is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RPIF36:** configurable event inputs 36 rising edge pending bit

When EXTI_PRIVCFGR.PRIV36 is disabled, RPIF36 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV36 is enabled, RPIF36 can only be accessed with privileged access. Unprivileged write to this RPIF36 is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **RPIF34:** configurable event inputs 34 rising edge pending bit

When EXTI_PRIVCFGR.PRIV34 is disabled, RPIF34 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV34 is enabled, RPIF34 can only be accessed with privileged access. Unprivileged write to this RPIF34 is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bits 1:0 Reserved, must be kept at reset value.

16.6.11 EXTI falling edge pending register 2 (EXTI_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF39	Res.	Res.	FPIF36	Res.	FPIF34	Res.	Res.
								rw			rw		rw		

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **FPIF39:** configurable event inputs x falling edge pending bit

When EXTI_PRIVCFGR.PRIV39 is disabled, FPIF39 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV39 is enabled, FPIF39 can only be accessed with privileged access. Unprivileged write to this FPIF39 is discarded, unprivileged read returns 0.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **FPIF36:** configurable event inputs x falling edge pending bit

When EXTI_PRIVCFGR.PRIV36 is disabled, FPIF36 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.FPIF36 is enabled, FPIF36 can only be accessed with privileged access. Unprivileged write to this FPIF36 is discarded, unprivileged read returns 0.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FPIF34:** configurable event inputs x falling edge pending bit

When EXTI_PRIVCFGR.PRIV34 is disabled, FPIF34 can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIV34 is enabled, FPIF34 can only be accessed with privileged access. Unprivileged write to this FPIF34 is discarded, unprivileged read returns 0.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bits 1:0 Reserved, must be kept at reset value.

16.6.12 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for privilege capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV39	PRIV38	PRIV37	PRIV36	PRIV35	PRIV34	PRIV33	PRIV32
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVx:** Privilege enable on event input x (x = 39 to 32)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

16.6.13 EXTI external interrupt selection register (EXTI_EXTICR1)

Address offset: 0x060

Reset value: 0x0000 0000

EXTI mux 0, 1, 2, 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection
These bits are written by software to select the source input for EXTI3 external interrupt.
When EXTI_PRIVCFGR.PRIV3 is disabled, EXTI3 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV3 is enabled, EXTI3 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[3] pin
0x01: PB[3] pin
0x02: PC[3] pin
0x03: PD[3] pin
0x04: PE[3] pin
0x05: PF[3] pin
0x06: PG[3] pin
0x07: PH[3] pin
Others: reserved

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.
When EXTI_PRIVCFGR.PRIV2 is disabled, EXTI2 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV2 is enabled, EXTI2 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[2] pin
0x01: PB[2] pin
0x02: PC[2] pin
0x03: PD[2] pin
0x04: PE[2] pin
0x05: PF[2] pin
0x06: PG[2] pin
0x07: PH[2] pin
Others: reserved

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.
When EXTI_PRIVCFGR.PRIV1 is disabled, EXTI1 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV1 is enabled, EXTI1 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[1] pin
0x01: PB[1] pin
0x02: PC[1] pin
0x03: PD[1] pin
0x04: PE[1] pin
0x05: PF[1] pin
0x06: PG[1] pin
0x07: PH[1] pin
Others: reserved

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI0 external interrupt.
When EXTI_PRIVCFGR.PRIV0 is disabled, EXTI0 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV0 is enabled, EXTI0 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[0] pin
0x01: PB[0] pin
0x02: PC[0] pin
0x03: PD[0] pin
0x04: PE[0] pin
0x05: PF[0] pin
0x06: PG[0] pin
0x07: PH[0] pin
Others: reserved

16.6.14 EXTI external interrupt selection register (EXTI_EXTICR2)

Address offset: 0x064

Reset value: 0x0000 0000

EXTI mux 4, 5, 6, 7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.
When EXTI_PRIVCFGR.PRIV7 is disabled, EXTI7 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV7 is enabled, EXTI7 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[7] pin
0x01: PB[7] pin
0x02: PC[7] pin
0x03: PD[7] pin
0x04: PE[7] pin
0x05: PF[7] pin
0x06: PG[7] pin
Others: reserved

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.
When EXTI_PRIVCFGR.PRIV6 is disabled, EXTI6 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV6 is enabled, EXTI6 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[6] pin
0x01: PB[6] pin
0x02: PC[6] pin
0x03: PD[6] pin
0x04: PE[6] pin
0x05: PF[6] pin
0x06: PG[6] pin
Others: reserved

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection

These bits are written by software to select the source input for EXTI5 external interrupt.
When EXTI_PRIVCFGR.PRIV5 is disabled, EXTI5 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV5 is enabled, EXTI5 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[5] pin
0x01: PB[5] pin
0x02: PC[5] pin
0x03: PD[5] pin
0x04: PE[5] pin
0x05: PF[5] pin
0x06: PG[5] pin
0x07: PH[5] pin
Others: reserved

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection

These bits are written by software to select the source input for EXTI4 external interrupt.
When EXTI_PRIVCFGR.PRIV4 is disabled, EXTI4 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV4 is enabled, EXTI4 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[4] pin
0x01: PB[4] pin
0x02: PC[4] pin
0x03: PD[4] pin
0x04: PE[4] pin
0x05: PF[4] pin
0x06: PG[4] pin
0x07: PH[4] pin
Others: reserved

16.6.15 EXTI external interrupt selection register (EXTI_EXTICR3)

Address offset: 0x068

Reset value: 0x0000 0000

EXTI mux 8, 9, 10, 11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.
When EXTI_PRIVCFGR.PRIV11 is disabled, EXTI11 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV11 is enabled, EXTI11 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[11] pin
0x01: reserved
0x02: PC[11] pin
0x03: PD[11] pin
0x04: PE[11] pin
0x05: PF[11] pin
0x06: PG[11] pin
Others: reserved

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.
When EXTI_PRIVCFGR.PRIV10 is disabled, EXTI10 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV10 is enabled, EXTI10 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[10] pin
0x01: PB[10] pin
0x02: PC[10] pin
0x03: PD[10] pin
0x04: PE[10] pin
0x05: PF[10] pin
0x06: PG[10] pin
Others: reserved

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.
When EXTI_PRIVCFGR.PRIV9 is disabled, EXTI9 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV9 is enabled, EXTI9 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[9] pin
0x01: PB[9] pin
0x02: PC[9] pin
0x03: PD[9] pin
0x04: PE[9] pin
0x05: PF[9] pin
0x06: PG[9] pin
Others: reserved

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection

These bits are written by software to select the source input for EXTI8 external interrupt.
When EXTI_PRIVCFGR.PRIV8 is disabled, EXTI8 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV8 is enabled, EXTI8 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[8] pin
0x01: PB[8] pin
0x02: PC[8] pin
0x03: PD[8] pin
0x04: PE[8] pin
0x05: PF[8] pin
0x06: PG[8] pin
Others: reserved

16.6.16 EXTI external interrupt selection register (EXTI_EXTICR4)

Address offset: 0x06C

Reset value: 0x0000 0000

EXTI mux 12, 13, 14, 15

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection

These bits are written by software to select the source input for EXTI15 external interrupt.
When EXTI_PRIVCFGR.PRIV15 is disabled, EXTI15 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV15 is enabled, EXTI15 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[15] pin
0x01: PB[15] pin
0x02: PC[15] pin
0x03: PD[15] pin
0x04: PE[15] pin
0x05: PF[15] pin
0x06: PG[15] pin
0x07: PH[15] pin
Others: reserved

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection

These bits are written by software to select the source input for EXTI14 external interrupt.
When EXTI_PRIVCFGR.PRIV14 is disabled, EXTI14 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV14 is enabled, EXTI14 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[14] pin
0x01: PB[14] pin
0x02: PC[14] pin
0x03: PD[14] pin
0x04: PE[14] pin
0x05: PF[14] pin
0x06: PG[14] pin
Others: reserved

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt.
When EXTI_PRIVCFGR.PRIV13 is disabled, EXTI13 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV13 is enabled, EXTI13 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[13] pin
0x01: PB[13] pin
0x02: PC[13] pin
0x03: PD[13] pin
0x04: PE[13] pin
0x05: PF[13] pin
0x06: PG[13] pin
Others: reserved

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.
When EXTI_PRIVCFGR.PRIV12 is disabled, EXTI12 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV12 is enabled, EXTI12 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[12] pin
0x01: PB[12] pin
0x02: PC[12] pin
0x03: PD[12] pin
0x04: PE[12] pin
0x05: PF[12] pin
0x06: PG[12] pin
Others: reserved

16.6.17 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0xFFFE 0000

Contains register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IMx:** CPU wake-up with interrupt mask on event input x [1] (x = 31 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with interrupt request from input event x is masked.

1: Wake-up with interrupt request from input event x is unmasked.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

16.6.18 EXTI CPU wake-up with event mask register (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **EMx:** CPU wake-up with event generation mask on event input x (x = 31 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with event generation from Line x is masked.

1: Wake-up with event generation from Line x is unmasked.

16.6.19 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x0000 000B (for STM32C53x/542/55x/562)

Reset value: 0x0000 007B (for STM32C59x/5A3)

Contains register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **IMx:** CPU wake-up with interrupt mask on event input x[(1)] (x = 39 to 32)

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with interrupt request from input event x is masked.

1: Wake-up with interrupt request from input event x is unmasked.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

16.6.20 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EMx:** CPU wake-up with event generation mask on event input x (x = 39 to 32)

When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with event generation from Line x is masked.

1: Wake-up with event generation from Line x is unmasked.

16.6.21 EXTI register map

Table 107. EXTI register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16	RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16	FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	Reserved																																
0x018	EXTI_PRIVCFGR1	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	Reserved																																
0x020	EXTI_RTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT39	Res.	Res.	RT36	Res.	RT34	Res.	Res.
	Reset value																									0			0		0		
0x024	EXTI_FTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT39	Res.	Res.	FT36	Res.	FT34	Res.	Res.
	Reset value																									0			0		0		
0x028	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI39	Res.	Res.	SWI36	Res.	SWI34	Res.	Res.
	Reset value																									0			0		0		
0x02C	EXTI_RPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF39	Res.	Res.	RPIF36	Res.	RPIF34	Res.	Res.
	Reset value																									0			0		0		

Table 107. EXTI register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	EXTI_FPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF39	Res.	FPIF36	Res.	FPIF34	Res.	Res.
	Reset value																										0		0		0		
0x034	Reserved																																
0x038	EXTI_PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV39	PRIV38	PRIV37	PRIV36	PRIV35	PRIV34	PRIV33	PRIV32
	Reset value																									0	0	0	0	0	0	0	0
0x03C - 0x05C	Reserved																																
0x060	EXTI_EXTICR1	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	EXTI_EXTICR2	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	EXTI_EXTICR3	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	EXTI_EXTICR4	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI12[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070 - 0x07C	Reserved																																
0x080	EXTI_IMR1	IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	EXTI_EMR1	EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088 - 0x08C	Reserved																																
0x090	EXTI_IMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
	Reset value																									0	1	1	0	1	0	1	1
0x094	EXTI_EMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

17 Cyclic redundancy check calculation unit (CRC)

17.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

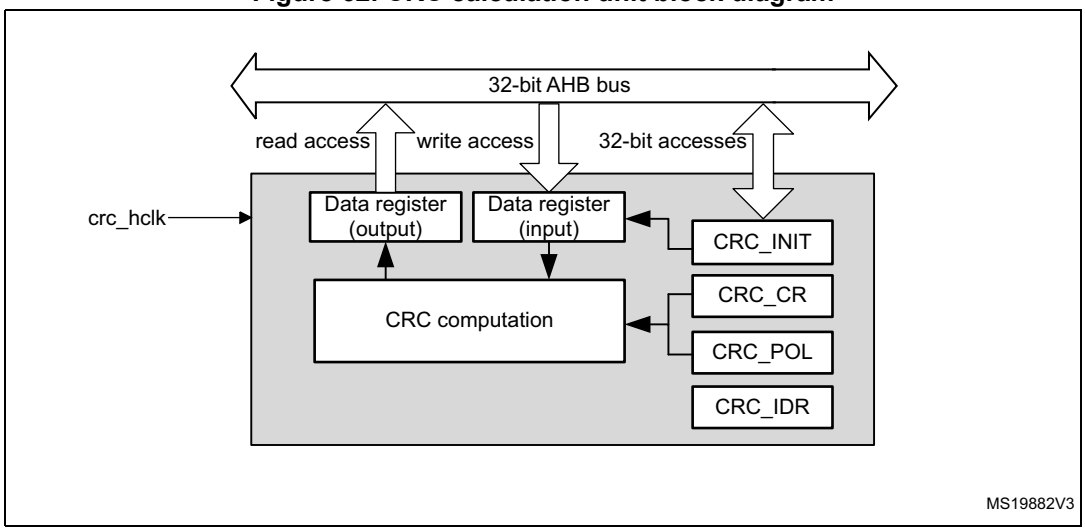
17.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility options on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

17.3 CRC functional description

17.3.1 CRC block diagram

Figure 62. CRC calculation unit block diagram



17.3.2 CRC internal signals

Table 108. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

17.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. When the RTYPE_IN bit is set in CRC_CR, non-word accesses to CRC_DR register are ignored. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on input data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register. If RTYPE_IN bit is set in this register, the input reversing operation can be performed with a byte or half-word granularity, but not with single bit granularity.

For example, with RTYPE_IN cleared, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte (REV_IN[1:0] = 01)
- 0xD458B23C with bit-reversal done by half-word (REV_IN[1:0] = 10)
- 0xB23CD458 with bit-reversal done on the full word (REV_IN[1:0] = 11)

With RTYPE_IN set, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x3C4D1A2B with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x4D3C2B1A with byte-reversal done by word (REV_IN[1:0] = 10)

The output data can also be reversed by setting the REV_OUT[1:0] bits in the CRC_CR register. If RTYPE_OUT is set in this register, the output reversing operation is performed with a byte or half-word granularity, but not with single bit granularity.

With RTYPE_OUT cleared, the operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

With RTYPE_OUT set, 0x11223344 output data are converted as:

- 0x33441122 with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x44332211 with byte-reversal done by word (REV_IN[1:0] = 10)

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC 7-, 8-, 16-, and 32-bit calculations. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7-, 8-, 16-, or 32-bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

17.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes when RTYPE bit is cleared in CRC_CR. When RTYPE is set, only word accesses are allowed. For the other registers only 32-bit accesses are allowed.

17.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

17.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

17.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
					rw	rw	rw	rw	rw	rw	rw	rw			rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RTYPE_OUT**: Reverse type output
This bit controls the reversal granularity of the output data.
0: Bit level output
1: Byte or half-word level output

Bit 9 **RTYPE_IN**: Reverse type input
This bit controls the reversal granularity of the input data.
0: Bit level input
1: Byte or half-word level input

Bits 8:7 **REV_OUT[1:0]**: Reverse output data
This bitfield controls the reversal of the bit order of the output data.
00: Bit order not affected (RTYPE_OUT = 0 or 1)
01: Bit-reversed output format (RTYPE_OUT = 0) or half-word reversal done by word (RTYPE_OUT = 1)
10: Bit order not affected (RTYPE_OUT = 0) or byte reversal done by word (RTYPE_OUT = 1)
11: Bit order not affected (RTYPE_OUT = 0 or 1)

Bits 6:5 **REV_IN[1:0]**: Reverse input data
This bitfield controls the reversal of the bit order of the input data
00: Bit order not affected (RTYPE_IN = 0 or 1)
01: Bit reversal done by byte (RTYPE_IN = 0) or half-word reversal done by word (RTYPE_IN = 1)
10: Bit reversal done by half-word (RTYPE_IN = 0) or byte reversal done by word (RTYPE_IN = 1)
11: Bit reversal done by word (RTYPE_IN = 0) or bit order is not affected (RTYPE_IN = 1)

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.

17.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value

This register is used to write the CRC initial value.

17.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

17.4.6 CRC register map

Table 109. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																						0	0	0	0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

18 CORDIC coprocessor (CORDIC)

18.1 CORDIC introduction

The CORDIC coprocessor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing, and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making possible the use of a lower operating frequency, or freeing up processor cycles to perform other tasks.

18.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atan2, atanh, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

18.3 CORDIC functional description

18.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle θ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle θ. The x and y cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of θ. Inversely, the angle of a vector [x, y] corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

18.3.2 CORDIC functions

The first step when using the coprocessor is to select the required function, by programming the FUNC field of the CORDIC_CR register.

Table 110 lists the functions supported by the CORDIC coprocessor.

Table 110. CORDIC functions

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	angle θ	modulus m	$m \cdot \cos\theta$	$m \cdot \sin\theta$
Sine	angle θ	modulus m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
Phase	x	y	atan2(y,x)	$\sqrt{x^2+y^2}$
Modulus	x	y	$\sqrt{x^2+y^2}$	atan2(y,x)
Arctangent	x	none	$\tan^{-1} x$	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	$\tanh^{-1} x$	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

Several functions take two input arguments (ARG1 and ARG2), and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: sin θ also generates cos θ, and cos θ also generates sin θ. Similarly for rectangular-to-polar conversion (phase(x,y), modulus(x,y)) and for hyperbolic functions (cosh θ, sinh θ).

Note: The exponential function, exp x, can be obtained as the sum of sinh x and cosh x. Furthermore, base N logarithms, $\log_N x$, can be derived by multiplying ln x by a constant K, where K = 1/ln N.

For certain functions (atan, log, sqrt) a scaling factor (see *Section 18.3.4*) can be applied to extend the range beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

Cosine

Table 111. Cosine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m • cos θ	[-1, 1]

Table 111. Cosine parameters (continued)

Parameter	Description	Range
RES2	m • sin θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

Table 112. Sine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m • sin θ	[-1, 1]
RES2	m • cos θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

Table 113. Phase parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Phase angle θ in radians, divided by π	[-1, 1]

Table 113. Phase parameters (continued)

Parameter	Description	Range
RES2	Modulus m	[0, 1]
SCALE	Not applicable	0

This function calculates the phase angle in the range -π to π of a vector **v** = [x y] (also known as atan2(y,x). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle θ of the vector **v**. RES1 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES2 is saturated to 1.

Modulus

Table 114. Modulus parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Modulus m	[0, 1]
RES2	Phase angle θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the magnitude, or modulus, of a vector **v** = [x y]. It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle θ of the vector **v**. RES2 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

Arctangent

Table 115. Arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-1, 1]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \tan^{-1} x$, in radians, divided by p	[-1, 1]
RES2	Not applicable	-
SCALE	n	[0 7]

This function calculates the arctangent, or inverse tangent, of the input argument x.

The primary argument, ARG1, is the input value, $x = \tan \theta$. If $|x| > 1$, a scaling factor of 2^{-n} must be applied in software such that $-1 < x \cdot 2^{-n} < 1$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the scale factor n must be programmed in the SCALE parameter.

Note that the maximum input value allowed is $\tan \theta = 128$, which corresponds to an angle $\theta = 89.55$ degrees. For $|x| > 128$, a software method must be used to find $\tan^{-1} x$.

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle $\theta = \tan^{-1} x$. RES1 must be multiplied by $2^n \cdot \pi$ to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 116. Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \cosh x$	[0.5 0.846]
RES2	$2^{-n} \cdot \sinh x$	[-0.683 0.683]
SCALE	n	1

This function calculates the hyperbolic cosine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. Since the minimum value of cosh x is 1, which is beyond the range of the q1.31 format, a scaling factor of 2^{-n} must be applied in software. The factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

Table 117. Hyperbolic sine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559, 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \sinh x$	[-0.683, 0.683]
RES2	$2^{-n} \cdot \cosh x$	[0.5, 0.846]
SCALE	n	1

This function calculates the hyperbolic sine of a hyperbolic angle x. It can also be used to calculate the exponential functions e^x = cosh x + sinh x, and e^{-x} = cosh x- sinh x.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. For all input values, a scaling factor of 2^{-n} must be applied in software, where n = 1. The scaled value x • 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

Table 118. Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x \bullet 2^{-n}$	[-0.403 0.403]
ARG2	Not applicable	-
RES1	$2^{-n} \bullet$atanh x	[-0.559 0.559]
RES2	Not applicable	-
SCALE	n	1

This function calculates the hyperbolic arctangent of the input argument x.

The primary argument is the input value x. Only values of x in the -0.806 to +0.806 range are supported. The value x must be scaled by a factor 2^{-n}, where n = 1. The scaled value

x • 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

Natural logarithm

Table 119. Natural logarithm parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.054 0.875]
ARG2	Not applicable	-
RES1	$2^{-(n+1)}.\ln x$	[-0.279 0.137]
RES2	Not applicable	-
SCALE	n	[1 4]

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor 2^{-n}, such that $x < 2^{-n} < 1-2^{-n}$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 120 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 120. Natural log scaling factors and corresponding ranges

n	x range	ARG1 range
1	$0.107 \leq x < 1$	$0.0535 \leq ARG1 < 0.5$
2	$1 \leq x < 3$	$0.25 \leq ARG1 < 0.75$
3	$3 \leq x < 7$	$0.375 \leq ARG1 < 0.875$
4	$7 \leq x \leq 9.35$	$0.4375 \leq ARG1 < 0.584$

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.

Square root

Table 121. Square root parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.027 0.875]
ARG2	Not applicable	-
RES1	$2^{-n}\sqrt{x}$	[0.04 1]
RES2	Not applicable	-
SCALE	n	[0 2]

This function calculates the square root of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.027 to 2.34 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$.

The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 122 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 122. Square root scaling factors and corresponding ranges

n	x range	ARG1 range
0	$0.027 \leq x < 0.75$	$0.027 \leq ARG1 < 0.75$
1	$0.75 \leq x < 1.75$	$0.375 \leq ARG1 < 0.875$
2	$1.75 \leq x \leq 2.341$	$0.4375 \leq ARG1 \leq 0.585$

The secondary argument is not used.

The primary result is the square root of x. RES1 must be multiplied by 2^n to obtain the correct value.

The secondary result is not used.

18.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to $1 - 2^{-31}$ (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to $1 - 2^{-15}$ (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

18.3.4 Scaling factor

Several of the functions listed in *Section 18.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output, or internal registers. If the scaling factor is

required, it must be calculated by software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

18.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see *Figure 63*.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see *Figure 64*). The square root function converges at roughly twice the speed of the hyperbolic functions (see *Figure 65*).

Figure 63. CORDIC convergence for trigonometric functions

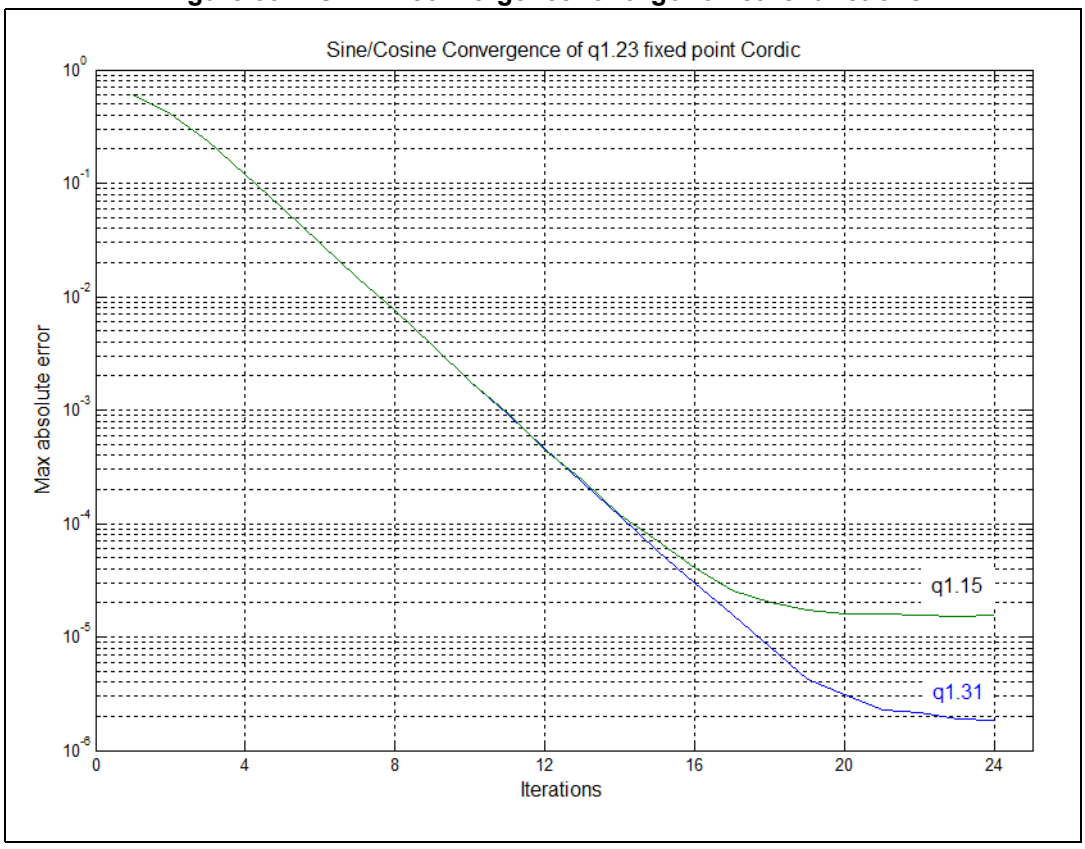


Figure 64. CORDIC convergence for hyperbolic functions

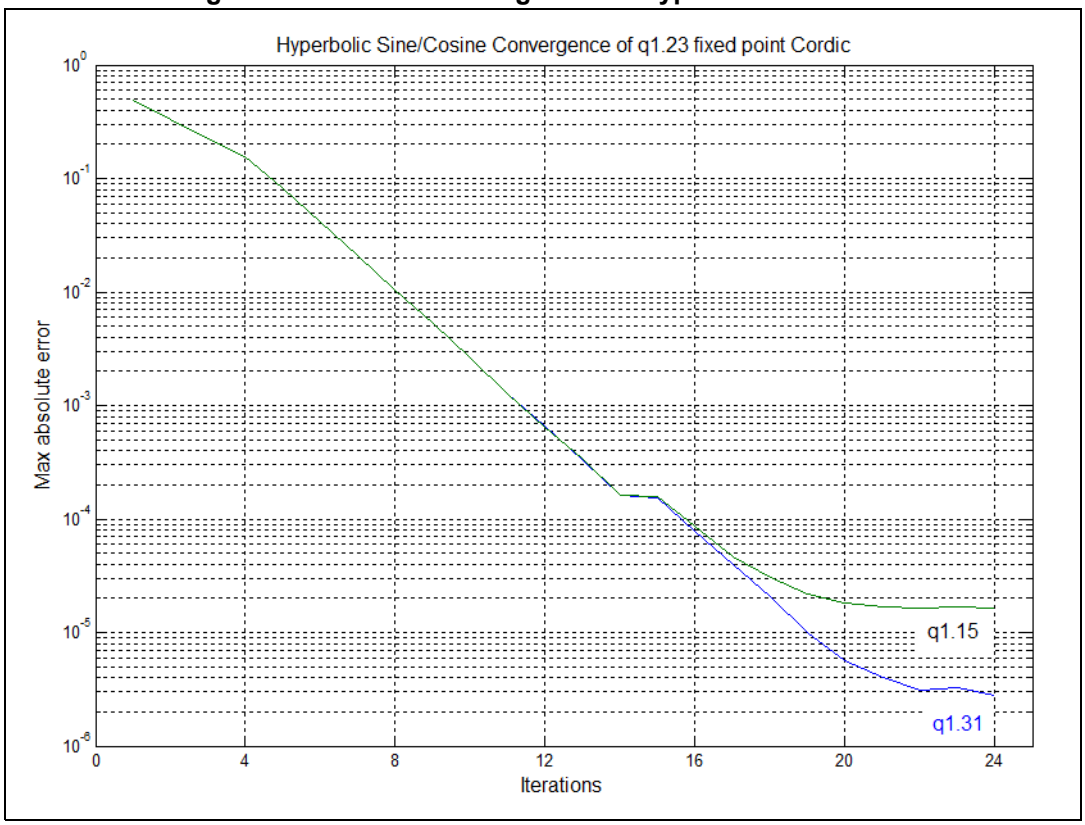
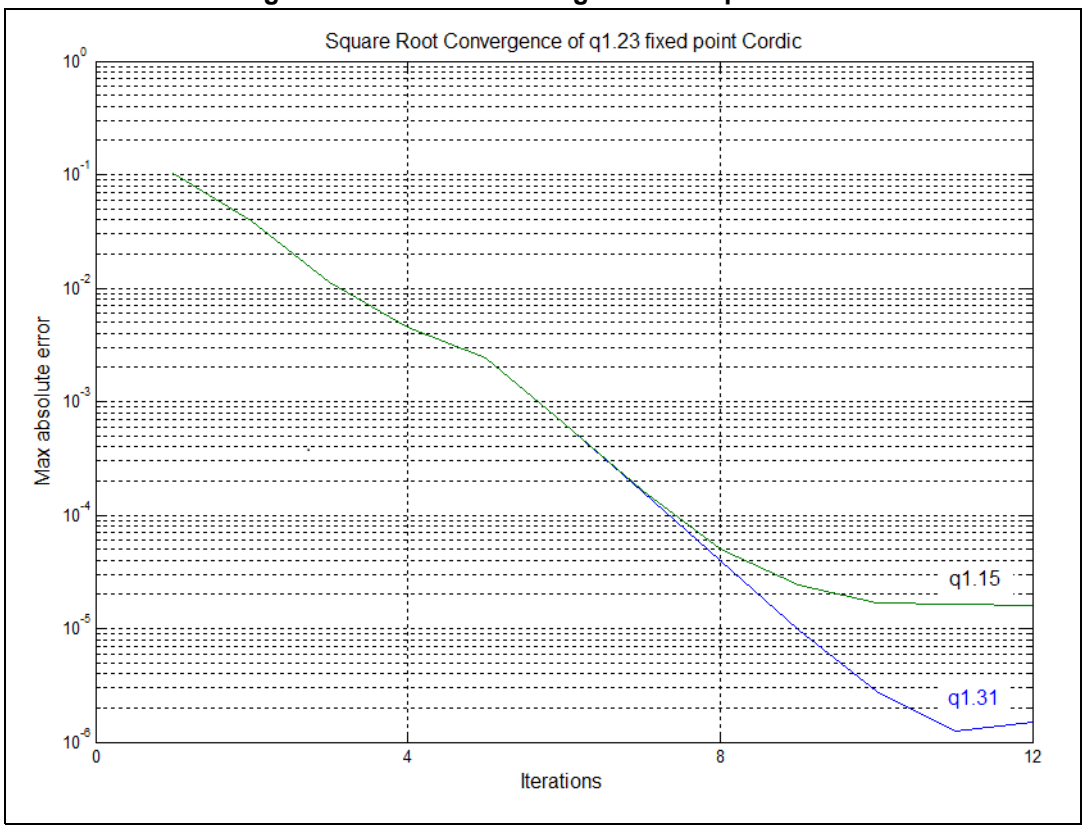


Figure 65. CORDIC convergence for square root



Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in *Table 123*, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The coprocessor stops as soon as the programmed number of iterations is completed, and the result can be read immediately.

Table 123. Precision vs. number of iterations

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sin, Cos, Phase(2), Mod, Atan(4)	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

Table 123. Precision vs. number of iterations (continued)

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sinh, Cosh, Atanh, Ln(3)	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt(4)	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to 2^{-16} for q15 format or 2^{-20} for q31 format.
2. For modulus > 0.5. The achievable precision reduces proportionally to the magnitude of the modulus, as quantization error becomes significant.
3. SCALE = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. SCALE = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

18.3.6 Zero-overhead mode

The fastest way to use the coprocessor is to preprogram the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

The calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished), a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to reprogram the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument (ARG1) must be written first, followed by the secondary argument (ARG2). If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.

Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait-states until the calculation is completed, before returning the result. It is then possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation, and prepare the next input data, if needed. The CORDIC_CSR register can be reprogrammed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous ones have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. IF NRES = 1, a second read of CORDIC_RDATA must be performed to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be a calculation in progress, or waiting for the results to be read, and an operation pending. Any further access to CORDIC_WDATA while an operation is pending cancels it and overwrites the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step *3*.

18.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice, depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. The processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (for example, if low latency interrupts must be serviced).

18.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However, it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

18.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention. This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. The sequence is started by the processor setting the DMAWEN flag, the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generates a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore useful only when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.

Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and may block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

18.4 CORDIC registers

The CORDIC registers can be accessed only in 32-bit word format

18.4.1 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARG SIZE	RES SIZE	N ARGS	NRES	DMA WEN	DMA REN	IEN
r									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SCALE[2:0]			PRECISION[3:0]				FUNC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RRDY**: Result ready flag

0: No new data in output register

1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data

0: 32-bit

1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format. The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data

0: 32-bit

1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.

Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register
0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.
1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.
Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register
0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.
1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.
Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel
0: Disabled. No DMA write requests are generated.
1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel
0: Disabled. No DMA read requests are generated.
1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt.
0: Disabled. No interrupt requests are generated.
1: Enabled. An interrupt request is generated whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor
The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor 2^{-n}, and/or the results need to be multiplied by 2^{n}. Refer to *Section 18.3.2* for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)
0: reserved
1 to 15: (Number of iterations)/4
To determine the number of iterations needed for a given accuracy refer to *Table 123*.
Note that for most functions, the recommended range for this field is 3 to 6.

Bits 3:0 **FUNC[3:0]**: Function

0: Cosine
1: Sine
2: Phase
3: Modulus
4: Arctangent
5: Hyperbolic cosine
6: Hyperbolic sine
7: Arctanh
8: Natural logarithm
9: Square root
10 to 15: Reserved

18.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARG[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to *Section 18.3.2* for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.

18.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RES[31:0]**: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.

A read from this register resets the RRDY flag in the CORDIC_CSR register.

18.4.4 CORDIC register map

Table 124. CORDIC register map and reset value

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CORDIC_CSR	RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARGSIZE	RESSIZE	NARGS	NRES	DMAWEN	DMAREN	IEN	Res.	Res.	Res.	Res.	SCALE [2:0]				PRECISION [3:0]				FUNC [3:0]			
	Reset value	0									0	0	0	0	0	0	0					0	0	0	0	0	1	0	1	0	0	0	0
0x04	CORDIC_WDATA	ARG[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x08	CORDIC_RDATA	RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

19 Extended-SPI interface (XSPI)

This section applies only for STM32C59xx/C5Axx devices.

19.1 XSPI introduction

The XSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the XSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The XSPI supports the following protocol with associated frame format:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

19.2 XSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad, octal communication
- Dual-memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- XSPI interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.

- Dual chip select support (NCS1 and NCS2)
- Extended external memory support: if two same size external memories (extmem1 and extmem2) are connected to the same I/O port, contiguously in the memory map and driven by a single XSPI. This XSPI automatically switches CS to extmem1 or extmem2, according to the address on interconnect side.
- Possibility to disable the automatic prefetch

19.3 XSPI implementation

Table 125. XSPI implementation

XSPI feature	XSPI1
Octal mode configuration (8-bit)	X
16-bit mode configuration	-
HyperBus standard compliant	X
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AHB compliant data interface	X
Dual AHB interface	X
Asynchronous AHB clock versus kernel clock	X
Functional modes: indirect, automatic status-polling, and memory-mapped(1)	X
Read and write support in memory-mapped mode	X
Dual-quad configuration	X
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS,DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8-, 16-, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Bus error bit	-
Address offset	-
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
GPDMA interface	X
Dual chip select	X
Extended external memory	X

Table 125. XSPI implementation (continued)

XSPI feature	XSPI1
Prefetch disable	X
Prefetch hardware software	-

1. Automatic status polling mode is not supported.

19.4 XSPI functional description

19.4.1 XSPI block diagram

Figure 66. XSPI block diagram for octal configuration

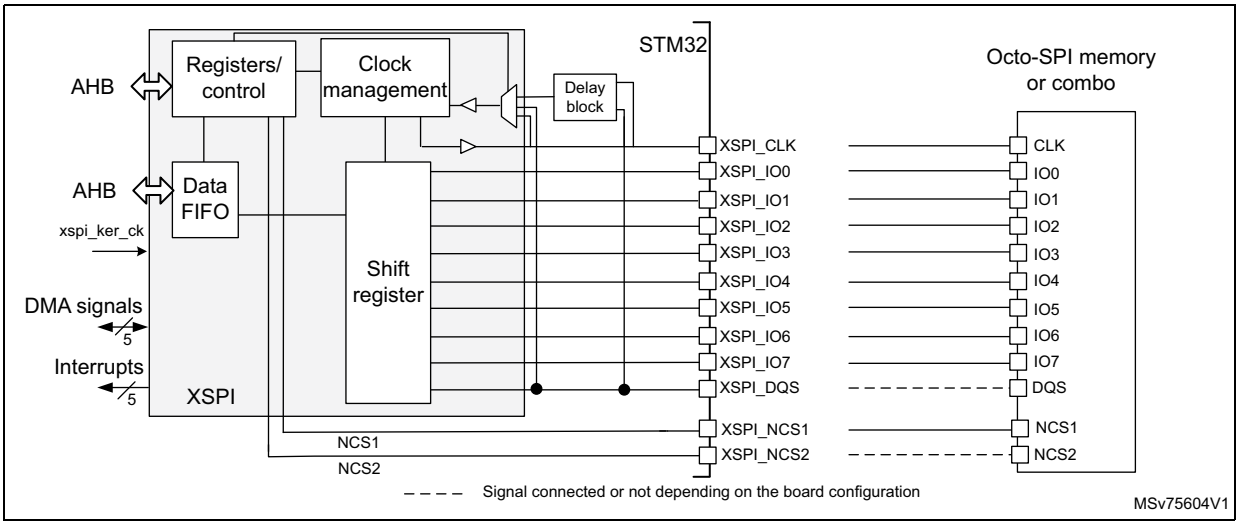


Figure 67. XSPI block diagram for quad configuration

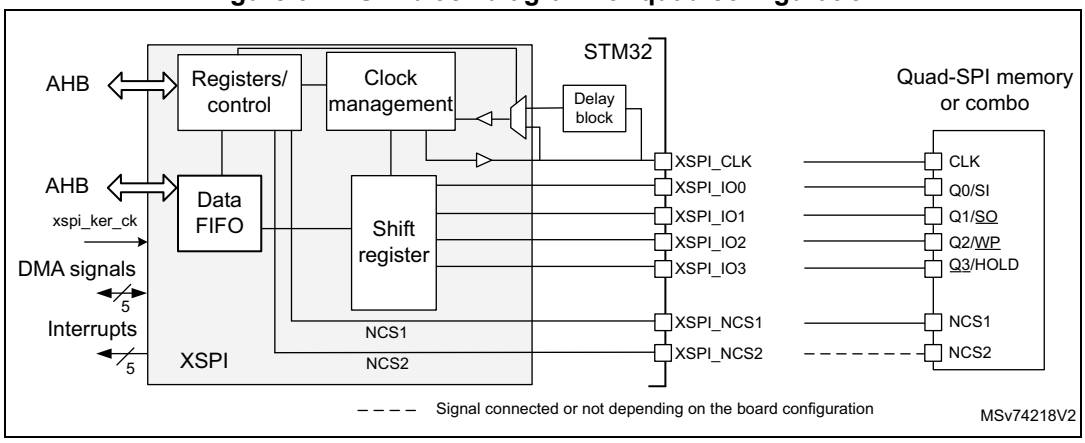
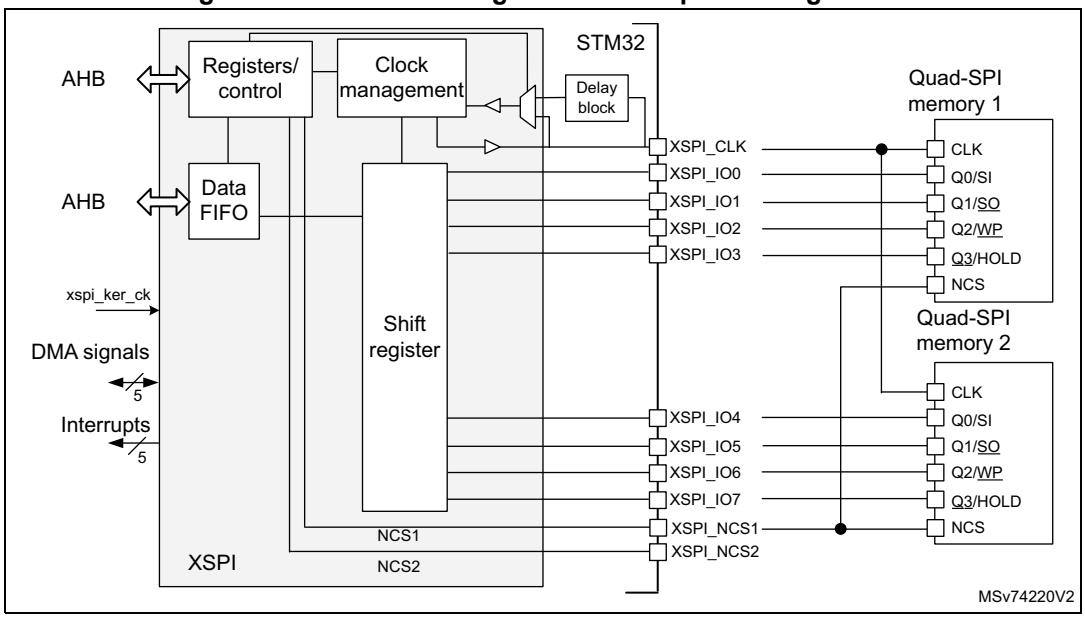


Figure 68. XSPI block diagram for dual-quad configuration



19.4.2 XSPI pins and internal signals

Table 126. XSPI input/output pins

Pin name	Type	Description
XSPI_CLK	Output	XSPI clock
XSPI_IOn (n = 0 to 7)	Input/output	XSPI data pins
XSPI_NCS1,2	Output	Chip select for the memory
XSPI_DQS	Input/output	Data strobe/write mask signal from/to the memory

Caution: Use the same configuration (output speed, HSLV[(a)]) for all XSPI input/output pins to avoid any data corruption.

Table 127. XSPI internal signals

Signal name	Type	Description
xspi_hclk	Input	XSPI AHB clock
xspi_ker_ck	Input	XSPI kernel clock
xspi_dma	N/A	DMA request signal
xspi_it	Output	Global interrupt line (see *Table 131* for the multiple sources of interrupt)

a. When applicable.

19.4.3 XSPI interface to memory modes

The XSPI supports the following protocol:

- regular-command protocol
- HyperBus protocol

The XSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select (in the following sections, unless otherwise specified, NCS applies for NCS1 or NCS2. Only one of these two signals is active at a given moment in time)
- CLK: communication clock
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

19.4.4 XSPI regular-command protocol

When in regular-command protocol, the XSPI communicates with the external device using commands. Each command can include the following phases:

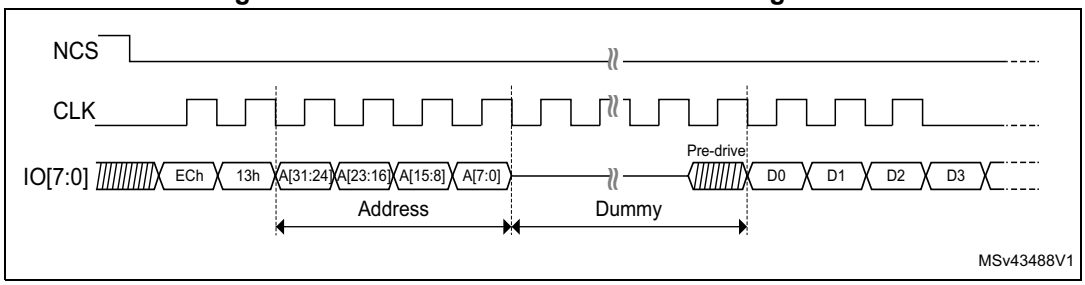
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 69. SDR read command in octal configuration



The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured by ISIZE[1:0] in XSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of XSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using IMODE[2:0] of XSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in XSPI_CCR.

When IMODE[2:0] = 000 in XSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the write operation is specified in XSPI_WIR, and the instruction format is specified in XSPI_WCCR. The instruction used for the read operation and the instruction format are specified in XSPI_IR and XSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in XSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified by ADDRESS[31:0] in XSPI_AR. In memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in XSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in XSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in XSPI_WCCR. The address format for the read operation is specified in XSPI_CCR.

Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in XSPI_CCR. The bytes to be sent are specified in XSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in XSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in XSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in XSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in XSPI_WABR, and the alternate byte format is specified in XSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in XSPI_ABR and XSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in XSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-, the quad-, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in XSPI_WTCR. The dummy cycles for the read operation are specified in XSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in XSPI_DLR. In this mode, the data to be sent to the external device must be written to XSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading XSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in XSPI_DLR, and the data received from the external device can be obtained by reading XSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AHB to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in XSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in XSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

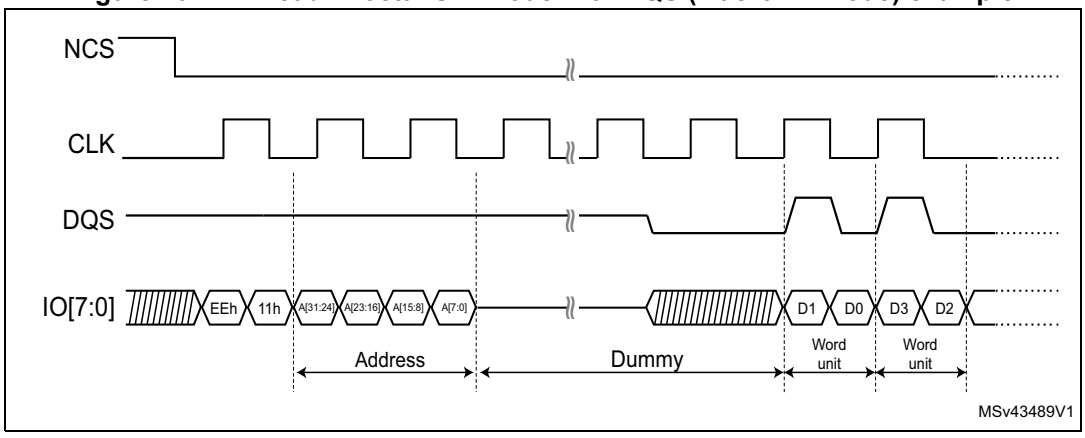
In memory-mapped mode, the data format for the write operation is specified in XSPI_WCCR. The data format for the read operation is specified in XSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in XSPI_CCR.

Figure 70. DTR read in octal-SPI mode with DQS (Macronix mode) example



19.4.5 XSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the XSPI generating the clock.

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 010, IO0 and IO1 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 011, IO0 to IO3 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase, when DMODE[2:0] = 100, IO[0:7] are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

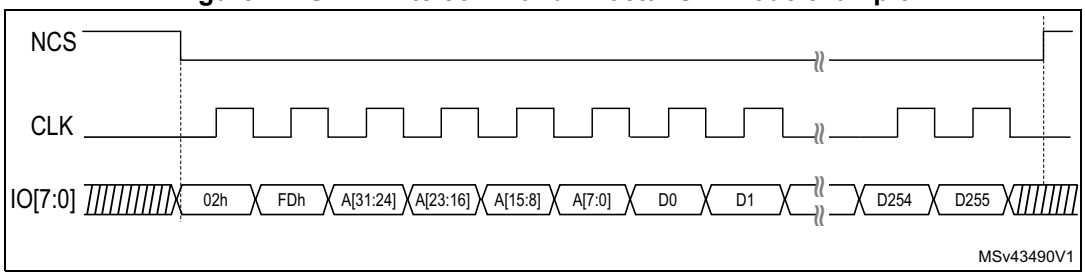
Single-data rate (SDR) mode

By default, all the phases operate in single-data rate (SDR) mode.

In SDR mode, when the XSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the XSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in XSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 71. SDR write command in octal-SPI mode example



Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in XSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in XSPI_WCCR. The DTR mode for each phase of the read operations is specified in XSPI_CCR.

In DTR mode, when the XSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

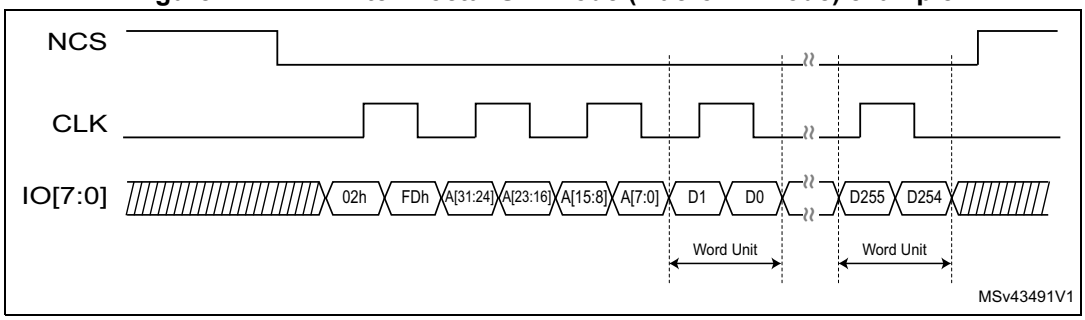
In DTR mode, when the XSPI drives the IO0 to IO7 signals in the data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the XSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in XSPI_CCR, the software must clear SSHIFT in XSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of XSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 72. DTR write in octal-SPI mode (Macronix mode) example



Dual-quad configuration

When DMM = 1 in XSPI_CR, the XSPI is in dual-memory configuration: if DMODE = 011, two external quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in XSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the XSPI gives for address X is the byte at the address X/2 of device A, and the byte that the XSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 2 bytes (16 bits), and the XSPI receives one byte from each device.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in XSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

19.4.6 HyperBus protocol

The XSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- DQS for read and write data strobe and latency insertion
- NCS
- CLK

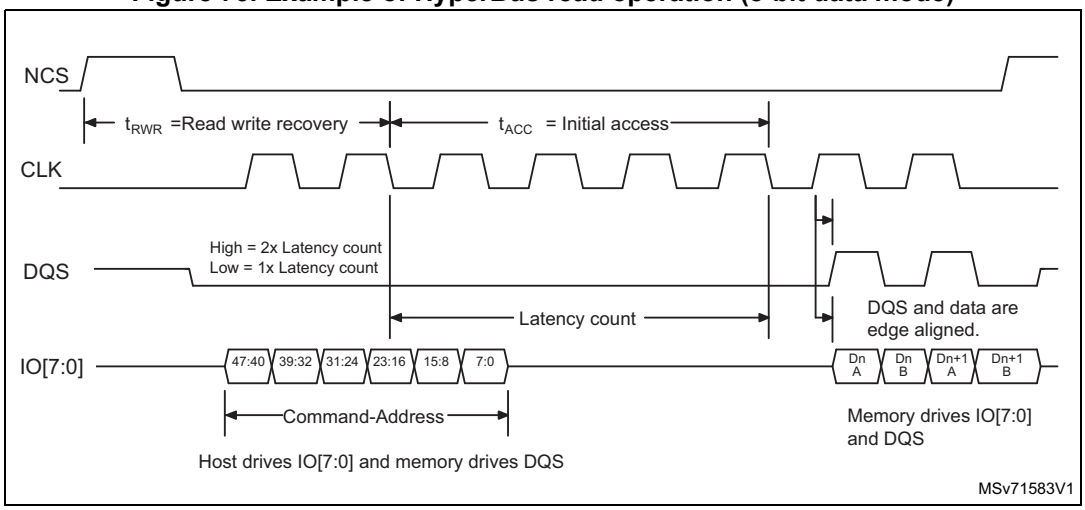
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 73. Example of HyperBus read operation (8-bit data mode)



The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the XSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 128. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write.
46	Address space	Indicates if the transaction accesses the memory or the register space.

Table 128. Command/address phase description

CA bit	Bit name	Description
45	Burst type	Indicates if the burst is linear or wrapped.
44-16	Row and upper column address	Selects the row and the upper column addresses.
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page.

The address space is configured through the memory type MTYP[2:0] in XSPI_DCR1.

The total size of the device must be configured through DEVSIZE[4:0] of XSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined by TRWR[7:0] in XSPI_HLCR)
- t_{ACC}: access time for the device (defined by TACC[7:0] in XSPI_HLCR) according to the memory latency

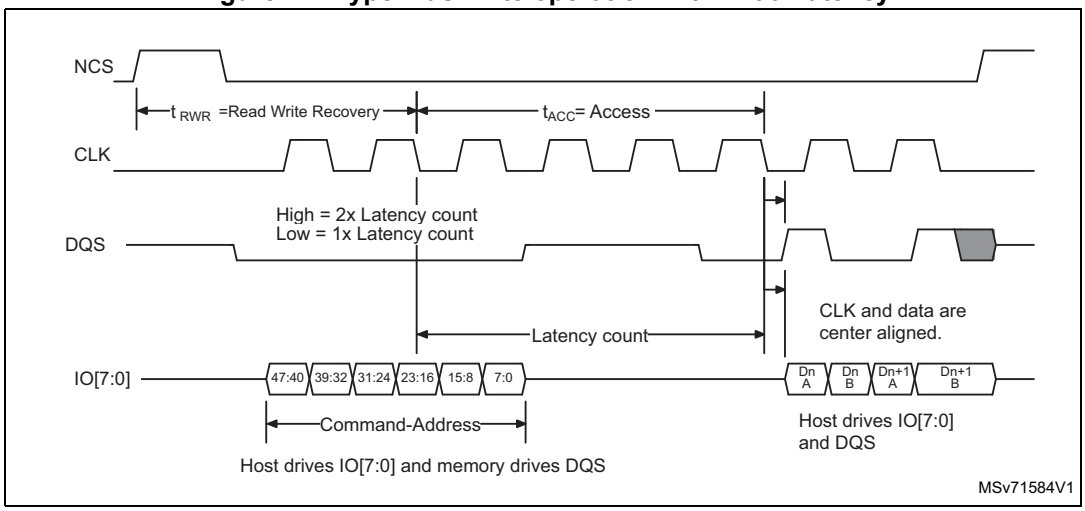
During the read operation, the DQS is used by the device, in two ways (see *Figure 73*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the DQS is used:

- by the device, during the command/address phase, to request an additional latency.
- by the XSPI, during the data phase, for write data masking.

Figure 74. HyperBus write operation with initial latency



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS must be tied to one during one of the DQS signals, during the command/address phase.

An additional t_{ACC} duration is added by the XSPI to meet the device request.

Figure 75. HyperBus read operation with additional latency

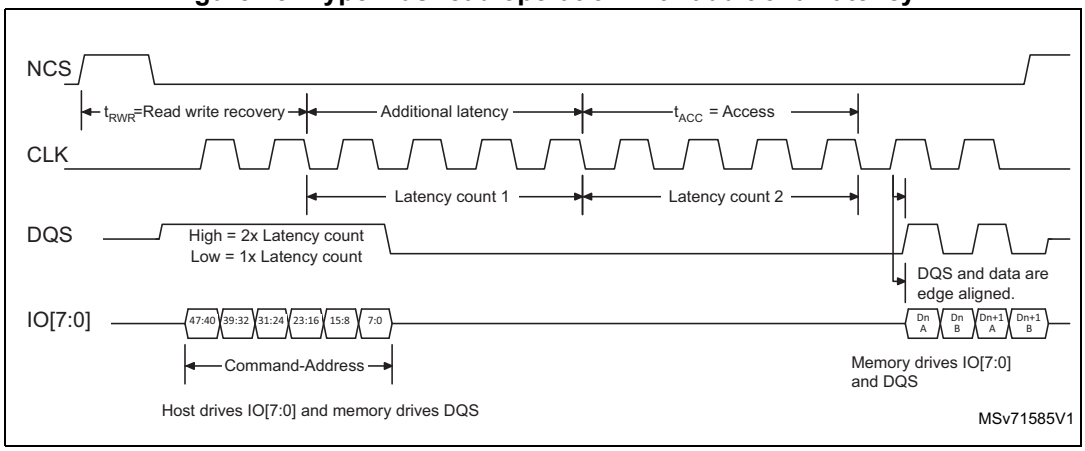
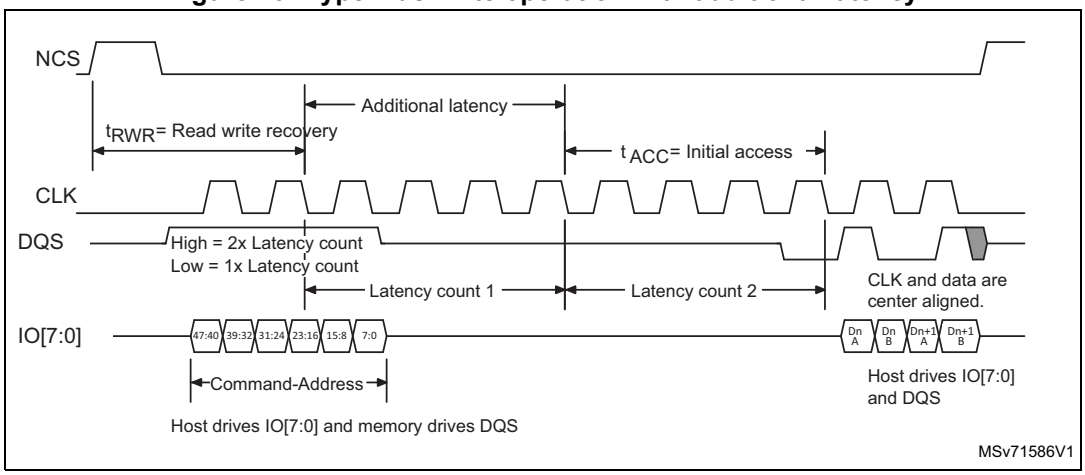


Figure 76. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

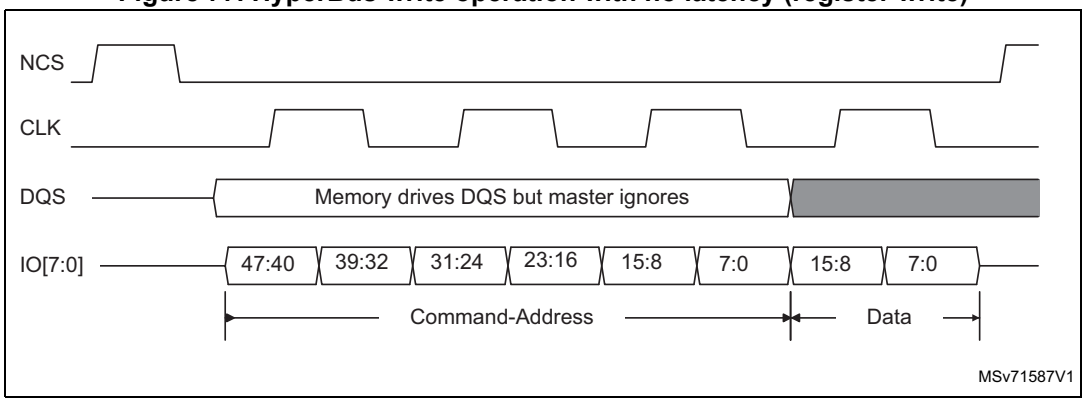
The latency can be forced to 2 x t_{ACC} by setting LM in XSPI_HLCR.

In this XSPI latency mode, the state of the DQS signal is not taken into account by the XSPI, and an additional latency is always added, leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in XSPI_HLCR.

Figure 77. HyperBus write operation with no latency (register write)

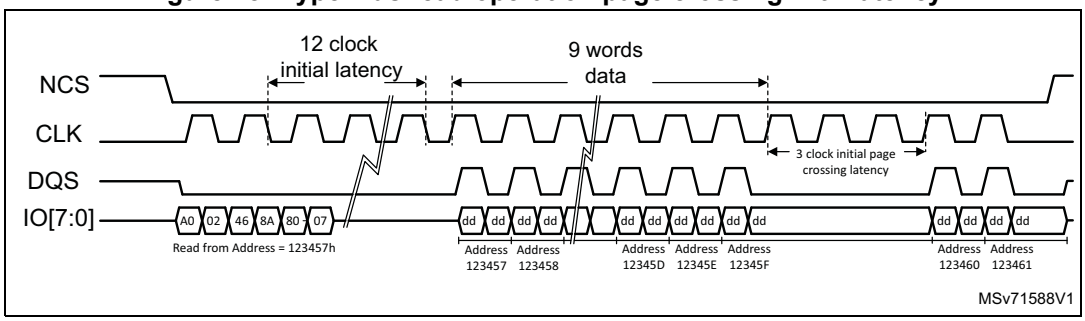


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 78. HyperBus read operation page crossing with latency



19.4.7 Specific features

The XSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The XSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in XSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 4 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated XSPI_WPxxx registers must be programmed according to the wrapped-read access characteristics. These registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (XSPI_CCR, XSPI_TCR, XSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the XSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured by CSBOUND[4:0] in XSPI_DCR3: it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. The NCS is then released each time the LSB address is equal to 0xF, and each time a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed by REFRESH[31:0] in XSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

19.4.8 XSPI operating mode introduction

The XSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

19.4.9 XSPI indirect mode

In indirect mode, the commands are started by writing to the XSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in XSPI_CR, the XSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to XSPI_DR.

When FMODE[1:0] = 01, the XSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading XSPI_DR.

In indirect mode, when the XSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The XSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in XSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the XSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in XSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the XSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in XSPI_SR, and an interrupt is generated if TCIE = 1 in XSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in XSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the XSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in XSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in XSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in XSPI_DR

A write to XSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in XSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the XSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication

in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (XSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01).
- when a write is performed to DATA[31:0] (XSPI_DR) in indirect-write mode (when FMODE = 00).
- when a (dummy) write is performed to INSTRUCTION[31:0] (XSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01).

As soon as a transfer is started, the BUSY bit (XSPI_SR[5]) is automatically set.

FIFO and data management

Data in indirect mode pass through a 32-byte FIFO that is internal to the XSPI. FLEVEL in XSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in XSPI_DR, and immediately transferred in the internal FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in XSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to XSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

19.4.10 XSPI automatic status-polling mode

In automatic status-polling mode, the XSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value. The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in XSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in XSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in XSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in XSPI_CR, the AND-match mode is activated: SMF is set in XSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in XSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in XSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the XSPI is disabled (EN = 0).

XSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in XSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. As a consequence, the memory must then be configured in fixed latency.

19.4.11 XSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256-Mbyte range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the XSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the XSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time. This prefetch feature can be disabled by setting the NOPREF bit in XSPI_DCR1. Disabling the prefetch can be beneficial for performance when the transfer requests are short and jumping frequently between non-consecutive addresses.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in XSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in XSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes. The indirect-write mode fulfills this operation.

However, if the application requires the use of the MCE for encryption (check MCE product availability), the memory-mapped write mode may be used to program encrypted data to external flash memory under the following conditions:

- Prefetch must be enabled.
- In block cipher mode, the CPU must write a complete 128-bit data block to prevent the MCE from initiating read-modify-write operations when only a few bytes need to be programmed. This precaution avoids incorrect programming operations. There are no specific constraints to respect if the MCE is used in stream cipher mode.
- Apply the abort sequence to exit memory-mapped mode when the data linked to the page has been written in the external memory buffers. The abort sequence triggers the start of the page programming.
- Switch to the automatic status-polling mode to monitor the completion of the page programing phase.
- Relaunch the write enable command in indirect mode, then switch back to the memory-mapped mode configuration to continue to program additional pages if any.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

The controller can apply an automatic address offset to simplify memory area code switching, such as during a firmware update (if the address offset feature is available in the product implementation table). To enable this feature, set the ADOFFEN bit in the XSPI_CR register and configure the offset value in the ADOFF[4:0] bitfield of the XSPI_DCR1 register. The offset is modulo 256 bytes.

19.4.12 XSPI configuration introduction

The XSPI configuration is done in three steps:

1. XSPI system configuration
2. XSPI device configuration
3. XSPI mode configuration

19.4.13 XSPI system configuration

The XSPI is configured using XSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- the DMA use with DMAEN
- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in XSPI_LPTR.

The DMA channel must not be enabled during the XSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and XSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

19.4.14 XSPI device configuration

The parameters related to the external device targeted are configured through XSPI_DCR1 and XSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the associated delay block DLYB connected externally to the peripheral
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{[\text{DEVSIZE}+1]}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the XSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for at least one CLK cycle by default.

If the external device requires more time between commands, CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the latency mode must be configured in XSPI_HLCR.

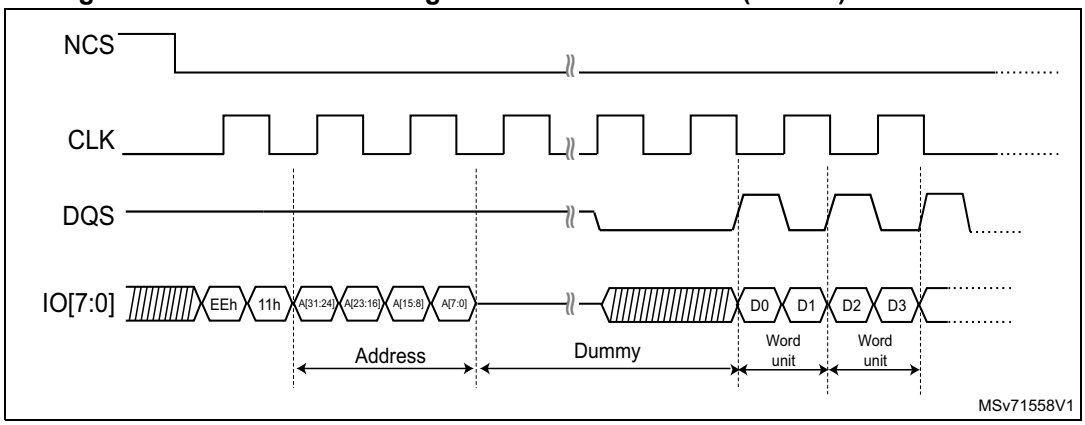
Memory types

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the XSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 79. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the memory. Refer to *Figure 70: DTR read in octal-SPI mode with DQS (Macronix mode) example*.
- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 129: OctaRAM command address bit assignment (based on 64 Mb OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock one when reading data from the memory.

Figure 80. OctaRAM read operation with reverse data ordering D1/D0

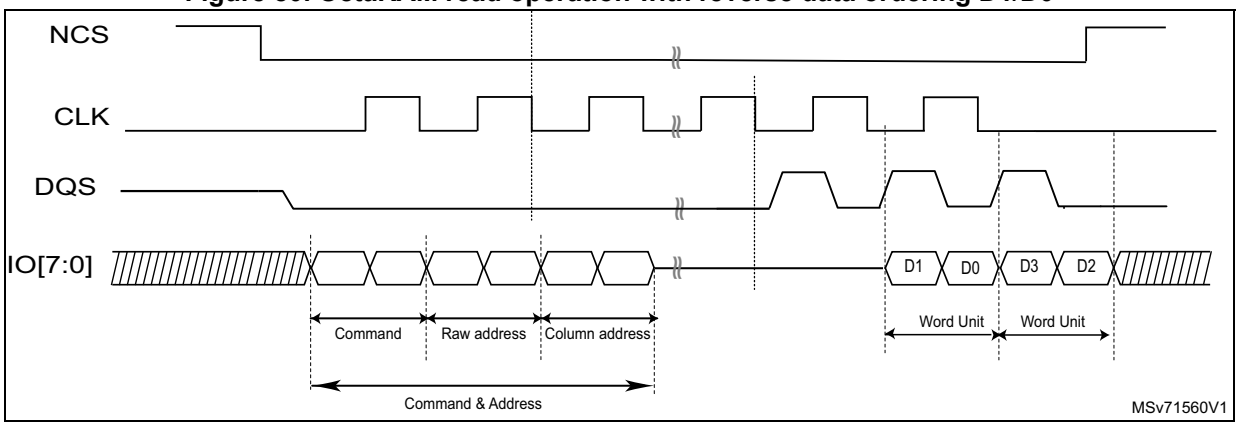


Table 129. OctaRAM command address bit assignment (based on 64 Mb(1) OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0(2)

1. Example of 64 Mb OctaRAM address assignment:
Row Address [RA12:RA0]: 8K. Column Address [CA9:CA0]: 1K. 64 Mb density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes must be used. The configuration to be set for this particular register space access is MTYP[2:0] = 101.
- Standard mode. It is the mode to use whenever the targeted memory is not corresponding to any other configurations described in this section. MTYP[2:0] = 010 for this standard mode.

19.4.15 XSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the frame timing in XSPI_TCR.
3. Specify the frame format in XSPI_CCR.
4. Specify the instruction in XSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
6. Specify the targeted address in XSPI_AR.
7. Enable the DMA channel if needed.

8. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

If neither the address register (XSPI_AR) nor the data register (XSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as XSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000),
the communication start is triggered by a write in the FIFO through XSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and data are retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in XSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in XSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in XSPI_PSMKR, and can be ORed or ANDed with the value stored in XSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The XSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in XSPI_DR.

When the XSPI is used in automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in XSPI_PSMKR.
2. Specify the comparison value in XSPI_PSMAR.
3. Specify the read period in XSPI_PIR.
4. Specify a number of data bytes to read in XSPI_DLR.
5. Specify the frame timing in XSPI_TCR.
6. Specify the frame format in XSPI_CCR.
7. Specify the instruction in XSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
9. Specify the optional targeted address in XSPI_AR.

If the address register (XSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as XSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in XSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

When the XSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in XSPI_TCR for read operation.
2. Specify the frame format in XSPI_CCR for read operation.
3. Specify the instruction in XSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR for read operation.
5. Specify the frame timing in XSPI_WTCR for write operation.
6. Specify the frame format in XSPI_WCCR for write operation.
7. Specify the instruction in XSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_WABR for write operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the XSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

XSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the XSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in XSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

XSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it is shifted precisely by one quarter of the SPI clock cycle, for all frequencies above freq_min, to compensate the data propagation delay in the “high-speed interface” when the product embeds one.

This shift is performed by an external delay block (DLYB) located outside the XSPI. This external delay block has to be calibrated (by enabling FRCK) before starting the first transfer, and recalibrated regularly. The control of this feature depends on the device implementation (see the delay block (DLYB) section in this document for more details about DLYB and calibration procedure, and the product datasheet for freq_min value).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in XSPI_DCR1.

19.4.16 XSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the targeted address in XSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through XSPI_DR.

Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the XSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit, or to abort by setting the ABORT bit.

19.4.17 XSPI error management

The following errors set the TEF flag in XSPI_SR and generates an interrupt if enabled (TEIE = 1 in XSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in XSPI_AR (according to the device size defined by DEVSIZE[4:0]).
- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.

In memory-mapped mode, the XSPI generates an AHB slave error in the following situations, setting the BERRF flag in XSPI_SR and generating an interrupt if enabled (BERRIE = 1 in XSPI_CR) (if the Bus error bit feature is available in the product implementation table, otherwise the Slave bus error response is generated at system level as transaction response with no dedicated bit into the controller status register):

- The memory-mapped mode is disabled and an AHB read or write request occurs.
- A read or write (including the address offset if ADOFFEN bit is set) address exceeds the size of the external memory.
- An abort is received while a read or write burst is ongoing when the abort condition is present at the moment of transfer phases preceding the data phase (In incremental or wrap burst type modes). To avoid the error, refer to *Section 19.4.18: XSPI BUSY and ABORT*.
- The XSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in XSPI_WCCR in octal DTR mode, in dual-memory configuration, in HyperBus mode or 16-bit mode.
- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means that the AHB bus transfer size (HSIZE) is different from 4 bytes (only for memory-mapped mode).
- Bit DMM is set in XSPI_CR (dual-memory configuration) and octal mode is set.

19.4.18 XSPI BUSY and ABORT

Once the XSPI starts an operation with the external device, BUSY is automatically set in XSPI_SR.

In indirect mode, BUSY is reset once the XSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in XSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in XSPI_DR, it is recommended to disable the DMA channel before aborting the SPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

19.4.19 XSPI reconfiguration or deactivation

Before any XSPI reconfiguration, the software must ensure that all the transactions are completed:

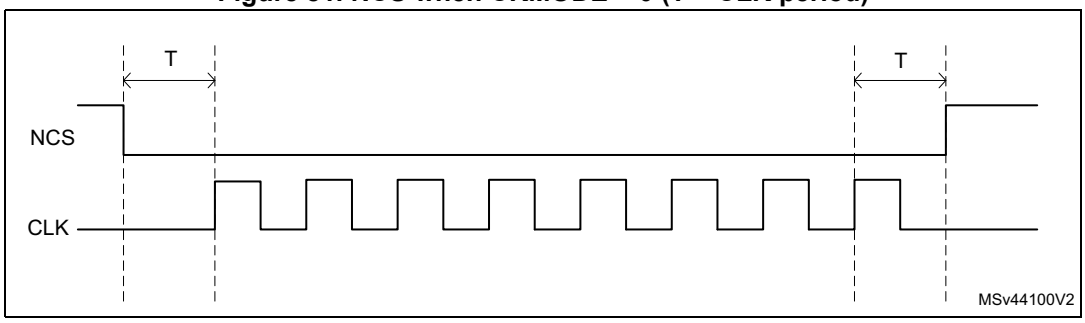
- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

19.4.20 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 81. NCS when CKMODE = 0 (T = CLK period)



Note: Clock mode 3 is not supported.

In dual-memory configuration, the NCS2 behaves exactly the same as the NCS1.
The quad-SPI/octal-SPI memory 2 can then use the NCS1, and the NCS2 pin can be used for other functions.

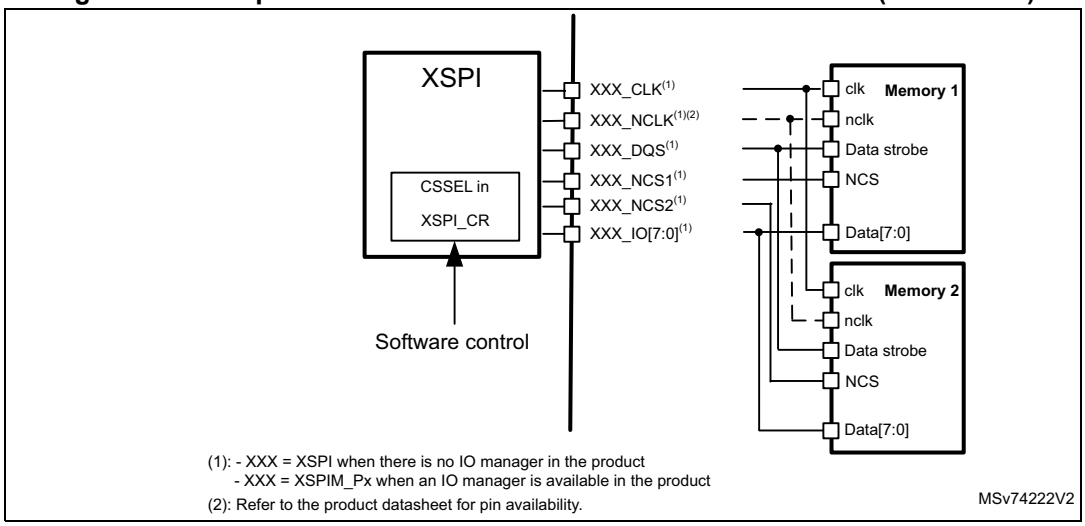
19.4.21 Software control of two external memories

One XSPI instance is able to control the generation of the two chip-select signals (NCS1 and NCS2). This allows two external memories to be connected to the same I/O port driven by the XSPI.

The XSPI generates one NCS signal and the user defines how the NCS is directed to NCS1 or NCS2, by setting the CSSEL bit in XSPI_CR. Intended operation is semistatic commutation, such as starting an application reading an external flash memory and then, for the rest of the application, use an external SRAM connected to the same I/O bus.

From a memory-map point of view, these two memories are located in the same address range.

Figure 82. Example of software control of two external memories (octal mode)



19.4.22 Hardware-controlled extended memory support

One XSPI instance is able to control autonomously the generation of the two chip-select signals (NCS1 and NCS2). This feature is activated by setting EXTENDMEM in XSPI_DCR1 and allows two external memories to be connected to the same I/O port driven by the XSPI.

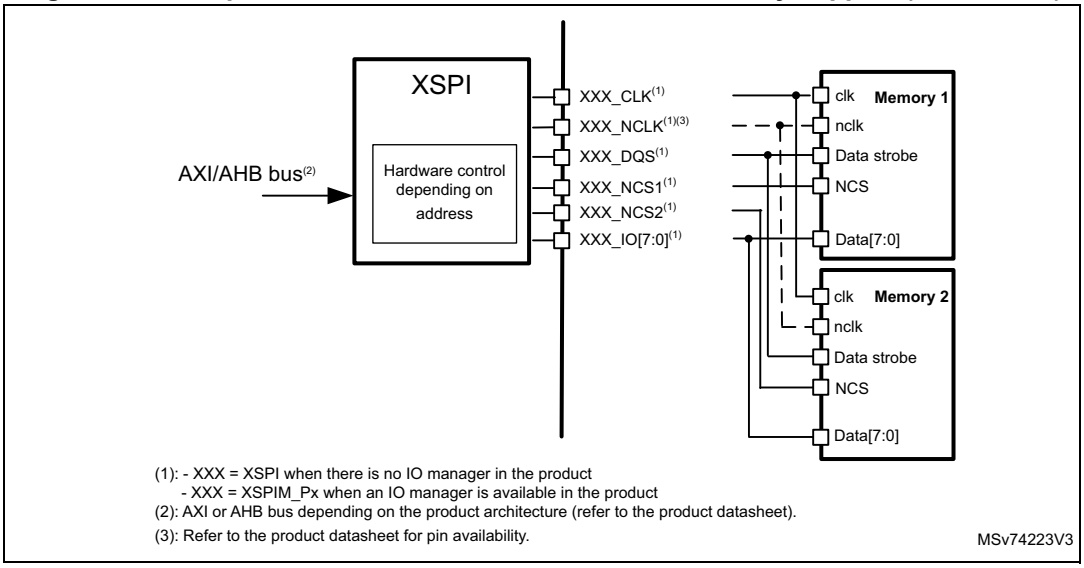
The aim is to extend by a factor two, the available external memory size. Both memories must be contiguous in the memory map. Both external memories must be of the same type and size. The size of one external memory must be set in DEVSIZE of XSPI_CR.

- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR in indirect mode is in the range [XSPI_base_address: XSPI_base_address + $2^{DEVSIZE}$ - 1], NCS1 is generated.
- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR in indirect mode is in the range [XSPI_base_address + $2^{DEVSIZE}$: XSPI_base_address + 2 × $2^{DEVSIZE}$ - 1], NCS2 is generated.

Note: If the access crosses the middle boundary of the external memory, CS changes automatically (no write into XSPI_AR in indirect mode, or new memory access in memory-mapped mode is needed).

The EXTENDMEM feature (set in bit 21 of XSPI_DCR1) and the Dual-memory mode (set in bit 6 of XSPI_CR) are exclusive: EXTENDMEM = 1 and DMM = 1 as this configuration is not supported.

Figure 83. Example of hardware-controller extended memory support (octal mode)



19.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 130. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				

Table 130. Address alignment cases (continued)

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Octal memory in DTR mode without WDM(6)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
HyperBus	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode.
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.
7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

19.6 XSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 131. XSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE

Table 131. XSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

19.7 XSPI registers

19.7.1 XSPI control register (XSPI_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MSEL	FMODE[1:0]		Res.	Res.	NOPREF	CSSEL	PMM	APMS	BERRIE	TOIE	SMIE	FTIE	TCIE	TEIE
	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	ADOFFEN	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **MSEL**: External memory select

This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).

0: External memory 1 selected (data exchanged over IO[3:0])

1: External memory 2 selected (data exchanged over IO[7:4])

This bit is ignored when DMM = 1 or when octal-SPI mode is selected.

Note: This bitfield can be modified only when BUSY = 0.

Bits 29:28 **FMODE[1:0]**: Functional mode

This bitfield defines the XSPI functional mode of operation.

00: Indirect-write mode

01: Indirect-read mode

10: Automatic status-polling mode (relevant in regular-command protocol only)

11: Memory-mapped mode

If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.

Note: This bitfield can be modified only when BUSY = 0.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **NOPREF**: No prefetch data

This bit disables the automatic prefetch in the external memory.

0: Automatic prefetch enabled

1: Automatic prefetch disabled

Bit 24 **CSSEL**: Chip-select selection
This bit indicates if it must copy NCS to NCS1 or NCS2.
0: NCS1 active
1: NCS2 active
Note: This bit can be modified only when BUSY = 0.

Bit 23 **PMM**: Polling match mode
This bit indicates which method must be used to determine a match during the automatic status-polling mode.
0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.
Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop
This bit determines if the automatic status-polling mode is stopped after a match.
0: Automatic status-polling mode is stopped only by abort or by disabling the XSPI.
1: Automatic status-polling mode stops as soon as there is a match.
Note: This bit can be modified only when BUSY = 0.

Bit 21 **BERRIE**: Bus error interrupt enable
This bit enables the bus error interrupt (if Bus error bit feature is available in the product implementation table).
0: Interrupt disabled
1: Interrupt enabled

Bit 20 **TOIE**: Timeout interrupt enable
This bit enables the timeout interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **SMIE**: Status-match interrupt enable
This bit enables the status-match interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level

This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in XSPI_SR, to be set.

00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.

00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.

...

11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in indirect-read mode.

Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DMM**: Dual-memory configuration

This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity

0: Dual-memory configuration disabled

1: Dual-memory configuration enabled

Note: This bit can be modified only when BUSY = 0.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ADOFFEN**: Address offset enable

An offset may be applied on the memory address. The offset value is set into ADOFF[4:0].

0: Address offset disabled

1: Address offset enabled

Note: This bit can be modified only when BUSY = 0. This bit is present only if the address offset feature is available in the product implementation table.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via XSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the XSPI.

0: XSPI disabled

1: XSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

19.7.2 XSPI device configuration register 1 (XSPI_DCR1)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADOFF[4:0]					MTYP[2:0]			Res.	Res.	EXTENDMEM	DEVSIZE[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
		rw	rw	rw	rw	rw	rw					rw		rw	rw

Bits 31:27 **ADOFF[4:0]**: Address offset

This bitfield defines the memory address offset value added to the current address requested to be sent by the controller to hit the external memory using the following formula:
Offset value = $2^{[ADOFF+8]}$. (modulo 256 bytes)
Maximum allowed value is 19 since the memory-mapped mode is limited to 256 Mbytes address space. For greater value, the address offset will not be applied to the address sent by the controller.
This bitfield is relevant only if ADOFFEN is set and if FMODE[1:0] = 11 (memory-mapped mode) and if the address offset feature is available in the product implementation table.

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.

Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.

010: Standard mode

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).

100: HyperBus memory mode, the protocol follows the HyperBus specification.

101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect-read/write modes must be used.

Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **EXTENDMEM**: Extended memory support

This bit is only relevant when one XSPI drives two same size external memories located in contiguous places in the memory map. In this case, DEVSIZE[4:0] represents the size of each external memory. When this bit is set, the XSPI automatically controls which of NCS1 or NCS2 is active.

0: NCS1 and NCS2 values depend from CSSEL bit in XSPI_CR (software controlled).

1: NCS1 and NCS2 values depend from the address of transfer (hardware controlled).

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:

Number of bytes in device = $2^{[DEVSIZE+1]}$.

DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

In regular-command protocol, if DMM = 1 or EXTENDMEM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.

0x0: NCS stays high for at least 1 cycle between external device commands.

0x1: NCS stays high for at least 2 cycles between external device commands.

...

0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the peripheral).

1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.

0: CLK is not free running.

1: CLK is free running (always provided).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set. Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0

This bit is read-only and is read as zero.

19.7.3 XSPI device configuration register 2 (XSPI_DCR2)

Address offset: 0x00C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This bitfield indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in the XSPI_WPIR register.

000: Wrapped reads are not supported by the memory.

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

Other: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as XSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the XSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

19.7.4 XSPI device configuration register 3 (XSPI_DCR3)

Address offset: 0x010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This bitfield enables the transaction boundary feature. When active, a minimum value of 3 is recommended. The NCS is released on each boundary of $2^{CSBOUND}$ bytes.

0: NCS boundary disabled

Other: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0 Reserved, must be kept at reset value.

19.7.5 XSPI device configuration register 4 (XSPI_DCR4)

Address offset: 0x014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFRESH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual- or quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled

Other: Maximum communication length is set to REFRESH + 1 clock cycles.

Note: REFRESH count is based on the memory clock period: if XSPI_DCR2 PRESCALER bitfield is changed, the REFRESH field must be updated accordingly.

19.7.6 XSPI status register (XSPI_SR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	BERRF	BUSY	TOF	SMF	FTF	TCF	TEF
		r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **BERRF**: Bus error flag

This bit is set when a bus error occurs. It is cleared by writing 1 to CBERRF. This bit has a meaning if Bus error bit feature is available in the product implementation table.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (XSPI_PSMAR).

It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

In automatic status-polling mode, this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

19.7.7 XSPI flag clear register (XSPI_FCR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBERRF	Res.	CTOF	CSMF	Res.	CTCF	CTEF
									w		w	w		w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CBERRF**: Clear bus error flag

Writing 1 clears the BERRF flag in the XSPI_SR register (if Bus error bit feature is available in the product implementation table).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag

Writing 1 clears the TOF flag in the XSPI_SR register.

Bit 3 **CSMF**: Clear status match flag

Writing 1 clears the SMF flag in the XSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag

Writing 1 clears the TCF flag in the XSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag

Writing 1 clears the TEF flag in the XSPI_SR register.

19.7.8 XSPI data length register (XSPI_DLR)

Address offset: 0x040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.

All 1's in indirect mode means undefined length, where XSPI continues until the end of the memory, as defined by DEVSIZE.

0x0000_0000: 1 byte is to be transferred.

0x0000_0001: 2 bytes are to be transferred.

0x0000_0002: 3 bytes are to be transferred.

0x0000_0003: 4 bytes are to be transferred.

...

0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.

0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.

0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.

DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.

This bitfield has no effect in memory-mapped mode.

19.7.9 XSPI address register (XSPI_AR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In dual-memory configuration, AR[0] is forced to 0.

Caution: Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value. So the software needs to multiply the address by two when accessing the memory registers.

19.7.10 XSPI data register (XSPI_DR)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.

Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

19.7.11 XSPI polling status mask register (XSPI_PSMKR)

Address offset: 0x080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.

19.7.12 XSPI polling status match register (XSPI_PSMAR)

Address offset: 0x088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MATCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MATCH[31:0]**: Status match

Value to be compared with the masked status register to get a match

19.7.13 XSPI polling interval register (XSPI_PIR)

Address offset: 0x090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval

Number of CLK cycles between a read during the automatic status-polling phases

19.7.14 XSPI communication configuration register (XSPI_CCR)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate- byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Other: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

19.7.15 XSPI timing configuration register (XSPI_TCR)

Address offset: 0x108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.

This bit allows the data to be sampled later in order to consider the external signal delays.

0: No shift

1: 1/2 cycle shift

The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

0: No delay hold

1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This bitfield defines the duration of the dummy phase according to the memory latency. In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

19.7.16 XSPI instruction register (XSPI_IR)

Address offset: 0x110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

19.7.17 XSPI alternate bytes register (XSPI_ABR)

Address offset: 0x120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address.

19.7.18 XSPI low-power timeout register (XSPI_LPTR)

Address offset: 0x130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period
After each access in memory-mapped mode, the XSPI prefetches the subsequent bytes and hold them in the FIFO.
This bitfield indicates how many CLK cycles the XSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

19.7.19 XSPI wrap communication configuration register (XSPI_WPCCR)

Address offset: 0x140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: no alternate bytes
001: alternate bytes on a single line
010: alternate bytes on two lines
011: alternate bytes on four lines
100: alternate bytes on eight lines
Other: reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

19.7.20 XSPI wrap timing configuration register (XSPI_WPTCR)

Address offset: 0x148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.

This bit allows the data to be sampled later in order to consider the external signal delays.

0: No shift

1: 1/2 cycle shift

The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.

0: No quarter cycle delay

1: 1/4 cycle delay inserted

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This bitfield defines the duration of the dummy phase according to the memory latency.

In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

19.7.21 XSPI wrap instruction register (XSPI_WPIR)

Address offset: 0x150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

19.7.22 XSPI wrap alternate bytes register (XSPI_WPABR)

Address offset: 0x160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

19.7.23 XSPI write communication configuration register (XSPI_WCCR)

Address offset: 0x180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Other: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

19.7.24 XSPI write timing configuration register (XSPI_WTCR)

Address offset: 0x188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

19.7.25 XSPI write instruction register (XSPI_WIR)

Address offset: 0x190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction

Instruction to be sent to the external SPI device

19.7.26 XSPI write alternate bytes register (XSPI_WABR)

Address offset: 0x1A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address

19.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR)

Address offset: 0x200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TRWR[7:0]:** Read-write minimum recovery time

Device read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7:0]**: Access time

Device access time according to the memory latency, expressed in number of communication clock cycles

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.
0: Latency on write accesses
1: No latency on write accesses

Bit 0 **LM:** Latency mode

This bit selects the latency mode.
0: Variable initial latency
1: Fixed latency

Note: This bit must be set when using the dual-octal HyperBus configuration.

19.7.28 XSPI register map

Table 132. XSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	XSPI_CR	Res.	MSEL	FMDOE[1:0]		Res.	Res.	NOPREF	CSSEL	PMM	APMS	BERRIE	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	ADOFFEN	TCEN	DMAEN	ABORT	EN
	Reset value		0	0	0			0	0	0	0	0	0	0	0	0	0				0	0	0	0	0		0		0	0	0	0	0
0x004	Reserved																																

Table 132. XSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	XSPI_DCR1	ADOFF[4:0]					MTYP [2:0]			Res.	Res.	EXTENDMEM	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0					0		0	0
0x00C	XSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x010	XSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x014	XSPI_DCR4	REFRESH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved																																
0x020	XSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	BERRF	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0		0	0	0	0	0	0	0
0x024	XSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBERRF	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																										0		0	0		0	0
0x028-0x03C	Reserved																																
0x040	XSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Reserved																															
0x048	XSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	XSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054-0x07C	Reserved																																
0x080	XSPI_PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	Reserved																																
0x088	XSPI_PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	Reserved																																

Table 132. XSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x090	XSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094-0x0FC	Reserved																																
0x100	XSPI_CCR	Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]			Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x104	Reserved																																
0x108	XSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																							0	0	0	0	0
0x10C	Reserved																																
0x110	XSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114-0x11C	Reserved																																
0x120	XSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124-0x12C	Reserved																																
0x130	XSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134-0x13C	Reserved	Reserved																															
0x140	XSPI_WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]			Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x144	Reserved	Reserved																															
0x148	XSPI_WPTCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																							0	0	0	0	0
0x14C	Reserved	Reserved																															
0x150	XSPI_WPIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154-0x15C	Reserved	Reserved																															
0x160	XSPI_WPABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x164-0x17C	Reserved																																

Table 132. XSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x180	XSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x184	Reserved																																
0x188	XSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x18C	Reserved																																
0x190	XSPI_WIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94-0x019C	Reserved																																
0x1A0	XSPI_WABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A4-0x1FC	Reserved																																
0x200	XSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

20 Delay block (DLYB)

This section applies only to STM32C59xx/C5Axx devices.

20.1 DLYB introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an XSPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

20.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

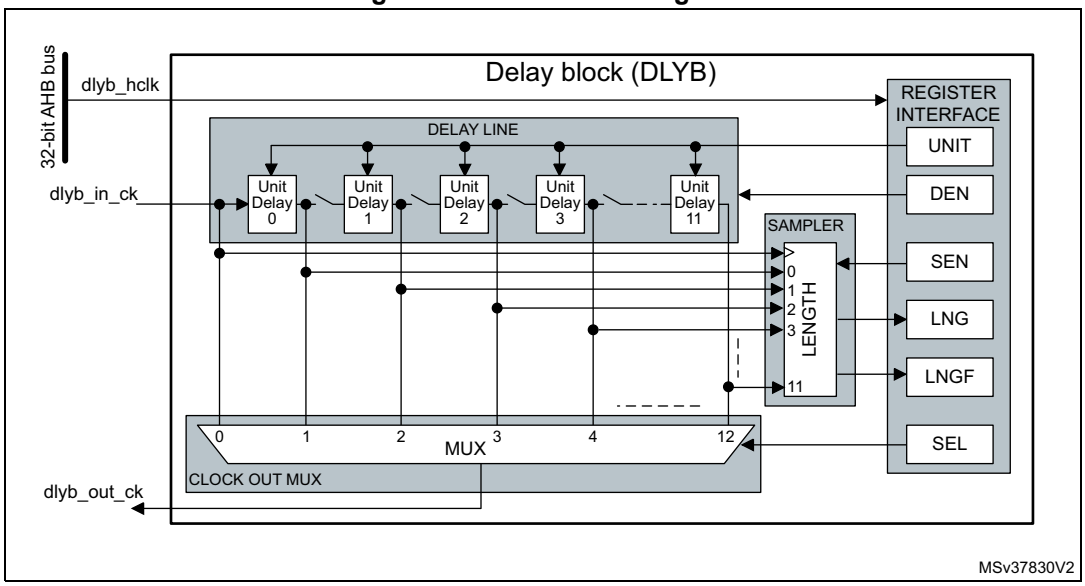
20.3 DLYB functional description

20.3.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 84. DLYB block diagram



20.3.2 DLYB pins and internal signals

Table 133 lists the DLYB internal signals.

Table 133. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

20.3.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 134. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don't care	Don't care	Don't care	Don't care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don't care	Don't care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

20.3.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

20.3.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

XSPI use case:

The delay block is used in conjunction with XSPI interface to allow shifting the input data sampling signal. This sampling signal can be the feedback clock or the data strobe (DQS) signal, which is delivered by certain type of devices. Note that in case DQS is used, the calibration procedure must be performed beforehand with a free running clock, as DQS is a discontinuous signal.

In case of SDR (single data rate) mode the user must typically shift the sampling signal by half period, so that the sampling edges are positioned in the middle of the valid data phase.

In case of DDR (dual data rate) mode, for which data are transitioning at start and middle of period, typical value must be close to N/4, once the calibration is completed.

In case of high frequencies and tight timing constraints, the delay setting granularity (10) might be too coarse. Since in most cases it is not necessary to have a possible delay value covering the whole sampling clock period, the "Unit" value can be overridden by application in order to improve the accuracy of the sampling edge position (example: providing a twice as small “Unit” gives twice better timing accuracy. The counterpart being that the maximum possible delay is divided by 2).

20.4 DLYB registers

All registers can be accessed in word, half-word and byte access.

20.4.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

20.4.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LNGF	Res.	Res.	Res.	LNG[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LNGF:** Length valid flag

This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.

0: Length value in LNG is not valid.

1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock. The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.

These bits can only be written when SEN = 1.

Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Phase for the output clock.

These bits can only be written when SEN = 1.

Output clock phase = input clock + SEL[3:0] x unit delay

20.4.3 DLYB register map

Table 135. DLYB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DLYB_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	DLYB_CFGR	LNGF	Res.	Res.	Res.	LNG[11:0]												Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

21 Analog-to-digital converters (ADC)

21.1 ADC introduction

This section describes the implementation of up to 3 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).
- ADC3 is controlled independently.

Each ADC consists of one 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 14 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan, or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned (default configuration) 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

21.2 ADC main features

- High-performance features
  - Up to 3 ADCs, out of which two of them can operate in dual mode:
    ADC1 is connected to 12 external channels + 2 internal channels
    ADC2 is connected to 14 external channels
    ADC3 is connected to 14 external channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - AHB slave bus interface to allow fast data handling
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Fixed latency for a trigger to start of sampling
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Four dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance

  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (autodelayed mode)
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024
  - Programmable data right and left shifts
- Data preconditioning
  - Gain compensation
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC): up to 14 GPIO pads
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for the internal temperature sensor (V_{SENSE})
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timer events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 85 shows the block diagram of one ADC.

21.3 ADC implementation

Table 136. ADC features

ADC modes/features	ADC1, ADC2(1)	ADC3(2)
Resolution	12 bits	
Maximum sampling speed	2 Msps (12-bit resolution)	
Dual mode operation	X	-
Offset calibration	X	
Single-end input	X	
Differential input	-	
Injected channel conversion	X	
Oversampling	up to x1024	
Data register	32 bits	
DMA support	X	
Parallel data output to MDF and ADF	-	
Offset compensation	X	
Gain compensation	X	
Number of analog watchdog	3	

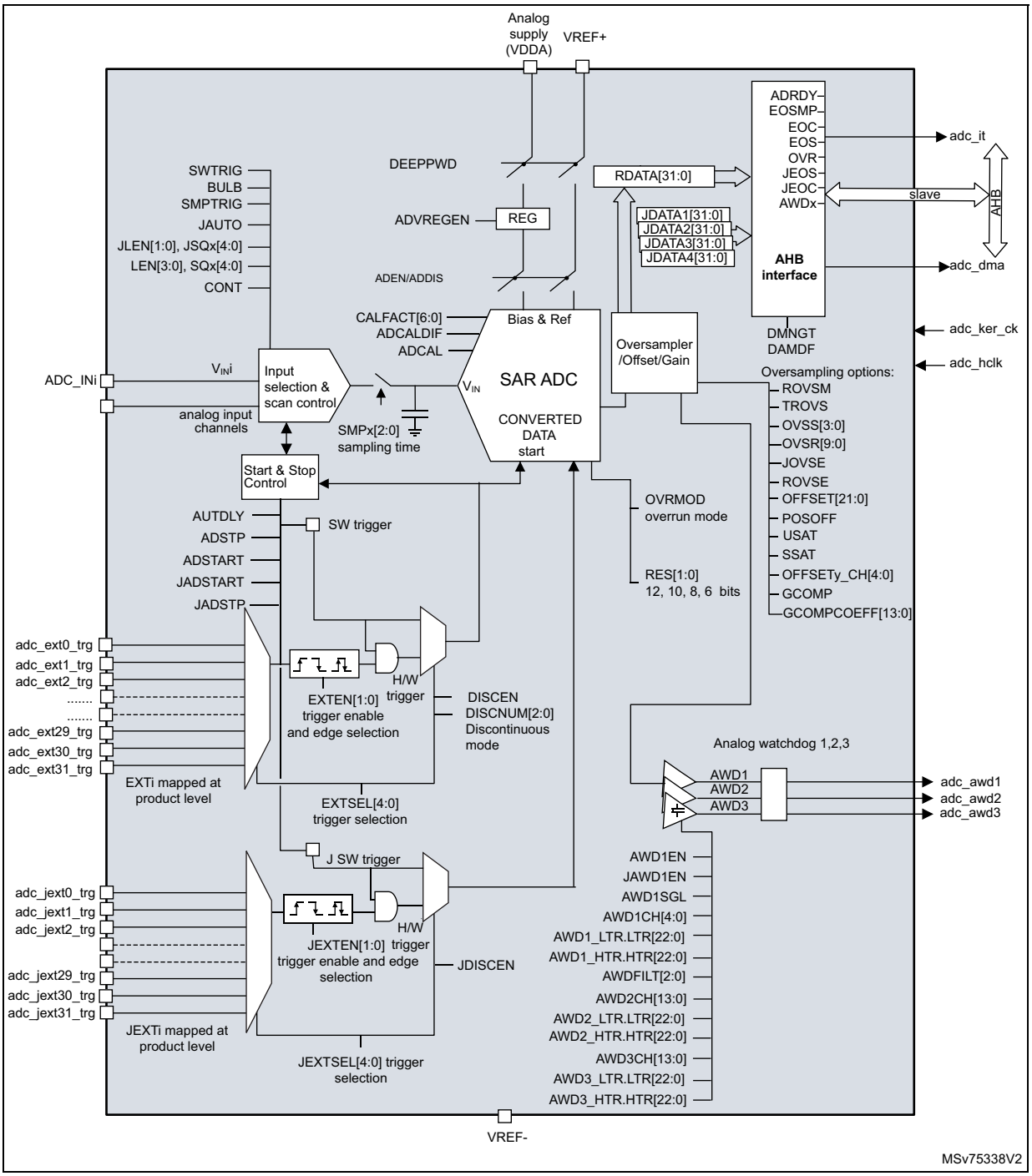
1. Only available on STM32C55x/562/59x/5A3 devices.
2. Only available on STM32C59x/5A3 devices.

21.4 ADC functional description

21.4.1 ADC block diagram

Figure 85 shows the ADC block diagram and *Table 137* gives the ADC pin description.

Figure 85. ADC block diagram



21.4.2 ADC pins and internal signals

Table 137. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC.
ADCx_INi	External analog input signals	14 external analog input channels (refer to *Section 21.4.4: ADC connectivity* for details)

Table 138. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{IN}i$	Analog input channels	Internal analog input channels connected either to ADCx_IN*i* external channels or to internal channels
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.
adc_awdy	Output	Internal analog watchdog output signal connected to on-chip timers. (y = Analog watchdog number 1,2,3)
adc_ker_ck	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request

Table 139. ADC1/2/3 interconnection

Signal name	Source/destination
ADC1 V_{IN}[13]	V_{REFINT} (output voltage from internal reference voltage or dac_int1)
ADC1 V_{IN}[12]	V_{SENSE} (internal temperature sensor output voltage)
adc_ext_trg0	exti11
adc_ext_trg1	tim1_oc1
adc_ext_trg2	tim1_oc2
adc_ext_trg3	tim1_oc3
adc_ext_trg4	tim1_trgo
adc_ext_trg5	tim1_trgo2

Table 139. ADC1/2/3 interconnection (continued)

Signal name	Source/destination
adc_ext_trg6	tim2_oc2
adc_ext_trg7	tim2_trgo
adc_ext_trg8	Reserved
adc_ext_trg9	Reserved
adc_ext_trg10	tim5_oc4(1)
adc_ext_trg11	tim5_trgo(1)
adc_ext_trg12	tim6_trgo
adc_ext_trg13	tim8_trgo
adc_ext_trg14	tim8_trgo2
adc_ext_trg15	tim15_trgo
adc_ext_trg16	lptim1_ch1
adc_ext_trg17	Reserved
adc_ext_trg18	Reserved
adc_ext_trg19	Reserved
adc_ext_trg20	Reserved
adc_ext_trg21	Reserved
adc_ext_trg22	Reserved
adc_ext_trg23	Reserved
adc_ext_trg24	Reserved
adc_ext_trg25	Reserved
adc_ext_trg26	Reserved
adc_ext_trg27	Reserved
adc_ext_trg28	Reserved
adc_ext_trg29	Reserved
adc_ext_trg30	Reserved
adc_ext_trg31	Reserved
adc_jext_trg0	exti15
adc_jext_trg1	tim1_oc4
adc_jext_trg2	tim1_trgo
adc_jext_trg3	tim1_trgo2
adc_jext_trg4	tim2_oc1
adc_jext_trg5	tim2_trgo
adc_jext_trg6	Reserved
adc_jext_trg7	Reserved
adc_jext_trg8	Reserved

Table 139. ADC1/2/3 interconnection (continued)

Signal name	Source/destination
adc_jext_trg9	Reserved
adc_jext_trg10	tim5_oc1(1)
adc_jext_trg11	tim5_oc2(1)
adc_jext_trg12	tim5_oc3(1)
adc_jext_trg13	tim5_trgo(1)
adc_jext_trg14	tim7_trgo
adc_jext_trg15	tim8_oc4
adc_jext_trg16	tim8_trgo
adc_jext_trg17	tim8_trgo2
adc_jext_trg18	tim12_trgo
adc_jext_trg19	tim15_trgo
adc_jext_trg20	lptim1_ch1
adc_jext_trg21	Reserved
adc_jext_trg22	Reserved
adc_jext_trg23	Reserved
adc_jext_trg24	Reserved
adc_jext_trg25	Reserved
adc_jext_trg26	Reserved
adc_jext_trg27	Reserved
adc_jext_trg28	Reserved
adc_jext_trg29	Reserved
adc_jext_trg30	Reserved
adc_jext_trg31	Reserved

1. Reserved on STM32C53x/542 devices.

21.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC kernel clock is independent from the AHB bus clock that is used to access ADC registers.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 86: ADC clock scheme*). This selection is done in the RCC (refer to section *Reset and clock control (RCC)* for more information):

1. The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous from the AHB clock.
2. The ADC clock can be derived from AHB clock (selected in RCC).

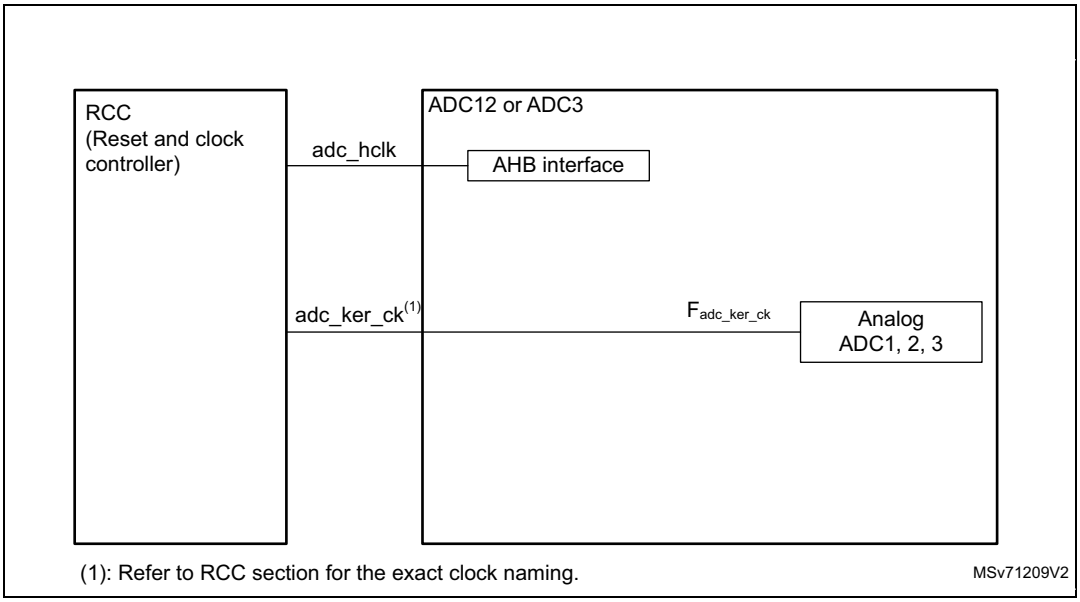
Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected.

Option **2** corresponds to a pseudosynchronous clock. This can be useful when the ADC is triggered by a timer and the application requires that the ADC is accurately triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains). This accurate trigger is supported only by trgo or trgo2 timer triggers.

For further details on the synchronization between ADC and timers, refer to section *ADC synchronization* in each timer section, and to section *Peripheral clock distribution* in the RCC section.

The clock is configured through the RCC and must be compliant with the operating frequency specified in the device datasheet.

Figure 86. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

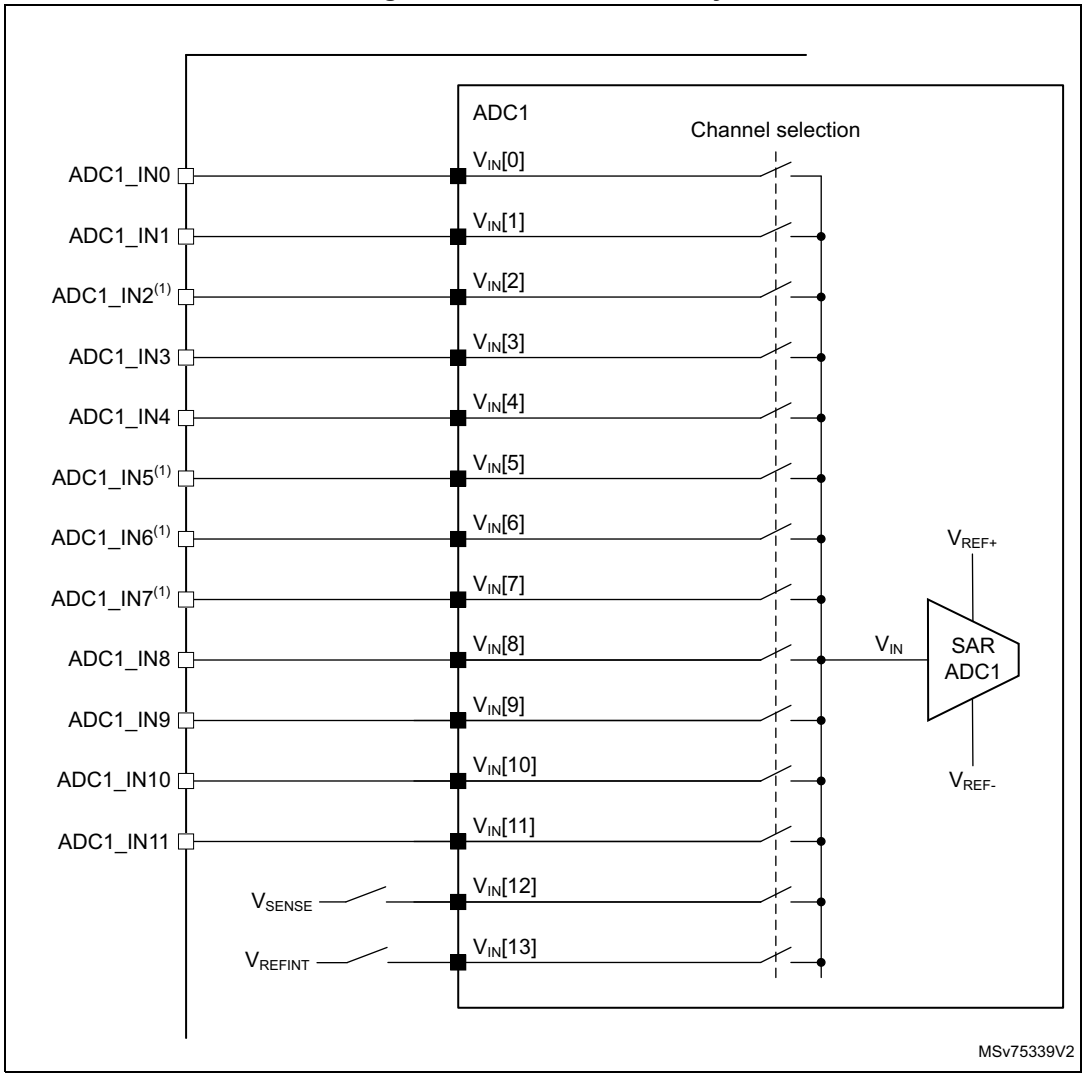
There are no constraints to respect for the ratio between the ADC clock and the AHB clock. However, the ratio must be carefully chosen to avoid any overrun especially if the clock AHB is much slower than the ADC clock.

When adc_hclk operates at a higher frequency than adc_ker_ck, writing to the AHB register does not immediately update the ADC configuration due to clock domain crossing delays. To guarantee that the configuration is properly applied within the adc_ker_ck domain, it is necessary to wait for four adc_ker_ck clock cycles after the AHB register update.

21.4.4 ADC connectivity

ADC inputs are connected to external channels as well as internal sources as described below.

Figure 87. ADC1 connectivity



1. On the STM32C53x/542 devices, these external channels can be connected to two GPIOs selected through the SBS_PMCR register.

Figure 88. ADC2 connectivity

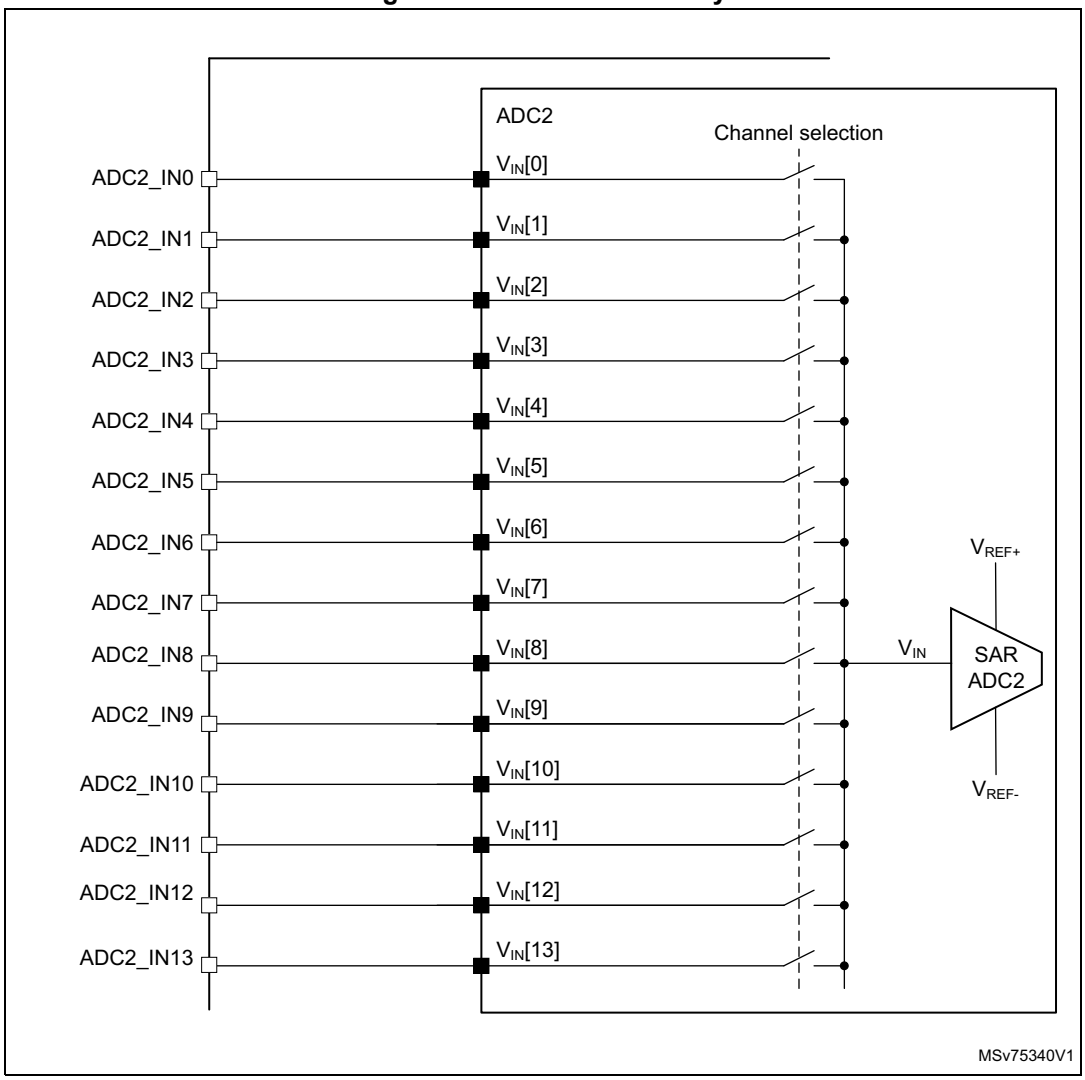
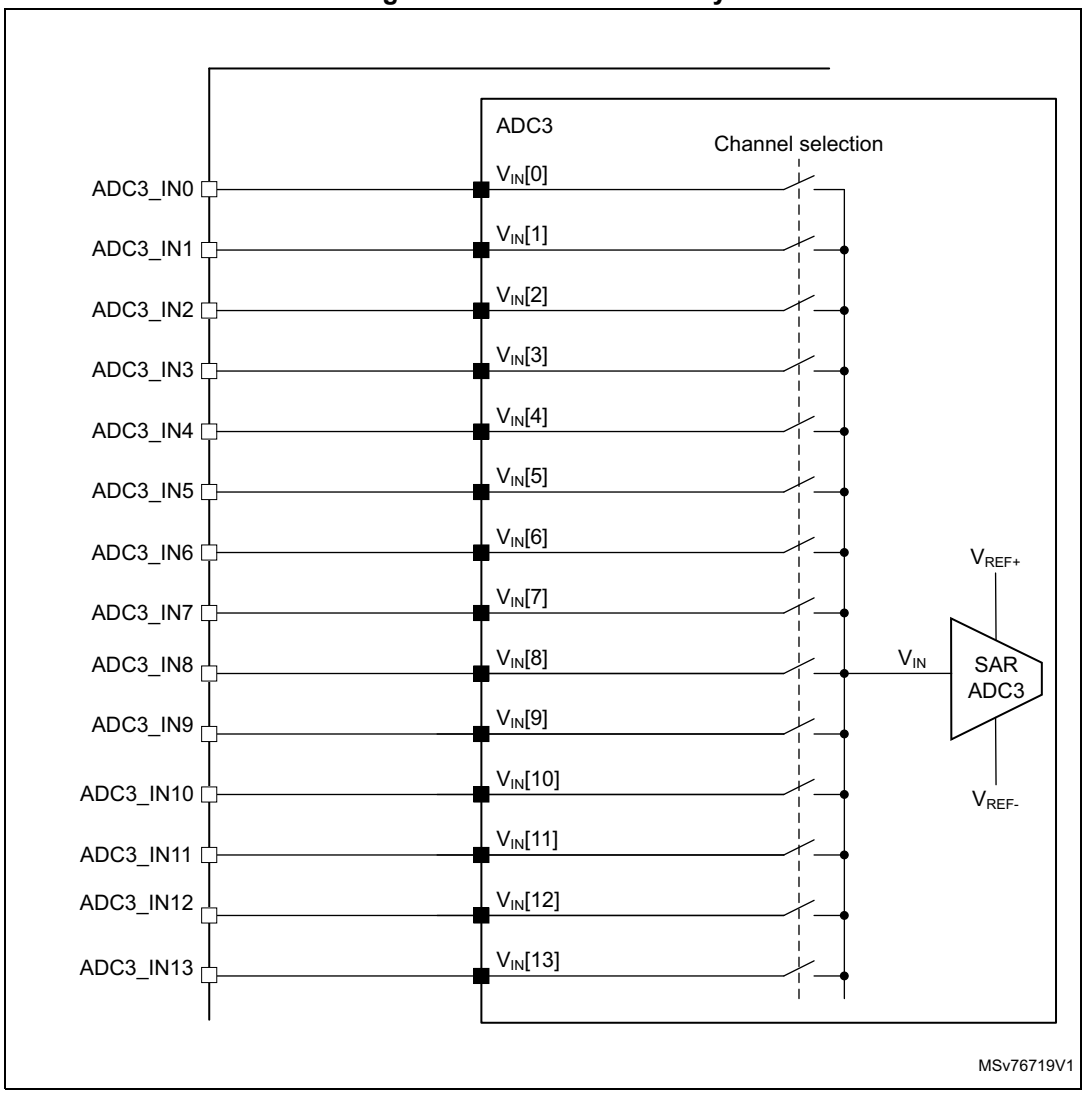


Figure 89. ADC3 connectivity



21.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and it can generate AHB error in case of wrong HSIZE.

21.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in deep-power-down mode where its supply voltage is internally switched off to reduce the leakage currents (the reset state of the DEEPPWD bit is 1 in the ADC_CR register).

To activate the ADC, first exit the Deep-power-down mode by clearing the DEEPPWD bit.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the ADVREGEN bit of the ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software. For the startup time of the ADC voltage regulator, refer to the device datasheet for $T_{ADCVREG_STUP}$ parameter.

When ADC operations are complete, the ADC can be disabled by clearing the ADEN bit of the ADC_CR register. Power can then be saved by disabling the ADC voltage regulator (by clearing ADVREGEN).

Additional power can be saved by entering ADC Deep-power-down mode again (by setting DEEPPWD of ADC_CR register). This is particularly interesting before entering Stop mode.

Note: Setting DEEPPWD automatically disables the ADC voltage regulator and the ADVREGEN bit is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN cleared), the internal analog calibration is kept.

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or reapply the calibration factor, which was previously saved (refer to *Section Calibration*).

21.4.7 Calibration (ADCAL, ADC_CALFACT)

Each ADC provides an automatic calibration procedure that drives all the calibration sequences including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor, which is 7-bit wide and is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration is initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). The ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT[6:0] of the ADC_CALFACT register.

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before reenabling the ADC.

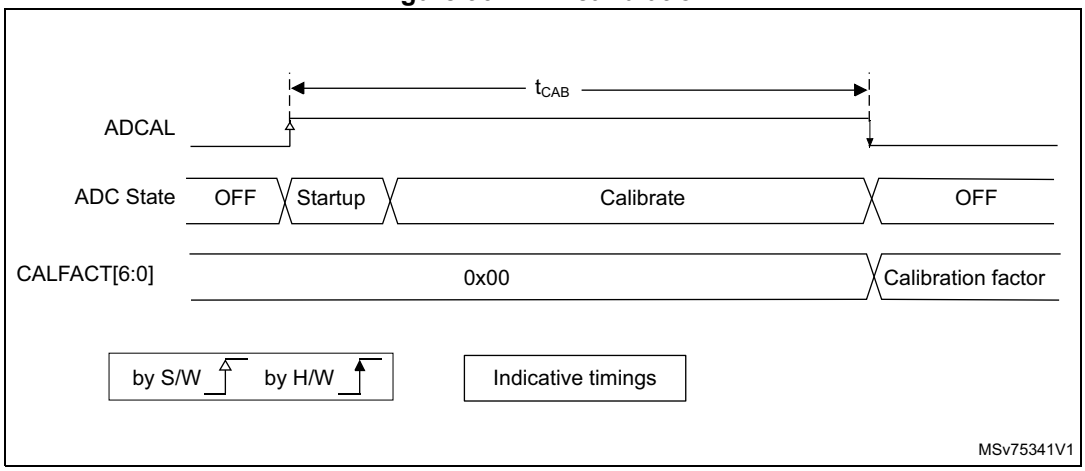
The internal analog calibration is lost each time the power of the ADC is removed (for example, when the product enters Standby). In this case, to avoid spending time recalibrating the ADC, it is possible to rewrite the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when the V_{REF+} voltage changed more than 10% (see *Section 21.4.31: Monitoring the internal voltage reference*).

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Set ADCAL.
4. Wait until ADCAL = 0.
5. The calibration factor can be read from the ADC_CALFACT register.

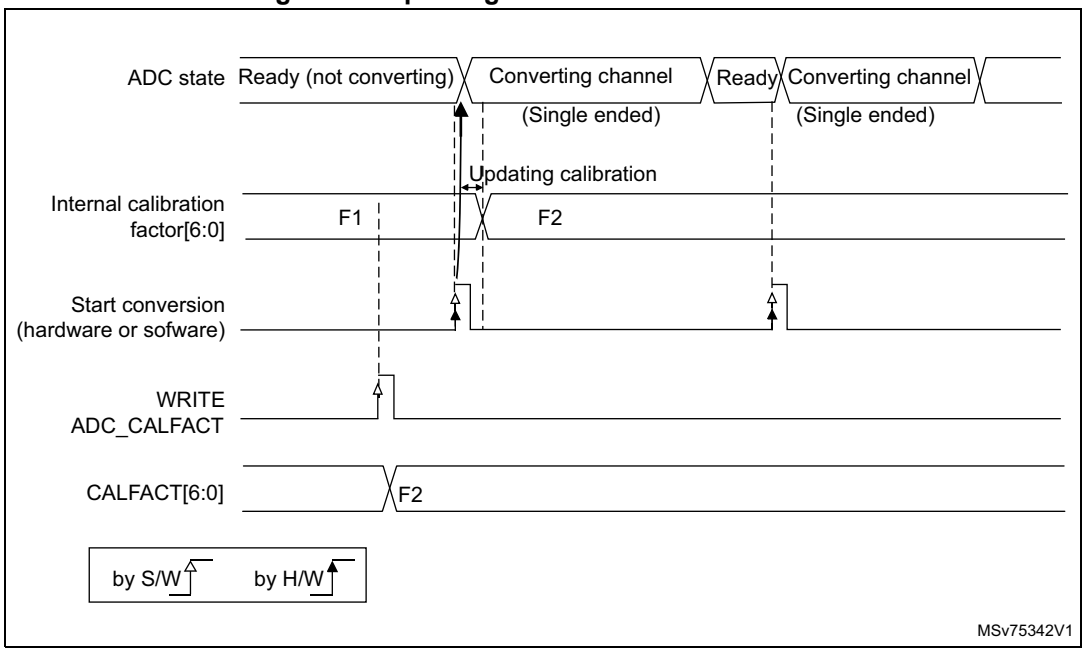
Figure 90. ADC calibration



Software procedure to reinject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT.

Figure 91. Updating the ADC calibration factor



21.4.8 ADC on-off control (ADEN, ADDIS, ADRDY)

First, follow the procedure described in *Section 21.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD is cleared and ADVREGEN is set, the ADC can be enabled. It requires a t_{STAB} stabilization time before starting converting accurately (see *Figure 92*).

Two control bits enable or disable the ADC:

- Set ADEN to enable the ADC. The ADRDY flag is set once the ADC is ready for operation.
- Set ADDIS to disable the ADC. ADEN and ADDIS are automatically deasserted by hardware as soon as the analog ADC is effectively disabled.

Regular conversions can then start either by setting ADSTART (refer to *Section 21.4.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs if regular triggers are enabled.

Injected conversions start by setting JADSTART or when an external injected trigger event occurs, if injected triggers are enabled.

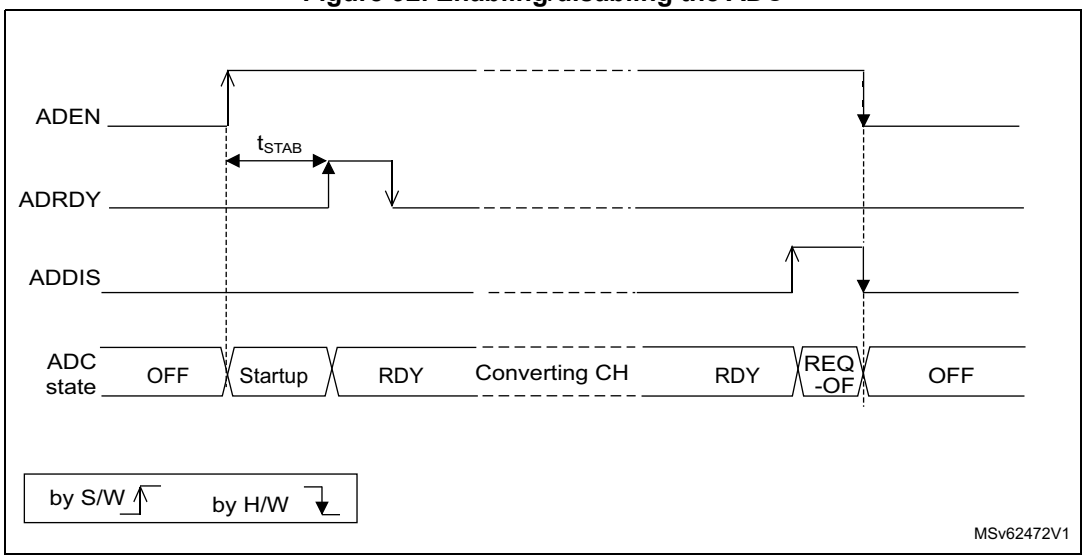
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by programming it to 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done by using the associated interrupt (ADRDYIE must be set).
4. Clear the ADRDY bit in the ADC_ISR register by programming it to 1 (optional).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop all regular and injected ongoing conversions by setting ADSTP and JADSTP. Then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait for ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 92. Enabling/disabling the ADC



21.4.9 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC section) and configure the ADEN bit in the ADC_CR register, only if the ADC is disabled (ADEN must be cleared).

The software can write the ADSTART, JADSTART and ADDIS control bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

The following constraints apply to all the other control bits of the ADC_CFGRx, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if no regular conversions are ongoing (ADSTART must be cleared).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if no injected conversions are ongoing (JADSTART must be cleared).
- ADC_TRy register thresholds can be modified when an analog-to-digital conversion is ongoing (refer to *Section 21.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)* for details).
- To modify the all above control bits when the software trigger function is used, an ADSTP or JADSTP must be issued to make sure all activities are stopped, even if ADSTART and JADSTART are cleared.

The software is allowed to write the ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and if there is no pending request to disable the ADC.

Note: Not all forbidden write accesses to ADC control bits are protected by hardware. In some cases, the ADC state may become unknown. To recover from this situation, the ADC must be disabled (ADEN = 0 as well as all the bits of the ADC_CR register).

21.4.10 Channel selection (ADC_SQRy, ADC_JSQR)

The ADC features up to 14 multiplexed channels per ADC, out of which:

- Up to 14 analog inputs coming from GPIO pads (ADCx_IN[i]). Depending on the device, not all of them are available on GPIO pads.
- The ADC is connected to up to 2 internal analog inputs:
  - internal reference voltage (V_{REFINT})
  - Internal temperature sensor (V_{SENSE})

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming VREFEN, TSEN in the ADCC_CCR registers.

Refer to *ADC interconnection* tables in *Section 21.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to ADC pins.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_IN3, ADCx_IN8, ADCx_IN2, ADCx_IN2, ADCx_IN0, ADCx_IN2, and ADCx_IN2.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the LEN[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to four conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the JLEN[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while a regular conversion is ongoing. ADC regular conversions must consequently be stopped by setting ADSTP (refer to *Section 21.4.18: Stopping an ongoing conversion (ADSTP, JADSTP)*).

21.4.11 Channel preselection register (ADC_PCSEL)

The PCSEL bit of the ADC_PCSEL register controls the analog switch integrated in the I/O.

For each channel selected through SQRx or JSQRx bits, the corresponding PCSEL bit must be configured in advance in the ADC_PCSEL register. The ADC input multiplexer selects the ADC input according to SQRx and JSQRx configuration with very high speed. The analog switch integrated in the I/O cannot react as fast as the ADC multiplexer. To avoid the delay due to the analog switch control on the I/O, it is necessary to preselect the input channels that are selected through the SQRx and JSQRx. The selection is based on the V_{IN} of each ADC input. For example, if the ADC converts ADCx_IN1, PCSEL1 bit must also be set in ADC_PCSEL register.

Note: Configuring the PCSEL bit is not necessary for the internal channels (such as V_{REFINT}).

21.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be sufficient for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time, which is programmable through the SMP[2:0] bits of the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 3 ADC clock cycles
- SMP = 001: 5 ADC clock cycles
- SMP = 010: 8 ADC clock cycles
- SMP = 011: 13 ADC clock cycles
- SMP = 100: 25 ADC clock cycles
- SMP = 101: 48 ADC clock cycles
- SMP = 110: 139 ADC clock cycles
- SMP = 111: 289 ADC clock cycles

The total conversion time is calculated as follows (12-bit mode):

$$T_{CONV} = \text{Sampling time} + 13 \text{ ADC clock cycles}$$

Example

With $F_{adc_ker_ck}$ = 32 MHz and a sampling time of 3 ADC clock cycles:

$$T_{CONV} = (3 + 13) \times \text{ADC clock cycles} = 16 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by asserting the EOSMP flag (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum and a maximum sampling times as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

The bulb sampling mode enables to obtain longer sampling time.

When the BULB bit is set in the ADC_CFGR2 register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPRx registers. The very first ADC conversion after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

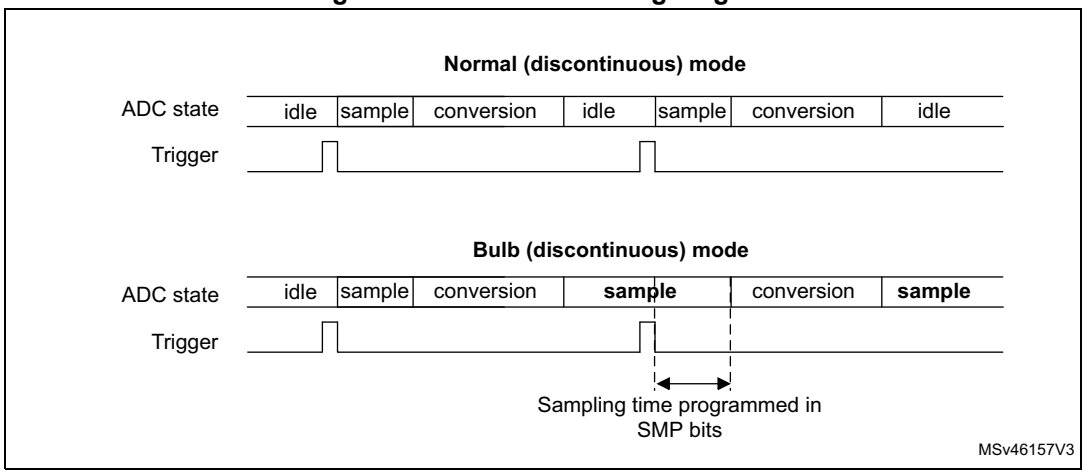
The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

Bulb mode is supported exclusively in regular conversion; dual mode is not supported.

The BULB bit can be modified only when the ADEN bit of the ADC_CR register is cleared.

When the BULB bit is set, it is not allowed to set the SMPTRIG bit in ADC_CFGR2.

Figure 93. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed through SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected (EXTEN[1:0] = 01), each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

Due to the synchronization mechanism, the minimum allowed pulse width is seven adc_ker_ck periods.

When a software trigger is selected (EXTEN[1:0] = 00), the software trigger is not the ADSTART bit of ADC_CR but the SWTRIG bit. The SWTRIG bit has to be set to start the sampling period, and it has to be cleared to end the sampling period and start the conversion. In this mode, the minimum sampling time is limited to seven ADC clock cycles.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is only compatible with the discontinuous mode. DISCNUM must be cleared for regular conversion mode. It is not supported for injected conversion mode and autoinjection mode.

When the SMPTRIG bit is set, setting the BULB bit is not allowed and only regular conversion is supported.

The sampling time control trigger mode is not compatible with the following dual modes: DUAL[4:0] = 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, and 0b01001.

21.4.13 Single conversion mode

To enable the single conversion mode for regular channels, clear both CONT and DISCEN bits in ADC_CFGR1 register. To enable it on injected channel, clear JDISCEN bit.

In single conversion mode, the ADC performs once all the regular conversions of the programmed channels. This mode is started by one of the following events:

- ADSTART bit set in the ADC_CR register (for a regular channel)
- JADSTART bit set in the ADC_CR register (for an injected channel)
- An external hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit 8-level FIFO accessible through the ADC_DR register,
- The EOC (end of regular conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers,
- The JEOC (end of injected conversion) flag is set, and
- An interrupt is generated if the JEOCIE bit is set.

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set, and
- An interrupt is generated if the JEOSIE bit is set.

The ADC then stops until a new external regular or injected trigger occurs or until the ADSTART or JADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

21.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts all the conversions of the sequence. This mode is started by setting the CONT bit of the ADC_CFGR1 register, either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the eight 32-bit word FIFO accessible through the ADC_DR register.
- The EOC (end of conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled, this means that it is forbidden to set both DISCEN and CONT bits.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit) (refer to Section : Autoinjection mode).

21.4.15 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN and CONT = 0 bit in the ADC_CFGR1 register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the LEN[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - First trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - Second trigger: channels converted are 6, 7, 8 (an EOC event is generated at the end of each conversion).
  - Third trigger: channels converted are 9, 10, 11 (an EOC event is generated at the end of each conversion) and an EOS event is generated after the conversion of channel 11.
  - Fourth trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - First trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all devices.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the fourth trigger reconverts the channels 1, 2 and 3 in the first subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR1 register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels with DISCNUM = 0.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JLEN[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - First trigger: channel 1 converted (a JEOC event is generated)
  - Second trigger: channel 2 converted (a JEOC event is generated)
  - Third trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all devices.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the fourth trigger reconverts the first injected channel 1.

It is not possible to use both autoinjection mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

21.4.16 Starting conversions (ADSTART, JADSTART)

ADC regular conversions can be started by setting ADSTART.

When ADSTART is set, the conversion starts:

- Immediately if EXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected regular hardware trigger, if EXTEN[1:0] is not equal to 00.

The application software starts ADC injected conversions by setting JADSTART.

When JADSTART is set, the conversion starts:

- Immediately if JEXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected injected hardware trigger, if JEXTEN[1:0] is not equal to 00.

Note: In autoinjection mode (JAUTO = 1), use the ADSTART bit to start regular conversions followed by autoinjected conversions (JADSTART must be kept cleared).

When the hardware trigger is enabled, ADSTART and JADSTART also indicate whether an ADC operation is ongoing. The ADC can be reconfigured while ADSTART and JADSTART are both cleared, indicating that the ADC is idle.

In the case of software triggering, ADSTP and/or JADSTP must be set before ADC reconfiguration.

ADSTART is deasserted by hardware:

- In single mode with software regular trigger (CONT = 0, EXTEN = 0x0): at any end of regular conversion sequence (EOS assertion)
- In discontinuous mode (CONT = 0, DISCEN = 1, DISCNUM= x): at any of subgroup sequence
- In all cases (CONT and EXTEN don't care), after execution of the ADSTP procedure asserted by the software

Note: In continuous mode (CONT = 1), ADSTART is not deasserted by hardware when EOS is asserted because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTEN ≠ 0x00), ADSTART is not deasserted by hardware when EOS is asserted to help the software that does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is deasserted by hardware:

- In single mode with software injected trigger (JEXTEN = 0x0): at the end of each injected conversion sequence (JEOS assertion)
- In discontinuous mode (JDISCEN = 1): at the end of any conversion
- In all cases (JEXTEN=x), after execution of the JADSTP procedure asserted by the software

Note: When the software trigger is selected, the ADSTART bit must not be set if the EOC flag is still high.

In bulb mode (BULB = 1), the previous end-of-conversion automatically starts the sampling, then setting ADSTART starts the conversion including the programmed sampling time.

In sampling time control trigger mode (SMPTRIG = 1), SWTRIG bit must be set to start the conversion.

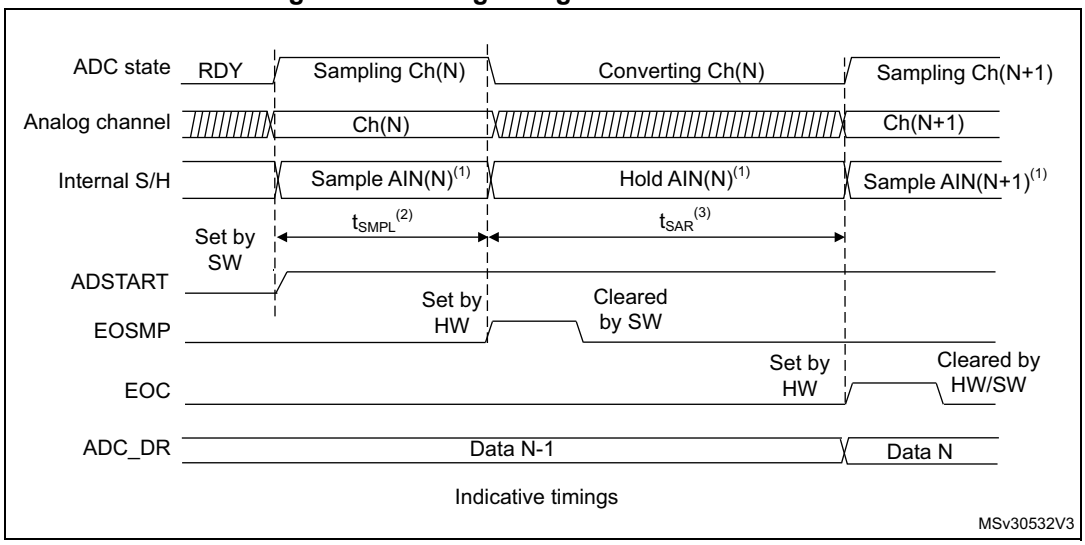
21.4.17 Timing

The elapsed time between the start of a conversion and the end of a conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [3\ _{|min} + 13\ _{|12bit}] \times T_{ADC_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 93.75\ ns\ _{|min} + 406.25\ ns\ _{|12bit} = 500\ ns$ (for $F_{adc_ker_ck} = 32$ MHz)

Figure 94. Analog-to-digital conversion time



1. AIN represents the input voltage captured by the internal Sample and Hold (S/H) capacitor.
2. t_{SMPL} depends on SMP[2:0].
3. t_{SAR} depends on RES[2:0].

Note: The bulb and the sampling time control trigger modes are not described in Figure 94: Analog-to-digital conversion time.

21.4.18 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop ongoing regular conversions by setting ADSTP and ongoing injected conversions by setting JADSTP.

Stopping conversions resets the ongoing ADC operation. The ADC can then be reconfigured (for example by changing the channel selection or the trigger). It is then ready for a new operation.

Injected conversions can be stopped while regular conversions are still ongoing and vice versa. This allows, for instance, the reconfiguration of the injected conversion sequence and triggers while regular conversions are still ongoing (and vice versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with the partial result discarded (the ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (the ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC restarts a new sequence).

Once this procedure is complete, ADSTP/ADSTART bits (for regular conversion), or JADSTP/JADSTART bits (for injected conversion) are deasserted by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In autoinjection mode (JAUTO = 1), setting the ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 95. Stopping ongoing regular conversions

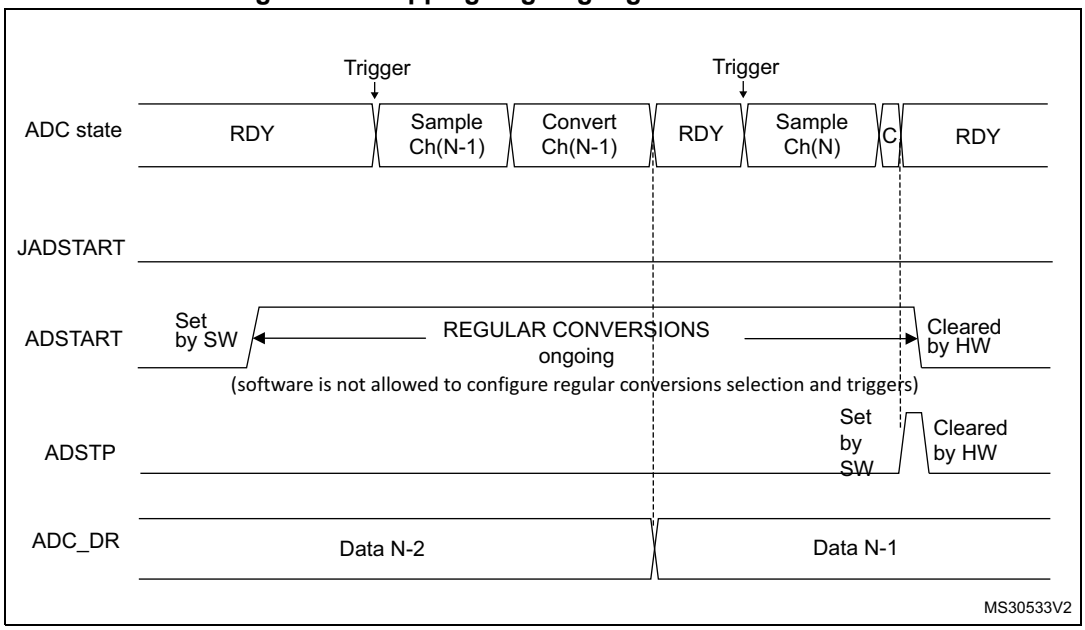
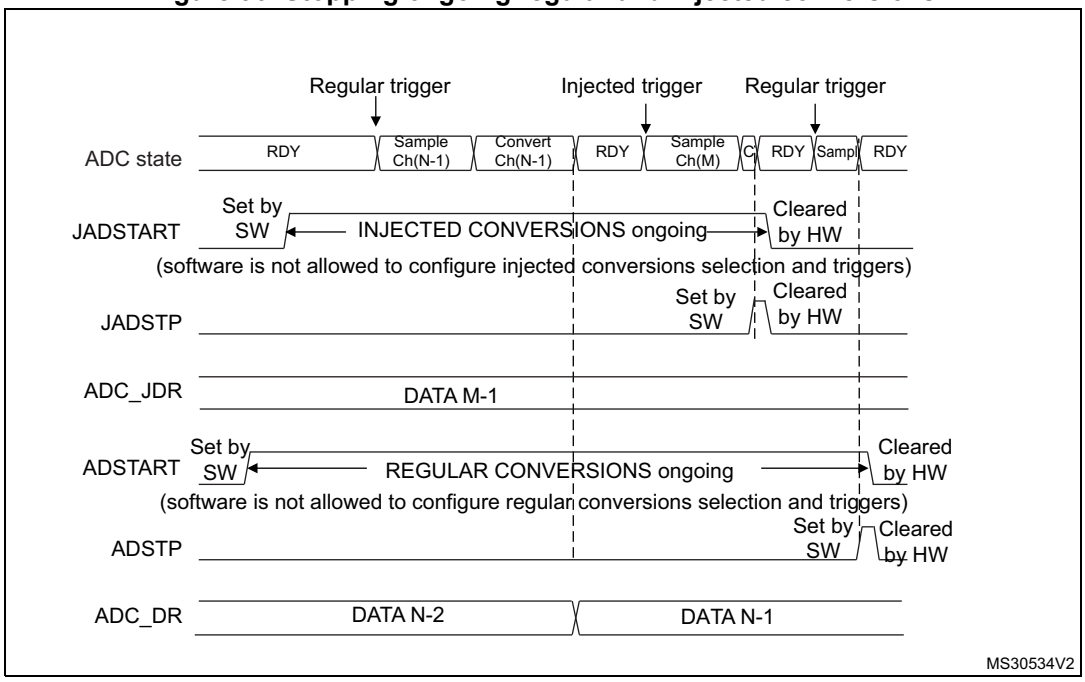


Figure 96. Stopping ongoing regular and injected conversions



21.4.19 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (for example timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once the software has set ADSTART while the injected trigger selection is effective once the software has set JADSTART bit.

All hardware triggers that occur while a conversion is ongoing are ignored:

- If ADSTART = 0, regular hardware triggers are ignored.
- If JADSTART = 0, injected hardware triggers are ignored.

Table 140 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 140. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware trigger detection disabled, software trigger detection enabled
01	Hardware trigger with detection on the rising edge (sampling time controlled in SMPTRIG mode)
10	Hardware trigger with detection on the falling edge
11	Hardware trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

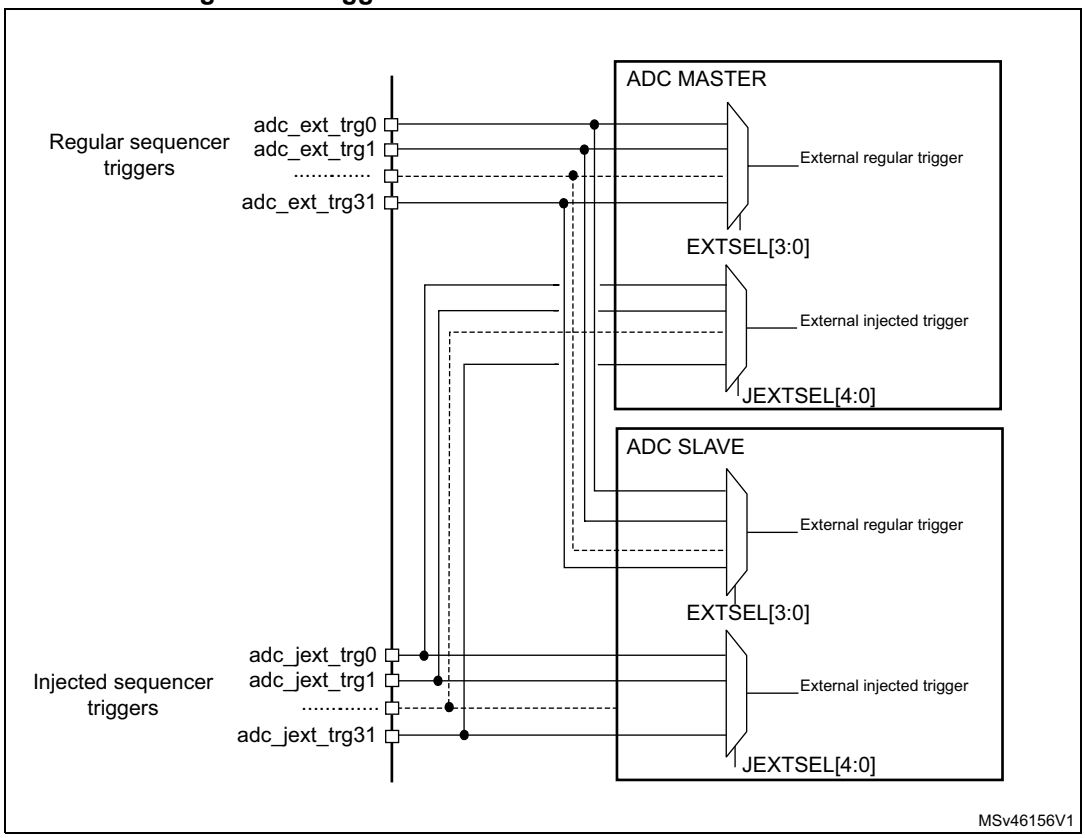
Table 141. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

The EXTSEL and JEXTSEL control bits select which, out of 32 possible events, can trigger the conversion of the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Figure 97. Triggers shared between ADC master and slave



Refer to *ADC interconnection* tables in *Section 21.4.2: ADC pins and internal signals* for the list of the external triggers.

21.4.20 Injected channel management

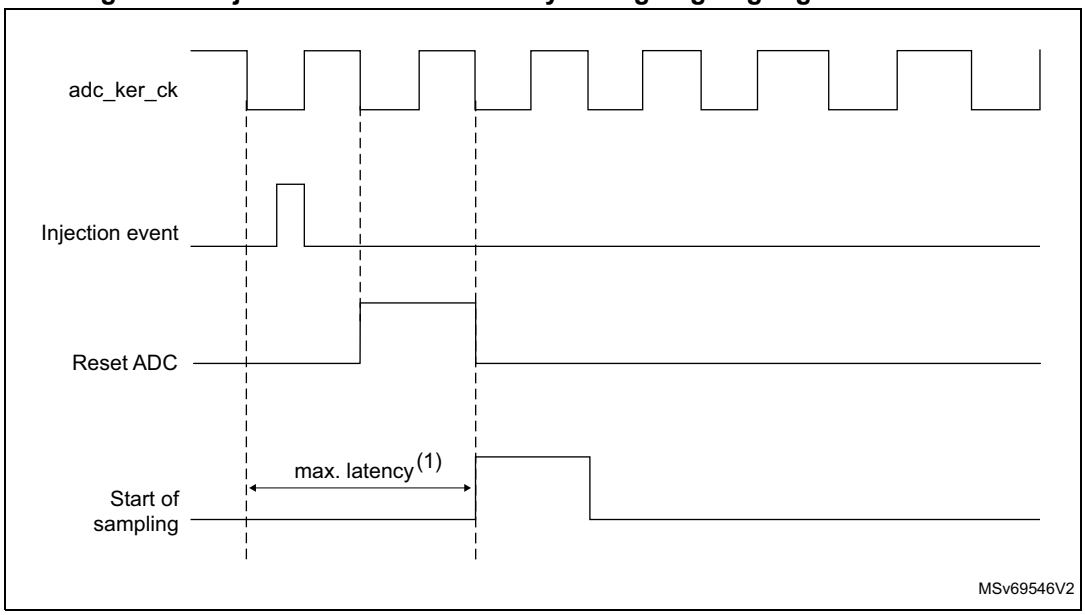
Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR1 register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once if JDISCEN = 0).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 98* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence.

Figure 98. Injected conversion latency during ongoing regular conversion



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

Autoinjection mode

If the JAUTO bit in the ADC_CFGR1 register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

Note: Autoinjection mode is compatible with discontinuous mode.

21.4.21 Programmable resolution (RES) - fast conversion mode

Faster conversions can be performed by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] control bits. *Figure 103*, *Figure 104*, *Figure 105* and *Figure 106* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 142*.

Table 142. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 32 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 3 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 32 MHz
12	13 ADC clock cycles	406.25 ns	16 ADC clock cycles	500.0 ns
10	11 ADC clock cycles	343.75 ns	14 ADC clock cycles	437.5 ns
8	9 ADC clock cycles	281.25 ns	12 ADC clock cycles	375.3 ns
6	7 ADC clock cycles	218.75 ns	10 ADC clock cycles	312.5 ns

21.4.22 End of conversion and end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application at the end of each regular conversion (EOC) event and injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set. The software can clear the EOC flag either by programming it to 1 or by reading ADC_DR. If ADC_DR FIFO is not emptied, the EOC flag is set again and an interrupt can be generated again.

The ADC sets the JEOC flag as soon as new injected conversion data is available in one of the ADC_JDRy registers. An interrupt can be generated if the JEOCIE bit is set. The software can clear the JEOC flag either by programming it to 1 or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of the sampling phase by setting the EOSMP flag (for regular conversions only). The EOSMP flag is cleared by software by programming it to 1. An interrupt can be generated if the EOSMPIE bit is set.

21.4.23 End of conversion sequence (EOS, JEOS)

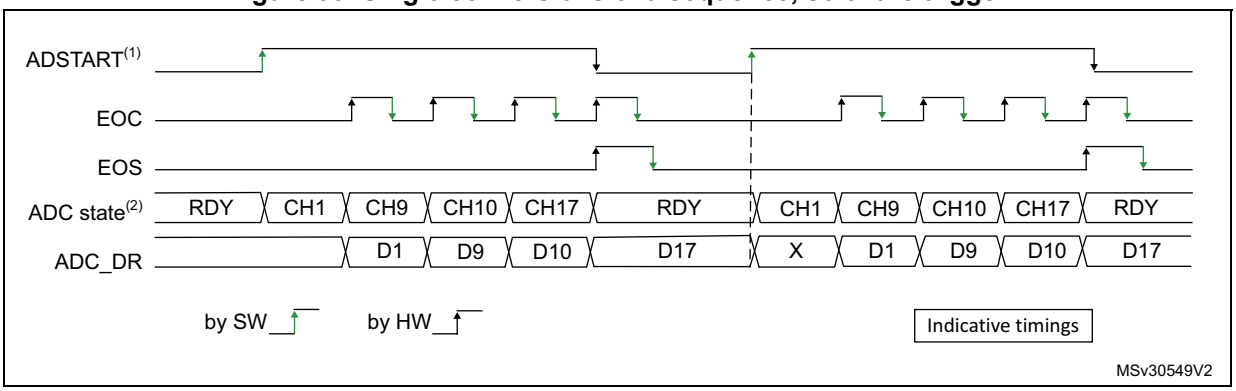
The ADC notifies the application at the end of each regular sequence (EOS) and injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the conversion of the last data of the regular sequence is complete. An interrupt can be generated if the EOSIE bit is set. The EOS flag is cleared by the software by programming it to 1.

The ADC sets the JEOS flag as soon as the conversion of the last data of the injected sequence is complete. An interrupt can be generated if the JEOSIE bit is set. The JEOS flag is cleared by the software by programming it to 1.

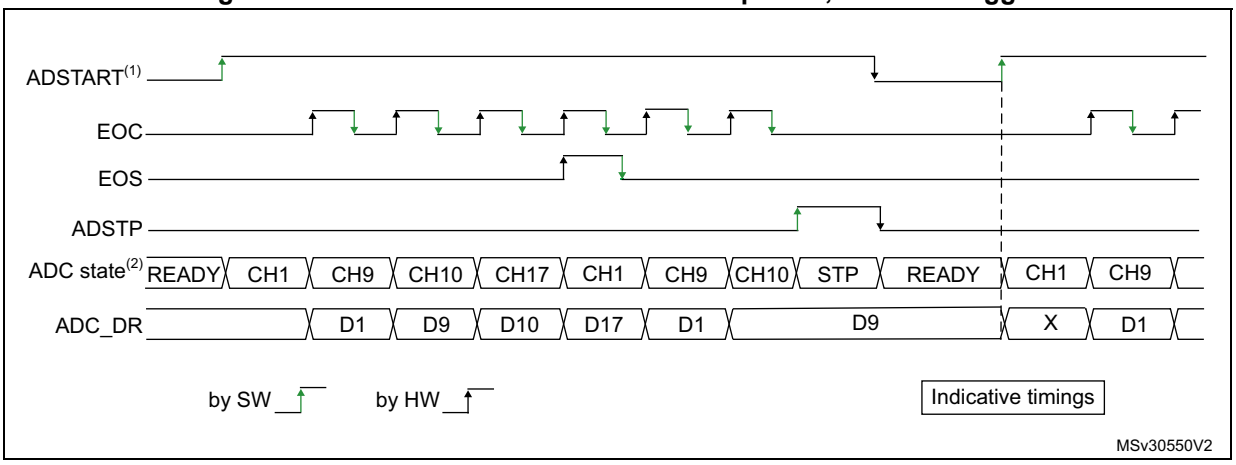
21.4.24 Timing diagram examples (single/continuous modes, hardware/software triggers)

Figure 99. Single conversions of a sequence, software trigger



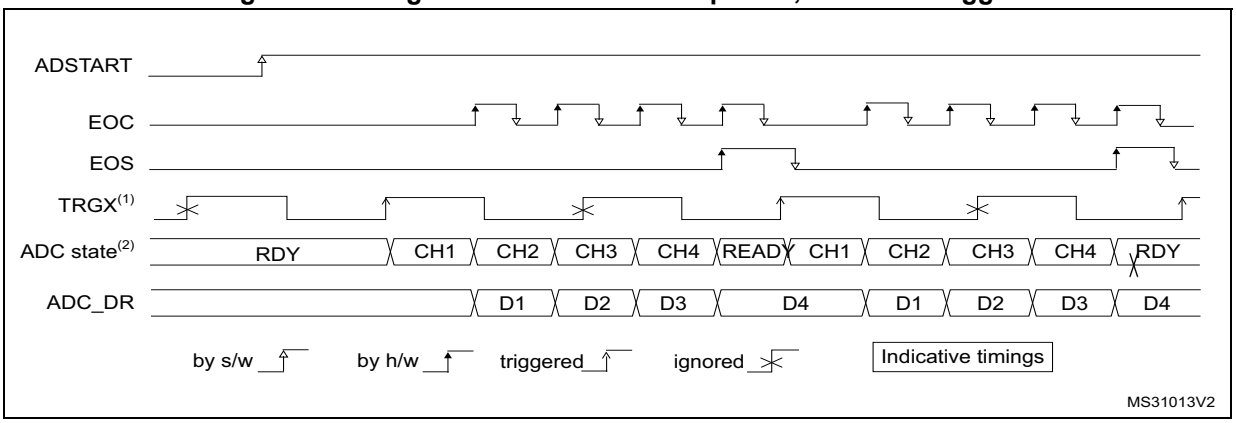
1. EXTEN[1:0] = 00, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 100. Continuous conversion of a sequence, software trigger



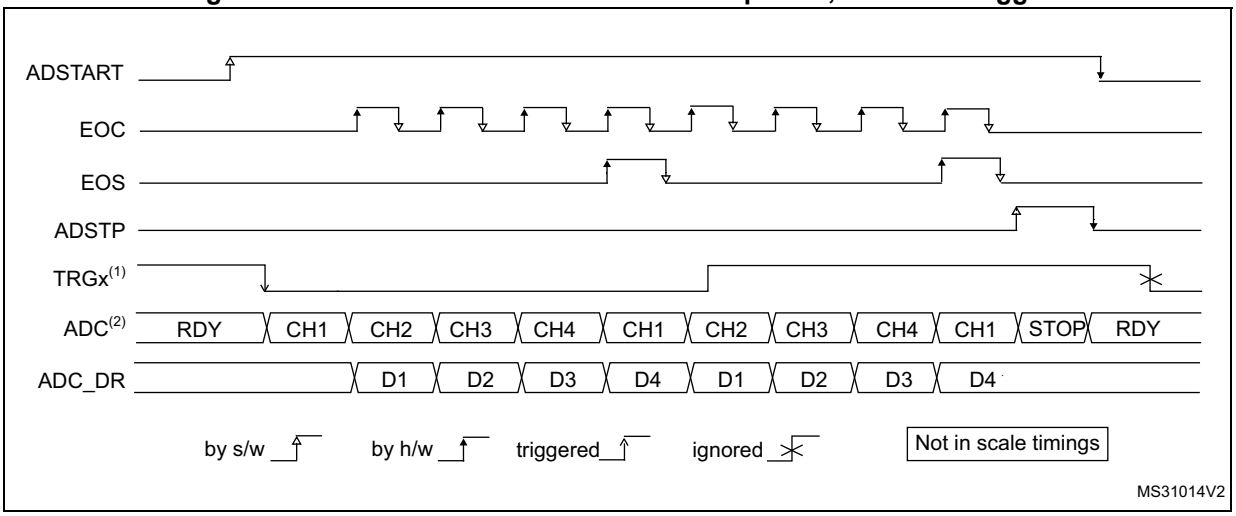
1. EXTEN[1:0] = 00, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 101. Single conversions of a sequence, hardware trigger



1. TRGx (overfrequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 102. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1.
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

21.4.25 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS[a], JLSHIFT[a], USAT, SSAT, POSOFF)

Data and alignment

At the end of each regular channel conversion (when an EOC event occurs), the result of the converted data is stored in the 32-bit wide ADC_DR data register with FIFO (when OVRMOD = 0). Since the data register has eight levels of FIFO, the EOC flag is raised again as far as the FIFO is not emptied.

At the end of each injected channel conversion (when a JEOC event occurs), the result of the converted data is stored in the 32-bit wide corresponding ADC_JDRy data register.

The OVSS[3:0], LSHIFT[3:0], JOVSS[3:0], and JLSHIFT[3:0] bitfields in the ADC_CFGR2 register select the alignment of the data stored after the conversion. By default, data are right-aligned. Refer to *Figure 103*, *Figure 104*, *Figure 105*, and *Figure 106* for examples of data alignment.

Note: The data can be aligned in normal and oversampling mode. The OVSS[3:0] bitfield is only effective in oversampling mode.

JOVSS[3:0] and JLSHIFT[3:0] is reserved for injected conversions. Their usage is the same as OVSS[3:0] and LSHIFT[3:0], unless otherwise specifically described.

Offset

An unsigned offset value y (y = 1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in the OFFSET[21:0] bitfield of the ADC_OFRy register. The channel to which the offset is applied is programmed to the OFFSETy_CH[4:0] bits of ADC_OFCFGRy register. The offset can be positive or negative depending on the value of POSOFF bit. When POSOFF is cleared, the converted value is subtracted by the user-defined offset written in OFFSET[21:0] bits. The result can be a negative value. The read data is consequently signed and the SEXT bit represents the extended sign value.

The offset value must be lower than the maximum conversion value (for example the maximum offset value is 0xFFF in 12-bit mode).

The offset can be used to convert unsigned data to signed data (for example the offset value is equal to 0x800 in 12-bit mode).

The offset correction is also supported in oversampling mode. In this mode, the offset is subtracted when all the oversampling operations are complete.

Note: If the OFFSET channel is selected but the offset value is 0x0000, this offset programming is ignored. If the same OFFSET channel is programmed in different registers multiple times, the first programmed register takes priority.

Table 148 describes how the computation is performed for all the possible ADC resolutions and offset settings.

a. Only for products that include this bit.

Table 143. Offset computation versus data resolution

Resolution (RES[1:0] bits)	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed or unsigned 24-bit data, right aligned to [11:0]	-
01: 10-bit	DATA[11:2],00	OFFSET[11:2]	Signed or unsigned 24-bit data, right aligned to [9:0]	The user must configure OFFSET[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	OFFSET[11:4]	Signed or unsigned 24 bit data right aligned to [7:0]	The user must configure OFFSET[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	OFFSET[11:6]	Signed or unsigned 24 bit data right aligned to [5:0]	The user must configure OFFSET[5:0] to 0b000000.

Figure 103, *Figure 104*, *Figure 105* and *Figure 106* show alignments for signed and unsigned data. JOVSS and JLSHIFT are not described here, but they behave the same as OVSS and LSHIFT.

Figure 103. Right alignment (offset disabled, unsigned value)

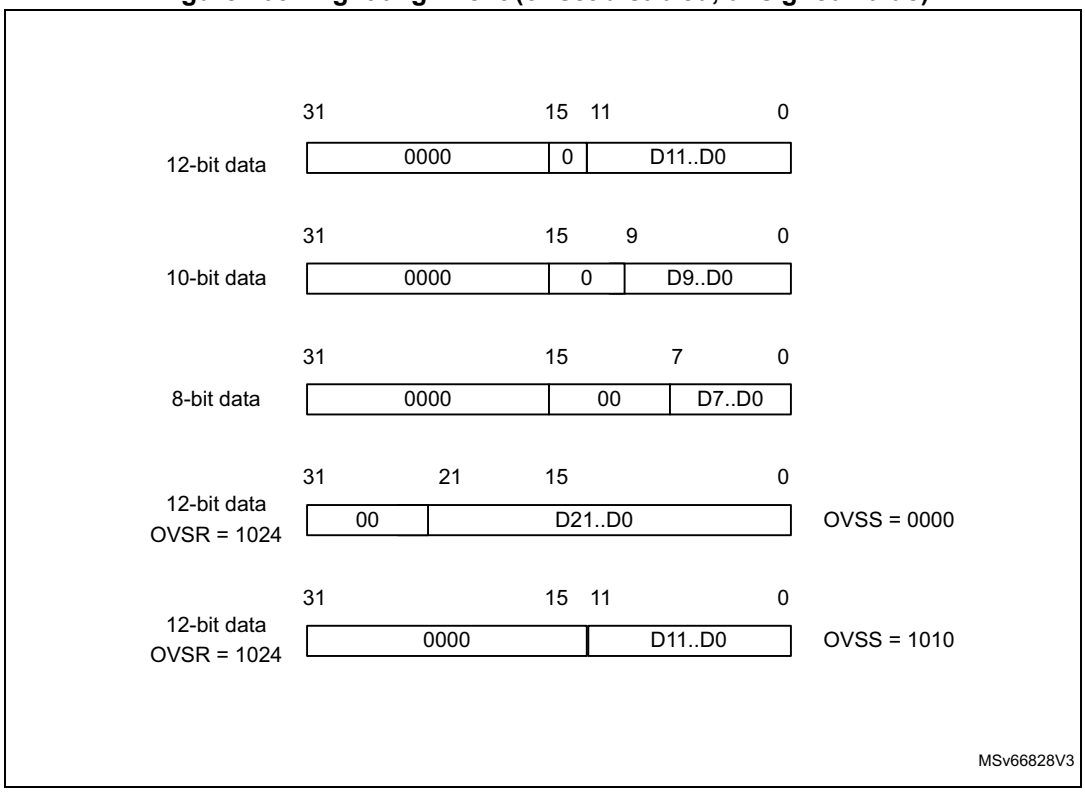


Figure 104. Right alignment (offset enabled, signed value)

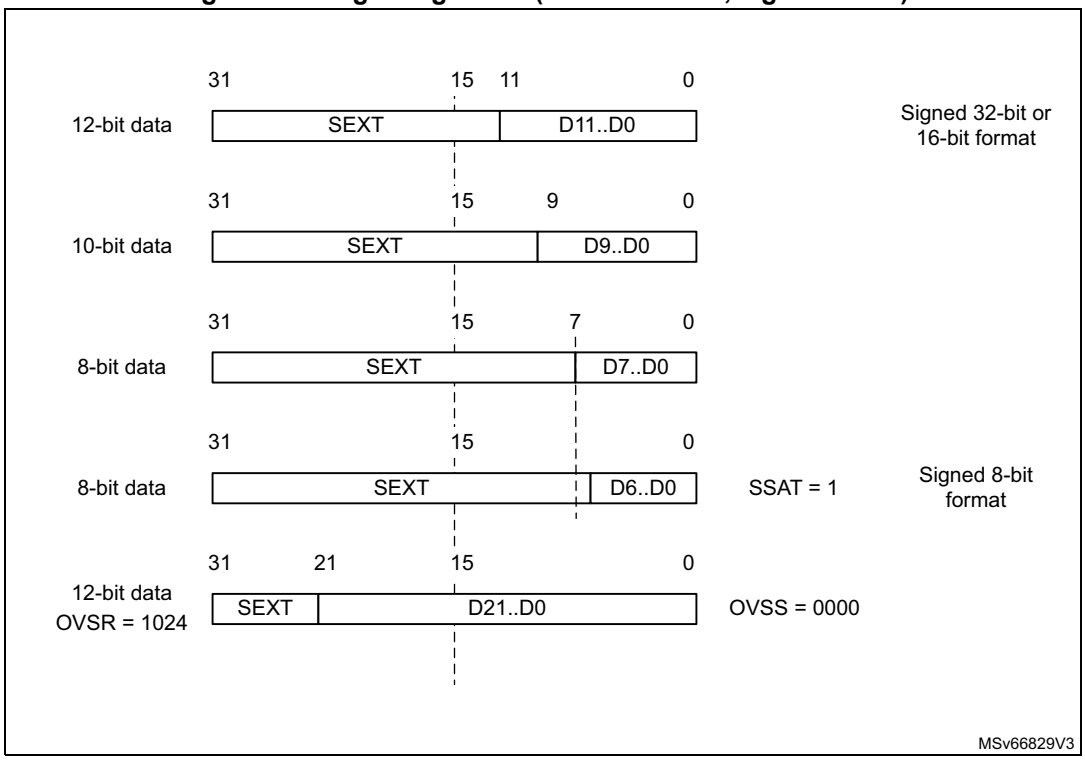


Figure 105. Left alignment (offset disabled, unsigned value)

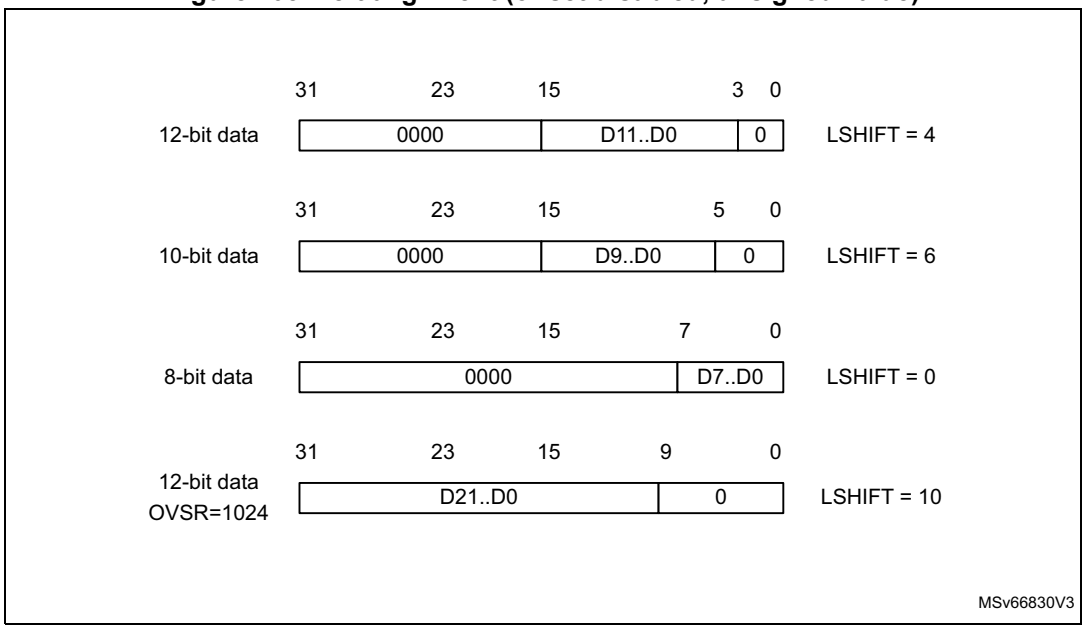
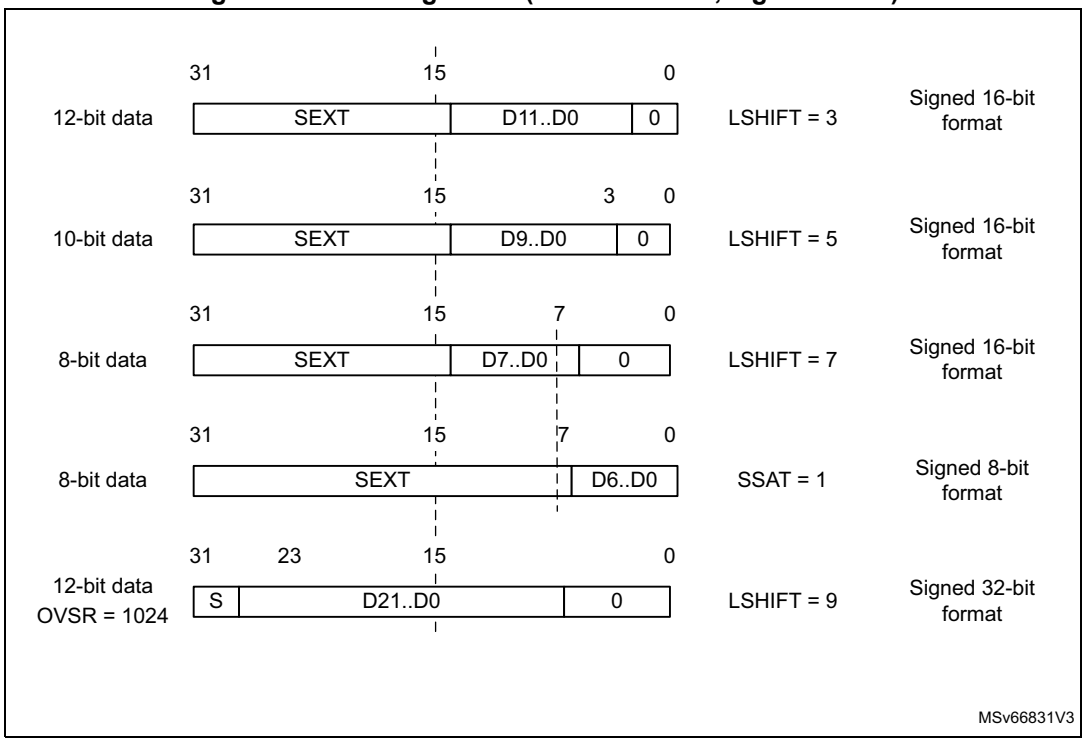


Figure 106. Left alignment (offset enabled, signed value)



Management of signed and unsigned saturation format (SSAT, USAT)

The offset correction might result in the data width to be wider than the original data.

To limit the original data width, the data saturation can be enabled through the SSAT and USAT bits of the ADC_OFCFGRy register.

Unsigned 12-bit data can be extended to 13-bit signed data by using an offset value different from 0x800.

The original data width can be preserved by setting SSAT bit to limit the data width to 12 bits.

Unsigned data can be saturated to the original data width by setting the USAT bit.

Table 144 shows the sign-extended data format corresponding to different resolutions.

Table 144.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	0	Sign-extended 13-bit significant data: SEXT[31:13] DATA[12:0]	0x0000 07FF - 0xFFFF F800
1	0	Sign-extended 12-bit significant data: SEXT[31:12] DATA[11:0]	0x7FF - 0x800

Table 144.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	1	Unsigned saturation 12-bit signification data: DATA[11:0]	0xFFF - 0x000
1	1	Reserved	-

Table 145 provides numerical examples for four different offset values.

Table 145. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0)

Raw conversion result	Offset value	Result SSAT = 0 USAT = 0	Result SSAT = 0 USAT = 1	Result SSAT = 1 USAT = 0
0xFFF	0x800	0x0000 07FF	0x07FF	0x07FF
0x800		0x0000 0000	0x0000	0x0000
0x000		0xFFFF F800	0x0000	0xF800
0xFFF	0x820	0x0000 07DF	0x07DF	0x07DF
0x800		0xFFFF FFE0	0x0000	0xFFE0
0x000		0xFFFF F7E0	0x0000	0xF7E0
0xFFF	0x7E0	0x0000 081F	0x081F	0x07FF
0x800		0x0000 0020	0x0020	0x0020
0x000		0xFFFF F820	0x0000	0xF820
0xFFF	0x020	0x0000 0FDF	0x0FDF	0x0FDF
0x800		0x0000 07E0	0x07E0	0x07E0
0x000		0xFFFF FFE0	0x0000	0xFFE0

Caution: SSAT must not be used in conjunction with USAT. No hardware check is performed to ensure that this condition is respected.

Gain compensation

When the GCOMP bit is set in the ADC_GCOMP register, the gain compensation is activated on all the converted data. After each conversion, data is calculated using the following formula.

$$DATA = DATA(\text{adc result}) \times (GCOMPCOEFF) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATAx or JDATAx bitfields, the LSB - 1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used in oversampling mode, the gain calculation is performed after the accumulation and

right-shift operations to minimize the power consumption (the gain calculation is done only once and not at each conversion).

The internal multiplier width is 32 bits and the input data width for the gain compensation must be less than 18 bits. When using oversampling with injected and regular conversion mode, the ROVSM bit of the ADC_CFGR2 register must be set to resume the pending conversion with the correct value.

When the gain compensation is activated, the ADC data latency is increased by one clock cycle. In continuous and scan conversion modes, the ADC conversion rate does not change.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) issues a notification that the regular converted data has not been read (by the CPU or the DMA) before the ADC_DR FIFO (eight stages) is overflowed, or adc_hclk clock is too slow to manage the data. If the CPU clears the EOC flag without reading the data register, the OVR flag may still be triggered.

The OVR flag is set when a new conversion completes while ADC_DR as single register or FIFO output is full. An interrupt is generated if the OVRIE bit is set.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP. The OVR flag is cleared by software by programming it to 1.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data are maintained up to ADC_DR FIFO depth (eight data) and the new conversion is discarded and lost. If OVR remains at 1, further conversions occur but the result data is also discarded. The FIFO can be emptied through ADC_DR read access by the CPU while the OVR flag is set.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data. In this mode, DMA burst mode cannot be supported.

Figure 107. Example of overrun (OVRMOD = 0)

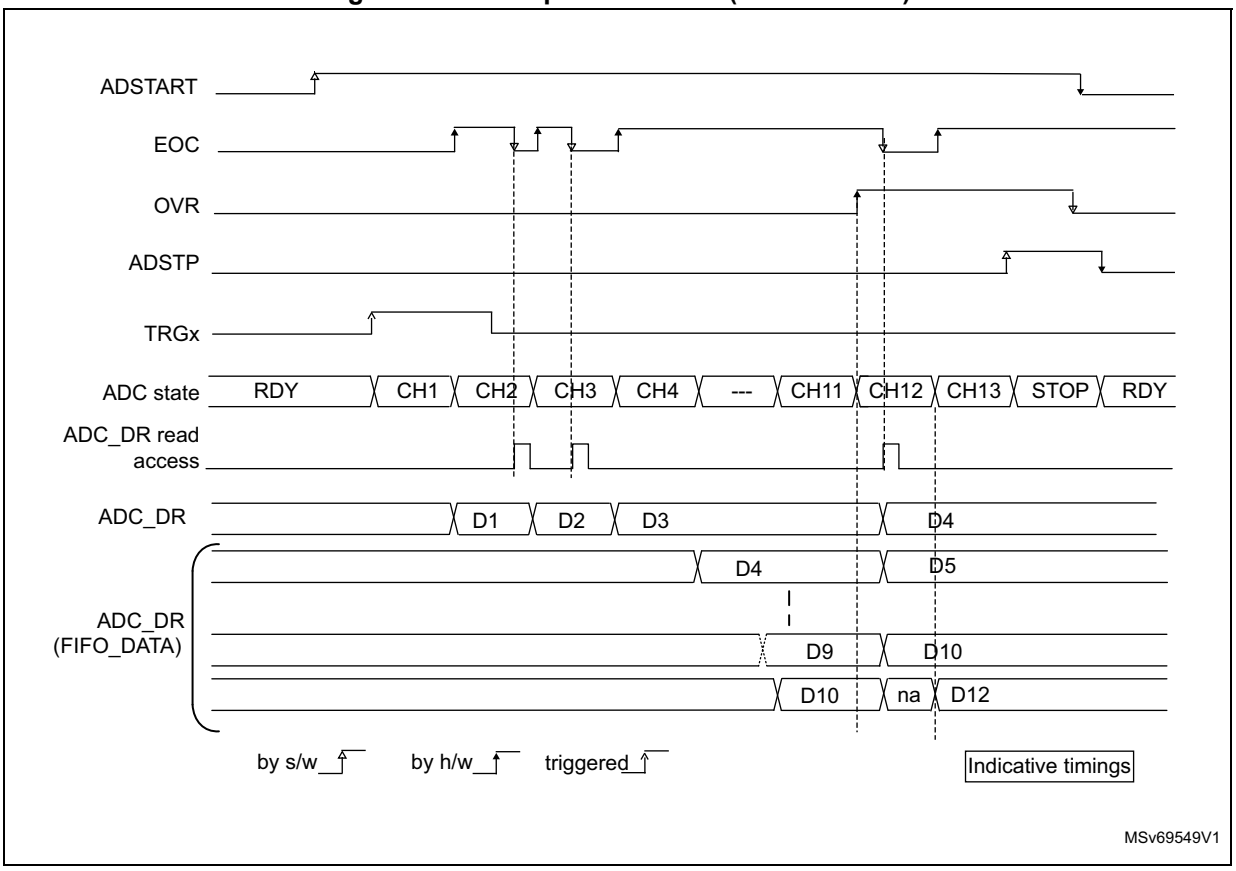
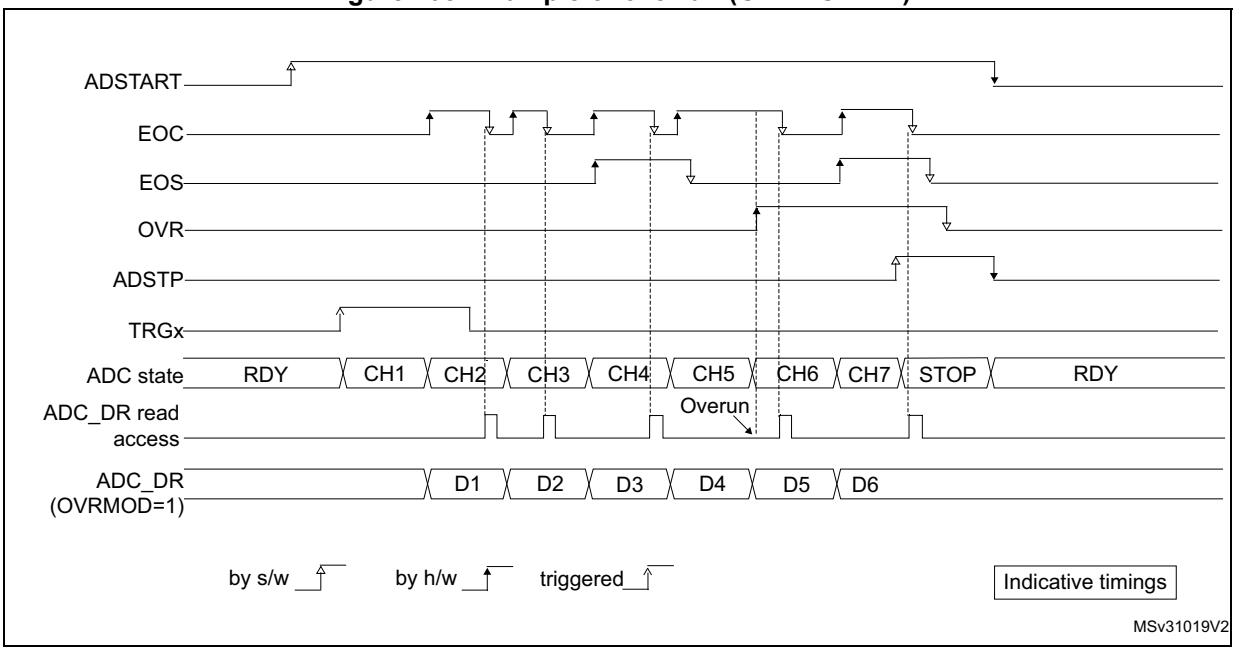


Figure 108. Example of overrun (OVRMOD = 1)



Note: Even if overrun may occur for injected channels, there is no overrun detection on these channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as errors.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and the OVR flag must be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

When the DMA mode is enabled (DMNGT[1:0] = 01 or 11 in the ADC_CFGR1 register in single ADC mode), a DMA request is generated after each channel conversion. This allows the transfer of the converted data from the ADC_DR register to the destination location configured in the DMA.

Despite this, if an overrun occurs (OVR = 1) because the DMA cannot serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA, which means that all the data transferred to the RAM can be considered as valid. (OVRMOD = 0).

When OVRMOD = 1, even if DMA does not transfer the data immediately, new data overwrite ADC_DR, causing DMA to potentially skip data. As a result, some conversion results may be missed in the data transferred to RAM.

Depending on the configuration of the OVRMOD bit, the data are either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*). If OVRMOD = 0 (data preserved), DMA transfer requests are blocked.

DMA can be configured through the DMNGT[1:0] bitfield of the ADC_CFGR1 register. Refer to *Section 21.4.29: Dual ADC modes* for information on DMA usage with dual ADC mode.

- DMA one-shot mode (DMNGT[1:0] = 01)

  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT[1:0] = 11)

  This mode is suitable when programming the DMA when the number of transfers is unknown.

DMA one-shot mode (DMNGT[1:0] = 01)

In one-shot mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (that is when DMA_LACK interrupt occurs, see *Direct-memory controller* section) even if a new conversion has been started.

When DMA transfers are complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- All ongoing conversions are aborted with the partial result discarded.
- The scan sequence is stopped and reset.
- The DMA and ADC are stopped.

DMA circular mode (DMNGT[1:0] = 11)

In circular mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register. This allows the configuration of the DMA in circular mode to handle a continuous analog input data stream.

DMA single-transfer or burst mode with FIFO (OVRMOD = 0)

The output data register features an eight-stage FIFO. Two different DMA requests are generated in parallel. When a data is available, an “SREQ single request” is generated. When four data are available, a “BREQ burst request” is generated. The DMA can be programmed either in single-transfer mode or in burst mode (four beats). The appropriate request line is selected by the DMA according to the mode selected. Refer to *Section Direct memory access controller* for further information.

Due to the existence of FIFO, DMA requests are generated until the FIFO is emptied. To stop DMA requests even when the FIFO is not emptied, set the ADSTP bit.

DMA for dual mode

Refer to *Section 21.4.29: Dual ADC modes*.

21.4.26 Dynamic low-power features

Autodelayed conversion mode (AUTDLY)

The ADC implements an autodelayed conversion mode controlled by the AUTDLY configuration bit in ADC_CFGR1. Autodelayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there must be risk of encountering an ADC overrun.

Regular conversion mode

When AUTDLY is set, a new conversion can start if the following conditions are met:

- The ADC_DR register is read or the EOC bit is cleared (see *Figure 109*).

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data and prevent overrun errors.

The delay is inserted after each regular conversion.

Injected conversion mode

When AUTDLY is set, a new conversion can start only if the JEOS bit has been cleared (see *Figure 110*). Since there are four ADC_JDRy registers, the autodelay is performed at the end on the injected conversion sequence.

There is no delay inserted between each conversion of the injected sequence, except after the last one.

A hardware trigger event occurring during this delay is ignored.

No delay is inserted between regular and injected conversions. The autodelay for regular and injected modes is managed separately.

Caution: The behavior is slightly different in autoinjection mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 113*).

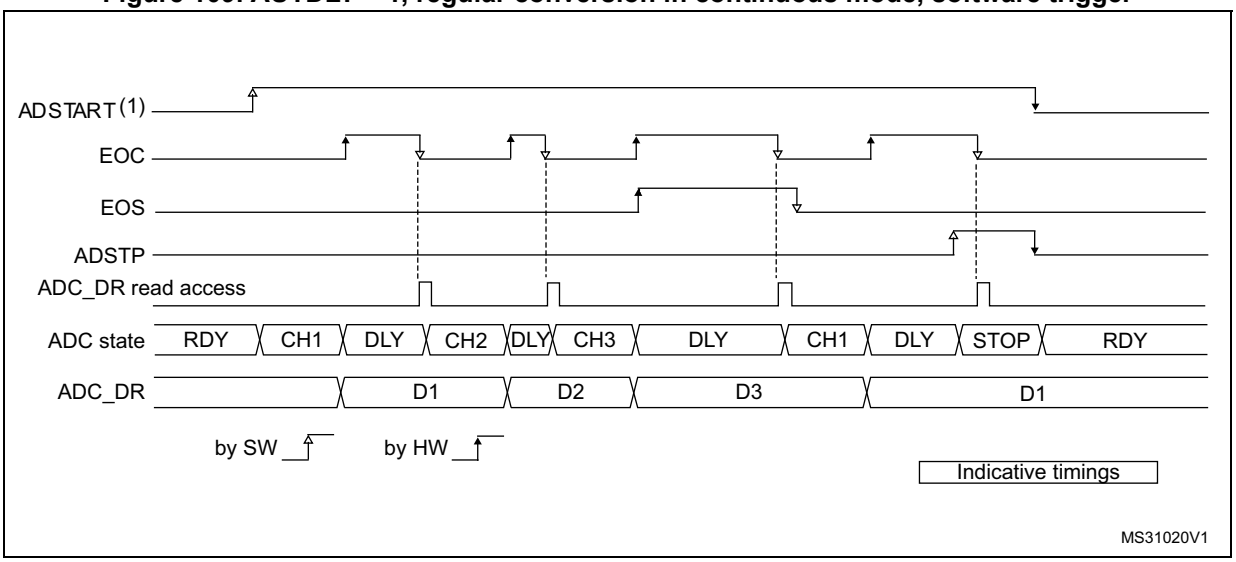
In autoinjection mode, there is no delay between the last regular conversion and the first injected conversion. The user must clear the last EOC flag before clearing the JEOS flag, otherwise the next sequence might start before EOC flag is cleared.

In autodelay mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In autodelay mode with JAUTO = 0, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

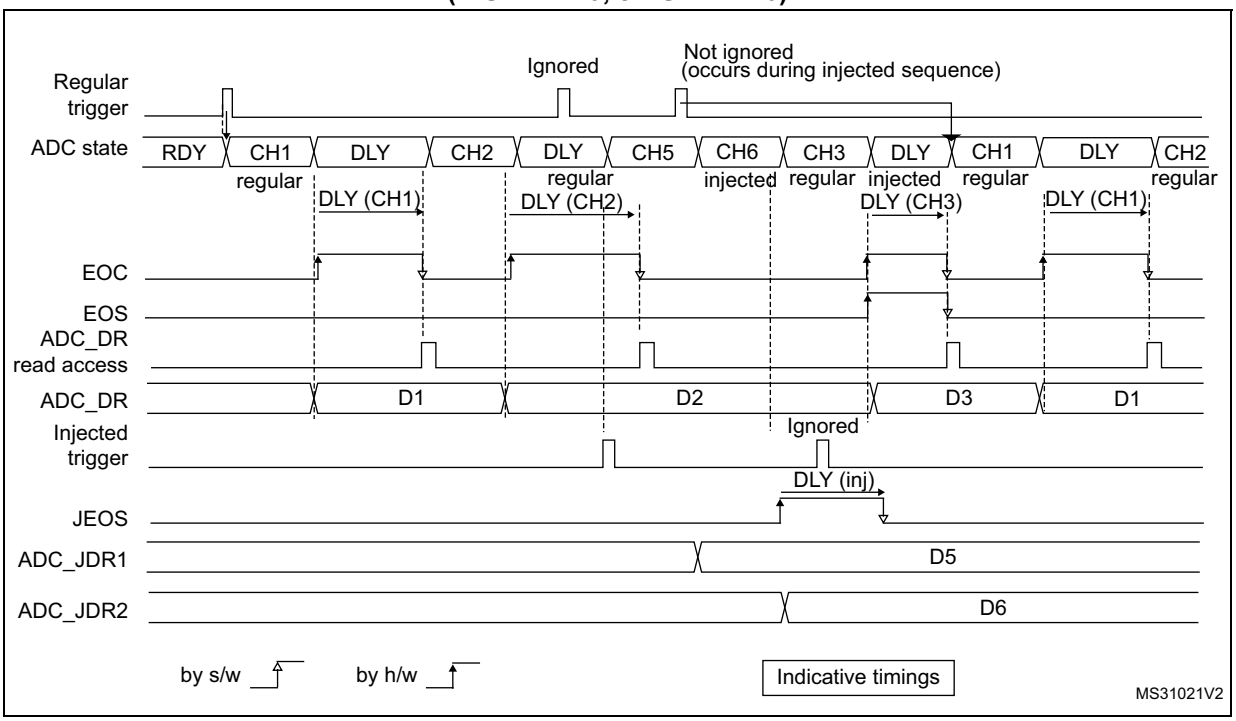
Note: AUTDLY mode is not supported with SMPTRIG or BULB mode.

Figure 109. AUTDLY = 1, regular conversion in continuous mode, software trigger



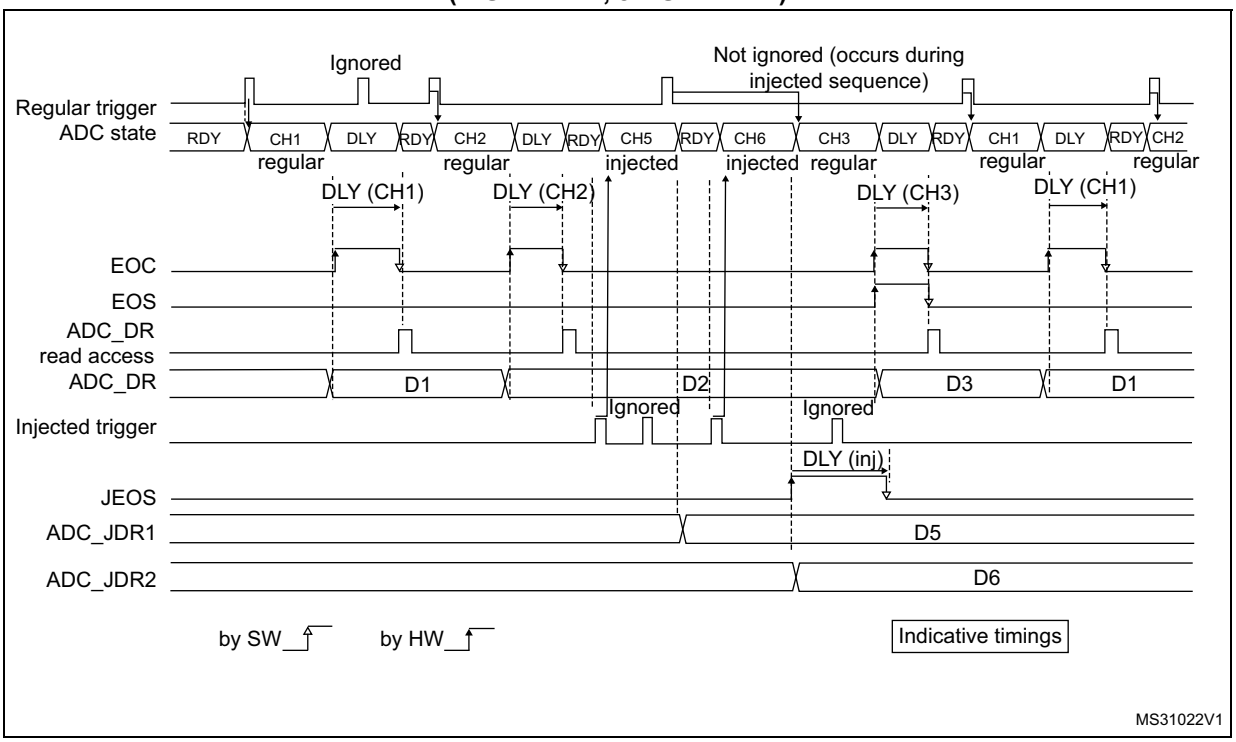
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, CHANNELS = 1, 2, 3.
3. Injected configuration DISABLED.

Figure 110. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



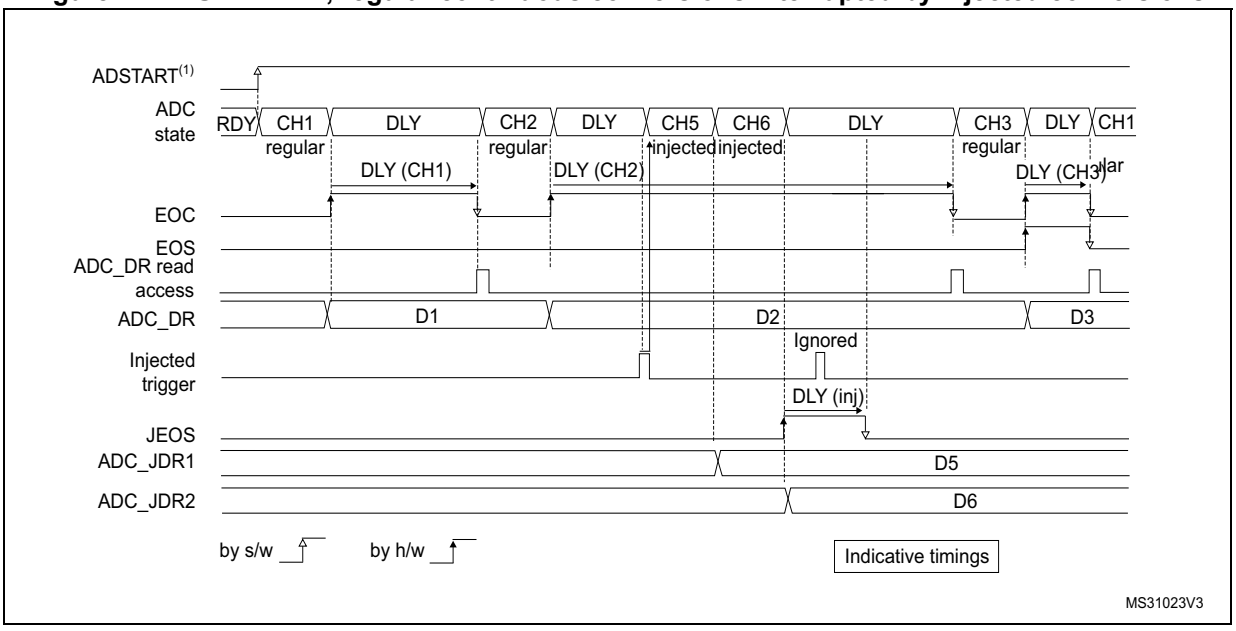
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 111. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



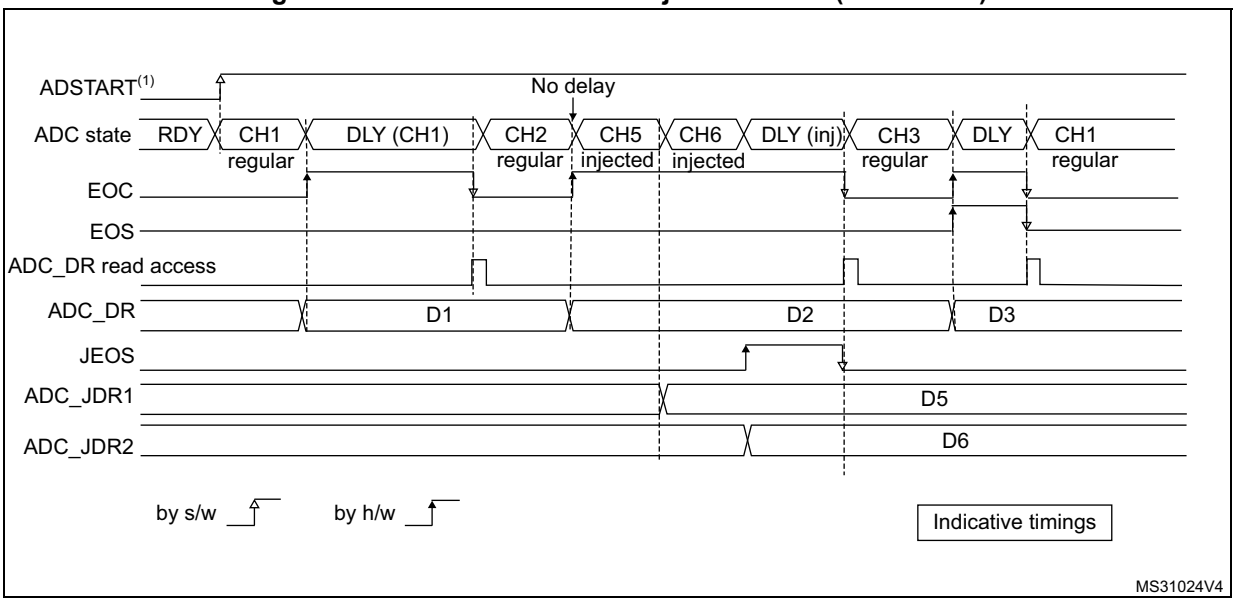
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 1, CHANNELS = 5, 6.

Figure 112. AUTDLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 113. AUTDLY = 1 in autoinjection mode (JAUTO = 1)



1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2.
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6.

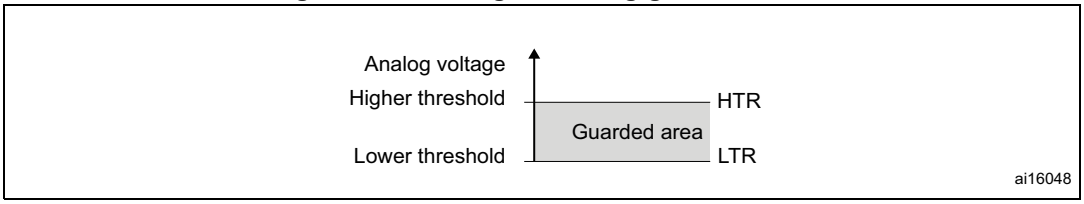
21.4.27 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)

Table 146. Analog window watchdog features

Feature	AWD1	AWD2	AWD3
Enable control	AWD1EN or JAWD1EN	ADC_AWD2CR	ADC_AWD3CR
Channel selection	AWD1CH[4:0]/AWD1SGL	ADC_AWD2CR	ADC_AWD3CR
Channel selection limitation	One or all channels	Any number of channels	
	Regular or injected	-	-
Threshold control	ADC_AWD1LTR / ADC_AWD1HTR	ADC_AWD2LTR / ADC_AWD2HTR	ADC_AWD3LTR / ADC_AWD3HTR
Filter configuration	AWDFILT[2:0] in ADC_AWDyHTR		
Interrupt	AWDy in ADC_ISR		

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 114. Analog watchdog guarded area



AWDy flag and interrupt

An interrupt can be enabled for each of the three analog watchdogs by setting AWDyIE in the ADC_IER register (y = 1, 2, 3).

AWDy (y = 1, 2, 3) flag is cleared by software by writing 1 to it, or by setting ADDIS.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN or JAWD1EN bit in the ADC_CFGR1 register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 147 shows how the ADC_CFGR1 registers must be configured to enable the analog watchdog on one or more channels.

Table 147. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog flag is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed through HTR[22:0] bits of the ADC_AWD1HTR register and LTR[22:0] bits of the ADC_AWD1LTR register for the analog watchdog 1.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024, offset and gain compensation)

When converting data with a resolution of less than 12 bits (defined by RES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data.

Table 148 describes how the comparison is performed for all the possible resolutions and for analog watchdog 1.

Table 148. Analog watchdog 1, 2, 3 comparison

Resolution (RES[1:0] bits)	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTR[22:0] and HTR[22:0]	-
01: 10-bit	DATA[11:2],00	LTR[22:0] and HTR[22:0]	The user must configure LTR[1:0] to 0b00 and HTR[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	LTR[22:0] and HTR[22:0]	The user must configure LTR[3:0] to 0b0000 and HTR[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	LTR[22:0] and HTR[22:0]	The user must configure LTR[5:0] to 0b000000 and HTR[5:0] to 0b000000.

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCH[13:0] (of ADC_AWD2CR and ADC_AWD3CR).

The corresponding watchdog is enabled when any bit of AWDCH[13:0] (of ADC_AWD2CR and ADC_AWD3CR) is set.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024 and offset compensation in signed format with gain compensation) and is programmed through the ADC_AWD2HTR, ADC_AWD2LTR, ADC_AWD3HTR and ADC_AWD3LTR registers.

When converting data with a resolution of less than 12 bits (defined by RES[1:0]) bits), the LSBs of the programmed threshold must be kept cleared, the internal comparison being performed on the full 12-bits converted data (left aligned).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated with an internal hardware signal ADC_AWDy_OUT (*y* being the watchdog number), which is directly connected to the ETR input (external trigger) of some of the on-chip timers. Refer to the on-chip timers sections to understand how to select the ADC_AWDy_OUT signal as ETR.

ADC_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDy_OUT is reset after the end of the next guarded conversion that is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS). Note that stopping regular or injected conversions (by setting ADSTP or JADSTP) has no influence on the generation of ADCx_AWDy_OUT.

Note: The AWDy flag is set by hardware and reset by software: the AWDy flag has no influence on the generation of ADC_AWDy_OUT (ex: ADC_AWDy_OUT can toggle while the AWDy flag remains at 1 if the software did not clear the flag).

The AWDy flag is cleared by programming it to 1 or by set ADDIS bit. ADSTP and JADSTP bits do not reset the AWDy flag.

Figure 115. ADC_AWDy_OUT signal generation (on all regular channels)

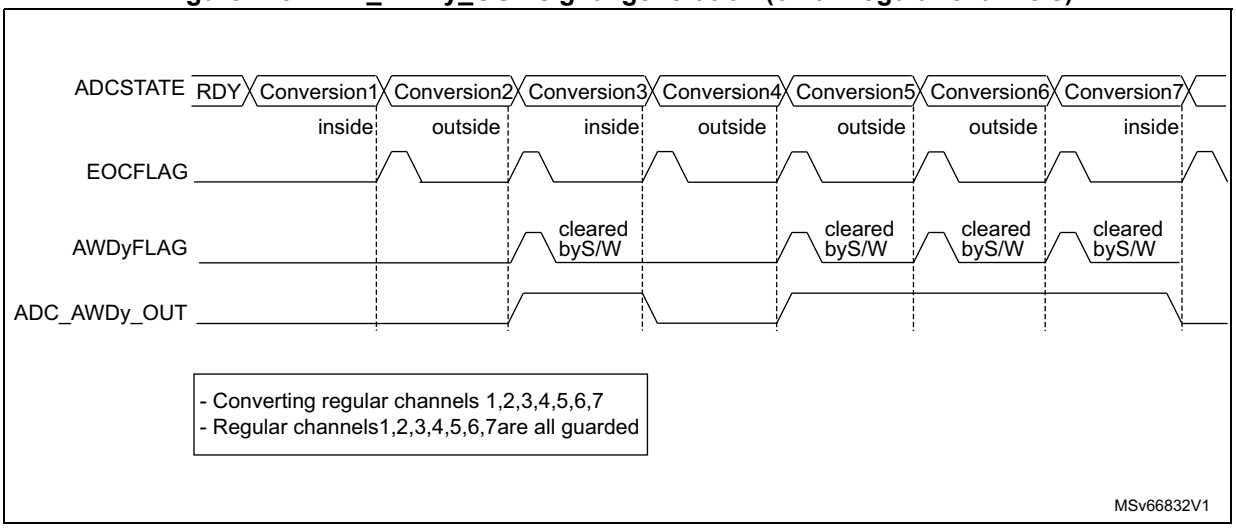


Figure 116. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software)

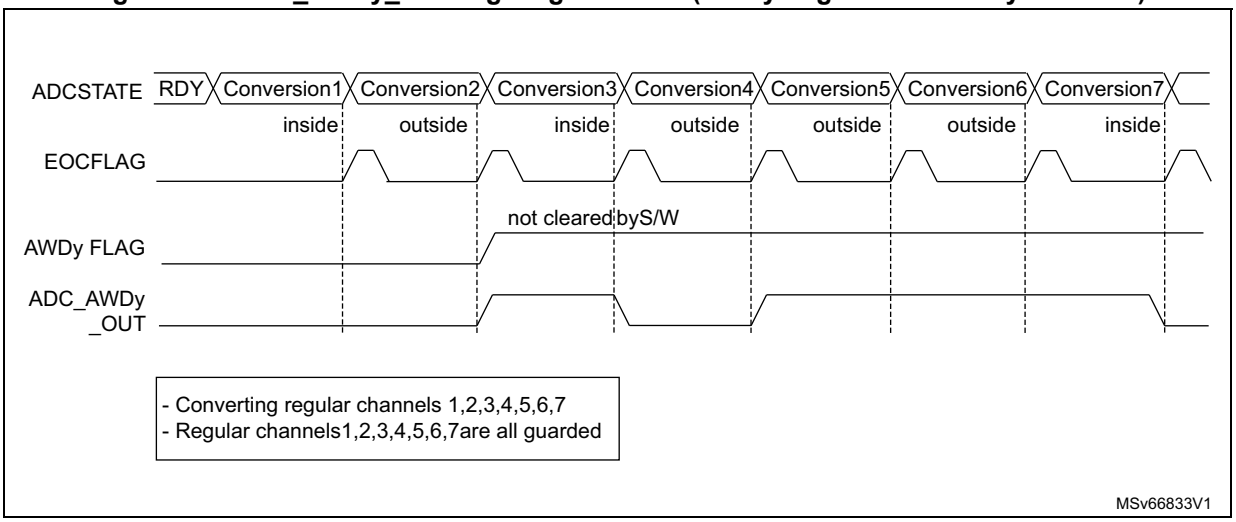


Figure 117. ADC_AWDy_OUT signal generation (on a single regular channel)

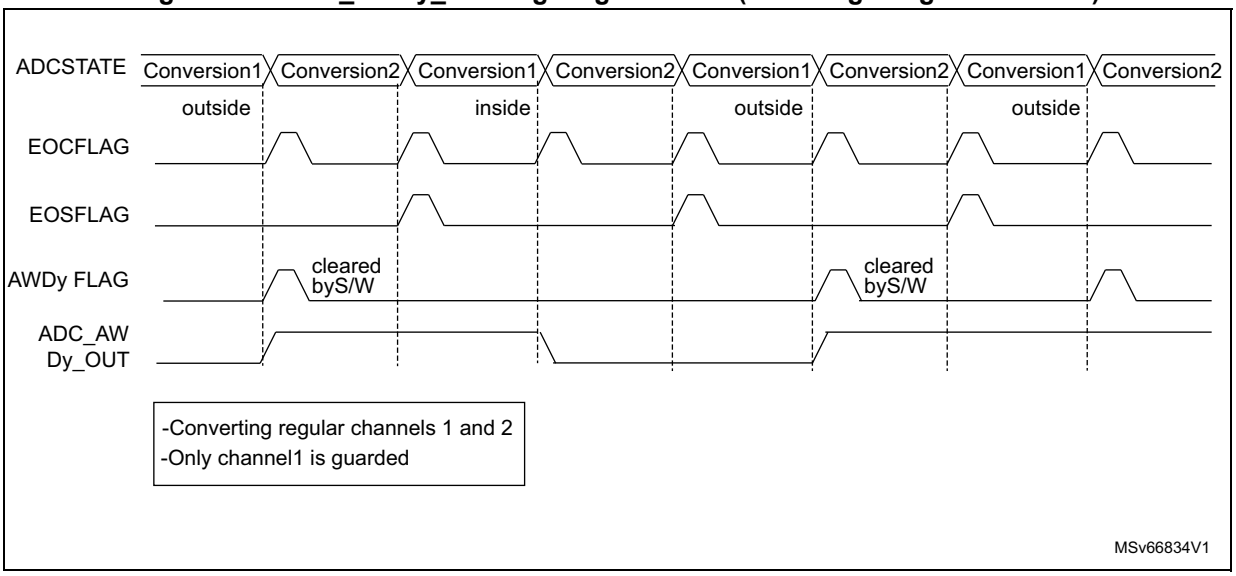
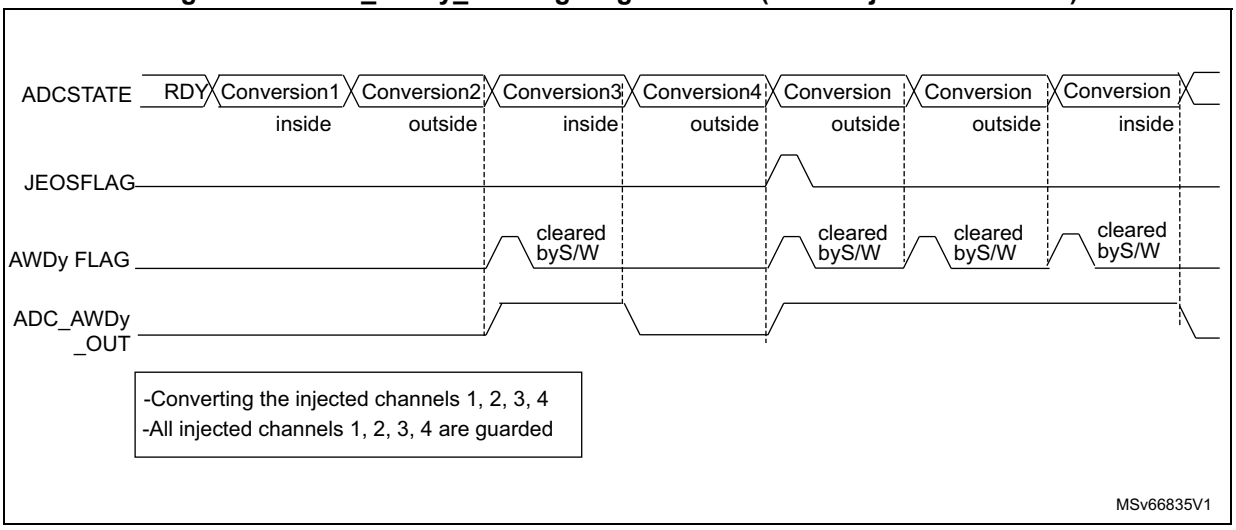


Figure 118. ADC_AWDy_OUT signal generation (on all injected channels)



Analog watchdog threshold control on the fly

LTR[22:0] and HTR[22:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTR[22:0] and HTR[22:0] are updated during the ADC conversion of the ADC guarded channel, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new modified threshold, no interrupt and AWDy_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion that started after the threshold update. If AWDy_OUT is already asserted, programming the new thresholds does not deassert the AWDy_OUT signal. However, the watchdog threshold must not be changed twice during the four adc_ker_ck periods.

Analog watchdog filter for watchdog 1

With analog watchdog, a valid ADC conversion data range can be configured through the ADC_AWD1LTR and ADC_AWD1HTR registers. When the conversion results have consecutively passed the threshold for more than the watchdog filter value (AWDFILT[2:0] + 1), AWD1_OUT is generated as well as the AWD1 flag. Before this value is reached, AWD1 output is not activated.

Analog watchdog with gain and offset compensation

When the gain and offset compensation are enabled, the analog watchdog compares the data after the compensation.

21.4.28 Oversampler

The oversampling unit performs data preprocessing to offload the device. It is able to handle multiple conversions and average them into a single data with increased data width, up to 22 bits.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It enables the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement and basic filtering.

The oversampling ratio N is defined by the OVSR[9:0] bits in the ADC_CFGR2 register, or by the JOVSS[3:0] bits in the ADC_CFGR3 register for parallel injected oversampling mode. They can range from 2x to 1024x. The division coefficient *M* consists of a right bit shift up to 10 bits. It is defined through the OVSS[3:0] bits of the ADC_CFGR2 register.

The summation unit can yield a result up to 22 bits, which can be left or right shifted at the end. When the right shift is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred to the ADC_DR data register.

Figure 119 gives a numerical example of the processing, from a raw 22-bit accumulated data to the final 12-bit result.

Figure 119. 12-bit result oversampling with 10-bit right shift and rounding

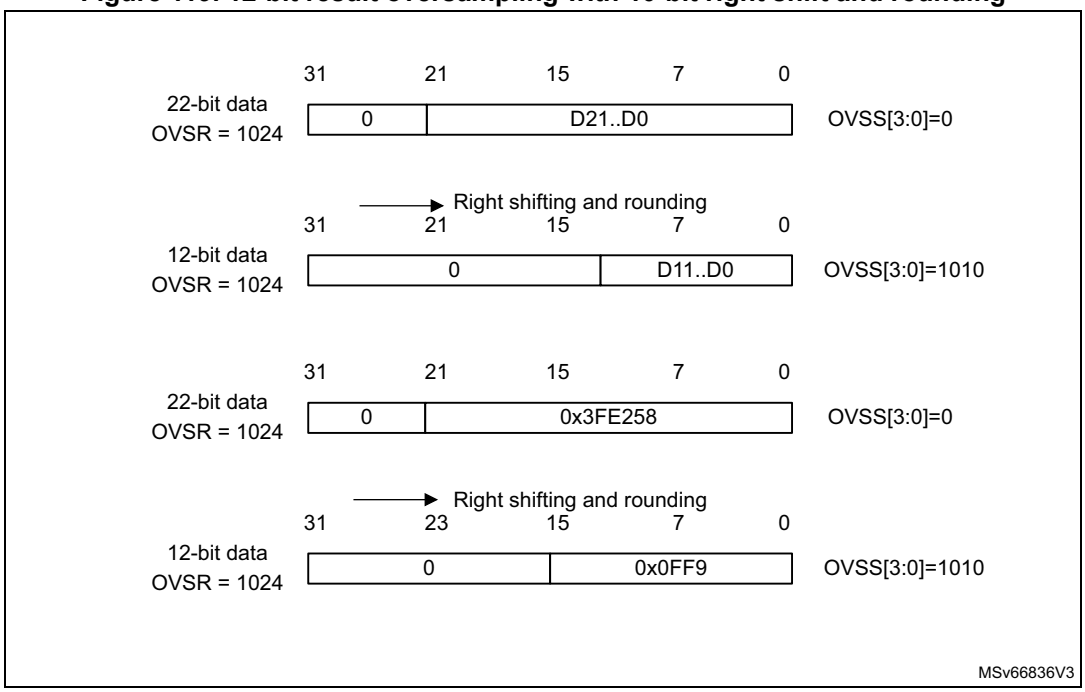


Table 107 gives data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 149. Maximum output results versus N and M

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS =0111	8-bit shift OVSS = 1000
x2	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
x4	0x3FFC	0x3FFC	x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
x16	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
x64	0x3FFC0	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
x256	0xFFF00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF
x1024	0x3FFC00	0x3FFC00	0x1FFE00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC

The conversion timing rate does not change in oversampling mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversion, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase for regular conversion.
- The end of conversion (EOC) occurs once every N conversion, when the oversampled result is available.
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after a total of N x sequence length conversions).

Unlike the conversion rate, the data latency changes in oversampling mode: each single conversion introduces an additional accumulation. As a result, an additional clock cycle is required when all the conversions are complete, after the accumulation and shift.
The EOC flag is set two additional clock cycles after the conversion is complete.

ADC operating modes supported in oversampling mode (single ADC mode)

In oversampling mode, most of the ADC operating modes are maintained:

- Single, discontinuous, or continuous conversion modes
- ADC conversions started either by software or triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits of the ADC_CFGR1 register) are accumulated, truncated, rounded, and shifted in the same way as 12-bit conversions are.
- Autoinjected mode (JAUTO) if JOVSPAR = 0 and oversampling ratio is the same for both regular and injected channels.

Analog watchdog

The analog watchdog functionality is maintained, with the following differences:

- The RES[1:0] bits are ignored and the comparison is always done using the full 23-bit values HTR[22:0] and LTR[22:0].
- The comparison is performed on the oversampled accumulated value before shifting.

Triggered mode

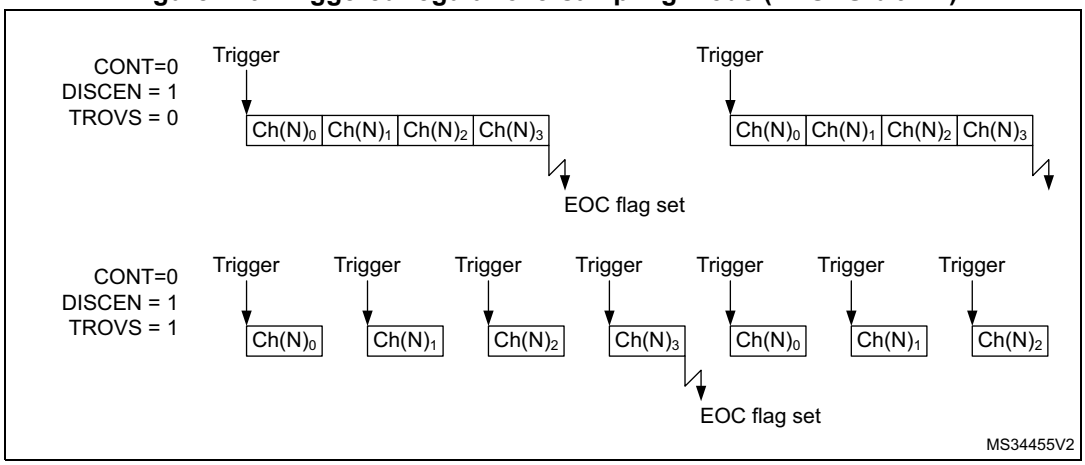
The oversampling can also be used for basic filtering purpose. Although it not a very powerful filter (slow roll-off and limited stop band attenuation), it can act as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched-mode power supply). For this purpose, a specific discontinuous mode can be enabled through the TROVS bit in ADC_CFGR2, to be able to achieve an oversampling frequency defined by the user and independent from the conversion time itself. The TROVS bit is supported with hardware trigger mode.

Figure 120 shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set and OVSR ≠ 0, the content of the DISCEN bit is ignored and considered as 1.

If the TROVS bit is set, neither software trigger nor JAUTO is supported.

Figure 120. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management in oversampling mode

In oversampling mode, injected and regular sequencers can have different behaviors. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

Oversampling regular channels only

The regular oversampling mode ROVSM bit defines how the regular oversampling sequence is resumed if it is interrupted by an injected conversion:

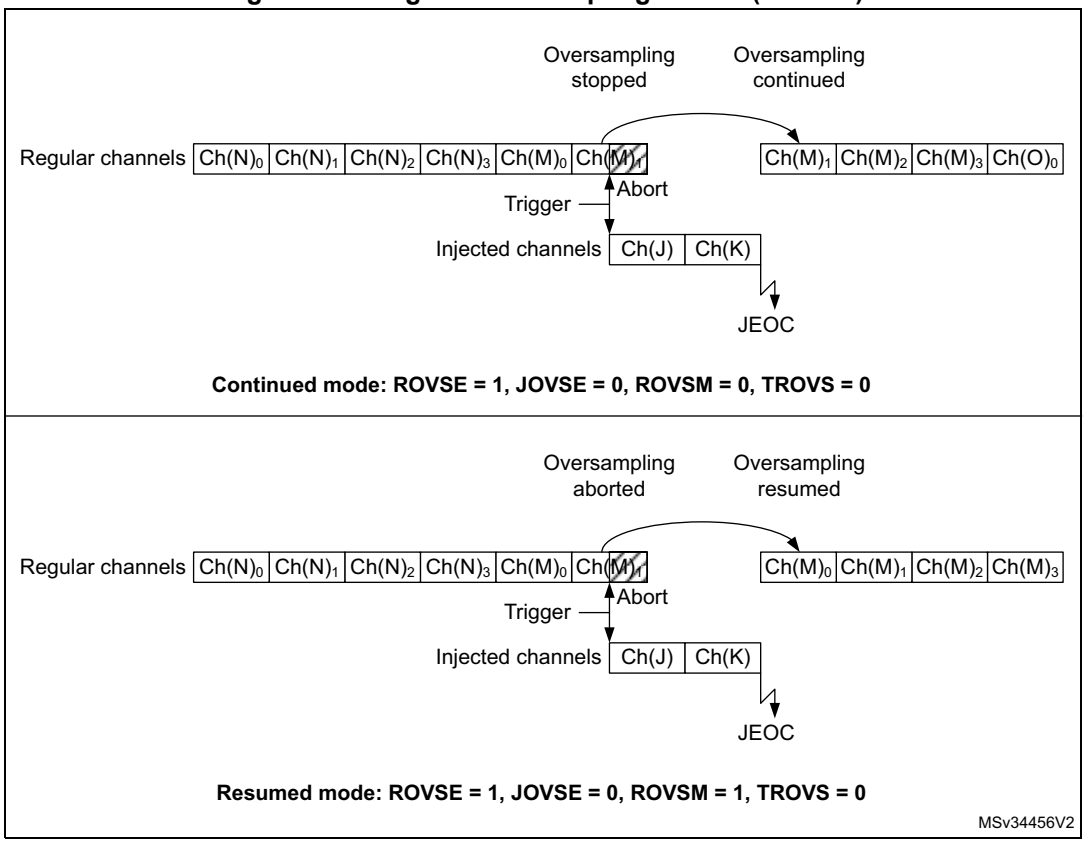
- In continuous mode (ROVSM = 0), the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that

oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);

- In resumed mode (ROVSM = 1), the accumulation restarts from 0 (previous conversion results are ignored). This mode guarantees that all the data used for oversampling were converted back-to-back within a single time slot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequence is blocked.

Figure 121 gives examples for a 4x oversampling ratio.

Figure 121. Regular oversampling modes (4x ratio)



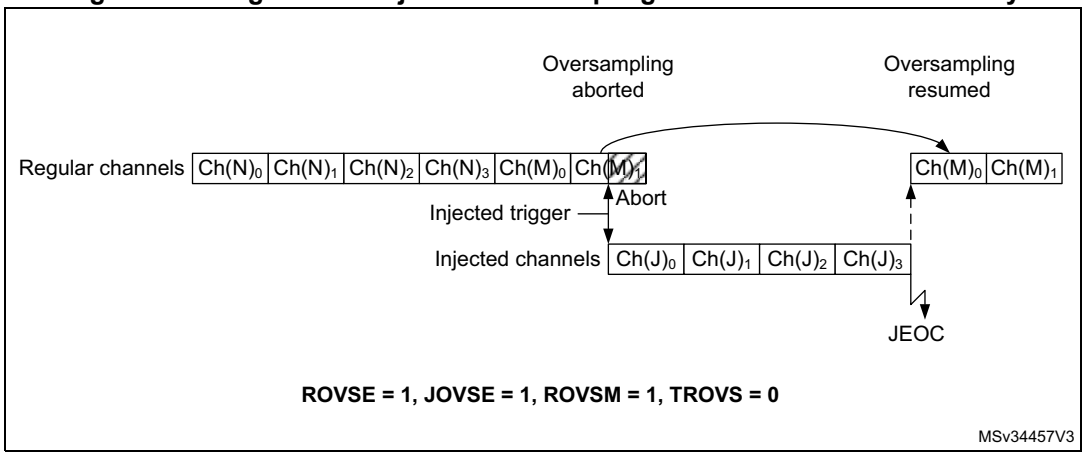
Oversampling injected channels only

The injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling must be resumed (ROVSM bit = 1), as shown in *Figure 122*. This limitation no longer applies in parallel injected oversampling mode.

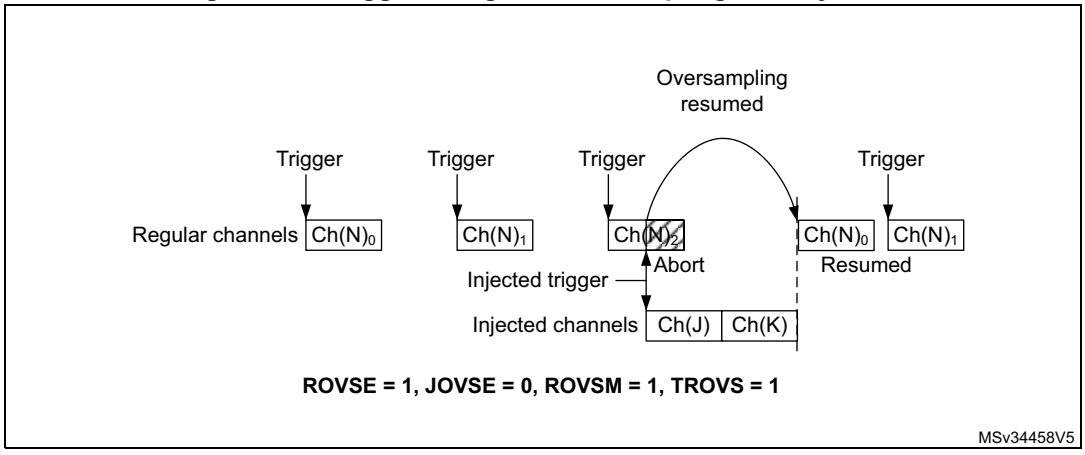
Figure 122. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected oversampling mode must be disabled and the ROVSM bit set (the resumed mode is forced). The JOVSE bit must be reset. The behavior is shown in *Figure 123*.

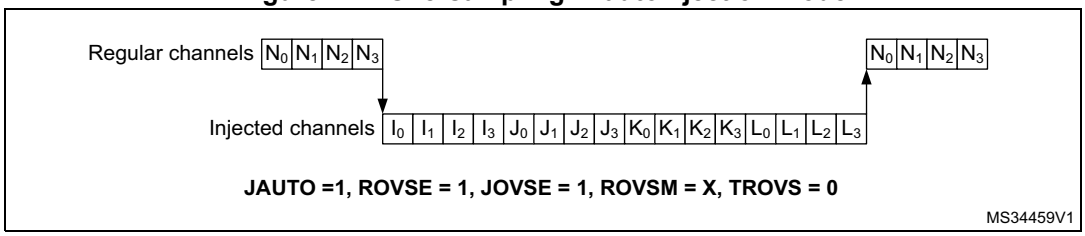
Figure 123. Triggered regular oversampling with injection



Autoinjection mode

It is possible to oversample autoinjected sequences and store all conversion results in registers. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1. Other combinations are not supported. The ROVSM bit is ignored in autoinjection mode. *Figure 124* shows how the conversions are sequenced.

Figure 124. Oversampling in autoinjection mode



Combined modes summary

Table 150 summarizes all combinations, including the modes that are not supported.

Table 150. Oversampler operating mode summary

Regular oversampling ROVSE	Injected oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

21.4.29 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 125*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADC master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when the DUAL[4:0] bits in the ADCC_CCR register are not equal to zero), the following bits are shared between the master and slave ADCs, with the slave ADC bits being ignored and always applying to the corresponding bits of the master ADC:

- ADC_CFGR1 register: DMNGT[2:0], RES[1:0], EXTEN[1:0], OVRMOD, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JAUTO, JDISCEN.
- ADC_CFGR2 register: ROVSE, JOVSE, TROVS, ROVSM, BULB, SWTRIG, SMPTRIG, OVSR[9:0].
- ADC_GCOMP register: GCOMP.
- ADC_CR register: ADSTART, ADSTP, JADSTART, JADSTP.
- ADC_CFGR1 register: CONT, DISCEN, DISCNUM[2:0].
- ADC_SQR1 register: LEN (for the regular conversion).
- ADC_JSQR register: JLEN[1:0], JEXTSEL[4:0], JEXTEN[1:0] (for the injected conversion).
- ADC_CFGR3 register: JOVSPAR, JOVSE, JOVSIVE, TJOVS, JOVSR[9:0] (for the injected conversion).

For dual mode usage, the sampling time of the master and slave channels must be equal.

To start a conversion in dual mode, the user must program the EXTEN[1:0], EXTSEL, JEXTEN[1:0] and JEXTSEL bits of the master ADC only to configure a software or hardware trigger and a regular or injected trigger. The EXTEN[1:0], JEXTEN[1:0], and JEXTSEL bits of the slave ADC are don’t care).

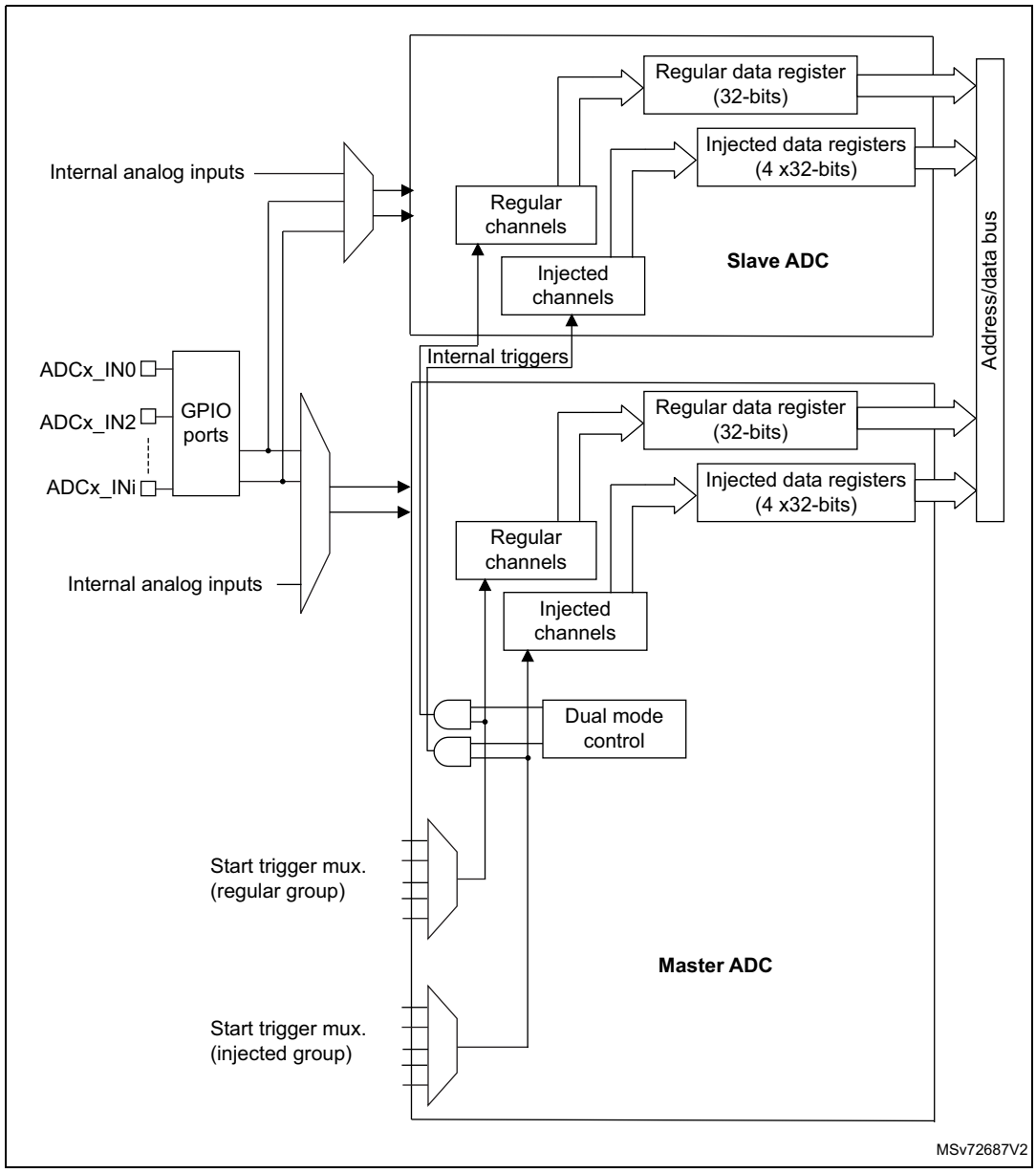
In regular simultaneous or interleaved modes, once the user sets the ADSTART or ADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, ADSTART or ADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes, once the user sets the JADSTART or JADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, JADSTART or JADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCC_CDR). The flags can also be read in parallel by reading the dual-mode status register (ADCC_CSR).

Note: In dual mode, BULB, JAUTO, SMPTRIG are not supported.

Figure 125. Dual ADC block diagram



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCC_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode with independent regular conversion

The injected simultaneous mode is selected by programming DUAL[4:0] bits to 0b00101.

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

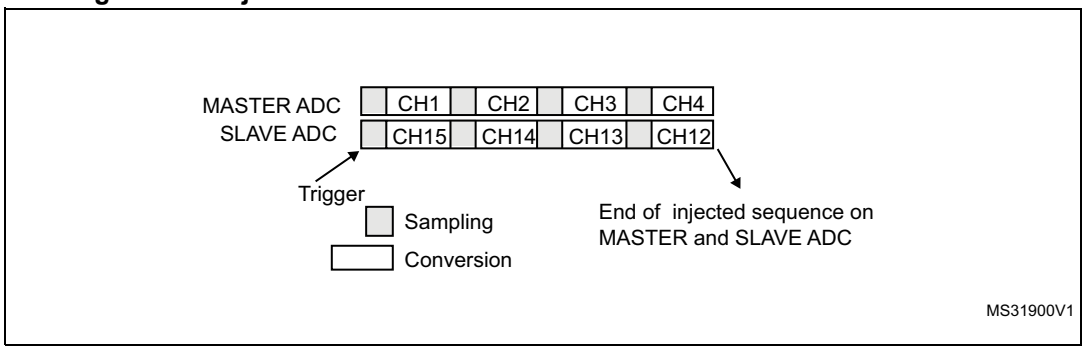
In simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

The software is notified by interrupts when it can read the data:

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master-injected sequence is equal to the duration of the slave injected one (as shown in *Figure 126*), the software can enable only one of the two JEOS interrupts (for example the master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 126. Injected simultaneous mode on four channels: Dual ADC mode



If JDISCEN is set, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with the AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored.
  There is the same behavior for regular sequences occurring on the slave ADC.

In this mode, the following register bits are shared between the master and slave ADCs:

JADSTART, JADSTP, JDISCEN, JOVSE, JLEN, JEXTEN, JEXTSEL, GCOMPEN, JOVSECT, JOVSISE, JOVSE, and TJOVS.

Regular simultaneous mode with independent injected conversions

The regular simultaneous mode is selected by programming DUAL[4:0] bits to 0b00110.

This mode applies to a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR1 register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on the master or on the slave) aborts the current simultaneous conversions both for master and slave. They are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

The software is notified by interrupts when it can read the data:

- As the duration of the master regular sequence is equal to the duration of the slave one (as shown in *Figure 127*), the software can enable only one of the two EOC interrupts (for example the master EOC) and read both converted data from the common data register (ADCC_CDR).
  The user can read data from either ADCC_CDR or ADCC_CDR2. Once software selects one of these registers, it must continue using the same register until the ADSTP procedure is executed.

The regular data can also be read using the DMA. Several methods are possible:

- Use two DMA channels (one for the master and one for the slave). In this case DAMDF[1:0] bits must be kept cleared:
  a) Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  b) Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configure the dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  a) Set DAMDF[1:0] = 0b10 or 0b11 (depending on resolution).
  b) A single DMA channel is used (the one corresponding to the master). Configure the DMA master ADC channel to read the common ADC data register (ADCC_CDR or ADCC_CDR2).
  c) A single DMA request is generated each time both the master and slave EOC events occur. At that moment:
  – The slave ADC converted data is available in the upper half-word of the 32-bit ADCC_CDR register
  – The master ADC converted data is available in the lower half-word of the ADCC_CDR register.

Alternatively, data can be read from ADCC_CDR2:

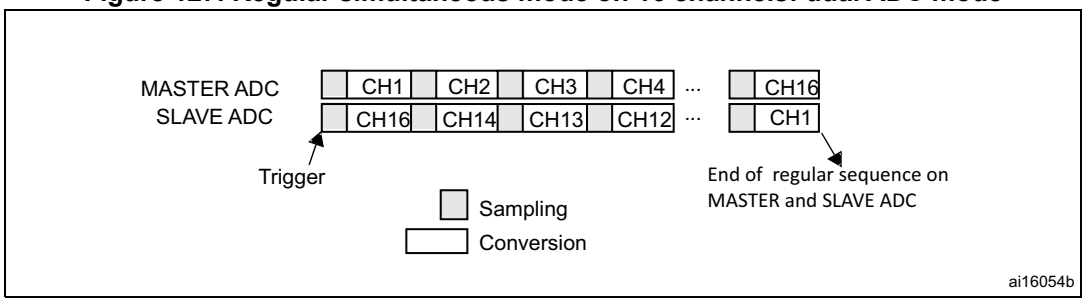
– The first read returns the master ADC data.

– The second read returns the slave ADC data.
After these reads, a DMA request is generated. The DMA transfers the slave data from the same data register and handles two transfers for dual mode operation.

d) Both EOC flags are cleared when the DMA reads the ADCC_CDR register.

Note: When DAMDF[1:0] = 0b10 or 0b11, the user must program the same number of conversions in the master and in the slave sequence. Otherwise, the remaining conversions do not generate a DMA request.

Figure 127. Regular simultaneous mode on 16 channels: dual ADC mode



If the DISCEN bit is set, each "*n*" simultaneous conversion of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with the AUTDLY mode:

- Once a simultaneous conversion of the sequence has completed, the next conversion in the sequence starts only after the EOC flag of both the master and the slave are cleared.
- Once a simultaneous regular sequence of conversions has completed, a new regular trigger event is accepted only after the EOC flag of both the master and the slave are cleared. Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

The DMA can be used to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

Interleaved mode with independent injected conversions

This mode is selected by programming DUAL[4:0] bits to 0b00111.

It can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately,
- The slave ADC starts after a delay of several ADC clock cycles after the sampling phase of the master ADC has completed (during the master ADC conversion period).

The minimum delay between two conversions in interleaved mode is configured in the DELAY bits in the ADCC_CCR register (see *Table DELAY bits versus ADC resolution*). This delay starts counting one half cycle after the end of the sampling phase of the master

conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However, the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

As the CONT bit is set on both the master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the master or slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Table 151. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1010	11 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1100	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
others	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$

Figure 128. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode

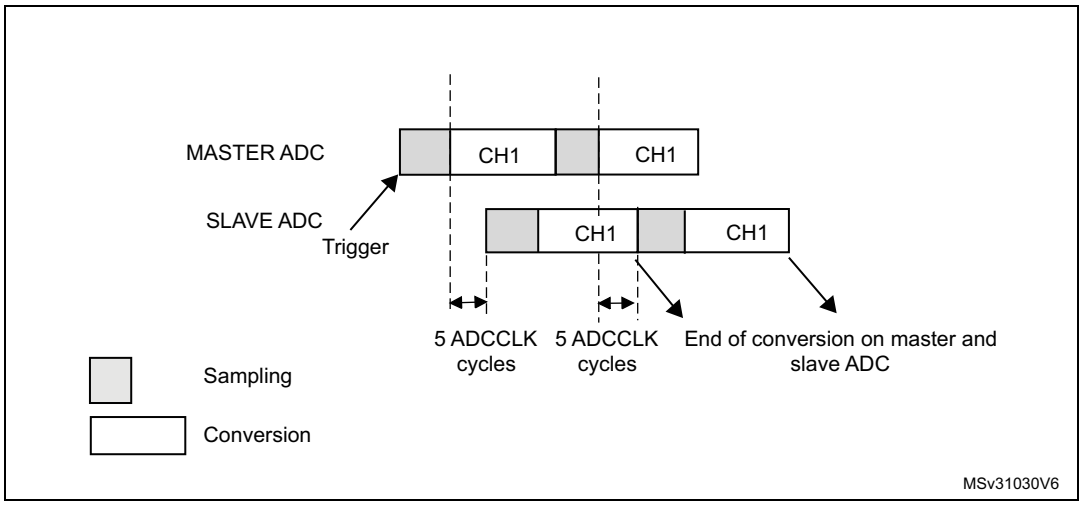
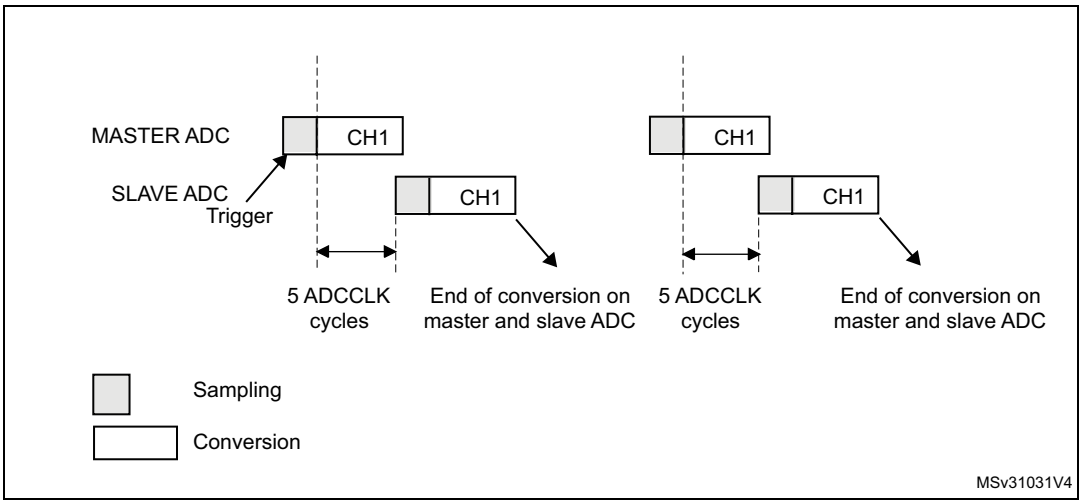


Figure 129. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode



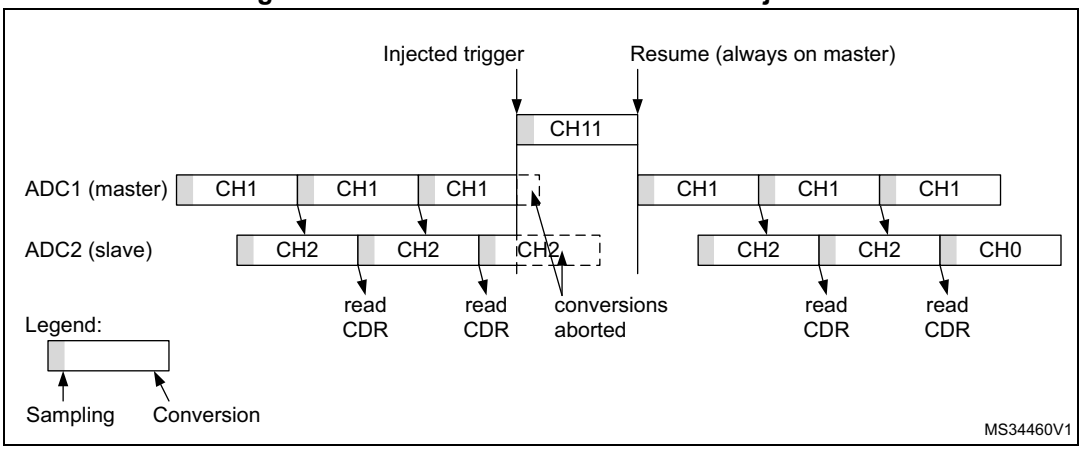
If the DISCEN bit is set, each "*n*" simultaneous conversion ("*n*" is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In interleaved mode, injected conversions are supported. When injection is done (either on master or slave), both master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 130*).

Note: In interleaved mode, the ADCx_AWDy_OUT signal for the master ADC is synchronized with the EOC of the master ADC, but the interrupt flag is synchronized with the EOC/JEOC flag of the slave ADC.

In addition, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 130. Interleaved conversion with injection



Alternate trigger mode with independent regular conversions

The alternate trigger mode is selected by programming DUAL[4:0] bits to 0b01001.

This mode can only be started on an injected group. The source of the external trigger comes from the multiplexer injected group of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN[1:0] must not be 0x0.

Injected discontinuous mode disabled (JDISCEN = 0)

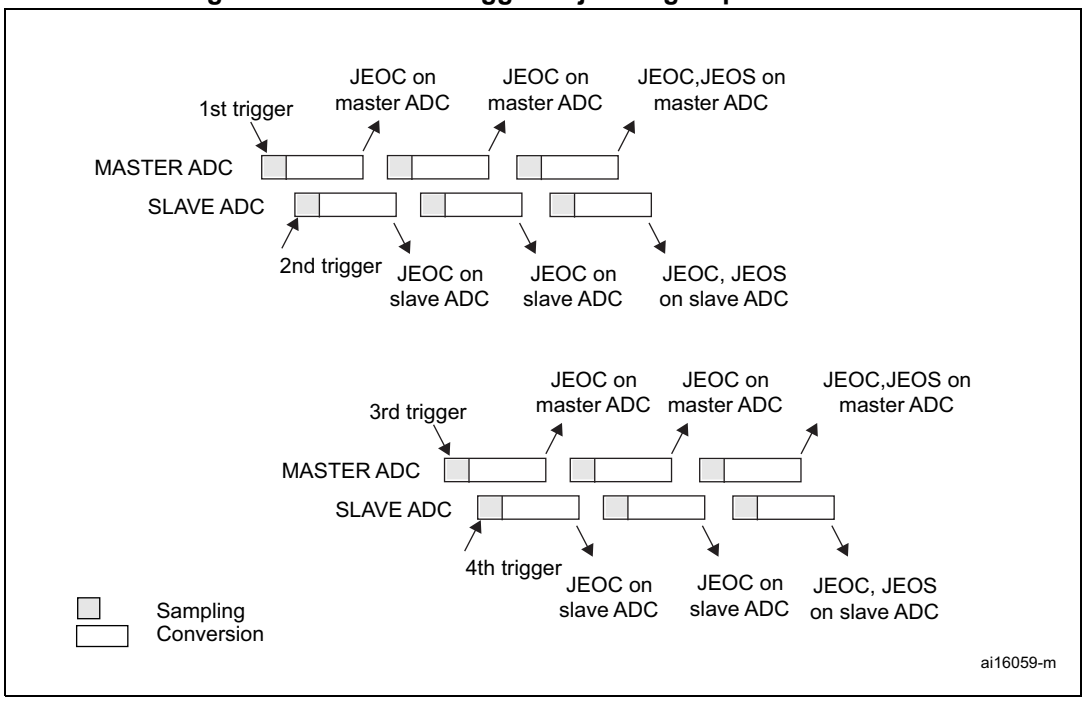
1. When the first trigger occurs, all injected master ADC channels in the group are converted.
2. When the second trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 131. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case, the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between two trigger events must be greater than or equal to one ADC clock period. The minimum time interval between two trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN = 1)

If the injected discontinuous mode is enabled for both master and slave ADCs:

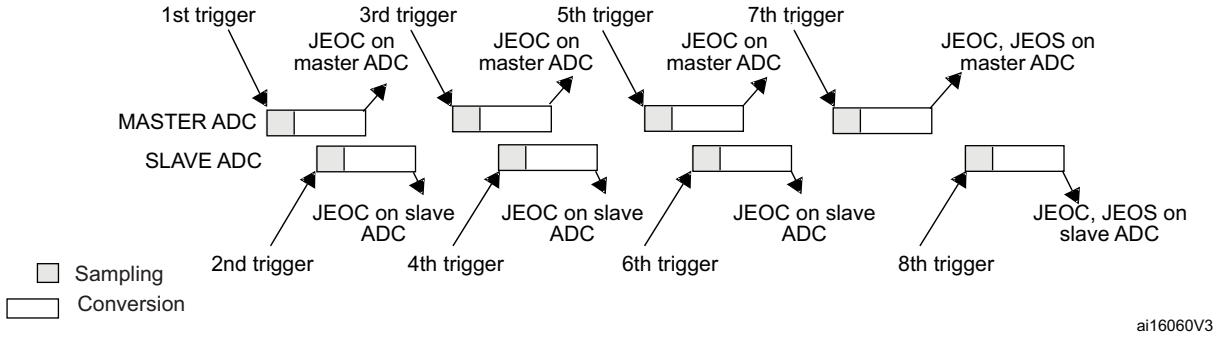
1. When the first trigger occurs, the first injected channel of the master ADC is converted.
2. When the second trigger occurs, the first injected channel of the slave ADC is converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 132. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming DUAL[4:0] bits to 0b00001.

The simultaneous conversion of a regular group can be interrupted to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, the user software must convert sequences with the same length.

Combined regular simultaneous and alternate trigger mode

This mode is selected by programming DUAL[4:0] bits to 0b00010.

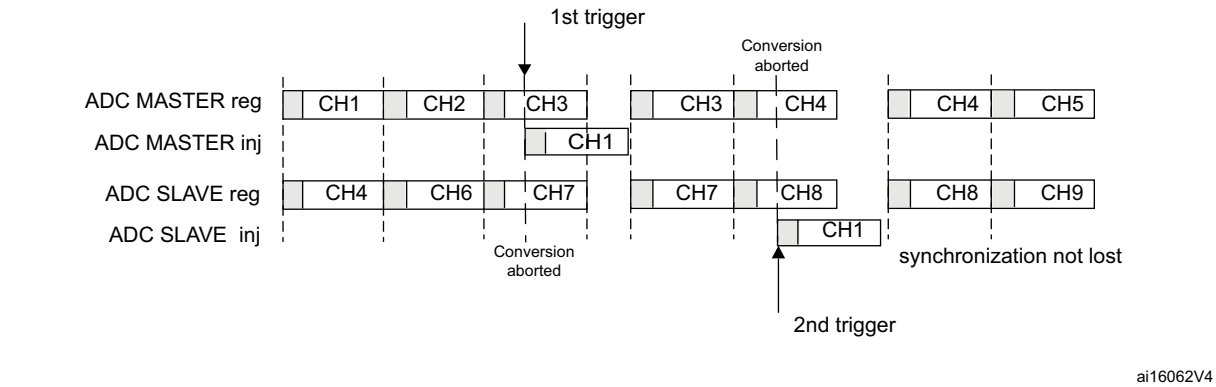
The simultaneous conversion of a regular group can be interrupted to start the alternate trigger conversion of an injected group. *Figure 133* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion immediately starts after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: In combined regular simultaneous + alternate trigger mode, the user software must convert sequences with the same length.

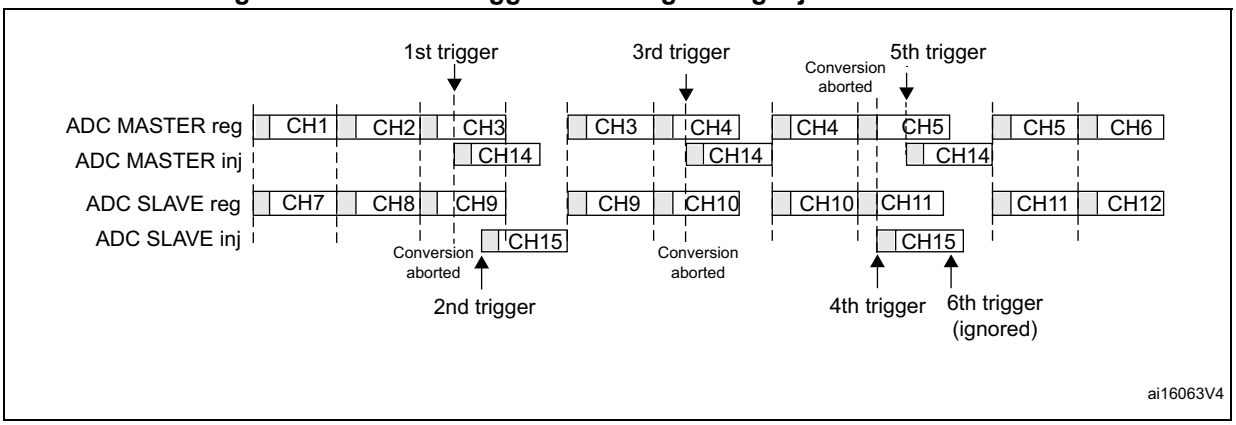
The software trigger is not supported for injected conversions.

Figure 133. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 134* shows the behavior in this case (note that the sixth trigger is ignored because the associated alternate conversion is not complete).

Figure 134. Case of trigger occurring during injected conversion



Combined injected simultaneous and interleaved mode

This mode is selected by programming DUAL[4:0] bits to 0b00011.

An interleaved conversion can be interrupted with a simultaneous injected event.

In this case, the interleaved conversion is immediately interrupted and the simultaneous injected conversion starts. At the end of the injected sequence, the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master's one. *Figure 135*, *Figure 136*, and *Figure 137* show examples of this behavior.

Note: In this mode, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 135. Interleaved single channel CH0 with injected sequence CH11, CH12

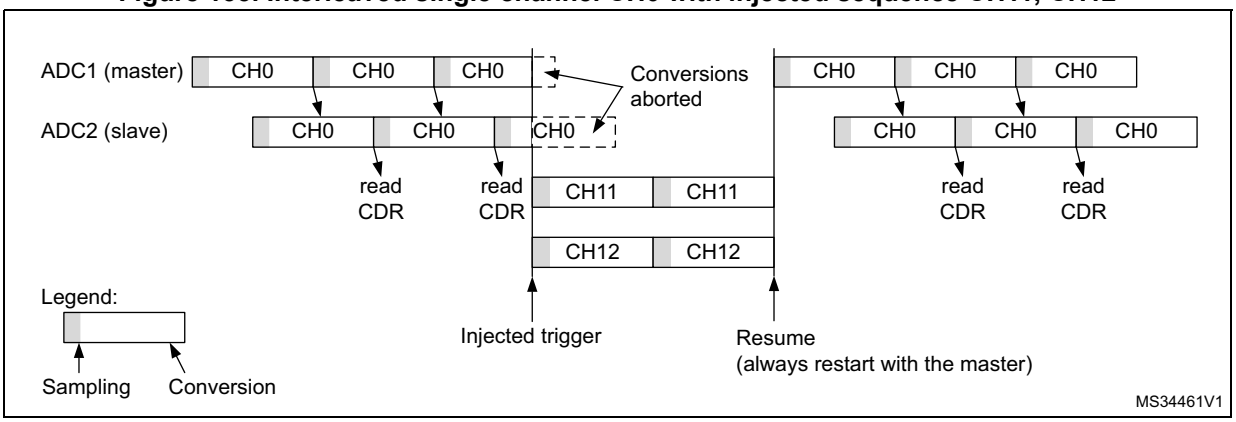


Figure 136. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

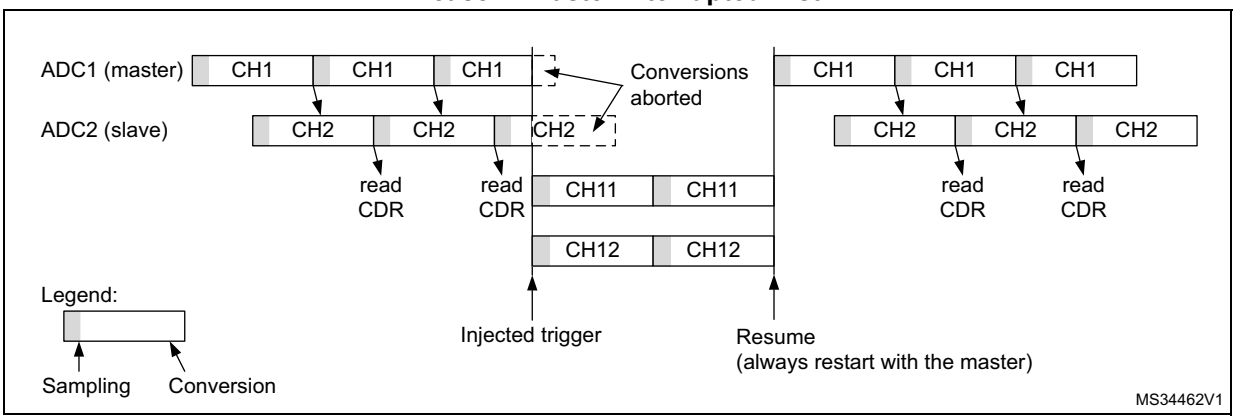
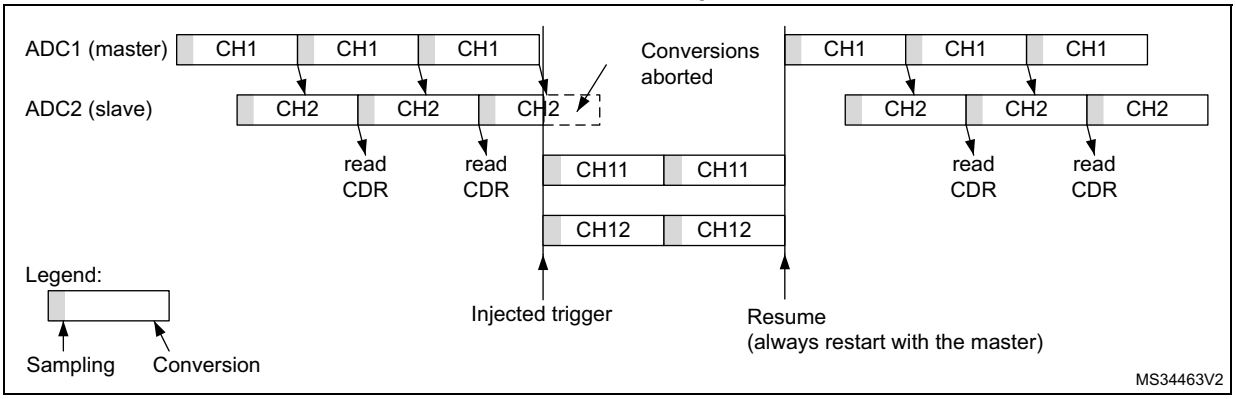


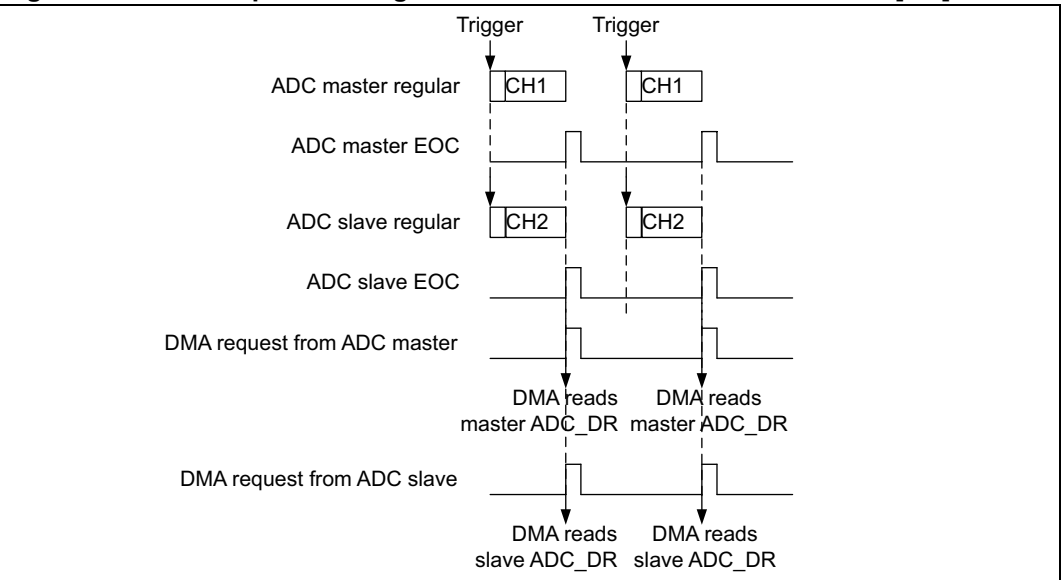
Figure 137. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 138: DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00*).

Figure 138. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00



Configuration where each sequence contains only one conversion. Master and slave timings are equal.

MSv31032V3

In interleaved mode or in simultaneous regular mode, it is also possible to save one DMA channel and transfer both data using a single DMA channel. To do this, the DAMDF[1:0] bits must be configured in the ADCC_CCR register:

- **DAMDF[1:0] = 0b10, 32-bit format:**

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, the data are alternatively available in the 32-bit register ADCC_CDR2 32-bit register.

  This mode is used when the data width is above 16 bits (conversion data width can exceed 16 bits when functions such as oversampling, compensation and left-bit shift, are used).

  Behavior: A DMA request is generated each time a new 32-bit data is available:

  – First DMA request: ADCC_CDR2[31:0] = MST_ADC_DR[31:0]

  – Second DMA request: ADCC_CDR2[31:0] = SLV_ADC_DR[31:0]

- **DAMDF[1:0] = 0b10, 16-bit format**:

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCC_CDR contains the two half-words representing two ADC-converted data. The slave ADC data takes the upper half-word and the master ADC data takes the lower half-word. DMA transfers two data by one request.

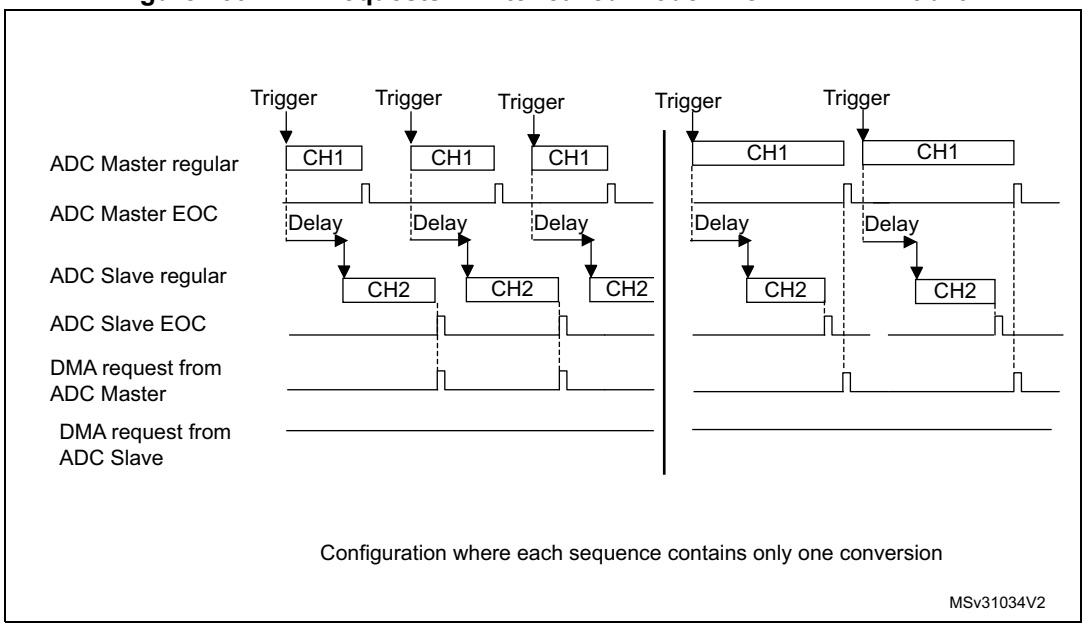
Any value above 16-bit in the master or the slave-converted data is truncated to the least 16 significant bits.

Behavior: A DMA request is generated each time a new 32-bit data is available:

– First DMA request:
ADCC_CDR[31:0] = (SLV_ADC_DR[15:0] << 16) | MST_ADC_DR[15:0]

– Second DMA request:
ADCC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 139. DMA requests in interleaved mode when DAMDF = 0b10



Note: When using dual ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion data is available.

- **DAMDF[1:0] = 0b11**:

  This mode is similar to the DAMDF[1:0] = 0b10 configuration. The only difference is that on each DMA request (two data are available), two bytes representing two ADC converted data are transferred as a half-word.

  This mode is used when the resolution is 6 bits or 8 bits and data are not signed.

  Behavior: A DMA request is generated each time two data items are available:

  – First DMA request:
  ADCC_CDR[15:0] = (SLV_ADC_DR[7:0] << 8)| MST_ADC_DR[7:0]

  – Second DMA request:
  ADCC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Note: DMA burst mode is not supported in ADC dual mode.

Overrun detection when OVRMOD = 0

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration). The EOC bit

corresponding to one ADC might remains set because the data register of this ADC contains valid data.

Overrun detection when OVRMOD = 1

In dual ADC mode, if an overrun occurs on either ADC, the data register is overwritten, causing continuous DMA requests. This results in loss of data coherency.

DMA one-shot mode/ DMA circular mode when multi-ADC mode is selected

When DAMDF mode is selected (0b10 or 0b11), DMNGT[1:0] bits of the ADCC_CCR register must also be configured to select between DMA one-shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*

Stopping the conversions in dual ADC modes

In dual ADC mode, the user software must set the ADSTP/JADSTP control bits in the master ADC to stop the conversions of both ADC. The other ADSTP control bit on the slave ADC has no effect in dual ADC mode.

Once both ADCs are effectively stopped, the ADSTART/JADSTART bits on the master and slave ADCs are both deasserted by hardware.

Analog watchdog in dual mode

The analog watchdog is supported in all dual modes.

Disabling the ADC in dual mode

During a dual-mode operation, the master ADC shares the ADC clock with the slave ADC. To disable the ADC correctly through the ADDIS bit of ADC_CR, set the ADSTP (or JADSP) bit on the master ADC. Once ADSTP (or JADSTP) becomes 0 both for the master and slave ADCs, set the ADDIS bit on the master ADC, then set the ADDIS bit on the slave ADC.

Autoinjection mode in dual mode

Autoinjection mode is not supported when DUAL[4:0] equals 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, or 0b01001.

Additionally, parallel injected oversampling mode is not supported in autoinjection mode.

Dual ADC modes supported in oversampling mode

It is possible to have oversampling enabled when working in dual ADC configuration. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

21.4.30 Temperature sensor

The temperature sensor can be used to measure the junction temperature (T_J) of the device.

The temperature sensor is internally connected to the ADC input channels, which are used to convert the sensor output voltage to a digital value (see *Table: ADC interconnection* in *Section 21.4.2: ADC pins and internal signals* for more details). When not in use, the sensor

can be put in power-down mode. It supports the temperature range specified in the product datasheet.

Figure 140 shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

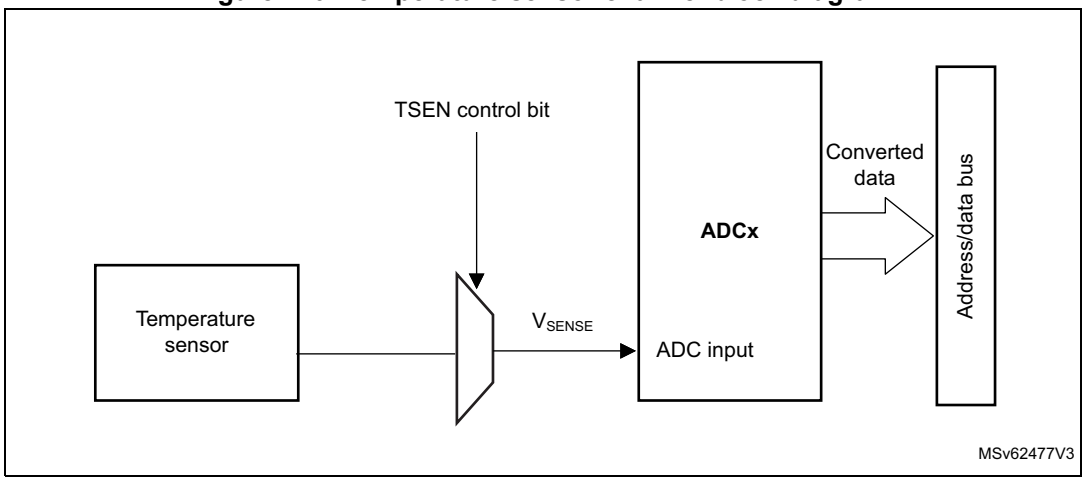
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel, which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 140 shows the block diagram of the temperature sensor.

Figure 140. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADC input channel that is connected to V_{SENSE}.
2. Program with the appropriate sampling time (refer to the electrical characteristics section of the device datasheet).
3. Set the TSEN bit in the ADCC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.

6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Caution: The above formula is valid only if the V_{REF+} condition specified in the table *Temperature sensor calibration values* of the datasheet are respected.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN, and TSEN bits must be set simultaneously.

21.4.31 Monitoring the internal voltage reference

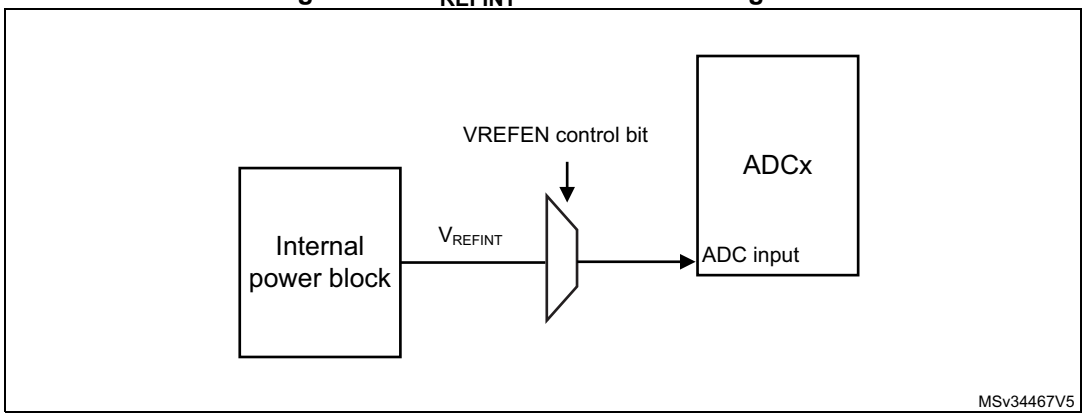
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to the *ADC interconnection* table in *Section 21.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 141 shows the block diagram of the V_{REFINT} sensing feature.

Figure 141. V_{REFINT} channel block diagram



1. The VREFEN bit of the ADCC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value.
- VREFINT_DATA is the actual VREFINT output value converted by ADC.

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For some applications, the V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

For applications where V_{REF+} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value.
- ADC_DATA is the value measured by the ADC on channel x (right-aligned).
- VREFINT_DATA is the actual VREFINT output value converted by the ADC.
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, we obtain $2^{12} - 1 = 4095$, or $2^8 - 1 = 255$ with 8-bit resolution.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

21.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (ADRDY flag)
- On the end of any conversion for regular groups (EOC flag)
- On the end of a sequence of conversion for regular groups (EOS flag)
- On the end of any conversion for injected groups (JEOC flag)
- On the end of a sequence of conversion for injected groups (JEOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2 and AWD3 flags)
- When the end of the sampling phase occurs (EOSMP flag)
- When the data overrun occurs (OVR flag)
- When the ADC internal voltage regulator output is ready (LDORDY flag)

Separate interrupt enable bits are available for flexibility.

Note: Interrupts are generated by the AHB clock domain. Since the ADC clock is adc_ker_ck, interrupt flags are delayed due to the synchronization between adc_ker_ck and AHB clock domains.

Table 152. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 flag is set	AWD1	AWD1IE			
	Analog watchdog 2 flag is set	AWD2	AWD2IE			
	Analog watchdog 3 flag is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			
	ADC internal voltage regulator ready	LDORDY	LDORDYIE			

21.6 ADC registers (for each ADC)

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

21.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY	Res.	Res.	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
			rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: ADC internal voltage regulator output ready flag

This bit is set by hardware and cleared by software. It indicates that the ADC internal voltage regulator output is ready and that the ADC can be enabled or calibrated.

0: ADC LDO internal voltage regulator disabled

1: ADC LDO internal voltage regulator enabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD3LTR and ADC_AWD3HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD2LTR and ADC_AWD2HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD1LTR and ADC_AWD1HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it or when ADEN = 0.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the corresponding ADC_JDRy register.

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel (a new conversion has completed while the EOC flag was already set) or when adc_hclk clock is too slow to manage the data.

It is cleared by software writing 1 to it or when ADEN = 0.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it or when ADEN = 0.

0: Regular conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the ADC_DR register.

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase. It is cleared by software writing 1 or when ADEN = 0.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it or when ADEN = 0.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

21.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDYIE	Res.	Res.	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDYIE**: ADC internal voltage regulator interrupt enable

This bit is set and cleared by software to enable/disable the ADC internal voltage regulator interrupt.

0: Internal voltage regulator interrupt disabled

1: Internal voltage regulator interrupt enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

21.6.3 ADC control register (ADC_CR)

Address offset: 0x008

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs		rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0 and ADEN =0).

Bit 28 **ADVREGEN**: ADC internal voltage regulator enable

This bits is set by software to enable the ADC internal voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC internal voltage regulator disabled

1: ADC internal voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 21.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be reconfigured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

When a software trigger is used, JADSTART bit is cleared by hardware, but JADSTP bit must be programmed to reconfigure the ADC.

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP command).

It is cleared by hardware when the conversion is effectively discarded. and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

When a software trigger is used, ADSTART bit is cleared by hardware. However it is necessary to write ADSTP bit to reconfigure the ADC.

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop ongoing regular conversions. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use ADSTP).

Note: In dual ADC regular simultaneous and interleaved modes, the ADSTP bit of the master ADC must be used to stop regular conversions on both ADCs. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (JEXTSEL = 0x0 and JDISEN = 1), this bit is cleared by hardware immediately after injected conversion starts.
- In all cases, after the execution of the JADSTP command, this bit and JADSTP are cleared by hardware simultaneously.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1, ADDIS = 0 and JAUTO = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared).

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (EXTSEL = 0x0 and DISEN = 1), this bit is cleared by hardware immediately after regular conversion starts.
- In all cases, after the execution of the ADSTP command, this bit and ADSTP are cleared by hardware simultaneously.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and place it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1, ADDIS = 0 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all the bits of ADC_CR registers are cleared (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0), except for ADVREGEN bit which must be set. In addition, the software must wait for the startup time of the voltage regulator.

21.6.4 ADC configuration register (ADC_CFGR1)

Address offset: 0x00C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM[2:0]			DISCEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

01101: ADC analog input channel 13 monitored by AWD1

Others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers. The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the JAUTO bit of the slave ADC is not valid and controlled by the JAUTO bit of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 Reserved, must be kept at reset value.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: When dual mode is enabled (DUAL bits of ADCC_CCR register are not equal to zero), the JDISCEN bit of the slave ADC is not valid and controlled by the JDISCEN bit of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCNUM[2:0] bits of the slave ADC are not valid and controlled by the DISCNUM[2:0] bits of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCEN bit of the slave ADC is not valid and controlled by the DISCEN bit of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the autodelayed conversion mode.

0: Autodelayed conversion mode off

1: Autodelayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the AUTDLY bit of the slave ADC is not valid and controlled by the AUTDLY bit of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the CONT bit of the slave ADC is not valid and controlled by the CONT bit of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0

00001: adc_ext_trg1

00010: adc_ext_trg2

00011: adc_ext_trg3

00100: adc_ext_trg4

00101: adc_ext_trg5

00110: adc_ext_trg6

00111: adc_ext_trg7

...

11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit

01: 10-bit

10: 8-bit

11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data management configuration

These bits are set and cleared by software to select how the ADC interface output data are managed.

00: Regular conversion data stored in DR only

01: DMA one-shot mode selected

10: Reserved

11: DMA circular mode selected

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

21.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	Res.	ROVSE
rw	rw	rw			rw	rw	rw	rw	rw	rw					rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OVSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled
1: Sampling time control trigger mode enabled

If EXTEN[1:0] bits are set to 01, the sampling time starts on the trigger rising edge, and the conversion starts on the trigger falling edge.

The SMPTRIG bit must not be set when the BULB bit is set.

When EXTEN[1:0] bits are set to 00, set SWTRIG to start the sampling.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with sampling time control trigger mode*.*

Bit 14 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to trigger the conversion in sampling time control trigger mode.

0: Software trigger starts the conversion for sampling time control trigger mode
1: Software trigger starts the sampling for *sampling time control trigger mode*

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing), SMPTRIG = 1, and EXTEN[0:1] = 00.

Bit 13 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

BULB bit must not be set when the SWTRIG bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with bulb mode.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing). It is recommended to clear both JOVSE and GCOMP when ROVSM = 0.

Bit 9 **TROVS**: Triggered regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift
0001: 1-bit shift
0010: 2-bit shift
0011: 3-bit shift
0100: 4-bit shift
0101: 5-bit shift
0110: 6-bit shift
0111: 7-bit shift
1000: 8-bit shift
1001: 9-bit shift
1010: 10-bit shift
Other: reserved, must not be used

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **ROVSE**: Regular oversampling enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular oversampling disabled
1: Regular oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

21.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 3 ADC clock cycles
001: 5 ADC clock cycles
010: 8 ADC clock cycles
011: 13 ADC clock cycles
100: 25 ADC clock cycles
101: 48 ADC clock cycles
110: 139 ADC clock cycles
111: 289 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

21.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **SMPx[2:0]:** Channel x sampling time selection (x = 13 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 3 ADC clock cycles
001: 5 ADC clock cycles
010: 8 ADC clock cycles
011: 13 ADC clock cycles
100: 25 ADC clock cycles
101: 48 ADC clock cycles
110: 139 ADC clock cycles
111: 289 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

21.6.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PCSEL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **PCSEL[13:0]**: Channel i ($V_{IN[i]}$) preselection

These bits are written by software to preselect the input channel I/O to be converted.

0: Input channel i ($V_{IN[i]}$) is not preselected for conversion, the result of the ADC conversion for this channel is wrong

1: Input channel i ($V_{IN[i]}$) is preselected for conversion

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Configuring the PCSEL bit is not necessary for the internal channels (such as VREFINT).

21.6.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 13) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 13) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 13) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 13) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **LEN[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 13) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 8th in the regular conversion sequence
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 7th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 6th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 5th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 14th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 13th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 12th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 11th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence
These bits are written by software with the channel number (0 to 13) assigned as the 10th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.12 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 13) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 13) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.13 ADC regular data register (ADC_DR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA[31:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 21.4.25: Data management*.

21.6.14 ADC injected sequence register (ADC_JSQR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 13) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 13) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 13) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 13) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JLEN[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

21.6.15 ADC offset y configuration register (ADC_OFCFGRy)

Address offset: 0x050 + 0x004 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET_CH[4:0]					SSAT	USAT	POS OFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:27 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed in bits OFFSET[21:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the data offset y.

Bit 26 **SSAT**: Signed saturation enable

This bit is set and cleared by software to enable the signed saturation feature.(see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*).

0: Offset is subtracted maintaining the data integrity and extending converted data size (13-bit signed format)

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **USAT**: Unsigned saturation enable

This bit is set and cleared by software to enable the unsigned saturation feature.

0: Offset is subtracted maintaining the data integrity

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **POSOFF**: Positive offset enable

This bit is set and cleared by software to enable the positive offset

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:0 Reserved, must be kept at reset value.

21.6.16 ADC offset y register (ADC_OFRy)

Address offset: 0x060 + 0x004 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:16]					
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **OFFSET[21:0]:** Data offset y for the channel programmed in OFFSETy_CH[4:0] bits

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the OFFSETy_CH[4:0] bits. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

21.6.17 ADC gain compensation register (ADC_GCOMP)

Address offset: 0x070

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GCOMPCOEFF[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GCOMP:** Gain compensation mode

This bit is set and cleared by software to enable the Gain compensation mode.

0: Regular ADC operation mode

1: Gain compensation enabled and applied on all channels.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing)

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]:** Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.

00 1000 0000 0000: gain factor of 0.5

...

01 0000 0000 0000: gain factor of 1

10 0000 0000 0000: gain factor of 2

11 0000 0000 0000: gain factor of 3

...

The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.9999756

Note: This gain compensation is only applied when GCOMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing).

21.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x080 + 0x004 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left- or right-aligned as described in *Section 21.4.25: Data management*.

21.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AWDCH[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **AWDCH[13:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWDCH[13:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

21.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AWDCH[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **AWDCH[13:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWDCH[13:0] = 000..0, the analog Watchdog 3 is disabled.

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

21.6.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 1 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 1.

Refer to *Section 21.4.27: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)*.

21.6.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR)

Address offset: 0x0AC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWDFILT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.

000: No filtering, one detection generates an AWD1 flag or an interrupt

001: two consecutive detections generate an AWD1 flag or an interrupt

...

111: Eight consecutive detections generate an AWD1 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 1 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 1. Refer to *Analog windows watchdog* section.

21.6.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR)

Address offset: 0x0B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 2 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

21.6.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR)

Address offset: 0x0B4

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 2 higher threshold
These bits are set and cleared by software to define the higher threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

21.6.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR)

Address offset: 0x0B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 3 lower threshold
These bits are set and cleared by software to define the lower threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

21.6.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR)

Address offset: 0x0BC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 3 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

21.6.27 ADC calibration factors (ADC_CALFACT)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT[6:0]**: Calibration factors

These bits can be written by hardware or by software.

Once the calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is applied once a new conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no conversion is ongoing).

21.6.28 ADC option register (ADC_OR)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

21.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs.

21.7.1 ADC common status register (ADCC_CSR)

Address offset: 0x000

Reset value: 0x0000 0000

This register provides an image of the flags of the different ADCs. Nevertheless it is read-only and does not allow to clear the different flags. Instead each flag must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LDORDY_SLV	Res.	Res.	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
			r			r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY_MST	Res.	Res.	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
			r			r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **LDORDY_SLV:** ADC internal voltage regulator flag of the slave ADC
This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY_MST:** ADC internal voltage regulator flag of the master ADC
This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

21.7.2 ADC common control register (ADCC_CCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSEN	VREFEN	Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw			rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TSEN**: Temperature sensor voltage enable

This bit is set and cleared by software to control V_{SENSE} channel.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Note: The temperature sensor is not available on all ADC instances. Refer to Section 21.4.4: ADC connectivity for details.

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Note: V_{REFINT} is not available on all ADC instances. Refer to Section 21.4.4: ADC connectivity for details.

Note: The software is allowed to write this bit only when ADEN = 0.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:14 **DAMDF[1:0]:** Dual ADC mode data format

This bitfield are set and cleared by software. It specifies the data format in the common data register ADCC_CDR and ADCC_CDR2.

00: Dual ADC mode without data packing (ADCC_CDR and ADCC_CDR2 registers not used).

01: Reserved

10: Data formatting mode for any data width (ADCC_CDR data register is used when the data width is less than 16 bits, otherwise ADCC_CDR2 register is used)

11: Data formatting mode for data width lower that 8 bits (ADCC_CDR data register is used)

Note: The software is allowed to write these bits only when ADEN = 0 (ADC is disabled).

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between two sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.
All the ADCs independent:
00000: Independent mode
The following settings apply to dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + Injected simultaneous mode
00010: Combined regular simultaneous + Alternate trigger mode
00011: Combined interleaved mode + Injected simultaneous mode
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
Others: Reserved
All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

21.7.3 ADC common regular data register for dual mode (ADCC_CDR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_SLV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_MST[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 21.4.29: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 21.4.29: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

21.7.4 ADC common regular data register for dual mode (ADCC_CDR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_ALT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_ALT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA_ALT[31:0]**: Regular data of the master/slave alternated ADCs.

In dual mode, these bits contain the regular 32-bit data of the master and the slave ADC. Refer to *Section 21.4.29: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

21.8 ADC register map

ADC1 and ADC3 are master ADCs.

ADC2 operates as a slave from ADC1.

Table 153. ADC register map and reset values for each ADC

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	Res.	Res.	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				0			0	0	0	0	0	0	0	0	0	0

Table 153. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDYIE	Res.	Res.	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																				0			0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR	ADCAL		DEEPPWD	ADVREGEN			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0		1	0																							0	0	0	0	0	0
0x00C	ADC_CFGR1	Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM[2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	Reset value	0x8000 0000																															
0x010	ADC_CFGR2	LSHIFT[3:0]				Res.	Res.	OVSR[9:0]										SMPTRIG	SWTRIG	BULB	Res.	Res.	ROMVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	Res.	ROVSE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0					0
0x014	ADC_SMPR1	Res.	Res.	SMP9[2:0]				SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ADC_SMPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ADC_PCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x02C	Reserved																																
0x030	ADC_SQR1	Res.	Res.	Res.	SQ4[4:0]				Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x034	ADC_SQR2	Res.	Res.	Res.	SQ9[4:0]				Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	SQ5[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x038	ADC_SQR3	Res.	Res.	Res.	SQ14[4:0]				Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	SQ10[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x03C	ADC_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x040	ADC_DR	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044-0x048	Reserved																																
0x04C	ADC_JSQR	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]				Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 153. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	ADC_OFCFGR1			OFFSET1_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x054	ADC_OFCFGR2			OFFSET2_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x058	ADC_OFCFGR3			OFFSET3_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x05C	ADC_OFCFGR4			OFFSET4_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x060	ADC_OFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	ADC_OFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	ADC_OFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	ADC_OFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	ADC_GCOMP	GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMPCOEFF[13:0]													
	Reset value	0																		0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x074-0x07C	Reserved																																
0x080	ADC_JDR1	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	ADC_JDR2	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	ADC_JDR3	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	ADC_JDR4	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090-0x09C	Reserved																																

Table 153. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	ADC_AWD1LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	ADC_AWD1HTR	AWDFILT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B0	ADC_AWD2LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	ADC_AWD2HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B8	ADC_AWD3LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	ADC_AWD3HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C0	Reserved																																
0x0C4	ADC_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved																																
0x0D0	ADC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0D4-0x0FC	Reserved																																

Table 154. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADCC_CSR	Res.	Res.	Res.	LDORDY_SLV	Res.	Res.	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	LDORDY_MST	Res.	Res.	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
								slave ADC																master ADC									
	Reset value				0			0	0	0	0	0	0	0	0	0	0				0			0	0	0	0	0	0	0	0	0	0
0x004	Reserved	Res.																															
0x008	ADCC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	TSEN	VREFEN	Res.				Res.	Res.	DAMDF[1:0]		Res.	Res.	DELAY[3:0]			Res.	Res.	Res.	DUAL[4:0]					
	Reset value									0	0							0	0			0	0	0	0				0	0	0	0	0
0x00C	ADCC_CDR	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	ADCC_CDR2	RDATA_ALT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	Reserved																																

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

22 Digital-to-analog converter (DAC)

22.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution.

The DACx_OUTy pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the sample and hold mode.

22.2 DAC main features

The DAC main features are the following (see *Figure 142: Dual-channel DAC block diagram*):

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx_OUTy output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREF+ pin

Figure 142 shows the block diagram of a DAC channel and *Table 156* gives the pin description.

22.3 DAC implementation

Table 155. DAC features

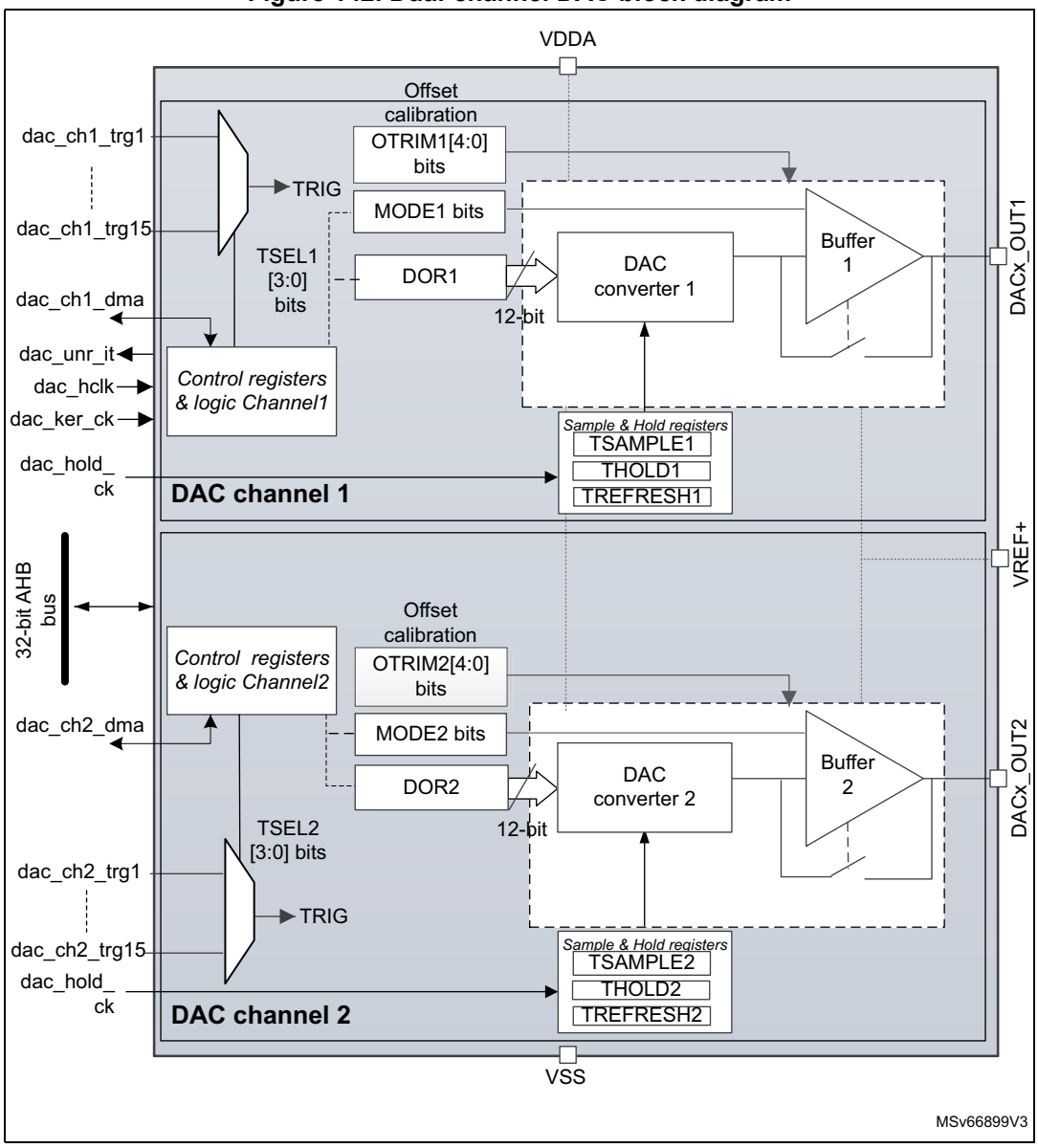
DAC features	DAC1
Dual channel	-
Output buffer	X
I/O connection	DAC1_OUT1 on PA4 DAC1_OUT2 on PA5(1)
Maximum sampling time	1 Msps
Autonomous mode	-
VREF+ pin	X

1. Only available on STM32C53x/54x devices.

22.4 DAC functional description

22.4.1 DAC block diagram

Figure 142. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.
2. Refer to *Section 22.3: DAC implementation* for channel2 availability.

22.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator and operational amplifier.
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI/LSE clock source (dac_hold_ck).

Table 156. DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog positive reference	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to the product datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

Table 157. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_ker_ck	Input	DAC kernel clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 158. DAC interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x =1, 2)	tim1_trgo	Internal signal from on-chip timer
dac_chx_trg2 (x =1, 2)	tim2_trgo	Internal signal from on-chip timer
dac_chx_trg3 (x =1, 2)	Reserved	-
dac_chx_trg4 (x =1, 2)	Reserved	-
dac_chx_trg5 (x =1, 2)	tim5_trgo(1)	Internal signal from on-chip timer
dac_chx_trg6 (x =1, 2)	tim6_trgo	Internal signal from on-chip timer
dac_chx_trg7 (x =1, 2)	tim7_trgo	Internal signal from on-chip timer
dac_chx_trg8 (x =1, 2)	tim8_trgo	Internal signal from on-chip timer
dac_chx_trg9 (x =1, 2)	tim12_trgo	Internal signal from on-chip timer
dac_chx_trg10 (x =1, 2)	tim15_trgo	Internal signal from on-chip timer
dac_chx_trg11 (x =1, 2)	lptim1_ch1	Internal signal from on-chip timer
dac_chx_trg12 (x =1, 2)	exti9	External pin
dac_chx_trg13 (x =1, 2)	Reserved	-
dac_chx_trg14 (x =1, 2)	Reserved	-
dac_chx_trg15 (x =1, 2)	Reserved	-

1. Not available on STM32C53x/54x devices.

22.4.3 DAC clocks

Two clock sources can be used to update the DAC:

- dac_hclk: DAC peripheral clock (AHB clock)
- dac_ker_ck: DAC kernel clock: this clock can be used to synchronize DAC and ADC.
- dac_hold_ck: low-power clock used in sample and hold mode

The DAC clock is selected in the RCC.

22.4.4 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

22.4.5 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

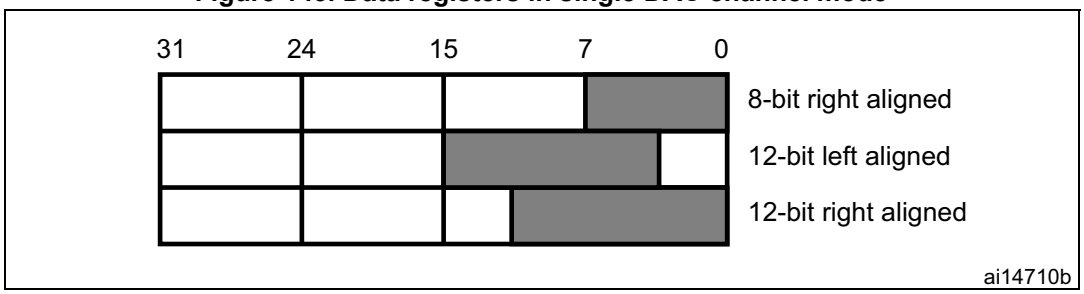
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC_DHRx (data holding registerx, which are internal non-memory-mapped registers). The DAC_DHRx register is then loaded into the DAC_DORx register either automatically, by software trigger or by an external event trigger.

Figure 143. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

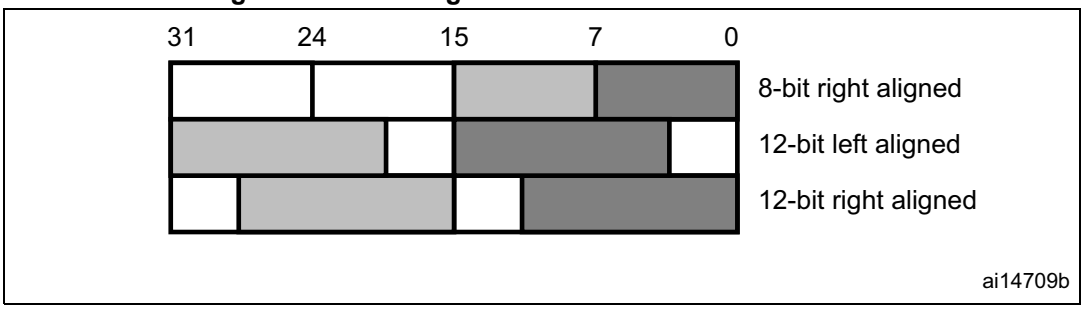
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 144. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2's complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB of the data written to DAC_DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB is inverted. The four LSBs are simply ignored.

Table 159. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DAC_DHRx register	DATA transfered to DAC_DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

22.4.6 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

To synchronize DAC and ADC, the same clock source can be used for both peripherals. This is done by selecting the dac_ker_ck clock instead of the dac_hclk clock (AHB clock) in the RCC.

HFSEL bits of DAC_MCR must be set when dac_ker_ck clock speed is faster than 80 MHz.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk or dac_ker_ck clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

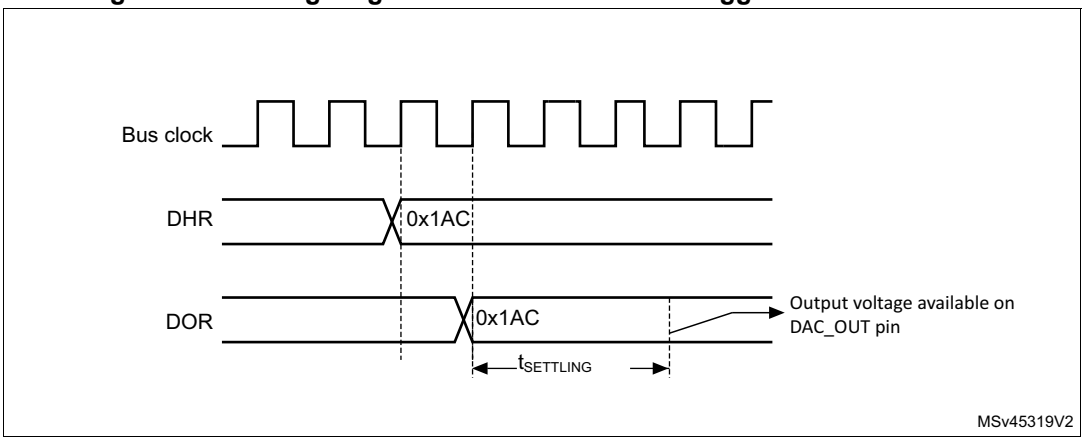
The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 160. HFSEL description

HFSEL [2:0]	Clock frequency	Latency using AHB clock (dac_hclk)	Latency using dac_ker_ck clock	Function
000	< 80 MHz	3	4	DAC_DOR update rate up to 3 AHB clock cycles or 4 dac_ker_ck cycles.
001	≥ 80 MHz(1)	5	5	DAC_DOR update rate up to 5 AHB clock or dac_ker_ck cycles.
Others	Reserved	-	-	-

1. Refer to the device datasheet for the value of the maximum dac_hclk or dac_ker_ck frequency.

Figure 145. Timing diagram for conversion with trigger disabled TEN = 0



22.4.7 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltage on the DAC channel pin is determined by the following equation:

$$\text{DAC output} = V_{REF} \times \frac{DOR}{4096}$$

where all voltages are expressed in Volt.

22.4.8 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_hclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

22.4.9 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA double data mode

When the DMA controller is used in normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA double data mode.

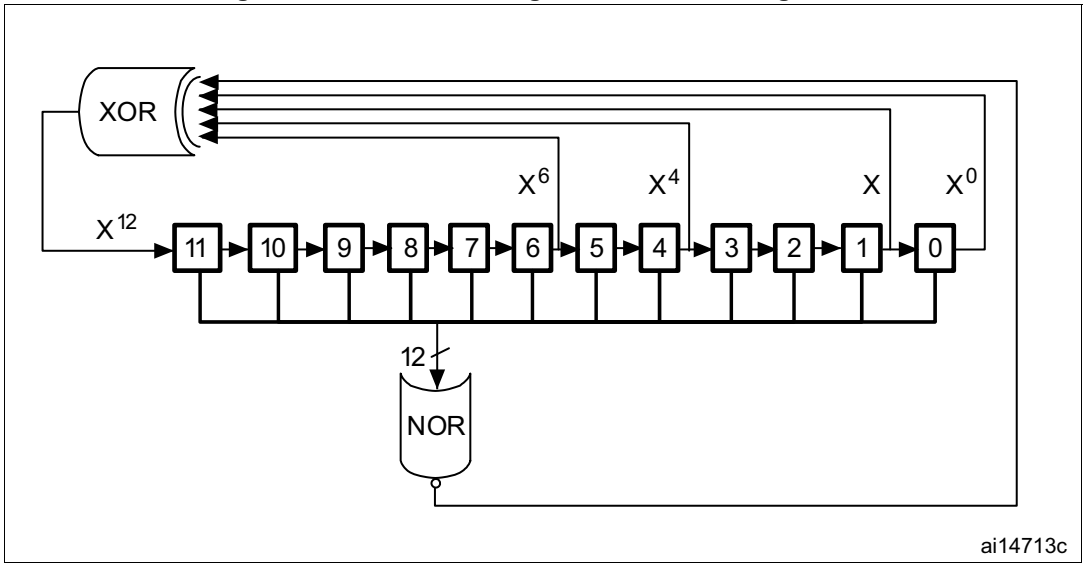
The following conditions must be met to change from double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

22.4.10 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 146. DAC LFSR register calculation algorithm

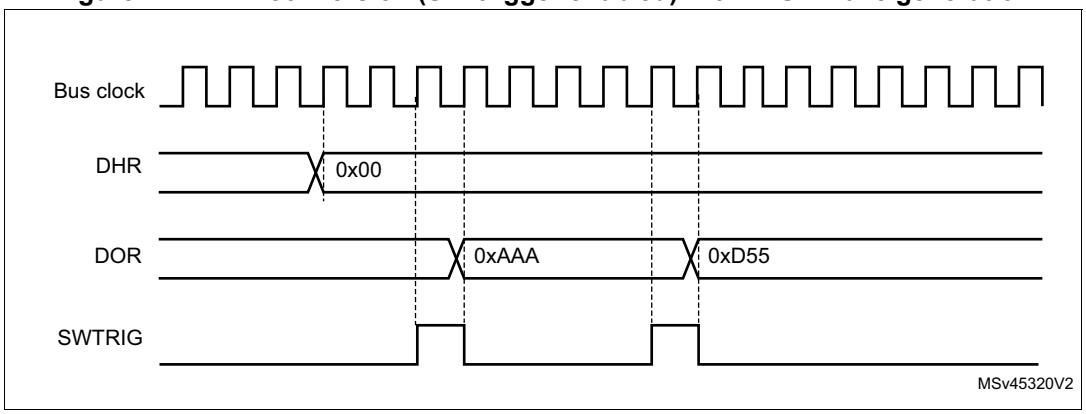


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a 1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 147. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

22.4.11 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 148. DAC triangle wave generation

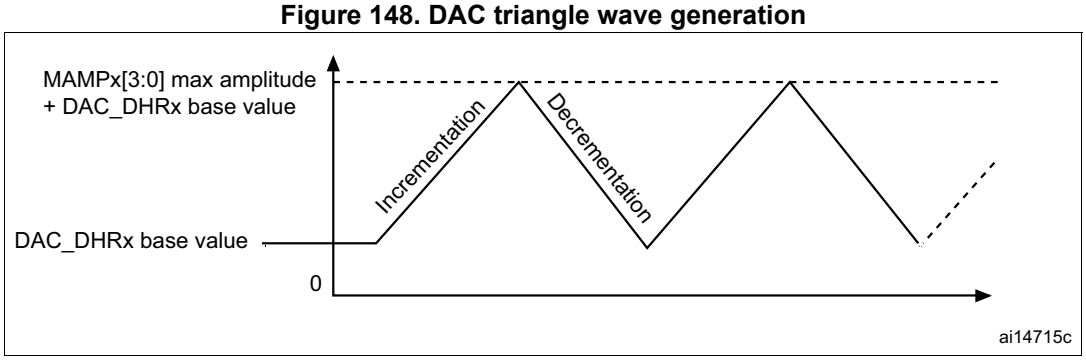
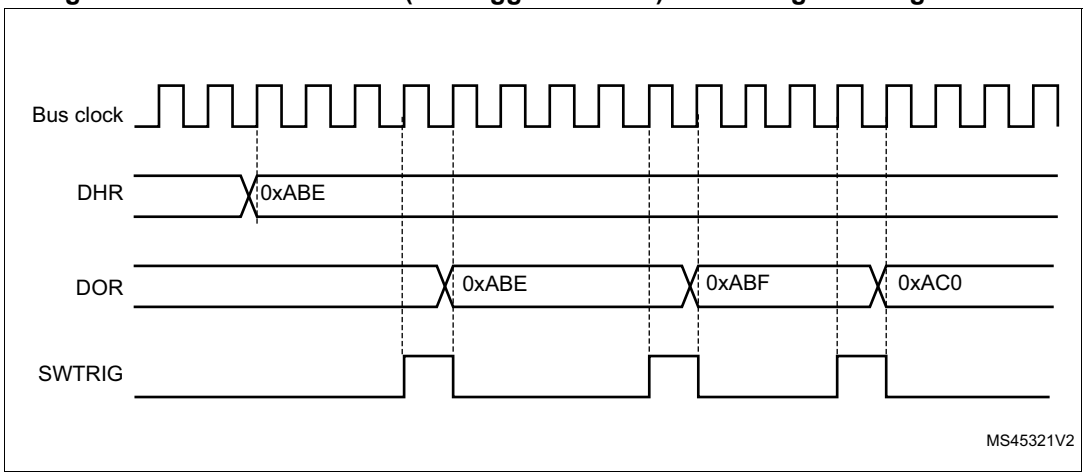


Figure 149. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

22.4.12 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI/LSE low-speed clock (dac_hold_ck) in addition to the dac_hclk clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI/LSE low-speed clock (dac_hold_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11.
- 62 cycles are required for hold phase: THOLDx[9:0] = 62.
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 161. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula, the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only (refer to the datasheet for product data):

C_{SH} = 100 nF

V_{REF+} = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs

(where N_{LSB} = 10 (10 LSB drop during the hold phase)

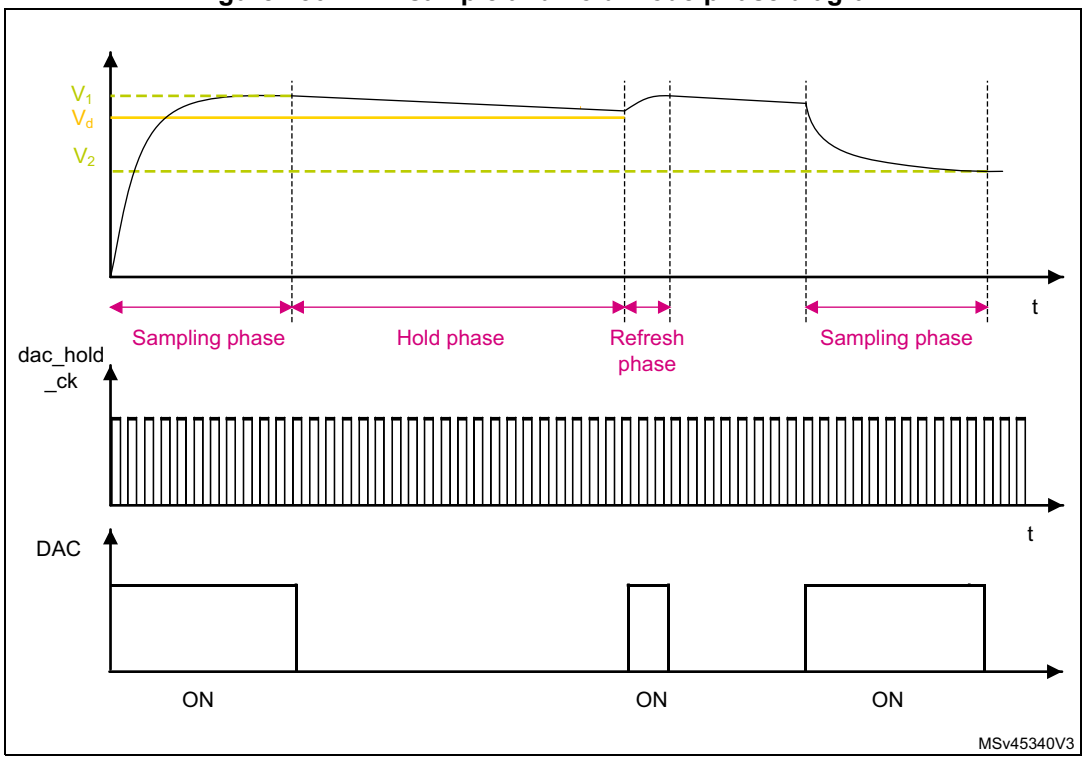
Hold phase:

$D_v = i_{leak} * t_{hold} / C_{SH}$ = 0.0073 V (10 LSB of 12bit at 3 V)

i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)

t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 150. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

  When MODEx[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

Table 162. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

22.4.13 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = \left((D / 2^N) \times G \times V_{REF}\right) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{REF} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and V_{OS} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to V_{REF+}/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, V_{REF+} values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when V_{DD} is removed (example the device enters in Standby mode) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) Until the CAL_FLAGx is read as 1 or the maximum trimming code is reached, increment OTRIMx[4:0] and repeat substeps from (b) to (d).

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, V_{REF+} and temperature conditions do not change during device operation while it enters more often in Standby mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

22.4.14 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DHRx register is transferred into DAC_DORx (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel triangle counter is then updated.

22.4.15 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

11 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DAC_DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.

2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

22.5 DAC in low-power modes

Table 163. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA.
Stop(1)	The DAC remains active with a static value. The sample and hold mode can be selected using LSE/LSI clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 22.3: DAC implementation* for information on the Stop modes supported by the DAC peripheral.

22.6 DAC interrupts

Table 164. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

22.7 DAC registers

Refer to *Section 1 on page 85* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

22.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Note: These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Reserved

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)
These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

0000: SWTRIG2
0001: dac_ch2_trg1
0010: dac_ch2_trg2
...
1111: dac_ch2_trg15

For details on trigger configuration and mapping, refer to *Section 22.4.2: DAC pins and internal signals*.

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_hclk clock cycle.
These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled

01: Noise wave generation enabled

10: Triangle wave generation enabled

11: Reserved

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
For details on trigger configuration and mapping, refer to *Section 22.4.2: DAC pins and internal signals*.

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_hclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

22.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

22.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

22.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

22.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHRB[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

22.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

22.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

22.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHRB[7:0]								DACC2DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

22.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

22.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

22.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

22.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

22.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

22.7.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

22.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

22.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL[2:0]			Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw	rw				rw	rw						rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA normal mode selected

1: DMA double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 22.3: DAC implementation.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2 = 0.

Refer to Section 22.3: DAC implementation for the availability of DAC channel2.

Bits 15:13 **HFSEL[2:0]**: High frequency interface mode selection
000: High frequency interface mode disabled
001: High frequency interface mode enabled for 80 MHz ≤ dac_ker_ck ≤ 160 MHz
Others: Reserved, must not be used.

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA normal mode selected
1: DMA double data mode selected

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
010: DAC channel1 is connected to external pin with Buffer disabled
011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
111: DAC channel1 is connected to on chip peripherals with Buffer disabled
Note: This register can be modified only when EN1 = 0.

22.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

22.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in sample and hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

22.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in sample and hold mode).
Hold time = (THOLD[9:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in sample and hold mode)
Hold time = (THOLD[9:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

22.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 22.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

22.7.21 DAC register map

Table 165 summarizes the DAC registers.

Table 165. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[3:1]			TSEL2[0]	TEN2	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:1]			TSEL1[0]	TEN1	EN1
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_DHR12L1	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_DHR12L2	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHRB[7:0]								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DAC_DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_DHR12LD	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x28	DAC_DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	DACC2DORB[11:0]												Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 165. DAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0												0	0	0	0	0											
0x38	DAC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4]	OTRIM2[3]	OTRIM2[2]	OTRIM2[1]	OTRIM2[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4]	OTRIM1[3]	OTRIM1[2]	OTRIM1[1]	OTRIM1[0]
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	DAC_MCR	Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			HFSEL[2]	HFSEL[1]	HFSEL[0]	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value							0	0						0	0	0	0	0	0				0	0						0	0	0
0x40	DAC_ SHSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	DAC_ SHSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	DAC_ SHHR	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	DAC_ SHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x50-0x54	Reserved																																
0x58-0x60	Reserved																																
0x64-0x68	Reserved																																

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

23 Comparator (COMP)

23.1 Introduction

The comparator can be used for a variety of functions including:

- Wake-up from a low-power mode triggered by an analog signal
- Analog signal conditioning
- Cycle-by-cycle current control loop when combined with a PWM output from a timer

23.2 COMP main features

- Selectable inverting analog inputs:
  - I/O pins
  - DAC channel outputs
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by the scaler (buffered voltage divider)
- I/O pins selectable as noninverting analog inputs
- Programmable hysteresis
- Programmable speed/consumption
- Mapping of outputs to I/Os
- Redirection of outputs to timer inputs for triggering:
  - Capture events
  - OCREF_CLR events (for cycle-by-cycle current control)
  - Break events for fast PWM shutdowns
- Blanking of comparator outputs
- Interrupt generation capability with wake up from Sleep and Stop modes (through the EXTI controller)
- Interrupt output through the EXTI controller
- Direct interrupt output to the CPU
- Window mode

23.3 COMP implementation

23.3.1 COMP implementation for STM32C55x/562 and STM32C59x/5A3 devices

Table 166. COMP implementation for STM32C55x/562 and STM32C59x/5A3 (1)

COMP features	COMP1
Window mode	-
Programmable power and speed	X
Programmable hysteresis	X

1. X = supported, - = not supported

Table 167. COMP implementation non-inverting input assignment for STM32C55x/562 and STM32C59x/5A3

INPSEL [2:0]	COMP1
000	COMP1_INP1
001	COMP1_INP2
010	COMP1_INP3
011	dac1_ch1
Others	Reserved

23.3.2 COMP implementation for STM32C53x/542 device

Table 168. COMP implementation for STM32C53x/542(1)

COMP features	COMP1	COMP2
Window mode	X	X
Programmable power and speed	X	X
Programmable hysteresis	X	X

1. X = supported, - = not supported

Table 169. COMP implementation non-inverting input assignment for STM32C53x/542

INPSEL [2:0]	COMP1	COMP2
000	COMP1_INP1	COMP2_INP1
001	COMP1_INP2	COMP2_INP2
100	COMP1_INP3	COMP2_INP3
101	dac1_ch1	dac1_ch2
Others	Reserved	Reserved

The following table lists the paired comparators required for window mode.

Table 170. Window paired comparators

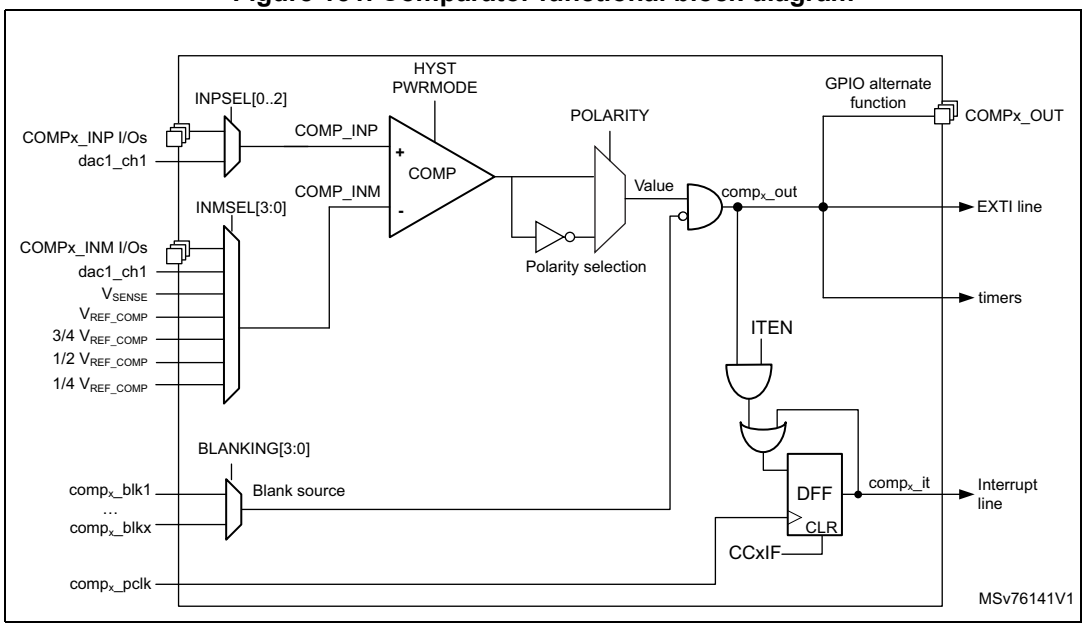
Window comparator	#1	#2
Upper threshold comparator	COMP1	COMP2
Lower threshold comparator	COMP2	COMP1

23.4 COMP functional description

23.4.1 COMP block diagram

The block diagram of the comparators is shown in the figure below.

Figure 151. Comparator functional block diagram



Note: The V_{SENSE} buffer is controlled by the ADC peripheral. To activate the buffer, set the TSEN bit in the ADCC_CCR register.

23.4.2 COMP pins and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

The comparator outputs can be connected to the I/Os through their alternate functions. Refer to the product datasheet for more information.

The outputs can also be internally redirected to a variety of timer inputs for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using ETR inputs of timers
- Input capture for timing measurements

The comparator output can be simultaneously addressed internally and to the I/O pins.

Table 171. COMP input/output internal signals

Signal name	Signal type	Description
dac1_out1	Analog input	Analog signal from DAC peripheral
compx_blkx	Digital output	See *Table 173: COMP blanking source assignment*
compx_out	Digital output	COMPx output
compx_it	Digital output	COMPx interrupt out

Note: For the COMP non-inverting input, refer to Table 167: COMP implementation non-inverting input assignment for STM32C55x/562 and STM32C59x/5A3 and Table 169: COMP implementation non-inverting input assignment for STM32C53x/542.

Table 172. COMP inverting input assignment

INMSEL [3:0]	COMP1	COMP2
0000	¼ V_{REFINT}	¼ V_{REFINT}
0001	½ V_{REFINT}	½ V_{REFINT}
0010	¾ V_{REFINT}	¾ V_{REFINT}
0011	V_{REFINT}	V_{REFINT}
0100	dac1_ch1	dac1_ch2
0101	COMP1_INM1	COMP2_INM1
0110	COMP1_INM2	COMP2_INM2
0111	COMP1_INM3	COMP2_INM3
0100	VENSE	opamp1_out

Note: Other combinations are reserved.

Table 173. COMP blanking source assignment

BLANKING[3:0]	COMP1	COMP2
0000	No blanking	No blanking
0001	tim1_oc5	tim1_oc6
0010	tim2_oc3	tim2_oc3
0011	tim5_oc3	-
0100	tim5_oc4	-
0101	tim8_oc5(1)	tim8_oc5
0110	tim15_oc2	lptim1_ch1
0111	-	lptim1_ch2

1. Not available for STM32C53x and STM32C542.

23.4.3 COMP reset and clocks

The clock comp_pclk, provided by the clock controller, synchronous with the APB clock.

Note: ***Important:*** *The polarity selection logic and the output redirection to the port work independently of the APB clock. This allows the comparator to operate even in Stop mode. The interrupt line, connected to the NVIC of the CPU, requires the APB clock (comp_pclk) to work. In absence of the APB clock, the interrupt signal comp_it cannot be generated.*

23.4.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as overcurrent or thermal protection. For applications with specific functional safety requirements, the comparator configuration can be protected against undesired alteration that could happen, for example, at program counter corruption.

For this purpose, the comparator configuration registers can be write-protected (read-only).

Upon configuring a comparator channel, its LOCK bit is set to 1. This causes the entire register set of the comparator channel to become read-only, including the LOCK bit.

The write protection can only be removed through the MCU reset.

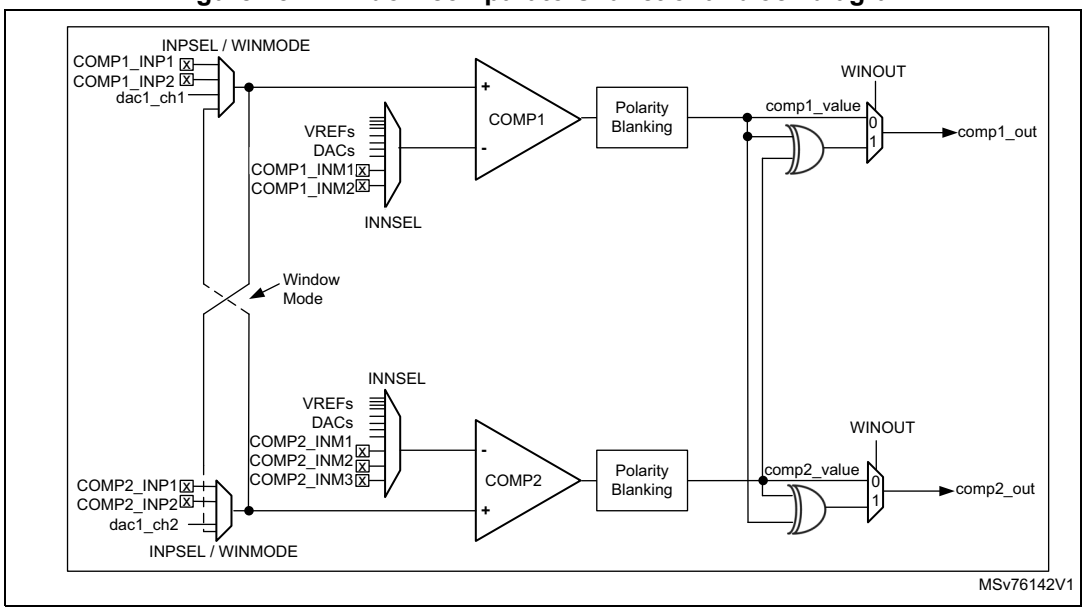
23.4.5 Window comparator (optional feature)

The purpose of the window comparator is to monitor the analog voltage to ensure it is within the voltage range defined by the lower and upper thresholds.

The two embedded comparators, COMP1 and COMP2, can be used to create a window comparator. The monitored analog voltage is connected to the noninverting (plus) inputs of comparators connected together (COMP1 or COMP2). The upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators.

The output of the window comparator (COMP1 or COMP2) is selected with the control bit WINOUT.

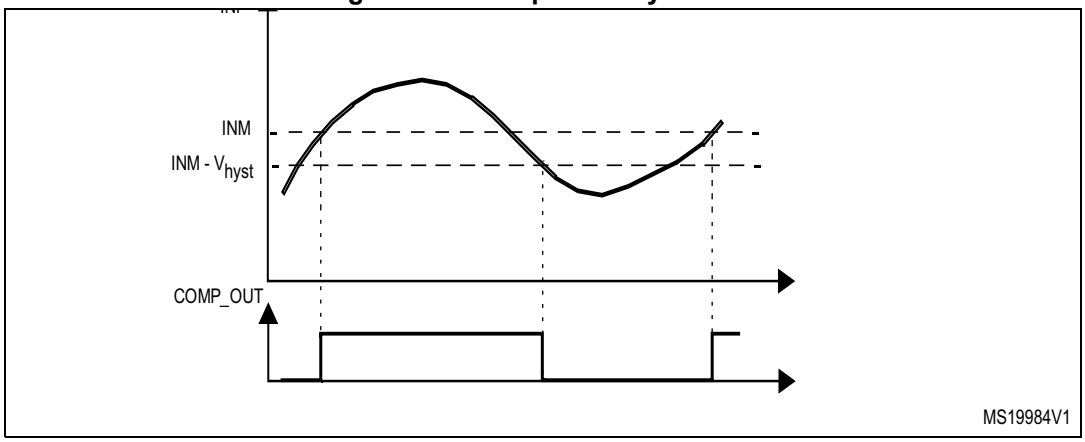
Figure 152. Window comparators functional block diagram



23.4.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

Figure 153. Comparator hysteresis

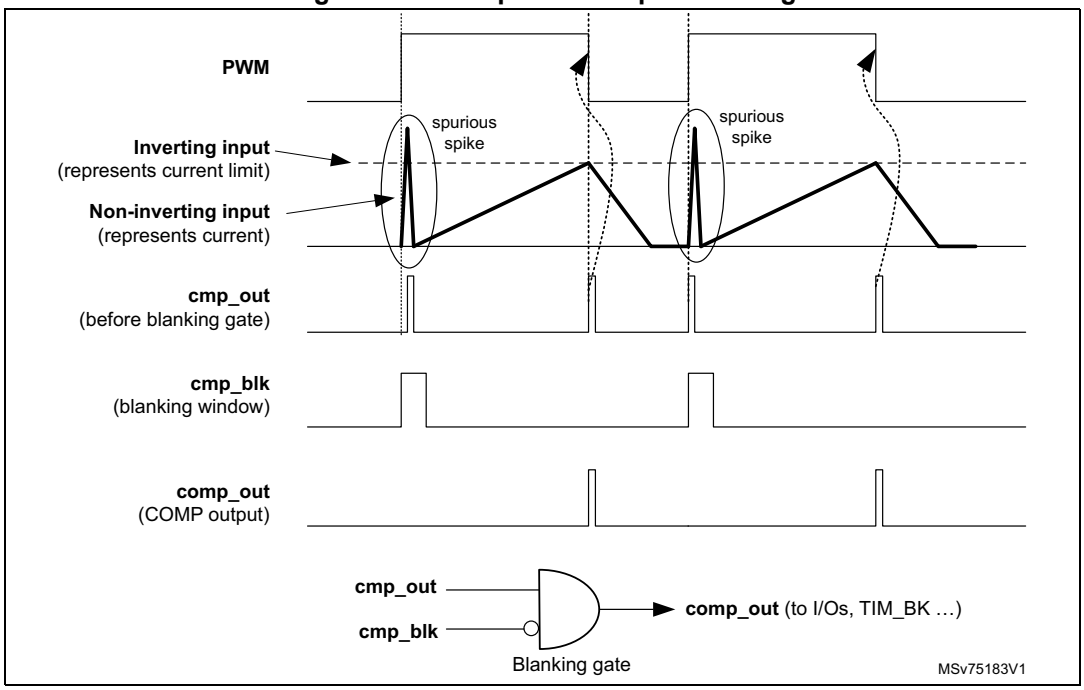


23.4.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It uses a blanking window defined with a timer output compare signal. Refer to the register description for selectable blanking signals.

The blanking signal gates the internal comparator output such as to clean the comp_out from spurious pulses due to current spikes, as depicted in the following figure (the COMP channel number is not represented).

Figure 154. Comparator output blanking



23.4.8 COMP startup stabilization time

The comparator parameters and its output is undefined after the COMP is enabled (power-on) for a period defined as startup stabilization time. See the product datasheet for more details. If the comparator output is driving any critical stage of the design, it is recommended to redirect the comparator output to the device I/O through the GPIO alternate function or through IP configuration (such as timers) after the stabilization time elapses.

23.4.9 COMP power and speed modes

The power consumption of the COMP channels versus propagation delay can be programmed with PWRMODE[1:0] bit field in COMP_CFGRx registers.

23.5 COMP low-power modes

Table 174. Comparator behavior in low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.

Note: The comparators cannot be used to exit the device from Sleep or Stop mode when the internal reference voltage is switched off.

23.6 COMP interrupts

There are two ways to use the comparator as interrupt source.

The comparator outputs are internally connected to the extended interrupt and event controller (EXTI). Each comparator has its own EXTI line and can generate either interrupts or events to make the device exit low-power modes.

The comparator also provides an interrupt line to the NVIC of the CPU. This functionality is used when the CPU is active to handle low-latency interrupt. It requires APB clock running.

23.6.1 COMP wake-up interrupt through EXTI

Refer to the *Section 16: Extended interrupts and event controller (EXTI)* for more details.

Sequence to enable the COMPx interrupt through EXTI block:

1. Enable the COMPx and wait for stabilization.
2. Configure the EXTI line, receiving the compx_wkup signal, in interrupt mode, select the rising, falling or either-edge sensitivity and enable the EXTI line.
3. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.

Table 175. COMP wake-up interrupt through EXTI control bits

Interrupt event	Event flag	Enable control bit	Exit Sleep mode	Exit Stop modes	Exit Standby mode
compx_wkup	Through EXTI	Through EXTI	Yes	Yes	N/A

23.6.2 COMP NVIC interrupt through EXTI

Refer to the *Section 16: Extended interrupts and event controller (EXTI)* and *Section 15: Nested vectored interrupt controller (NVIC)* for more details.

Sequence to enable the COMPx interrupt through the EXTI block:

1. Enable the COMP and wait for stabilization.
2. Configure the EXTI line, receiving the comp_out signal, in interrupt mode, select the rising, falling, or either-edge sensitivity and enable the EXTI line.
3. Configure and enable the IMx in EXTI_IMRx.

Table 176. COMP NVIC interrupt through EXTI control bits

Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit Sleep mode	Exit Stop modes
compx_it	RPIFx or FPIFx in EXTI_RPRx or EXTI.FPRx	RTx or FTx in EXTI.RTSRx or EXTI.FTSRx and IMx in EXTI.IMRx	RPIFx or FPIFx in EXTI_RPRx or EXTI.FPRx	Yes (with APB clock)	Yes

Note: It is mandatory to enable the APB clock to use this interrupt. If the clock is not enabled, an interrupt is not generated.

23.6.3 COMP interrupt through NVIC of the CPU

Refer to the *Section 15: Nested vectored interrupt controller (NVIC)* for more details.

Sequence to enable the COMP interrupt through NVIC of the CPU:

1. Enable the COMP and wait for stabilization.
2. Configure and enable the NVIC IRQ channel mapped to the comp_it line.
3. Configure and enable the ITEN in COMP_CFGRx.

Table 177. COMP interrupt through NVIC of the CPU control bits

Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit Sleep mode	Exit Stop modes
comp_it	C1IF in	ITEN in COMP_CFGR1	CC1IF	Yes (with APB clock)	No
	C2IF in	ITEN in COMP_CFGR2	CC2IF	Yes (with APB clock)	No

Note: It is mandatory to enable APB clock to use this interrupt. If clock is not enabled, interrupt is not generated.
The interrupt is active on level and not automatically clear by hardware. It must be cleared in the comparator interrupt handler by firmware with ITEN = 0 (COMP_CFGR1 register).
The interrupt signal comp_it is generated on comparator level, for this reason: upon interruption occurrence, the comp_it signal must be set to low as soon as possible to avoid comparator interrupt handler re-entrance. This can be done by setting ITEN = 0 or changing the comparator POLARITY bit.

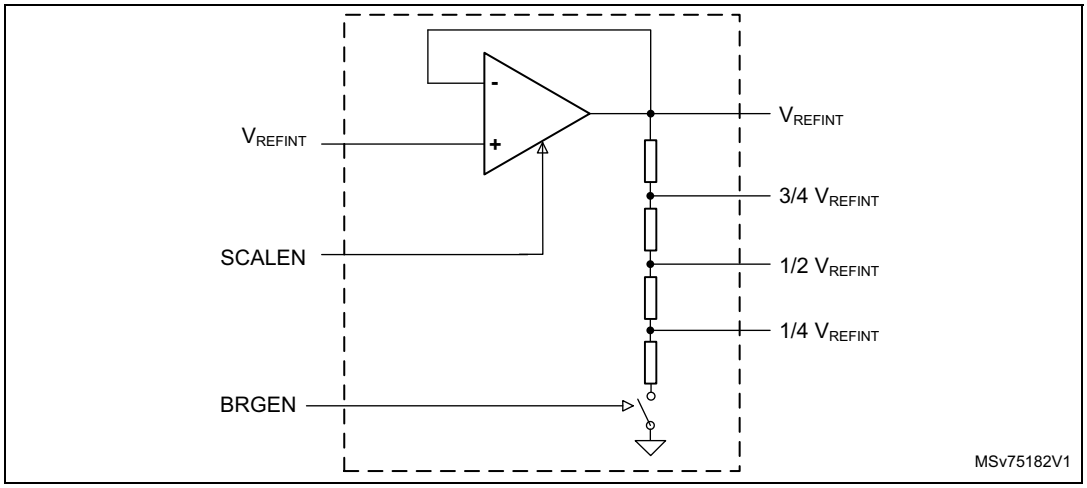
23.7 SCALER function

The scaler block is available to provide the different voltage reference levels to the comparator inputs. It is based on an amplifier driving a resistor bridge. The amplifier input is connected to the internal voltage reference V_{REFINT}.

The amplifier and the resistor bridge can be enabled separately. The amplifier is enabled by the SCALEN bits of the COMP_CFGRx registers. The resistor bridge is enabled by the BRGEN bits of the COMP_CFGRx registers.

When the resistor-divided voltage is not used, the resistor bridge disconnect to reduce the consumption.

Figure 155. Scaler block diagram



23.8 COMP registers

23.8.1 Comparator status register (COMP_SR)

The COMP_SR is the comparator status register.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2IF	C1IF
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2VAL	C1VAL
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **C2IF**: COMP2 channel interrupt flag
This bit is set by hardware when the COMP2 channel output is set.
This bit is cleared by software by writing CC12F = 1 in the COMP_ICFR register.

Bit 16 **C1IF**: COMP1 channel interrupt flag
This bit is set by hardware when the COMP1 channel output is set.
This bit is cleared by software by writing CC1IF = 1 in the COMP_ICFR register.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **C2VAL:** COMP2 channel output status bit
This bit is read-only. It reflects the current COMP1 channel output taking into account POLARITY and BLANKING bits effect.

Bit 0 **C1VAL:** COMP1 channel output status bit
This bit is read-only. It reflects the current COMP1 channel output taking into account POLARITY and BLANKING bits effect.

23.8.2 Comparator interrupt clear flag register (COMP_ICFR)

The COMP_ICFR is the comparator interrupt clear flag register.

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	CC1IF
														rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC2IF**: Clear COMP2 channel interrupt flag
Writing 1 clears the C2IF flag in the COMP_SR register.

Bit 16 **CC1IF**: Clear COMP1 channel interrupt flag
Writing 1 clears the C1IF flag in the COMP_SR register.

Bits 15:0 Reserved, must be kept at reset value.

23.8.3 Comparator configuration register 1 (COMP_CFGR1)

The COMP_CFGR1 is the COMP1 configuration register.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	BLANKING[3:0]				Res.	INPSEL[2:0]			INMSEL[3:0]			
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WINOUT	PWRMODE[1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	WINMODE	POLARITY	SCALEN	BRGEN	EN
	rw	rw	rw			rw	rw		rw		rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock
This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP channel configuration register COMP_CFGR1[31:0]
0: COMP_CFGR1[31:0] register is read/write
1: COMP_CFGR1[31:0] is read-only

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **BLANKING[3:0]:** COMP1 channel blanking source selection
Bits of this field are set and cleared by software (only if LOCK not set).
The field selects the input source for COMP1 channel output blanking.
Note: See Table 173: COMP blanking source assignment for more details.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **INPSEL[2:0]**: COMP1 non-inverting input selection
These bits are set and cleared by software (only if LOCK not set).
They select which input is connected to the non-inverting input of COMP1 channel.
Note: See Table 167: COMP implementation non-inverting input assignment for STM32C55x/562 and STM32C59x/5A3 and Table 169: COMP implementation non-inverting input assignment for STM32C53x/542 for more details.

Bits 19:16 **INMSEL[3:0]**: COMP1 channel inverting input selection
These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the inverting input of COMP1 channel.
Note: See Table 172: COMP inverting input assignment for more details.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WINOUT**: COMP1 output selector
This bit is controlled by software (if not locked). It selects the COMP1 output.
0: COMP1_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 152: Window comparators functional block diagram*

Bits 13:12 **PWRMODE[1:0]:** Power mode of the COMP1 channel
These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the COMP channel.
00: High speed/high power
01: Medium speed/medium power
All others values: Reserved

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **HYST[1:0]:** COMP channel hysteresis selection
These bits are set and cleared by software (only if LOCK not set). They select the hysteresis voltage of the COMP channel.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ITEN:** COMP channel interrupt enable
This bit is set and cleared by software (only if LOCK not set). This bit enable the interrupt generation of the COMP channel.
0: Interrupt generation disabled for COMP channel
1: Interrupt generation enabled for COMP channel

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WINMODE**: COMP1 non-inverting input selector for window mode
This bit is controlled by software (if not locked). It selects the COMP1_INP input of COMP1.
0: Signal selected with INPSEL[1:0]
1: COMP2_INP signal of COMP2 (required for window mode, see *Figure 152: Window comparators functional block diagram*

Bit 3 **POLARITY:** COMP channel polarity selection
This bit is set and cleared by software (only if LOCK not set). It inverts COMP channel polarity.
0: COMP channel output is not inverted
1: COMP channel output is inverted

Bit 2 **SCALEN**: Voltage scaler enable

This bit is set and cleared by software (only if LOCK not set). This bit enables the V_{REFINT} scaler for the COMP channels.

0: V_{REFINT} scaler disabled

1: V_{REFINT} scaler enabled

Bit 1 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enables the bridge of the scaler.

0: Scaler resistor bridge disabled

1: Scaler resistor bridge enabled

Bit 0 **EN:** COMP channel enable

This bit is set and cleared by software (only if LOCK not set). It enables the COMP channel.

0: Disable

1: Enable

23.8.4 Comparator configuration register 2 (COMP_CFGR2)

The COMP_CFGR2 is the COMP2 configuration register.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	BLANKING[3:0]				Res.	INPSEL[2:0]			INMSEL[3:0]			
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WINOUT	PWRMODE[1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	WINMODE	POLARITY	SCALEN	BRGEN	EN
	rw	rw	rw			rw	rw		rw		rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP channel configuration register COMP_CFGR1[31:0]

0: COMP_CFGR2[31:0] register is read/write

1: COMP_CFGR2[31:0] is read-only

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **BLANKING[3:0]:** COMP2 channel blanking source selection

Bits of this field are set and cleared by software (only if LOCK not set).

The field selects the input source for COMP1 channel output blanking.

Note: See Table 173: COMP blanking source assignment for more details.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **INPSEL[2:0]**: COMP2 non-inverting input selection

These bit are set and cleared by software (only if LOCK not set).

They select which input is connected to the non-inverting input of COMP2 channel.

Note: See Table 169: COMP implementation non-inverting input assignment for STM32C53x/542 for more details.

Bits 19:16 **INMSEL[3:0]**: COMP2 channel inverting input selection
These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the inverting input of COMP1 channel.
Note: See Table 172: COMP inverting input assignment for more details.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WINOUT**: COMP2 output selector
This bit is controlled by software (if not locked). It selects the COMP1 output.
0: COMP2_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 152: Window comparators functional block diagram*

Bits 13:12 **PWRMODE[1:0]:** Power mode of the COMP2 channel
These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the COMP channel.
00: High speed/high power
01: Medium speed/medium power
All others values: Reserved

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **HYST[1:0]:** COMP channel hysteresis selection
These bits are set and cleared by software (only if LOCK not set). They select the hysteresis voltage of the COMP channel.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ITEN:** COMP channel interrupt enable
This bit is set and cleared by software (only if LOCK not set). This bit enable the interrupt generation of the COMP channel.
0: Interrupt generation disabled for COMP channel
1: Interrupt generation enabled for COMP channel

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WINMODE**: COMP2 non-inverting input selector for window mode
This bit is controlled by software (if not locked). It selects the COMP2_INP input of COMP2.
0: Signal selected with INPSEL[1:0]
1: COMP2_INP signal of COMP2 (required for window mode, see *Figure 152: Window comparators functional block diagram*

Bit 3 **POLARITY:** COMP channel polarity selection
This bit is set and cleared by software (only if LOCK not set). It inverts COMP channel polarity.
0: COMP channel output is not inverted
1: COMP channel output is inverted

Bit 2 **SCALEN**: Voltage scaler enable
This bit is set and cleared by software (only if LOCK not set). This bit enables the V_{REFINT} scaler for the COMP channels.
0: V_{REFINT} scaler disabled
1: V_{REFINT} scaler enabled

Bit 1 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enables the bridge of the scaler.

0: Scaler resistor bridge disabled

1: Scaler resistor bridge enabled

Bit 0 **EN:** COMP channel enable

This bit is set and cleared by software (only if LOCK not set). It enables the COMP channel.

0: Disable

1: Enable

23.8.5 COMP register map

Table 178. COMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**COMP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2IF	C1IF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2VAL	C1VAL
	Reset value															0	0															0	0
0x04	**COMP_ICFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	CC1IF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0																
0x0C	**COMP_ CFGR1**	LOCK	Res.	Res.	Res.	BLANKING [3:0]				Res.	INPSEL [2:0]			INMSEL [3:0]				Res.	WINMOUT	PWRMODE [1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	WINMODE	POLARITY	SCALEN	BRGEN	EN
	Reset value	0				0	0	0	0		0	0	0	0	0	0	0		0	0	0			0	0		0		0	0	0	0	0
0x10	**COMP_ CFGR2**	LOCK	Res.	Res.	Res.	BLANKING [3:0]				Res.	INPSEL [2:0]			INMSEL [3:0]				Res.	WINMOUT	PWRMODE [1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	WINMODE	POLARITY	SCALEN	BRGEN	EN
	Reset value	0				0	0	0	0		0	0	0	0	0	0	0		0	0	0			0	0		0		0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

24 Operational amplifier (OPAMP)

This section applies only to STM32C53x/54x devices.

24.1 Introduction

The devices embed one operational amplifier instance, OPAMP1.

OPAMP contains an operational amplifier, an internal resistor divider, and multiplexers to set the OPAMP peripheral into a specific operating mode. In one of these, the operational amplifier inputs (INP and INM) and the output (OUT) are connected to external pins, which allows the use of OPAMP as a standalone operational amplifier device. The other OPAMP configurations allow the operation as a follower and as noninverting or inverting operational amplifier with programmable gain (with or without external filtering and/or biasing).

The programmable gain and the operating mode can be controlled through primary and secondary control register sets that can be activated in a time multiplex by the timer.

The operational amplifier output can be connected, through a multiplexer, to the OPAMP_VOUT pin or internally to a COMP input channel.

The INP input of OPAMP1 can be connected to an output of the internal DAC.

The operational amplifier operates at normal and high speed (slew rate). Its input offset is compensated through factory trimming, which can be modified by the user if needed.

As long as OPAMP is not required, it can be disabled to save power.

24.2 OPAMP main features

- Rail-to-rail input voltage range
- Low input bias current
- Low input offset voltage (factory compensated for)
- High frequency bandwidth
- Normal and high speed (slew rate)
- Timer-controlled operation

Note: Refer to the product datasheet for detailed OPAMP characteristics.

24.3 OPAMP implementation

The device has one OPAMP instance, OPAMP1. *Table 179* shows the pin and internal signal specific naming, and provides the assignment to I/Os and the possible connection of the internal signals:

- *Generic name* provides the pin and internal signal names as per *Figure 156*.
- *Instance-specific name* provides pin and internal signal names pertaining to each instance.
- *Assignment or possible connection* provides the assignment to I/O pins for signals on pins (in upper-case letters), and the possible connections of internal signals (in lower-case letters).

Table 179. OPAMP pin and signal assignment per instance

	Generic name	Instance-specific name	Assignment or possible connection	Comment
OPAMP1	OPAMP_VINM0	OPAMP1_VINM0	PA2	Selected as additional I/O function
	OPAMP_VINM1	OPAMP1_VINM1	PB1	
	OPAMP_VINP0	OPAMP1_VINP0	PA1	
	OPAMP_VINP1	OPAMP1_VINP1	PB0	
	OPAMP_VINP2	OPAMP1_VINP2	PB15	
	opamp_dac_in	opamp1_dac_in	dac2_int	-
	OPAMP_VOUT	OPAMP1_VOUT	PA6	Selected as additional I/O function
	opamp_out	opamp1_out	comp2_inn8	Selected in the COMP
	opamp_pclk	opamp1_pclk1	-	Enabled/disabled through RCC

24.4 OPAMP functional description

Each OPAMP instance can individually be enabled. When disabled, its output is in a high-impedance state. When enabled, it can operate in functional or in calibration mode.

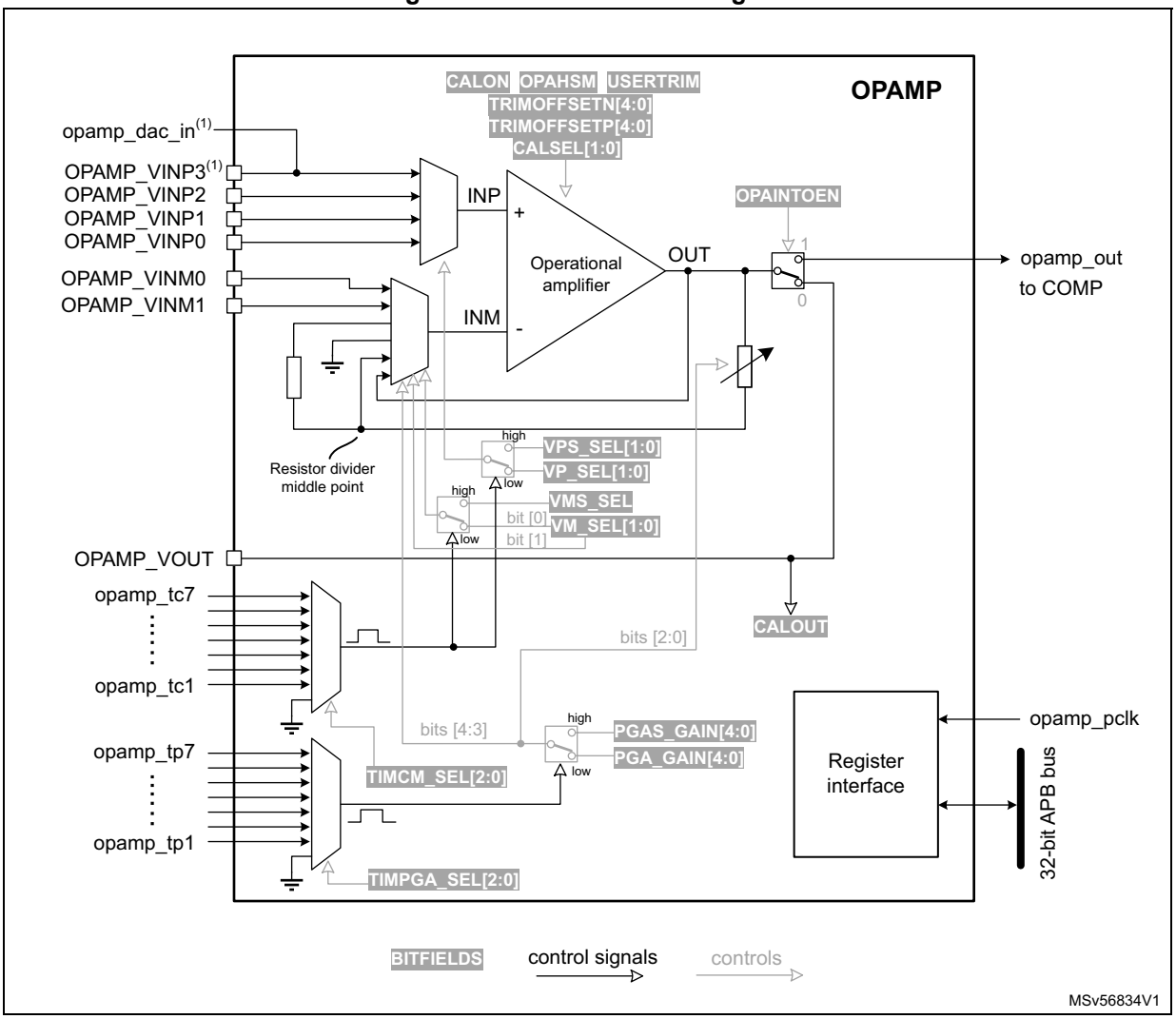
In functional mode, the inputs and the output of the operational amplifier are connected as described in *Section 24.4.8*. In calibration mode, they are all disconnected.

Caution: The GPIOs connected to the operational amplifier terminals through the OPAMP multiplexers must be configured in analog mode (default state), through the corresponding GPIOx_MODER register.

24.4.1 OPAMP block diagram

Figure 156 shows the block diagram of the OPAMP peripheral. The calibration multiplexers and some control bitfields are not shown.

Figure 156. OPAMP block diagram



1. Either OPAMP_VINP3 pin or DAC output, depending on the OPAMP instance. Refer to *Section 24.3*.

24.4.2 OPAMP pins and internal signals

Table 180 shows the pins used by the OPAMP peripheral. Their names correspond to the configuration for use as inputs or outputs. Refer to *Section 24.3* for the assignment of each to an I/O.

Table 180. OPAMP pins

Name	Type	Description
OPAMP_VINP0 OPAMP_VINP1 OPAMP_VINP2 OPAMP_VINP3	I	Selectable as INP noninverting input of the operational amplifier.
OPAMP_VINM0 OPAMP_VINM1	I	Selectable as INM inverting input of the operational amplifier.

Table 180. OPAMP pins (continued)

Name	Type	Description
OPAMP_VOUT	O	Operational amplifier output. Usable for other purposes when output OUT is redirected to an internal COMP channel.

Caution: The input and output pins connected, through the OPAMP multiplexers, to the operational amplifier terminals INP, INM, and OUT must be configured in analog mode (default state), through the corresponding GPIOx_MODER register.

Table 181 lists the OPAMP internal signals. Refer to *Section 24.3* for pthe ossible connection of each to internal resources and for specialization of the signal name.

Table 181. OPAMP internal input/output signals

Name	Type	Description
opamp_tc1 opamp_tc2 opamp_tc3 opamp_tc4 opamp_tc5 opamp_tc6 opamp_tc7	I	Signal from a timer channel, selectable as control signal switching between primary and secondary controls for the input multiplexers.
opamp_tp1 opamp_tp2 opamp_tp3 opamp_tp4 opamp_tp5 opamp_tp6 opamp_tp7	I	Signal from a timer channel, selectable as control signal switching between primary and secondary controls for the programmable gain and for setting the PGA flavor.
opamp_dac_in	I	Output of a DAC channel, selectable as noninverting operational amplifier input. Exclusive with the OPAMP_VINP3 pin.
opamp_pclk	I	Clock for OPAMP registers.
opamp_out	O	Operational amplifier output to an internal COMP channel, when redirected through the OPAINTOEN bit.

24.4.3 OPAMP reset and clocks

The OPAMP clock opamp_pclk provided by the clock controller is generated from the PCLK1 (APB1 clock). To access the OPAMP control registers, it is necessary to set the corresponding OPAMPxEN bit of the RCC_APB1ENR register.

The OPAMP registers can be reset through the corresponding OPAMPxRST bit of the RCC_APB1RSTR register.

24.4.4 OPAMP output redirection to an internal COMP channel

The operational amplifier output OUT can be redirected internally to a COMP channel, by setting the OPAINTOEN bit of the OPAMP_CSR register. In this case, the I/O pin OPAMP_VOUT on which the operation amplifier output OUT is mapped can be used for another purpose. See *Table 179* for possible COMP input channel assignment to OPAMP instances.

When OUT is connected to a COMP input channel, it is not connected to the OPAMP_VOUT pin.

24.4.5 OPAMP initial configuration

By default (upon reset):

- The operational amplifier inputs INP/INM and output OUT are connected to I/O pins.
- The registers for offset compensation contain factory trimming values and are in read-only access mode.
- The speed (slew rate) is set to normal.
- The timer-controlled operation of multiplexers is disabled.
- OPAMP is disabled.

24.4.6 OPAMP operational amplifier speed

The initial operational amplifier speed (slew rate, response) is normal. Setting the OPAHSM bit of the OPAMP_CSR register increases the speed. Both normal and high-speed mode characteristics are defined in the datasheet.

24.4.7 OPAMP enable

When disabled (default upon reset), the OPAMP control and calibration registers can still be modified. The operational amplifier output OUT is in high-Z state.

It is recommended to update the OPAMP registers while the OPAMP peripheral is disabled, to avoid spurious effects on the output.

OPAMP is enabled by setting the OPAEN bit of the OPAMP_CSR register. The INP and INM inputs and the OUT output are connected through the embedded multiplexers according to the control register settings and the timer control signals.

24.4.8 OPAMP operating modes

OPAMP can be configured to different operating modes or use cases, through:

- OPAMP control registers
- GPIOx_MODER registers, configuring the I/Os selected as OPAMP inputs and output
- COMP registers, selecting a COMP input channel when the operational amplifier output is redirected to the COMP
- External components, defining the gain, bias, or filtering parameters

The principal operating modes are:

- Stand-alone operational amplifier
- Follower
- Programmable-gain amplifier (PGA)

Refer to the following subsections for details and for different PGA flavors.

Through the OPAINTOEN bit of the *OPAMP control/status register (OPAMP_CSR)*, the operational amplifier output OUT can be connected to the I/O pin selected as OPAMP output (default) or to an internal COMP input channel. However, the latter option cannot be used when the operational amplifier output on the pin is required.

The OPAHSM bit (0 by default) allows increasing the operational amplifier speed (slew rate).

The USERTRIM bit can enable writing to the input offset trimming register, to allow modifying the by-default factory trimming values by the user.

The impedance of the input signal must be low enough to prevent the OPAMP input current from creating a voltage drop in the source, thus altering the input voltage. Refer to the electrical characteristics section in the datasheet for further details.

Standalone operational amplifier

For this use, ensure the following settings.

- Select a pin for the INP input (as opposed to a DAC output), with VP_SEL[2:0] and VPS_SEL[2:0] bitfields.
- Set the VM_SEL[1:0] bitfield to 0X, to ensure that a pin is selected for the INM input.
- Clear OPAINTOEN bit, to direct the operational amplifier output to the OPAMP_VOUT pin.

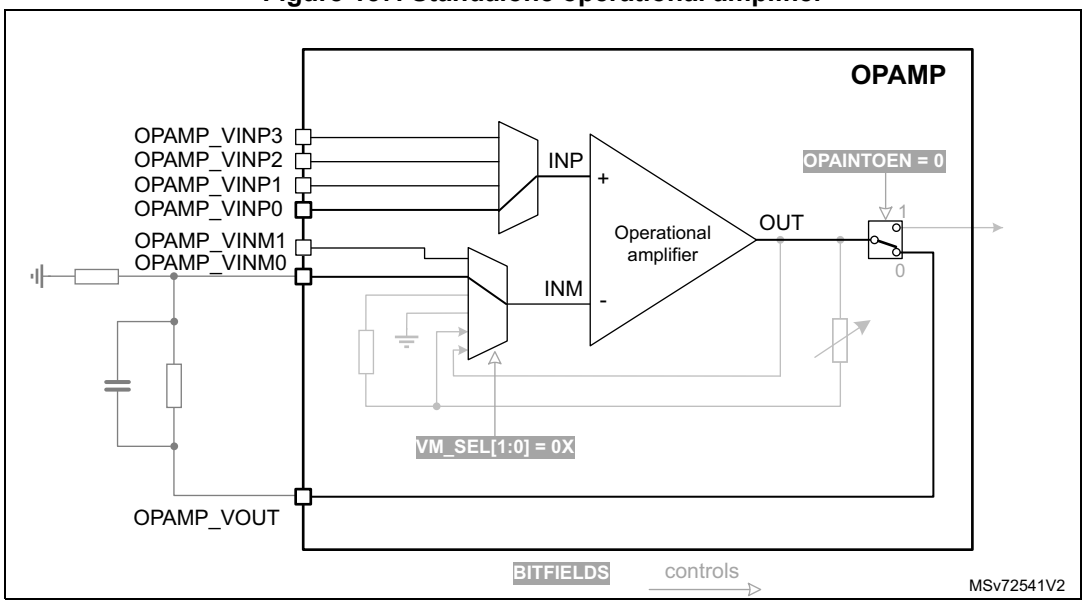
With these settings, the INP, INM inputs, and the OUT output of the operational amplifier are connected to pins and to no other internal structure.

Note: The OPAMP_CSR, OPAMP_TCMR, and GPIOx_MODER registers in their default state configure OPAMP as standalone operational amplifier, with normal speed, factory offset trimming values, and with OPAMP_VINP0 and OPAMP_VINM0 selected for INP and INM, respectively. The OPAMP is disabled so the OUT and OPAMP_OUT are in high-Z state.

Once enabled, OPAMP operates as a standalone operational amplifier, with the gain, filtering, and biasing determined by external routing and components.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin, and not to the internal COMP channel.

Figure 157. Standalone operational amplifier



Follower

To use OPAMP as a follower with an internal feedback, set the following bitfields as indicated:

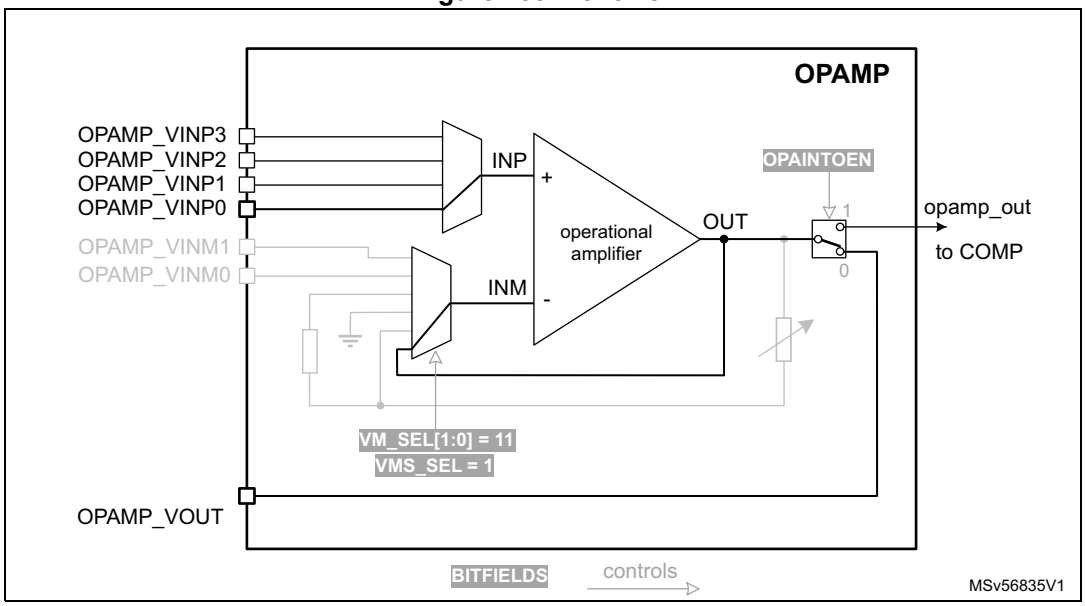
- VM_SEL[1:0] to 11 and VMS_SEL to 1, to loop the operational amplifier output back to its inverting input INM
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select for the noninverting input INP

Once enabled, the operational amplifier buffers the voltage selected on its noninverting input INP.

Note: The GPIOs corresponding to OPAMP_VINMy are free for another use.

In this mode, OPAMP takes two pins if a pin is selected for INP noninverting input. It takes a single pin if an internal DAC output is selected for INP noninverting input (on instances supporting this). Refer to Section 24.3.

Figure 158. Follower



Programmable-gain amplifier

When OPAMP is configured as a programmable-gain amplifier (PGA), the internal resistor divider with a programmable resistor value determines the amplifier gain. Refer to the datasheet for the values of the internal resistors.

The following sections describe different PGA flavors.

Noninverting PGA

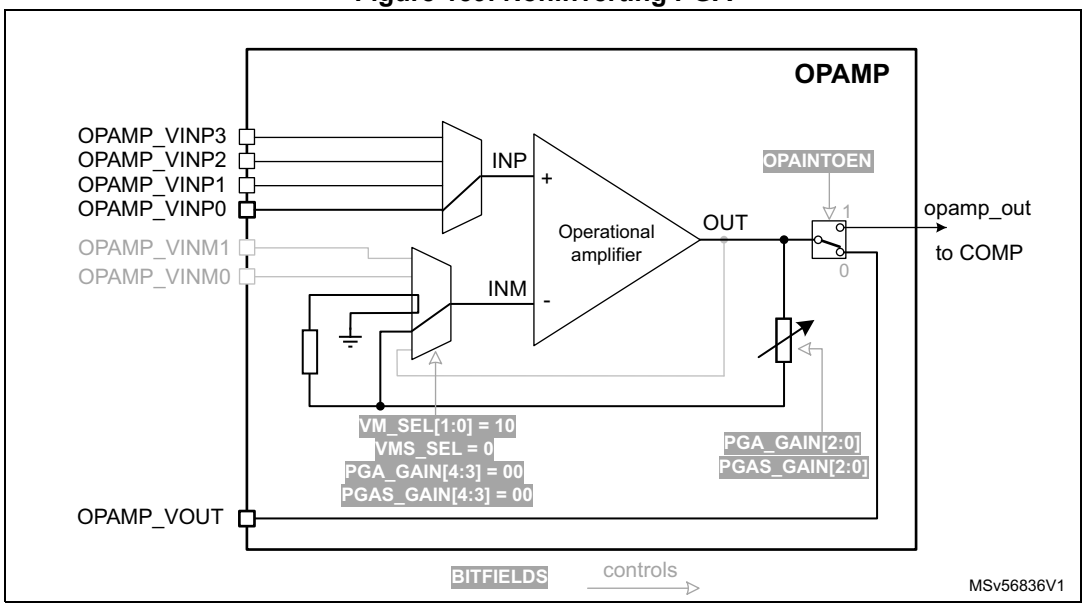
To use OPAMP as a noninverting PGA without filtering, set the following bitfields as indicated:

- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 00XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

Once enabled, OPAMP outputs a voltage that corresponds to the selected input pin voltage multiplied by the selected gain value.

Note: To avoid saturation, keep the product of the input voltage and the selected gain value below V_{DDA}.

Figure 159. Noninverting PGA



Noninverting PGA with filtering

To use OPAMP as a noninverting PGA with filtering, set the following bitfields as indicated:

- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 10XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

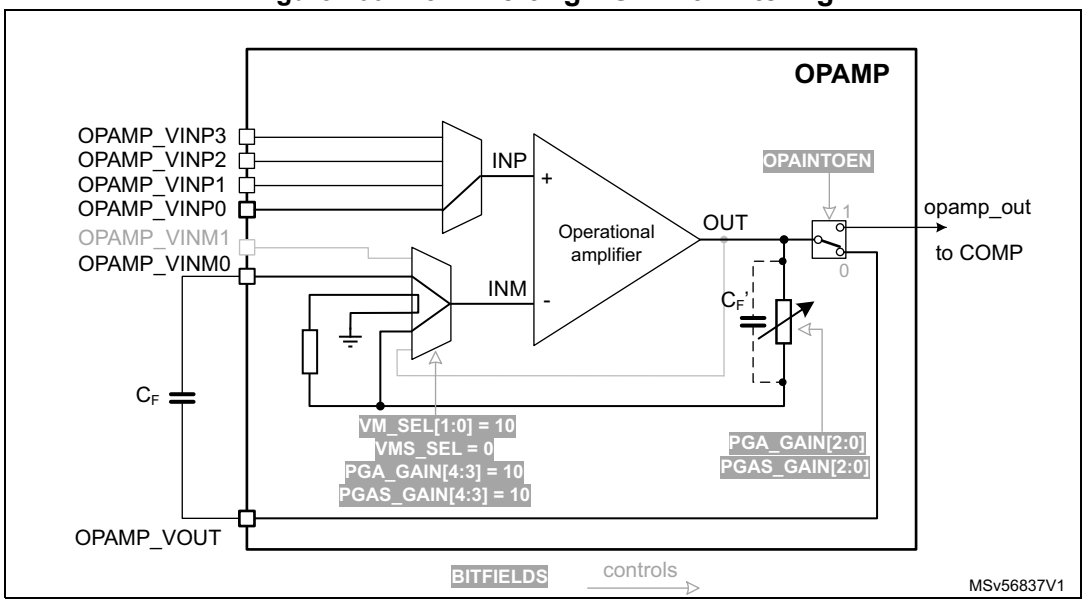
Once enabled, OPAMP outputs a voltage that corresponds to the selected input pin voltage multiplied with the selected gain (for static signals).

External components connected between the OPAMP_VOUT pin and the OPAMP_VINM0 pin are put in parallel with the internal programmable resistor. For example, connecting a capacitor C_F as shown in *Figure 160* has a filtering effect equivalent to C_F' of the same

capacitance. The gain for dynamic signals then also depends on the signal frequency and the cutoff frequency of the filter.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin and not to the internal COMP channel.

Figure 160. Noninverting PGA with filtering



Noninverting PGA with external bias, or inverting PGA

To use OPAMP as a noninverting PGA with external bias or as an inverting PGA, set the following bitfields as indicated:

- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 01XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

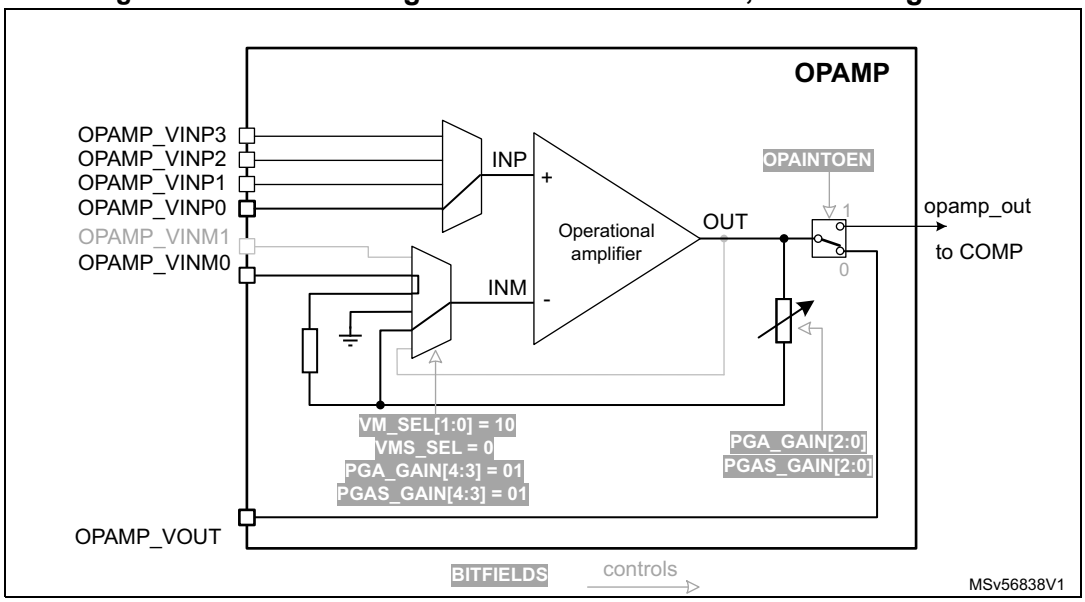
For the signal on the selected OPAMP_VINPy pin, OPAMP operates as a noninverting amplifier with the selected positive gain. The OPAMP_VINM0 pin is used as a reference (bias).

For the signal on the OPAMP_VINM0 pin, OPAMP operates as an inverting amplifier with the selected negative gain. The selected OPAMP_VINPy pin is used as a reference.

Note: The final gain depends upon the output impedance of the source connected to the OPAMP_VINM0 pin.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin, and not to the internal COMP channel.

Figure 161. Noninverting PGA with external bias, or inverting PGA



Inverting or noninverting PGA with external bias or/and filtering

To use OPAMP as a noninverting PGA with external bias, or as an inverting PGA with filtering, set the following bitfields as indicated:

- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 11XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

For the signal on the selected OPAMP_VINPy pin, OPAMP operates as a noninverting amplifier with the selected positive gain. The voltage on the OPAMP_VINM0 pin is used as a reference (bias).

For the signal on the OPAMP_VINM0 pin, OPAMP operates as an inverting amplifier with the selected negative gain. The voltage on the selected OPAMP_VINPy pin is used as a reference (bias). The output impedance of the biasing voltage source impacts the total gain.

In both cases, any external components connected between the OPAMP_VOUT pin and the OPAMP_VINM1 pin are put in parallel with the internal programmable resistor. For example, connecting a capacitor C_F as shown in the following figure has a filtering effect equivalent to C_F' of the same capacitance. The gain for dynamic signals then also depends on the signal frequency and the cutoff frequency of the filter.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin and not to the internal COMP channel.

Figure 162. Inverting or noninverting PGA with external bias or/and filtering

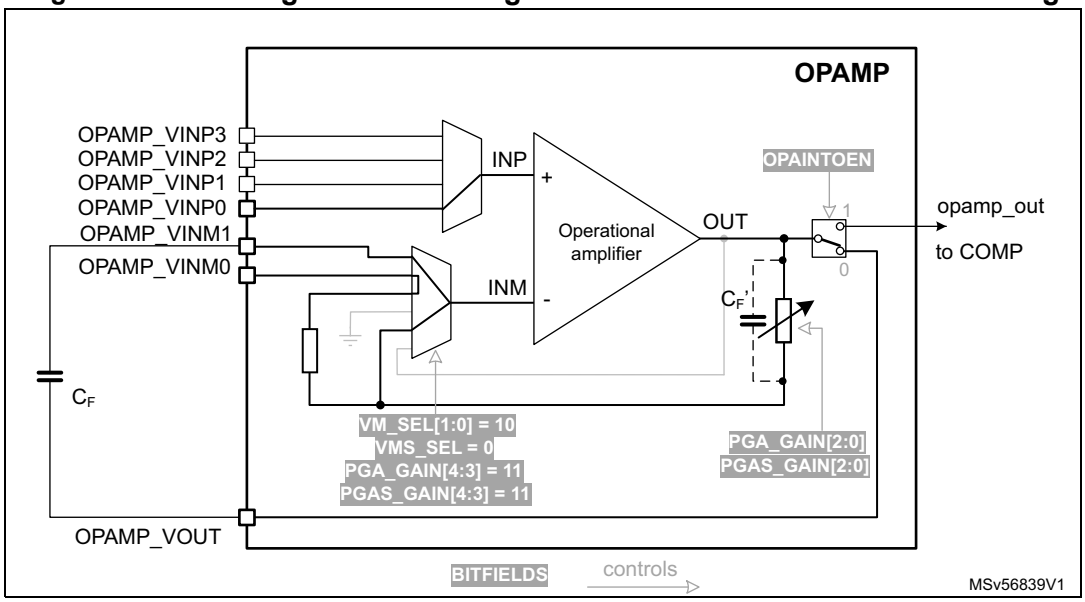
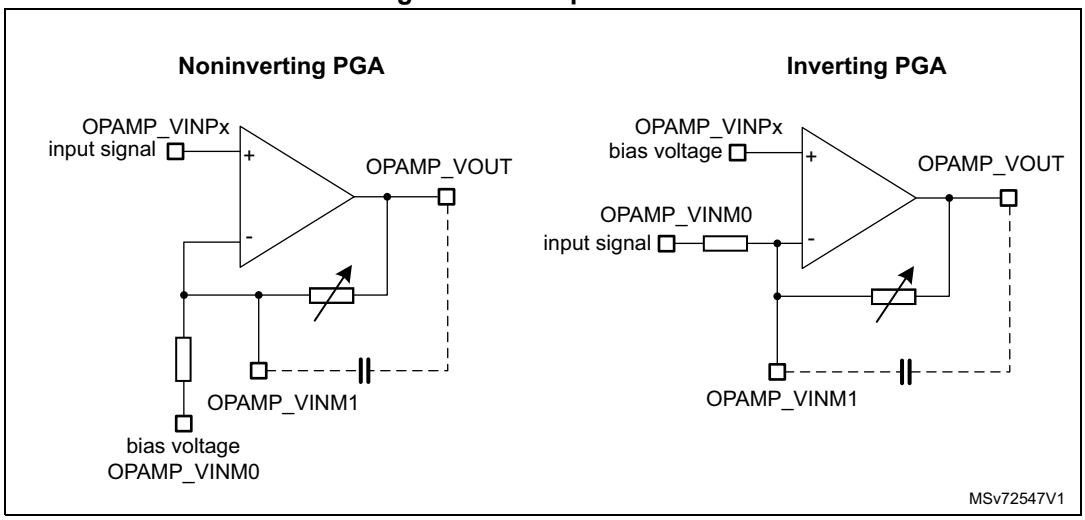


Figure 163. Simplified view



24.4.9 OPAMP programmable gain

In PGA mode, OPAMP allows selecting one of discrete positive (noninverting configuration) or negative (inverting configuration) gain values. The selection is done through the bits [2:0] of the PGA_GAIN[4:0] bitfield of the OPAMP_CSR register or through the bits [2:0] of the PGAS_GAIN[4:0] bitfield of the OPAMP_TCMR register. The control bitfield is selected with the control signal at the output of the opamp_tpX selector.

In noninverting mode, the programmable gain depends only upon the internal resistive divider. In inverting mode, depends upon the internal resistors, and upon the output impedance of the signal source. The higher it is, the greater impact it has on the final gain.

24.4.10 OPAMP operational amplifier offset compensation

The peripheral incorporates a trimming circuitry that impacts the operational amplifier input offset. The trimming circuitry is controlled through the TRIMOFFSETN[4:0] (for the N differential pair) and TRIMOFFSETP[4:0] (for the P differential pair) bitfields of the *OPAMP control/status register (OPAMP_CSR)*.

The TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] bitfields keep the trimming value for the N differential pair and P differential pair, respectively.

Upon reset, these registers are loaded with factory trimming values stored in the nonvolatile memory. As the USERTRIM bit is low, the TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] control bitfields are read-only.

Note: The factory trimming values result from a calibration procedure run on the production line in specific production conditions.

When the offset compensation obtained with the factory trimming values is not acceptable for a specific use case (supply voltage, operating temperature), the user software can apply different trimming values.

To modify the trimming values, the user software has first to set the USERTRIM bit to allow writing into the TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] control bitfields.

To retain the user trimming values, the user software has to write them into a nonvolatile memory. It typically retrieves them from the nonvolatile memory and writes into TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] at the startup, before the first use of OPAMP.

The resolution of the offset trimming allows reducing the input offset voltage to a value defined in the datasheet, at a defined supply voltage and die temperature.

24.4.11 OPAMP calibration

The calibration allows finding the trimming values that minimize the operational amplifier offset.

Setting the CALON bit isolates the operational amplifier INP and INM inputs and connects them to the internal reference voltage level selected through the CALSEL[1:0] bitfield. The operational amplifier is in a comparator (high gain, open loop) mode in which its output transits from one rail to another when the voltage level on one input crosses the voltage level on the other input.

The CALOUT flag reflects this digital value (lower rail or upper rail) of the OPAMP_VOUT pin.

The CALSEL[1:0] bitfield values 01 (low reference voltage) and 11 (high reference voltage) are intended for use in the calibration procedure. The value 01 is for trimming the PMOS through the TRIMOFFSETP[4:0] bitfield, the value 11 for trimming the NMOS through the TRIMOFFSETN[4:0] bitfield.

The calibration is carried out by software. Starting from the trimming bitfield at 0 and the CALOUT flag at one, the procedure consists in incrementing the trimming value until the CALOUT flag toggles to 0. The last trimming value is the one that compensates for the operational amplifier offset of the given OPAMP instance, and for the given supply voltage, die temperature, and differential pair stage (PMOS or NMOS). Upon each trimming value incrementation, the software must wait for at least the $t_{OFFTRIM}$max period defined in the product datasheet, before reading the CALOUT flag and potentially proceeding to the next incrementation step.

Note: The closer is the trimming value to the offset compensation point, the longer the output voltage takes to respond (without ever exceeding $t_{OFFTRIM}max$).

The full calibration consists of four calibration loops, as listed in *Table 182*. They can be executed in any order. The CALON bit must be set for executing any calibration loop. The calibration loops for the operational amplifier speed (slew rate) not required in the application can be omitted.

Table 182. Calibration loops

Calibration loop	Control bitfield		
	CALSEL[1:0]	TRIMOFFSETN[4:0]	TRIMOFFSETP[4:0]
NMOS	11	Increment	Keep as is
PMOS	01	Keep as is	Increment

Different OPAMP instances can be calibrated at any time, even at the same time.

Calibration initialization

To initialize the offset trimming value calibration:

- Enable OPAMP, by setting OPAEN.
- Make the trimming bitfields writable, by setting USERTRIM.
- Write TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] bitfields with 0b00000.

Calibration loop sequence

To execute a calibration loop, go through the following steps:

1. Select the calibration loop, with the CALSEL[1:0] bitfield. That is, the speed (slew rate) and the differential pair end (NMOS or PMOS) to calibrate.
2. Increment the appropriate trimming bitfield (TRIMOFFSETN[4:0] for NMOS, TRIMOFFSETP[4:0] for PMOS), keeping the other trimming bitfield as is.
3. Wait for at least the $t_{OFFTRIMmax}$ period.
4. Check the CALOUT flag (OPAINTOEN = 0). If high, go to step *2* (loop back).
5. The trimming bitfield is calibrated. Store the calibrated trimming value if required.

For a given speed (slew rate), execute both the NMOS and the PMOS calibration loops.

Note: During the calibration loop execution, the external connection of the operational amplifier output must not pull up or down currents exceeding 500 µA.

Calibrating when operational amplifier output goes to a COMP input

With the operational amplifier output redirected to an internal COMP input channel (OPAINTOEN bit set), the CALOUT flag does not reflect the operational amplifier output. To address this, it is advised to clear the OPAINTOEN bit for the time of calibration to make the CALOUT flag operate. When the calibration is completed, set OPAINTOEN to redirect the operational amplifier output back to a COMP input channel. As during the calibration the internal signal sent to the OPAMP_VOUT pin toggles, take care to prevent any conflict on the pin. For example, use the pin in the application as an input.

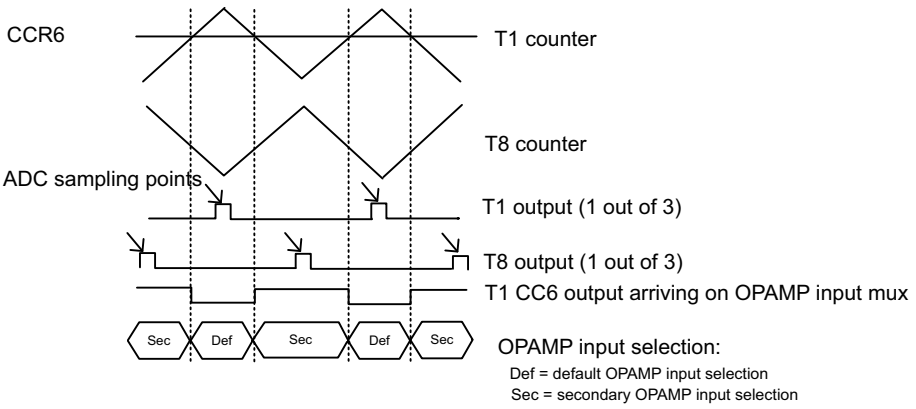
24.4.12 Timer-controlled operation

Certain timer channels (see *Table 183*) can toggle the control of the operational amplifier multiplexers between the primary configuration defined through the OPAMP_CSR register and the secondary configuration defined through the OPAMP_TCMR register.

Once configured, the mechanism operates even when the OPAMP clock is disabled.

As an example, this can be used for dual motor control. *Figure 164* illustrates the use of a timer compare output toggling the OPAMP primary (default) and secondary configurations, to measure alternatively the currents of the three phases of a motor, then of the other motor.

Figure 164. Timer-controlled operation for a dual motor application



Timer-controlled input selection

This operation is activated by writing a value other than 0 to the TIMCM_SEL[2:0] bitfield of the OPAMP_TCMR register, thus selecting one of the opamp_tcX signals. As long as the selected signal is low, the input selectors are controlled with the VP_SEL[1:0] and VM_SEL[1:0] bitfields of the OPAMP_CSR register. As long as it is high, the input selectors are controlled with the VPS_SEL[1:0] and VMS_SEL bitfields of the OPAMP_TCMR register. Writing 0 to TIMCM_SEL[2:0] (default upon reset) makes the input selectors steadily controlled with VP_SEL[1:0] and VM_SEL[1:0].

Table 183 gives the correspondence between the opamp_tcX and the timer signals.

Table 183. Timer-controlled input selection

TIMCM_SEL[2:0]	Signal name	Timer connection
000	-	None
001	opamp_tc1	tim1_oc6
010	opamp_tc2	tim2_oc4
011	opamp_tc3	tim12_oc1
100	opamp_tc4	tim15_oc2
101	opamp_tc5	Reserved
110	opamp_tc6	Reserved
111	opamp_tc7	Reserved

Timer-controlled programmable gain and PGA flavor

This operation is activated by writing a value other than 0 to the TIMPGA_SEL[2:0] bitfield of the OPAMP_TCMR register, thus selecting one of opamp_tpX signals. As long as the selected signal is low, the programmable gain and PGA flavor selectors are controlled with the PGA_GAIN[4:0] bitfield of the OPAMP_CSR register. As long as it is high, the input selectors are controlled with the PGAS_GAIN[4:0] bitfield of the OPAMP_TCMR register. Writing 0 to TIMPGA_SEL[2:0] (default upon reset) makes the programmable gain and PGA flavor selectors steadily controlled with PGA_GAIN[4:0].

The following table gives the correspondence of the opamp_tpX signals to timer signals.

Table 184. Timer-controlled programmable gain selection

TIMPGA_SEL[2:0]	Signal name	Timer connection
000	-	None
001	opamp_tp1	tim1_oc6
010	opamp_tp2	tim2_oc4
011	opamp_tp3	tim12_oc2
100	opamp_tp4	tim15_oc2
101	opamp_tp5	Reserved
110	opamp_tp6	Reserved
111	opamp_tp7	Reserved

24.5 OPAMP in low-power modes

Table 185. Effect of low-power modes on OPAMP

Mode	Description
Sleep	No effect.
Stop	No effect, OPAMP register contents are kept.
Standby	The OPAMP registers are powered down and must be re-initialized after exiting Standby mode.

24.6 OPAMP registers

24.6.1 OPAMP control/status register (OPAMP_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	CAL OUT	TSTREF	TRIMOFFSETN[4:0]					TRIMOFFSETP[4:0]					PGA_GAIN[4:2]		
rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN[1:0]		CALSEL[1:0]		CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL[1:0]		USERTRIM	VP_SEL[1:0]		FORCE_VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP_CSR register lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.
This bit is used to configure the OPAMP_CSR register as read-only.
0: OPAMP_CSR is read-write
1: OPAMP_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit reflects the digital value of OPAMP output when OPAINTOEN bit is low. During the input offset calibration procedure, it provides the indication that the target trimming value is reached, by transiting from 1 to 0.
The bit cannot be used as long as the OPAINTOEN bit is high.

Bit 29 **TSTREF:** OPAMP calibration reference voltage output control

This bit is reserved for test. When set, OPAMP outputs the calibration reference voltage.
0: Does not output calibration reference voltage
1: Outputs calibration reference voltage

Bits 28:24 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs

Upon reset, this bitfield is loaded with the factory trimming value. It can only be modified when the USERTRIM bit is set.

Bits 23:19 **TRIMOFFSETP[4:0]**: Trim for PMOS differential pairs

Upon reset, this bitfield is loaded with the factory trimming value. It can only be modified when the USERTRIM bit is set.

Bits 18:14 **PGA_GAIN[4:0]:** Operational amplifier programmable gain and PGA flavor primary control
This bitfield selects the programmable gain (bits [2:1]) and the PGA flavor ([bits 4:3]). Positive numbers represent positive gain (noninverting PGA), negative numbers negative gain (inverting PGA).
00000: gain 2
00001: gain 4
00010: gain 8
00011: gain 16
01000: gain -1 / gain 2 with bias on OPAMP_VINM0
01001: gain -3 / gain 4 with bias on OPAMP_VINM0
01010: gain -7 / gain 8 with bias on OPAMP_VINM0
01011: gain -15 / gain 16 with bias on OPAMP_VINM0
10000: gain 2 with filtering on OPAMP_VINM0
10001: gain 4 with filtering on OPAMP_VINM0
10010: gain 8 with filtering on OPAMP_VINM0
10011: gain 16 with filtering on OPAMP_VINM0
11000: gain -1 / gain 2 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11001: gain -3 / gain 4 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11010: gain -7 / gain 8 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11011: gain -15 / gain 16 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
Other: Not used

Bits 13:12 **CALSEL[1:0]:** Calibration selection
This bitfield selects the offset calibration reference voltage applied on the operational amplifier inputs when CALON and FORCE_VP bits are set.
00: $0.033*V_{DDA}$
01: $0.1*V_{DDA}$ (for PMOS calibration)
10: $0.5*V_{DDA}$
11: $0.9*V_{DDA}$ (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enable
This bitfield disables or enables the calibration mode.
0: Calibration mode disabled
1: Calibration mode enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable
This bit determines where the operational amplifier output is connected, either to the OPAMP_VOUT pin or internally to a COMP channel.
0: OPAMP_VOUT pin
1: COMP channel

Bit 7 **OPAHSM:** Operational amplifier high-speed mode
This bit determines the speed (slew rate) of the operational amplifier.
0: Normal speed
1: High speed
Only change this configuration when OPAMP is disabled.

Bits 6:5 **VM_SEL[1:0]:** Inverting input primary selection
This bitfield determines where the operational amplifier inverting input is connected to.
00: OPAMP_VINM0 pin
01: OPAMP_VINM1 pin
10: Internal resistor divider middle point
11: Operational amplifier output
The setting 10 is for the PGA use cases. With this setting, the connection of the OPAMP_VINM0 and OPAMP_VINM1 pins is determined by the PGA_GAIN[4:0] bitfield.
The setting 11 sets OPAMP in follower mode. With the other two settings, OPAMP is configured as a stand-alone operational amplifier.

Bit 4 **USERTRIM**: User trimming enable
This bit enables the user trimming of the operational amplifier input offset. When set, the TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] bitfields, loaded with factory trimming values upon reset, become writable. This allows modifying the input offset trimming values by the user.
0: Disable (trimming by user not possible)
1: Enable (trimming by user possible)

Bits 3:2 **VP_SEL[1:0]:** Noninverting input primary selection
This bitfield determines where the operational amplifier noninverting input is connected to.
00: OPAMP_VINP0 pin
01: OPAMP_VINP1 pin
10: OPAMP_VINP2 pin
11: OPAMP_VINP3 pin or DAC output channel
Refer to *Section 24.3: OPAMP implementation* for OPAMP instance-dependent information.

Bit 1 **FORCE_VP:** Force internal reference on noninverting input
This bit is reserved for test. It allows forcing the calibration reference voltage selected through the CALSEL[1:0] bitfield on the operational amplifier noninverting input.
0: Do not force (normal operation)
1: Force

Bit 0 **OPAEN:** Operational amplifier enable
This bit enables and disables the operational amplifier.
0: Disable
1: Enable

24.6.2 OPAMP timer-controlled mode register (OPAMP_TCMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMPGA_SEL[2:0]			PGAS_GAIN[4:0]					Res.	Res.	TIMCM_SEL[2:0]			VPS_SEL[1:0]		VMS_SEL
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP_TCMR register lock
This bit is write-once. It is set by software. It can only be cleared by a system reset.
This bit determines whether the OPAMP_TCMR register is read-write or read-only.
0: Read-write
1: Read-only (OPAMP_TCMR locked)

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:13 **TIMPGA_SEL[2:0]**: Timer toggle signal selection for programmable gain control
This bitfield is set and cleared by software. It selects the signal (coming from timer to OPAMP multiplexers) that toggles the programmable gain control bitfields between the primary (PGA_GAIN[4:0], signal at low level) and secondary (PGAS_GAIN[4:0], signal at high level).
000: None (gain permanently controlled through PGA_GAIN[4:0] and VM_SEL[1:0])
001: opamp_tp1
010: opamp_tp2
011: opamp_tp3
100: opamp_tp4
101: opamp_tp5
110: opamp_tp6
111: opamp_tp7

Bits 12:8 **PGAS_GAIN[4:0]**: Operational amplifier programmable gain and PGA flavor secondary control
This bitfield selects the programmable gain (bits [2:1]) and the PGA flavor ([bits 4:3]) when the timer signal selected through TIMPGA_SEL[2:0] is high. Positive numbers represent positive gain (noninverting PGA), negative numbers negative gain (inverting PGA).
00000: gain 2
00001: gain 4
00010: gain 8
00011: gain 16
01000: gain -1 / gain 2 with bias on OPAMP_VINM0
01001: gain -3 / gain 4 with bias on OPAMP_VINM0
01010: gain -7 / gain 8 with bias on OPAMP_VINM0
01011: gain -15 / gain 16 with bias on OPAMP_VINM0
10000: gain 2 with filtering on OPAMP_VINM0
10001: gain 4 with filtering on OPAMP_VINM0
10010: gain 8 with filtering on OPAMP_VINM0
10011: gain 16 with filtering on OPAMP_VINM0
11000: gain -1 / gain 2 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11001: gain -3 / gain 4 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11010: gain -7 / gain 8 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11011: gain -15 / gain of 16 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
Other: Not used

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:3 **TIMCM_SEL[2:0]**: Timer toggle signal selection for operational amplifier input control
This bitfield is set and cleared by software. It selects the signal (coming from timer to OPAMP multiplexers) that toggles the operational amplifier input configuration control bitfields between the primary (VP_SEL[1:0] / VM_SEL, signal at low level) and secondary (VPS_SEL[1:0] / VMS_SEL, signal at high level).
000: None (input configuration permanently controlled through VP_SEL[1:0] and VM_SEL)
001: opamp_tc1
010: opamp_tc2
011: opamp_tc3
100: opamp_tc4
101: opamp_tc5
110: opamp_tc6
111: opamp_tc7

Bits 2:1 **VPS_SEL[1:0]**: OPAMP noninverting input secondary selection.
This bitfield is set and cleared by software. It selects the signal to connect to the operational amplifier noninverting input as long as the level of the timer signal selected through the TIMCM_SEL[2:0] bitfield is high.
00: OPAMP_VINP0
01: OPAMP_VINP1
10: OPAMP_VINP2
11: OPAMP_VINP3 pin or DAC output channel
Refer to *Section 24.3: OPAMP implementation* for OPAMP instance-dependent information.

Bit 0 **VMS_SEL**: OPAMP inverting input secondary selection
This bitfield is set and cleared by software. It selects the signal to connect to the operational amplifier inverting input as long as the level of the timer signal selected through the TIMCM_SEL[2:0] bitfield is high. The effect also depends on the VM_SEL[1:0] bitfield.
Condition: VM_SEL[1:0] = 00 or 01
0: OPAMP_VINM0 (operation as stand-alone operational amplifier)
1: OPAMP_VINM1 (operation as stand-alone operational amplifier)
Condition: VM_SEL[1:0] = 10 or 11
0: Internal resistor divider middle point (operation as PGA)
1: Operational amplifier output (operation as follower)

24.6.3 OPAMP register map

Table 186. OPAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OPAMP_CSR	LOCK	CALOUT	Res.	TRIMOFFSETN [4:0]					TRIMOFFSETP [4:0]					PGA_GAIN[4:0]					CALSEL	CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL [1:0]		FORCE_VP	OPAEN
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0
0x004	OPAMP_TCMR	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMPGA _SEL [2:0]			PGAS_GAIN [4:0]					Res.	Res.	TIMCM _SEL [2:0]			VPS_SEL [1:0]		VMS_SE
	Reset value	0																0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x008 - 0x3FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.2 on page 90* for the register boundary addresses.

25 Coupling and chaining bridge (CCB)

This section only applies to STM32C59x/5A3 products.

25.1 CCB introduction

The coupling and chaining bridge (CCB) is a sophisticated security mechanism designed to ensure that private keys are protected, even against CPU access, and are wrapped with unique device keys, and stored securely in a secure storage.

The CCB can be programmed to implement special coupling and chaining operations required to protect private keys used in PKA protected operations.

These coupling and chaining operations involve the PKA, the SAES, and sometimes the RNG peripherals.

25.2 CCB main features

The principal features of the CCB are listed below:

- AHB system slave port, mapping multiple peripherals (see *Section 25.3*)
- AHB configuration slave port (CCB peripheral), accessible in write through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored). There is no limitation for reads.
- Optional coupling of PKA RAM writes to SAES input data register
- Support for multiple optional chaining options:
  - Read and write chaining between PKA RAM and SAES
  - SAES to CCB read chaining, with comparison to a 128-bit reference tag
  - RNG output chaining with either PKA RAM or SAES_IVR registers
- Dedicated sequences to support three PKA protected operations (modular exponentiation, scalar multiplication, ECDSA signature)
  - One-time sequence to prepare a PKA protected operation (blob creation)
  - Many-time sequence to execute PKA protected operation (blob usage)
- Optional private key generation for ECDSA signature, and for ECC private key cryptography (key never accessible to the application in clear-text)
- Possibility to encrypt, with AES-256, any PKA blob encryption key using the device unique secret key DHUK (in the SAES), which makes the PKA encrypted blob usable only on this device
- Software-reset capability
- Repository for cryptographic subsystem tamper event flags

25.3 CCB implementation

The CCB connects the peripherals listed in *Table 187*.

Table 187. Peripherals connected by CCB interconnect

Peripheral	Coupling and chaining feature(1)	Tamper status flag in CCB_SR
RNG	Yes	TAMP_EVT0
SAES	Yes	TAMP_EVT1
PKA	Yes	TAMP_EVT2 (registers + PKA RAM)
		TAMP_EVT3 (PKA operations)
CCB	Yes	TAMP_EVT4
AES	No	TAMP_EVT5
HASH	No	-

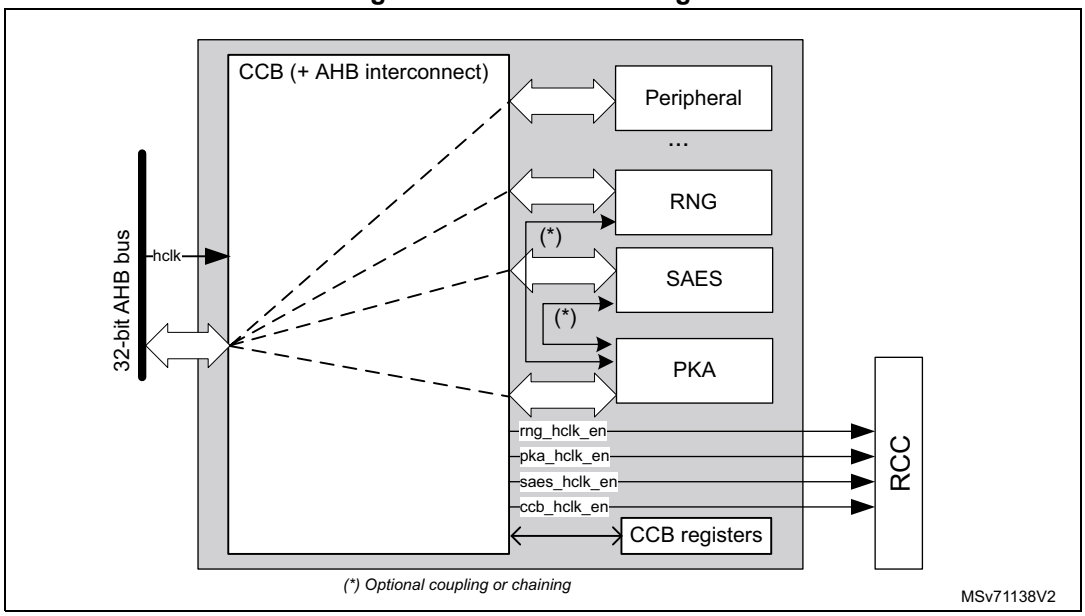
1. See *Section 25.4.4* for details.

25.4 CCB functional description

25.4.1 CCB block diagram

Figure 165 shows the CCB block diagram. Peripherals connected to the CCB interconnect, with optional coupling/chaining features, are defined in *Section 25.3*.

Figure 165. CCB block diagram



25.4.2 CCB internal signals

Table 188 describes the relevant internal signals interfacing the CCB peripheral. Signals targeting peripherals connected to the CCB are not listed.

Table 188. CCB internal input/output signals

Signal name	Signal type	Description
hclk	Digital input	Interconnect system clock input
ccb_hclk_en	Digital output	CCB clock request to the RCC
x_hclk_en	Digital output	Peripheral x clock request to the RCC (x = SAES, PKA, or RNG)

25.4.3 CCB reset and clocks

The CCB shares the same clock used by the AHB bus. Once the CCB clock is activated in the RCC, and coupling and chaining activity is activated by setting CCOP ≠ 0 in CCB_CR, the CCB automatically requests the RCC to enable RNG, SAES, and PKA clocks . These clocks remain enabled until the application gates them in the RCC, with CCOP cleared in the CCB_CR register.

The CCB has a dedicated reset bit in the RCC, and a system reset input. Locally to the CCB it is also possible to trigger a reset of the peripheral based on IPRST (do not use it when the application does not use CCB)in the CCB_CR register (refer to the bitfield description for details). Status register CCB_SR has a specific reset behavior (see *Section 25.7.2*).

When at least a cryptographic peripheral raises a tamper event, the CCB requests its clock to the RCC. The tamper status TAMP_EVTi of these peripherals can be found in the CCB_SR register, as defined in *Section 25.3*.

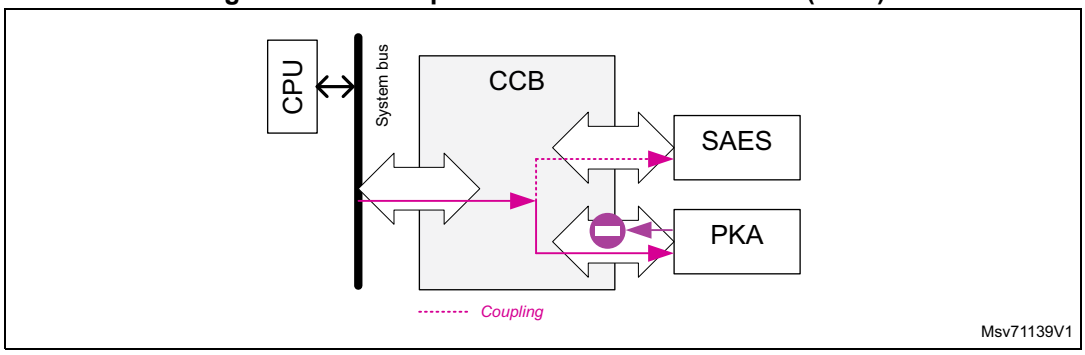
25.4.4 CCB coupling and chaining

Note: The selection of these chaining modes is defined in Section 25.5. In most of these operations, a read of the PKA RAM returns 0 (shown with a purple circle in the following figures).

Coupling CPU → PKA + SAES (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and sends them to SAES_DINR, as shown in *Figure 166*. Write to SAES_DINR is not coupled to the PKA. The CCB sends both data and address to SAES_DINR.

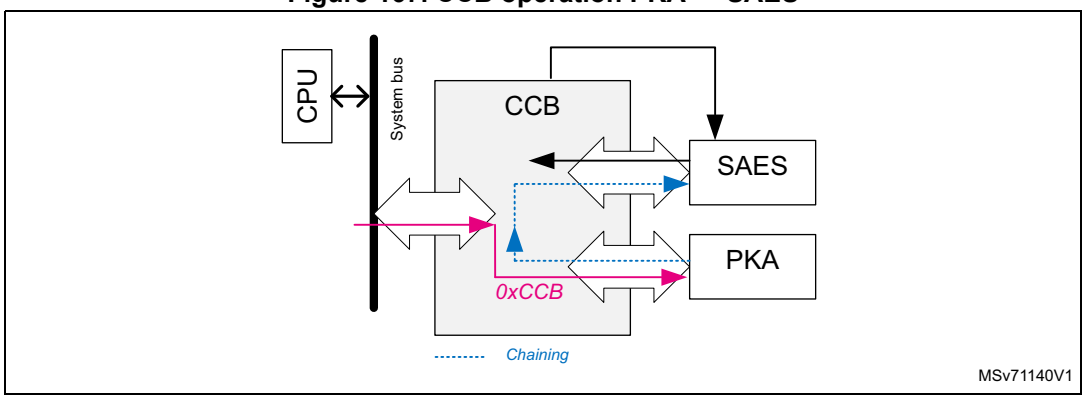
Figure 166. CCB operation CPU → PKA + SAES (write)



Chaining PKA → SAES

When the corresponding OPSTEP is selected, for each 32-bit write request to the PKA RAM of magic number 0xCCB, the CCB generates a read request from the write target (in the PKA RAM) to SAES_DINR.

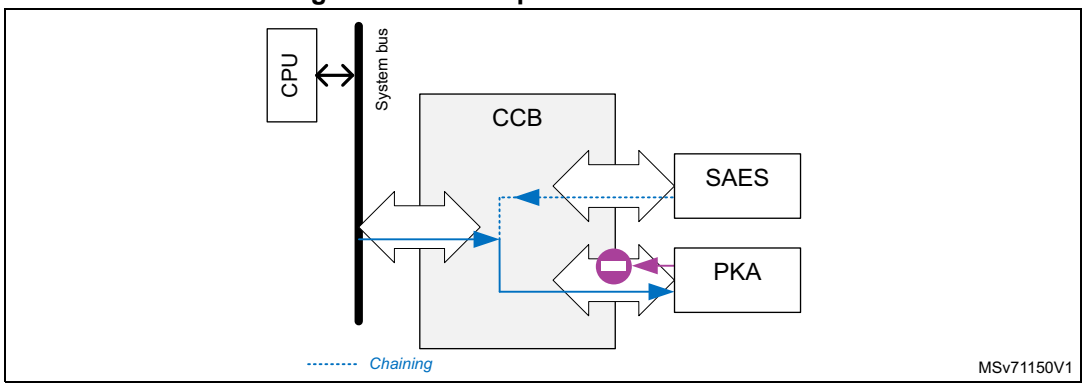
Figure 167. CCB operation PKA → SAES



Chaining SAES → PKA

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and substitutes these data by a 32-bit word coming from SAES_DOUTR. When available, this chaining is activated when the application writes 0xCCB to PKA RAM.

Figure 168. CCB operation SAES → PKA

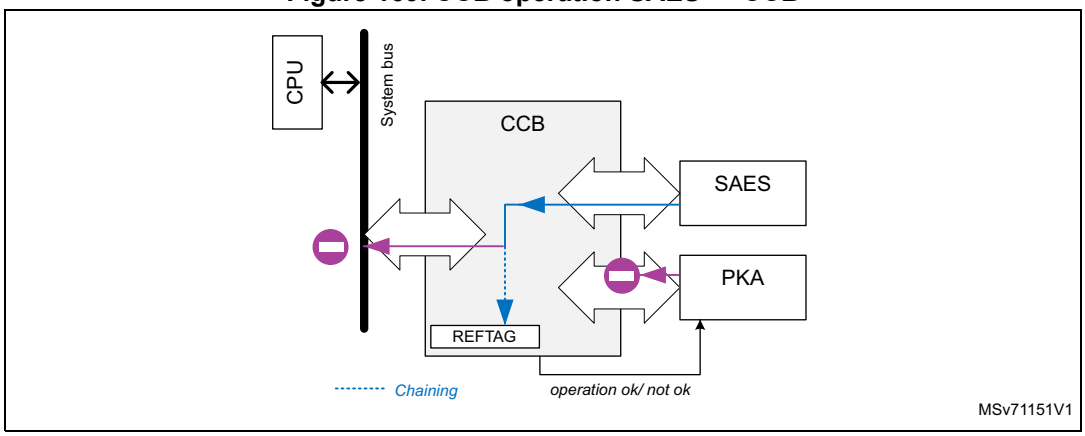


Chaining SAES → CCB

When the corresponding OPSTEP is selected, the CCB takes the read requests to SAES_DOUTR, and sends them to the system interconnect and to CCB_REFTAGRx holding the reference TAG information. For confidentiality reason, read data toward system bus are zeroed by the CCB.

If the CCB finds that the transfered value matches its reference information, it signals to the PKA that it can proceed with the protected PKA operation (START bit can be set).

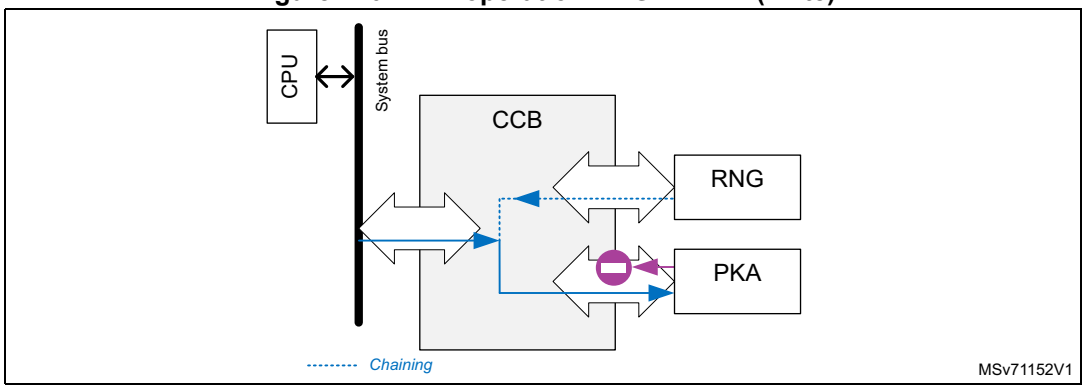
Figure 169. CCB operation SAES → CCB



Chaining RNG → PKA (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and substitutes these data with random numbers coming from RNG_DR. For consistency, the application can write 0xCCB to activate this chaining.

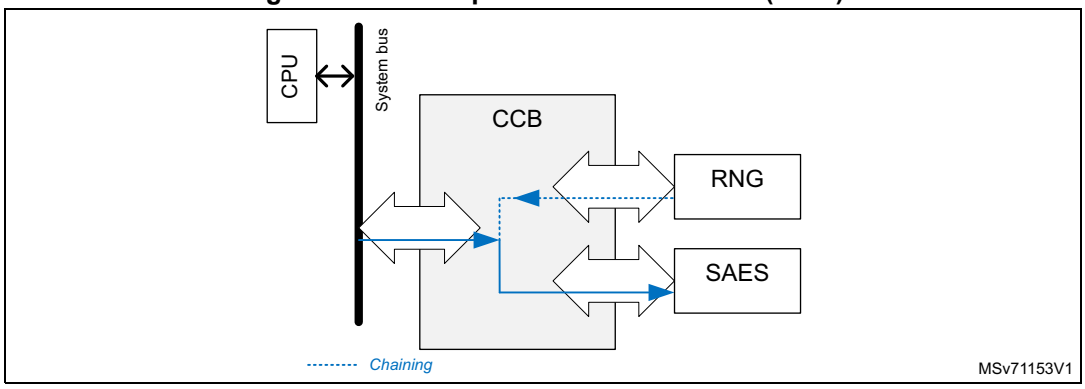
Figure 170. CCB operation RNG → PKA (write)



Chaining RNG → SAES (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to SAES_IVR1 to IVR3, and substitutes these data with random numbers coming from RNG_DR. For consistency, the application can write 0xCCB to activate this chaining.

Figure 171. CCB operation RNG → SAES (write)



25.4.5 CCB error management

When the CCB detects an error during a coupling and chaining operation (see *Section 25.5*), it sets accordingly OPERR[5:0] in CCB_SR (see *Section 25.7.2* for the possible errors).

When OPERR[5:0] ≠ 0 in CCB_SR, the CCB raises IPRST in CCB_CR, sets OPSTEP[4:0] to 0x1F, and sets CCB_BUSY bit to 1. The application must clear IPRST to be able to use again the CCB and the peripheral(s) used by the coupling and chaining operation.

25.5 CCB coupling and chaining operations

When an application must enforce private key confidentiality by wrapping them with a 256-bit hardware unique key, the following steps are needed:

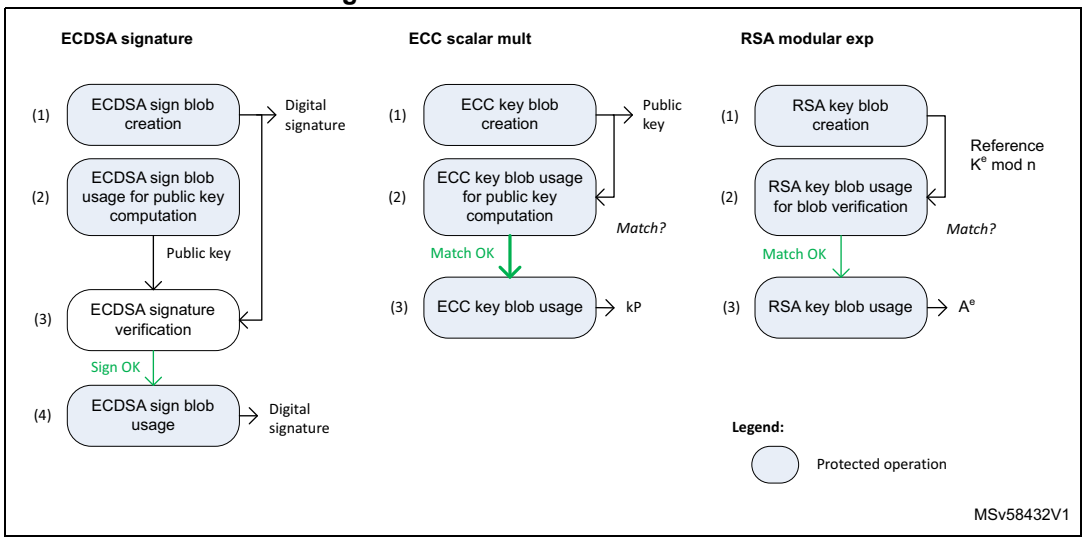
1. Select the protected operation in the PKA that uses this private key (ECDSA signature, scalar multiplication, modular exponentiation). Then, choose the CCOP value in the CCB that corresponds to the selected operation, as defined in this section. Then, as multiple cryptographic peripherals are used, make sure to follow the guidelines described in *Section 25.5.1*
2. When CCB is active (OPSTEP=0x1), select in the SAES the DHUK (with optional XOR with BHK), or unwrap in the SAES an existing blob encryption key that has been encrypted (in wrapped-key mo.de) using the DHUK. A 256-bit KEYSIZE must be selected.
   As the DHUK value is different when CCB is active or inactive, the operation of wrapping or unwrapping blob encryption key in SAES (with KMOD[1:0]=0x1) must be done when OPSTEP=0x1. Refer to operation with wrapped keys in *Section 28: Secure AES coprocessor (SAES)* for details.
3. When the application uses this protected operation in the PKA, it must load the related blob information in the PKA RAM and in SAES_DINR.
   – If the information exactly matches the conditions of the blob generation (choice and order of the writes of protected parameters in the PKA RAM, selection of blob

encryption key, use of original encrypted secrets), the CCB allows the application to set START bit in PKA_CR.

– If the blob information does not match the conditions of the blob generation, the application can only reset the sequence by setting IPRST bit in CCB_CR. CCB, PKA, and SAES become available when CCB_BUSY is cleared in CCB_SR.

4. At any time, the application can abort the executing protected operation by using the IPRST bit in CCB_CR.
5. When creating a blob, always use it once before trusting it to be stored in non-volatile memory or given to a less trusted code (see *Figure 172*). More specifically, each time a blob is created:
   – For ECDSA signature blob: use it to compute the public key, and use this key to verify the signature computed during the blob creation.
   – For ECC key blob: use it to compute the public key, and verify key coordinates match those of the key computed during the blob creation.
   – For RSA key blob: use it to compute a reference modular exponentiation and verify it matches the one computed during the blob creation.

Figure 172. Blob creation scenarios



This section describes how to do the above steps, for all available protected operations in the PKA. Note that:

- **Bold** elements in the following tables are key inputs that the application must save in non-volatile memory after the newly created blob has been confirmed by its first usage.
- The CCB automatically goes to the next step, updating the OPSTEP value, when the application clears or performs the end-step event described in tables of *Section 25.5.2* to *Section 25.5.4*.
- When an incremental read or write is mentioned, the application must write four bytes in the PKA memory, with an incremental memory address (modulo 0x8). Otherwise, the CMF error flag is triggered, and the process must be restarted.
- When writing to SAES_CR register the application must write the configuration expected by the CCB, otherwise an OPERR might be raised. Refer to the OPSTEP “CPU action” instructions in the application tables (*Table 189* through *Table 195*).
- When the CCB uses the RNG, the application must check in RNG_SR that no error occurred (SEIS = CEIS = 0), and that a random number is ready (DRDY = 1). In case of issue, RNGOKF is not set, and the application must ensure that the RNG is functional before running again the OPSTEP sequence that uses the RNG.
- For each OPSTEP>0, the application must manage the error flags that could occur in the RNG, SAES or PKA blocks. During a normal protected operation, no error flags are expected. Refer to *Section 25.4.5: CCB error management* for details on CCB error management.
- When CCB draws random number to compute ECC private keys, it implements the testing candidates method defined by NIST in FIPS PUB 186-4 section B.4.

Note: It is not possible to generate RSA key pairs with the CCB, they must be generated outside the device.

25.5.1 Preliminary steps before any protected operation

The following steps are required before running any operation described in this section:

1. Initialize the GTZC using the same security/privilege configuration protections for the SAES, PKA, RNG and CCB peripherals *Section 5: Global TrustZone® controller (GTZC)*. Otherwise OPERR=0x1A occurs (see *Section 25.7.2*).
2. Activate both RNG clocks, then initialize the peripheral to be fully functional, without pending errors.
3. Activate the PKA clock, then release the PKA reset. The PKA is functional once its RAM is automatically erased, and it has fetched the necessary random numbers. For further details refer to enabling/disabling PKA in *Section 30: Public key accelerator (PKA)*.
4. Activate the SAES clock, then release the SAES reset. The SAES is functional once it has fetched necessary random numbers.
5. Activate the CCB clock, set IPRST bit in CCB_CR, then clear it when CCB_BUSY=0x0 in CCB_SR.

Note: Step 4 can be done before step 3, as CCOP = 0x0.

25.5.2 ECDSA signature

This PKA operation is protected against side-channel attacks. When CCOP = 0x0, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0xC3, the private key d can be manipulated by the application only if encrypted, using a special blob called “ECDSA signature blob”.

ECDSA signature blob creation

The ECDSA signing operation is prepared by generating and then encrypting the private key d, and computing the reference GCM tag. This process is summarized *Table 189*, with coupling and chaining operations described in *Section 25.4.4*. This sequence starts by setting CCOP to 0xC0 or 0xC2 in CCB_CR.

Table 189. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0)

OP STEP	MODE in PKA	GCMPH in SAES	Coupling or chaining	End-step event(1)	CPU action in the step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x24 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt the blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set AES-GCM up in 256-bit encryption mode, with the DHUK or the decrypted key in register (if applicable). – Write hash z in the PKA RAM.
0x2		0x0	RNG -> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. SAES_IVR1 to SAES_IVR3 are loaded with randoms. – Read back **blob IV** in SAES_IVR registers.
0x3	0x24	0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in the SAES triggered after each 2x32-bit write.
0x6 (<i>=1)	0x24	0x2	RNG -> PKA	RNGOKF set	Write incrementally d in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if sanity check fails).
0x4 (<i>=0)			n/a	DATAOKF set	Write incrementally d in the PKA.
0x8	0x24		PKA -> SAES	DATAOKF set	– Read incrementally d from the PKA (writing 0xCCB magic numbers), and run AES encryption rounds. Special rule must be used for the last 64-bit word (see step 7 after the table). – Read **encrypted d** in SAES_DOUTR. – Repeat for each 16 bytes block of data.
0x9			RNG -> PKA	RNGOKF set, GCMPH = 0x3	– Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0xA(2)		0x3	N/A	START= 0x1	– Complete GCM final. – Read **reference tag** in the SAES. – Start the operation in PKA setting START bit in PKA_CR For the first two steps keep the SAES enabled (EN bit set).

Table 189. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0) (continued)

OP STEP	MODE in PKA	GCMPH in SAES	Coupling or chaining	End-step event(1)	CPU action in the step
0x19(3)	0x24	N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM reads as 0
0x1A		N/A	N/A	IPRST set in the CCB	– Read signature (r,s) in PKA RAM, and save it in SRAM. It is used (at least once) for blob creation verification. – Set IPRST in CCB
0x0	N/A	N/A	N/A	N/A	– Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. – Verify the newly created blob, running the *ECDSA signature blob use for public-key computation* with it.

1. The application must clear and/or perform the described event for the CCB to automatically go to the next step.
2. Once START bit is set SAES is reset, and OPERR linked to SAES are deactivated. Also when START bit is set, CCB sets OPSTEP to 0x19 after a few cycles.
3. SAES cannot be used for any operations, for security reasons.

The sequence described in *Table 189* is detailed below. Refer to the table for OPSTEP values and related coupling/chaining modes.

1. Write CCOP = 0xC0 or 0xC2 in CCB_CR.
2. Enable the PKA in the correct mode, writing in PKA_CR MODE=0x24 and EN=1 (repeat if the EN bit stays clear). Wait for INITOK=1 in PKA_SR.
3. Write the hash of the message in the PKA RAM
4. Initialize the ECC signature blob key in the SAES. This key can be the DHUK (KEYSEL = 0x1 or 0x4), or a key decrypted in 256-bit wrapped key mode by the DHUK (KEYSEL = 0x1 or 0x4, KMOD = 0x1, MODE = 0x1 or 0x2). See *Section 28: Secure AES coprocessor (SAES)* for details.
5. In the SAES, set the GCM chaining mode in encryption mode (CHMOD = 0x3, GCMPH = 0x0, MODE = 0x0, DATATYPE = 0x0). Then:
   a) Configure the 256-bit key size using KEYSIZE.
   b) Write SAES_IVRx, with any value (except SAES_IVR0 that must be equal to 0x2). For SAES_IV1 to IV3, random values are loaded into the SAES by the CCB. If an error occurs, RNGERRF is set in CCB_SR.
   c) Read back SAES_IVR1 to SAES_IVR3, and **save the value** in non-volatile memory after blob verification.
   d) Set EN in SAES_CR, and wait until CCF = 1 in SAES_ISR (end of GCM init).
   e) Clear CCF flag in SAES_ICR.
6. Select the header phase of the running GCM chaining mode (GCMPH = 0x1). Write the protected operation (ECDSA signature) mandatory parameters in the PKA RAM, as described in *Section 30: Public key accelerator (PKA)*. Do the following when writing data in the PKA:
   a) Write two 32-bit word in the PKA RAM, then wait until CCF is set in SAES_ISR. An SAES interrupt can be triggered when this flag is set.
   b) Clear CCF in SAES_ICR.

7. Select the payload phase of the running GCM chaining mode (GCMPH = 0x2). Write the private key *d* in the PKA RAM using the following sequence:
   a) If CCOP = 0xC0, write the private key d value in the PKA RAM. If CCOP = 0xC2, write the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The CCB then draws the value of *d* from the RNG, with PKA running a special sanity check to verify random compliance with NIST standard.
   b) Complete parameter information with the mandatory two 32-bit zeroed words. At the end of this process, the private key d of size *d_size* is written in the PKA RAM. Clear the error flag, then restart step 6 if the PKA sets RNGERRF when BUSY is cleared.
8. When the PKA sets its completion flag (RNGOKF if CCOP = 0xC2, DATAOKF if CCOP = 0xC0), clear it, then read the private key d in the PKA RAM using the sequence below (starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8):
   a) For each 64-bit word in the PKA RAM, write twice the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The target content of the PKA memory is transfered to SAES_DINR by the CCB. Wait until CCF is set in SAES_ISR (end of AES encryption). An SAES interrupt can be triggered when this flag is set.
   b) Clear CCF flag in SAES_ICR. Then read the encrypted private key d in SAES_DOUT, **to save it** in the non-volatile memory after blob verification.
   c) For the final 64-bit word (mandatory zeroed word): if the *d_size* is proportional to 16 bytes (for example, for 192- and 521-bit curves), the final 64 bits must be written to with two 32-bit magic numbers 0xCCB. If the *d_size* is not proportional to 16 bytes (for example, for 256- and 512-bit curves), the final 64 bits must be written with 0s.
9. When DATAOKF = 1 in PKA_SR (private key d ok), clear this flag. Then write to the integer k parameter addresses in the PKA memory, starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8. The CCB draws the value of k from the RNG. The PKA then runs a special sanity check to verify random compliance with NIST standard. Clear the error flag then restart this step if the PKA sets RNGERRF when BUSY is cleared.
10. When RNGOKF = 1 in PKA_SR (random k OK), clear this flag. Then select the final phase of the running GCM chaining mode (GCMPH = 0x3). Write the last GCM block in SAES_DINR. Then wait until CCF is set in SAES_ISR (end of GCM final).
11. Read SAES_DOUTR four times to retrieve the ECDSA signature blob reference tag. Save it in the non-volatile memory after blob verification. Do not disable SAES by clearing the EN bit.
12. Set START bit in PKA_CR to run the ECDSA signature operation.
13. When the PROCENDF completion flag is set save the ECDSA signature (r,s) of the hash z in SRAM to use it after using the new blob for public-key computation.
14. Clear CCOP by setting IPRST bit in CCB_CR. After CCB_BUSY is cleared in CCB_SR, clear the IPRST bit. The SAES, and PKA can again be used for any tasks. They are unprotected by the CCB until CCOP is written different from 0.

ECDSA signature blob usage for ECDSA signature

Table 190 summarizes the ECDSA signing operation using the encrypted private key d, the reference blob IV, and the computed GCM tag of *Table 189*. Coupling and chaining traffic options are described in *Section 25.4.4*. This sequence starts by setting CCOP to 0xC3 in the CCB_CR register.

Table 190. Blob use steps for ECDSA signature (CCOP = 0b11000011)

OPSTEP	MODE in PKA	GCMPH	Coupling or chaining operations	End-step event(1)	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x24 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt the blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set AES-GCM up in 256-bit encryption mode, with the DHUK or the decrypted key in register (if applicable). – Write hash z.
0x12		0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13	0x24	0x1	CPU –> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in the SAES, triggered after each 64-bit write.
0x14		0x2	SAES –> PKA	DATAOKF set	– Write encrypted d in SAES_DINR, and runs AES decryption round. – Write incrementally private key d in the PKA, using 0xCCB magic number. A special rule must be used for the last 64-bit write (see step 7 in the sequence detailed after this table). – Repeat for each 16-byte block of data.
0x16			RNG –> PKA	RNGOKF set, GCMPH = 0x3	Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0x17		0x3	SAES –> CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x24	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)		N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM is read as 0.
0x1A		N/A	N/A	N/A	Read **signature (r,s)** in the PKA RAM.

1. The application must clear and/or perform the described event for the CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons..

The sequence described in *Table 190* is detailed below. Refer to the table for the OPSTEP values and related coupling and chaining modes.

1. Write CCOP = 0xC3 in CCB_CR.
2. Enable the PKA in the correct mode, writing in PKA_CR MODE=0x24 and EN=1(repeat if the EN bit stays clear). Wait for INITOK=1 in PKA_SR.
3. Write the hash of the message in the PKA RAM.
4. Initialize the ECC signature blob key in the SAES. This key can be the DHUK (KEYSEL = 0x1 or 0x4), or a key decrypted in 256-bit wrapped key mode by the DHUK (KEYSEL = 0x1 or 0x4, KMOD = 0x1, MODE = 0x1). Refer to *Section 28: Secure AES coprocessor (SAES)* for details.
5. Set the GCM chaining mode in decryption in the SAES (CHMOD = 0x3, GCMPH = 0x0, MODE = 0x2, DATATYPE = 0x0). Then:
   a) Configure the 256-bit key size using KEYSIZE.
   b) Write to SAES_IVRx registers the value saved in non-volatile memory.
   c) Set EN in SAES_CR, and wait until CCF is set in SAES_ISR (end of GCM init).
   d) Clear CCF flag in SAES_ICR.
   e) Write to CCB_REFTAGR registers, the ECDSA signature blob reference tag value saved in non-volatile memory.
6. Select the header phase of the running GCM chaining mode (GCMPH = 0x1). Write the protected operation (ECDSA signature) mandatory parameters in the PKA RAM, exactly as done during the blob creation. See step 5 in *ECDSA signature blob creation* for details.
7. Select the payload phase of the running GCM chaining mode (GCMPH = 0x2). Write the protected private key d in the PKA RAM, following the below sequence (starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8):
   a) Writetwo 64-bit words of encrypted private key d in SAES_DINR. Wait until CCF is set in SAES_ISR (end of AES decryption). An SAES interrupt can be triggered when this flag is set.
   b) Clear the CCF flag in the SAES_ICR register. Then, for both 64-bit words, write four times the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The CCB transfers the clear-text content of SAES_DOUTR to the PKA.
   c) For the final 64-bit word (mandatory zeroed word): If *d_size* defined in step 6 in *ECDSA signature blob creation* is not proportional to 16-bytes (for example, for 256- and 512-bit curves), the final 64 bits must be written with 0s.
8. When DATAOKF = 1 in PKA_SR (private key d ok), clear this flag. Then write to the integer k parameter addresses in the PKA memory, starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8. The CCB draws the value of k from the RNG. The PKA then runs a special sanity check to verify random compliance with NIST standard. Clear the error flag, then restart step 7 if the PKA sets RNGERRF when BUSY is cleared.
9. When RNGOKF = 1 in PKA_SR (random k ok), clear this flag. Then select the final phase of the running GCM chaining mode (GCMPH = 0x3). Write the last GCM block in SAES_DINR. Then wait until CCF is set in SAES_ISR (end of GCM final).
10. Read four times SAES_DOUTR (returns 0s). If PKA parameters were written exactly like during the blob creation, and the correct private key d was used, and all expected DATAOKF/RNGOKF flags were raised then cleared, the application can set START bit

in PKA_CR to run the ECDSA signature operation. Otherwise, IPRST in CCB_CR must be used to restart the sequence from step 1.

11. Clear CCOP by setting IPRST in CCB_CR. After CCB_BUSY is cleared in CCB_SR, clear the IPRST bit. The SAES, and PKA can again be used for any tasks. They are unprotected by the CCB until CCOP in CCB_CR is written different from 0.

ECDSA signature blob use for public-key computation

The ECC public key computation using the encrypted private key d, the reference blob IV, and the computed GCM tag of *Table 189* is summarized in *Table 191*. The additional ECDSA verification at the end of the sequence must be used before saving a new ECDSA signature blob in non-volatile memory. Shown coupling and chaining traffic options are described in *Section 25.4.4*. This sequence starts by setting CCOP to 0x81 in the CCB_CR register.

Table 191. Blob use steps for signing public key computation (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable).
0x12	0x20	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13		0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in SAES, triggered after each 2x32-bit write.
0x14		0x2	SAES -> PKA	DATAOKF set, GCMPH = 0x3	– Write encrypted d in SAES DINR and run AES decryption round. – Write incrementallyscalar multiplier k in the PKA, using 0xCCB magic number. – Repeat for each 16 bytes block of data.
0x17		0x3	SAES -> CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x20	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)		N/A	N/A	PROCENDF set	The PKA computes the operation. PKA RAM is read as 0.
0x1A	0x20	N/A	N/A	N/A	– Read **public key point Q** in PKA RAM

Table 191. Blob use steps for signing public key computation (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1A	0x20	N/A	N/A	IPRST set in the CCB	– Read public key point Q in PKA RAM – Set IPRST in CCB
0x0	0x26	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. Before writing new ECDSA signature blob in non-volatile memory do the following: – Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x26 and EN=1. Wait for INITOK=1. – Write the hash z used during blob creation (see *Table 189*) – Write ECDSA verification parameters in PKA, including the public key coordinates computed in step 0x1A. – Start the operation setting START bit in PKA_CR – Wait for PROCENDF bit is set. If the signature is valid save the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

25.5.3 ECC scalar multiplication

This operation consists in the computation of k x P, where P is a point on a curve over prime fields, and “x” is the elliptic curve scalar point multiplication. The computation result is a point that belongs to the same curve, or a point at infinity.

The scalar multiplication operation is protected against side-channel attacks. When CCOP = 0x0 in CCB_CR, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0x81, the scalar multiplier k can be manipulated by the application only if encrypted, using a special blob called “ECC key blob”.

Scalar multiplication “ECC key” blob creation

The scalar multiplication operation is prepared by generating then encrypting the scalar multiplier k, and computing the reference GCM tag. This process is summarized in *Table 192*. Coupling and chaining operations described in *Section 25.4.4*. This sequence starts by setting CCOP to 0x80 or 0x82 in CCB_CR register.

Table 192. Blob creation steps for scalar multiplication (CCOP = 0b100000<i>0)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event[1]	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in 256-bit encryption mode with DHUK or the decrypted key in register (if applicable) – Write base point G coordinates as input in PKA RAM.
0x2		0x0	RNG –> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. Registers IVR1 to 3 are loaded with randoms. – Read back **blob IV** in IVR registers.
0x3	0x20	0x1	CPU –> PKA + SAES	GCMPH = 0x2	Write ECC scalar multiplication parameters in the PKA RAM, clearing CCF flags in SAES, triggered after each 64-bit write.
0x6 (<i>=1)	0x20	0x2	RNG –> PKA	RNGOKF set	Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0x4 (<i>=0)			N/A	DATAOKF set	Write incrementally k in the PKA.
0x8	0x20	0x2	PKA –> SAES	DATAOKF set, GCMPH = 0x3	– Read incrementally k from the PKA (writing 0xCCB magic numbers), and runs AES encryption rounds. – Read **encrypted k** in SAES DOUTR. – Repeat for each 16-byte block of data..
0xA[2]		0x3	N/A	START= 0x1	– Complete GCM final. – Read **reference tag** in the SAES. – Start the operation in PKA setting START bit in PKA_CR For the first two steps keep the SAES enabled (EN bit set).
0x19[3]	0x20	N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM reads as 0
0x1A		N/A	N/A	IPRST set in the CCB	– Read reference public key point Q in PKA RAM, and save it in SRAM to verify the new blob. – Set IPRST in CCB.
0x0	N/A	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. Once START bit is set SAES is reset, and OPERR linked to SAES are deactivated. Also when START bit is set, CCB sets OPSTEP to 0x19 after a few cycles.
3. SAES cannot be used for any operations, for security reasons.

Scalar multiplication blob use for example for ECC decryption

ECC scalar multiplication computation using the encrypted k, the reference blob IV and the computed GCM tag of *Table 192* is summarized in *Table 193*. Shown coupling and chaining traffic options are described in *Section 25.4.4*. This sequence starts by setting CCOP to 0x81 in CCB_CR register.

Table 193. Blob use steps for scalar multiplication (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally, decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable). – Write input point P coordinates in the PKA RAM. When computing a public key, write basepoint G coordinates.
0x12	0x20	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CBB_REFTAGR.
0x13		0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC scalar multiplication parameters in the PKA RAM, clearing CCF flags in the SAES, triggered after each 64-bit write.
0x14		0x2	SAES -> PKA	DATAOKF set, GCMPH = 0x3	– Write encrypted k in SAES_DINR, and runs AES decryption round. – Write incrementally scalar multiplier k in the PKA, using 0xCCB magic number. – Repeat for each 16 bytes block of data.
0x17		0x3	N/A	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x20	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR. Do not use SAES during this step (under reset).
0x19(2)				PROCENDF set	The PKA computes the operation. The PKA RAM is read as 0.
0x1A				N/A	Read **kP point coordinates** in the PKA RAM. If this point is the public key verify it matches the reference public key computed in *Table 192* before saving the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

25.5.4 RSA modular exponentiation

The modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod n.

This modular exponentiation operation (MODE = 0x3) is protected against side-channel attacks. When CCOP = 0x0, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0x45, the exponent e and phi value can only be manipulated encrypted by application, using a special blob called “RSA key blob”.

Modular exponentiation “RSA key” blob creation

Modular exponentiation operation is prepared by generating then encrypting the scalar multiplier *k* and computing the reference GCM tag. This process is summarized in *Table 194*. The coupling and chaining operations described in *Section 25.4.4*. This sequence starts by setting CCOP to 0x44 in CCB_CR register.

Table 194. Blob creation steps for modular exponentiation (CCOP = 0b01000100)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x3 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in 256-bit encryption mode with DHUK or the decrypted key in register (if applicable). – Write a constant K as operand A in the PKA RAM.
0x2		0x0	RNG -> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. SAES_IVR1 to 3 registers are loaded with randoms. – Read back blob IV in SAES_IVR registers.
0x3	0x03	0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write protected modular exponentiation parameters (PKA mode 0x3) in the PKA RAM, clearing CCF flags in the SAES triggered after each 2x32-bit write.
0x4	0x3	0x2	N/A	DATAOKF set	Write incrementally clear-text exponent e.
0x5				DATAOKF set	Write incrementally clear-text phi.
0x8			PKA -> SAES	DATAOKF set	– Read incrementallye from the PKA (writing 0xCCB magic numbers), and runs AES encryption rounds. – Read **encrypted e** in SAES_DOUTR. – Repeat for each 16 bytes block of data.
0x9				DATAOKF set, GCMPH = 0x3	– Read incrementally phi from the PKA (writing 0xCCB magic numbers), and run AES encryption rounds. – Read **encrypted phi** in SAES_DOUTR. – Repeat for each 16 bytes block of data
0xA(2)	0x3	0x3	N/A	START= 0x1	– Complete GCM final. – Read **reference tag** in the SAES. – Start the operation in PKA setting START bit in PKA_CR. For the first two steps keep the SAES enabled (EN bit set).

Table 194. Blob creation steps for modular exponentiation (CCOP = 0b01000100)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in step
0x19(3)	0x3	N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM is read as 0
0x1A		N/A	N/A	IPRST set in the CCB	– Read the reference modular exponentiation K^e mod *n* in PKA RAM, and save it in SRAM to verify the new blob. – Set IPRST in CCB
0x0	N/A	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. Once START bit is set SAES is reset, and OPERR linked to SAES are deactivated. Also when START bit is set, CCB sets OPSTEP to 0x19 after a few cycles.
3. SAES cannot be used for any operations, for security reasons.

Modular exponentiation blob use (for example for RSA decryption)

The modular exponentiation computation using the encrypted *e* and *phi*, the reference blob IV, and the computed GCM tag of *Table 194* is summarized on *Table 195*. Shown coupling and chaining traffic options are described in *Section 25.4.4*. This sequence starts by setting CCOP to 0x45 in CCB_CR register.

Table 195. Blob use steps for modular exponentiation (CCOP = 0b01000101)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x3 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable). – Write operand A in the PKA RAM. Write constant K as operand A defined in *Table 194* when verifying the created blob.

Table 195. Blob use steps for modular exponentiation (CCOP = 0b01000101)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x12	0x03	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13		0x1	CPU –> PKA + SAES	GCMPH = 0x2	Write protected modular exponentiation parameters (PKA mode 0x3) in the PKA RAM, clearing CCF flags in the SAES, triggered after each -bit write.
0x14		0x2	SAES –> PKA	DATAOKF set	– Write encrypted e in SAES_DINR, and runs AES decryption round. – Write incrementally exponent e in the PKA, using 0xCCB magic number. – Repeat for each 16-byte block of data.
0x15				DATAOKF set, GCMPH = 0x3	– Write encrypted phi in SAES_DINR, and runs AES decryption round. – Write incrementally phi in the PKA, using 0xCCB magic number. – Repeat for each 16-byte block of data.
0x17		0x3	SAES –> CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x03	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)				PROCENDF set	The PKA computes the operation, PKA RAM is read as 0.
0x1A				N/A	Read $\mathbf{A^e}$ **mod** n in the PKA RAM. If the result is K^e mod n verify it matches the reference computed in *Table 194* before saving the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

25.6 CCB interrupts

The CCB does not support any interrupt.

25.7 CCB registers

The following section provides information on registers.

Reserved bits (marked as Res.) must be kept at their reset values, unless otherwise specified.

For general information relative to register description, refer to Document conventions.

These registers are accessible through the AHB interface of the CCB (32-bit words only). All reads are supported. Writes accesses must be 32-bit word only (other access types generate an AHB error).

25.7.1 CCB control register (CCB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Following a CCB hardware reset, the above reset value applies if OPERR[5:0] = 0x0 and no TAMP_EVT bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: CCB reset

Set this bit to start or abort any PKA protected operation defined in *Section 25.5*. IPRST bit must be low while writing to any CCB register.

After setting this bit, the application must clear it when CCB_BUSY and OPSTEP are cleared in CCB_SR (IPRST clear is ignored otherwise). After this clear all CCB registers are read with their reset valuesSetting IPRST clears any data stored in the SAES and the PKA.

When IPRST and CCB_BUSY are set, SAES registers are not accessible (read as 0, write ignored), and writes to EN and START bits in PKA_CR are ignored.

This bit is set by CCB when OPERR[5:0] ≠ 0 or any TAMP_EVT bit is set.

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **CCOP[7:0]**: Coupling and chaining operation

These bits must be set according to the chosen PKA protected operation (see *Section 25.5*).

0x044: Modular exponentiation blob creation with priv key from the CPU

0x045: Modular exponentiation blob use

0x080: ECC scalar multiplication blob creation with priv key from the CPU

0x082: ECC scalar multiplication blob creation with priv key from the RNG

0x081: ECC scalar multiplication blob use, or ECDSA signature blob use for public key computation

0x0C0: ECDSA signature blob creation with priv key from the CPU

0x0C2: ECDSA signature blob creation with priv key from the RNG

0x0C3: ECDSA signature blob use for ECDSA signature

Note: When an operation not listed here is written by the application, the write to CCOP[7:0] is ignored. All these operations implement protections against side-channel attacks. Write to CCOP is ignored when CCB_BUSY is set in CCB_SR.

25.7.2 CCB status register (CCB_SR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is reset by a system reset, or after clearing IPRST in CCB_CR. The CCB reset input from the RCC has no effect on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TAMP_EVT5	TAMP_EVT4	TAMP_EVT3	TAMP_EVT2	TAMP_EVT1	TAMP_EVT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCB_BUSY
		r	r	r	r	r	r								r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OPERR[5:0]						Res.	Res.	Res.	OPSTEP[4:0]				
		r	r	r	r	r	r				r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **TAMP_EVTi**: Tamper i flag (i = 5 to 0)

0: No event

1: Tamper event i has been raised. Global tamper event is raised. See *Section 25.3* for details on the events.

This bit is cleared by writing 0 to IPRST bit in CCR_CR. When at least one TAMP_EVT bit is set, CCB sets IPRST bit.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **CCB_BUSY**: CCB busy

This flag is set when CCB is busy, or when OPSTEP[4:0] ≠ 0 following a CCB reset (hardware reset or use of IPRST in CCB_CR).

0: CCB idle

1: CCB busy or PKA RAM is being cleared following a peripheral reset.

When IPRST and CCB_BUSY are set, the SAES registers are not accessible (write ignored, read returns 0), and writes to EN or START bits in PKA_CR are ignored.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **OPERR[5:0]**: Operation error

When an error is detected during a protected PKA operation, the CCB raises IPRST in CCB_CR, and OPSTEP[4:0] is set to 0x1F. The bitfield is cleared by setting IPRST to 0.

0x0: No error detected during protected PKA operation (if applicable)

0x1: CCB RCC reset has been detected when OPSTEP > 0x0

0x2: SAES RCC reset has been detected when OPSTEP > 0x0

0x3: PKA RCC reset has been detected when OPSTEP > 0x0

0x4: RNG RCC reset has been detected when OPSTEP > 0x0

0x5: The RCC has gated the CCB clock when OPSTEP > 0x0.

0x6: The RCC has gated the SAES clock when OPSTEP > 0x0.

0x7: The RCC has gated the PKA clock when OPSTEP > 0x0.

0x8: The RCC has gated the RNG clock when OPSTEP > 0x0.

0x9: Random in SAES_IV1 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xA: Random in SAES_IV2 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xB: Random in SAES_IV3 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xC: When OPSTEP = 0x17 SAES GCM TAG comparison with CCB_REFTAG failed, or a register different from SAES_DOUT was read in the SAES.

0xD: KEYSIZE ≠ 256 bits in SAES_CR when the SAES is enabled, and OPSTEP > 0x0.

0xE: CHMOD ≠ GCM in SAES_CR when 0x1 < OPSTEP < 0x18, or the SAES is not enabled, or KEYVALID is cleared when GCMPH > 0x0.

0xF: KMOD different from wrapped key mode in SAES_CR when the SAES is enabled, and OPSTEP = 0x1, or 0x1A.

0x10: KEYSEL ≠ 0x1 or 0x4 in SAES_CR when the SAES is enabled, KEYVALID is set, and OPSTEP = 0x1.

0x11: GCMPH not correct (not incremented) in SAES_CR when OPSTEP > 0x1.

0x12: MODE in SAES_CR does not match the expected value (selected by CCOP in CCB_CR) when OPSTEP > 0x1.

0x13: BUSY = 1 in SAES_SR when OPSTEP= 0x18 or 0x19. SAES must not be used when PKA is computing a protected operation.

0x14: EN = 0 in PKA_CR when OPSTEP > 0x2 (except 0x12). In other words, PKA must be enabled in any OPSTEP except 0x1, 0x2, or 0x12.

0x15: INITOK = 0 in PKA_SR when OPSTEP > 0x2 (except 0x12).

0x16: Authorized by local firewall, but unexpected AHB error detected when OPSTEP > 0x0.

0x17: LMF = 1 in PKA_SR (only ECDSA verification available) when OPSTEP > 0x0.

0x18: CCB_REFTAGRx registers are equal to 0 when OPSTEP > 0x12.

0x1A: Security, privilege protections inconsistent (see *Section 5: Global TrustZone® controller (GTZC)* for the peripherals SAES, PKA, RNG, and CCB when OPSTEP > 0x0. This error is also triggered when the protections change while OPSTEP > 0x0,

0x1B: MODE in PKA_CR does not match the expected value (selected by CCOP in CCB_CR) when OPSTEP > 0x0.

0x1C: OPSTEP end-step event not found at OPSTEP= 0x0 (SAES KEYVALID is set). Use CCB IPRST before starting a coupling and chaining operation.

0x1D: OPSTEP end-step event not found at OPSTEP = 0x1.

0x1E: OPSTEP end-step event not found at OPSTEP = 0x2.

0x1F: OPSTEP end-step event not found at OPSTEP = 0x3.

0x20: OPSTEP end-step event not found at OPSTEP = 0xA.

0x21: OPSTEP end-step event not found at OPSTEP = 0x12.

0x22: OPSTEP end-step event not found at OPSTEP = 0x13.

0x23: OPSTEP end-step event not found at OPSTEP = 0x17.

For error codes 0x1C to 0x23, details on end-step events are found in *Section 25.5*. Those errors are also triggered when values written in SAES_CR register do not correspond to the selected CCOP and the expectations in the current OPSTEP.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **OPSTEP[4:0]**: Operation step

This read-only information indicates the current step of the coupling and chaining operation (indicated by CCOP). See *Section 25.5* for details. .

This bitfield can be cleared only using IPRST in CCB_CR. OPSTEP[4:0] = 0x1F when the IPRST procedure is in progress.

25.7.3 CCB reference tag register (CCB_REFTAGRx)

Address offset: 0x010 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

Writes to this register are ignored when CCB_BUSY is set in the CCB_SR register. The register is cleared when IPRST is set and CCB_BUSY is cleared.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFTAG{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFTAG{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REFTAG{i + 32 * x}**: Reference tag bits {i + 32 * x} (i = 31 to 0)

The application must write in these registers, the reference GCM tag of the required operation blob (respectively ECC signature, ECC key, or RSA key). This enables the PKA protected operation (respectively ECC signature/public key computation, scalar multiplication, or modular exponentiation). Refer to *Section 25.5* for details.

Write to this register is ignored unless OPSTEP = 0x12 in CCB_SR.

25.7.4 CCB register map

Table 196. CCB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CCB_CR	IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOP[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0x004	CCB_SR	Res.	Res.	TAMP_EVT5	TAMP_EVT4	TAMP_EVT3	TAMP_EVT2	TAMP_EVT1	TAMP_EVT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCB_BUSY	Res.	Res.	OPERR[5:0]						Res.	Res.	Res.	OPSTEP[4:0]				
	Reset value			0	0	0	0	0	0								0			0	0	0	0	0	0				0	0	0	0	0
0x010 + 0x4 * x (x=0 to 3)	CCB_ REFTAGRx	REFTAG{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

26 True random number generator (RNG)

26.1 RNG introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is an NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP 800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

26.2 RNG main features

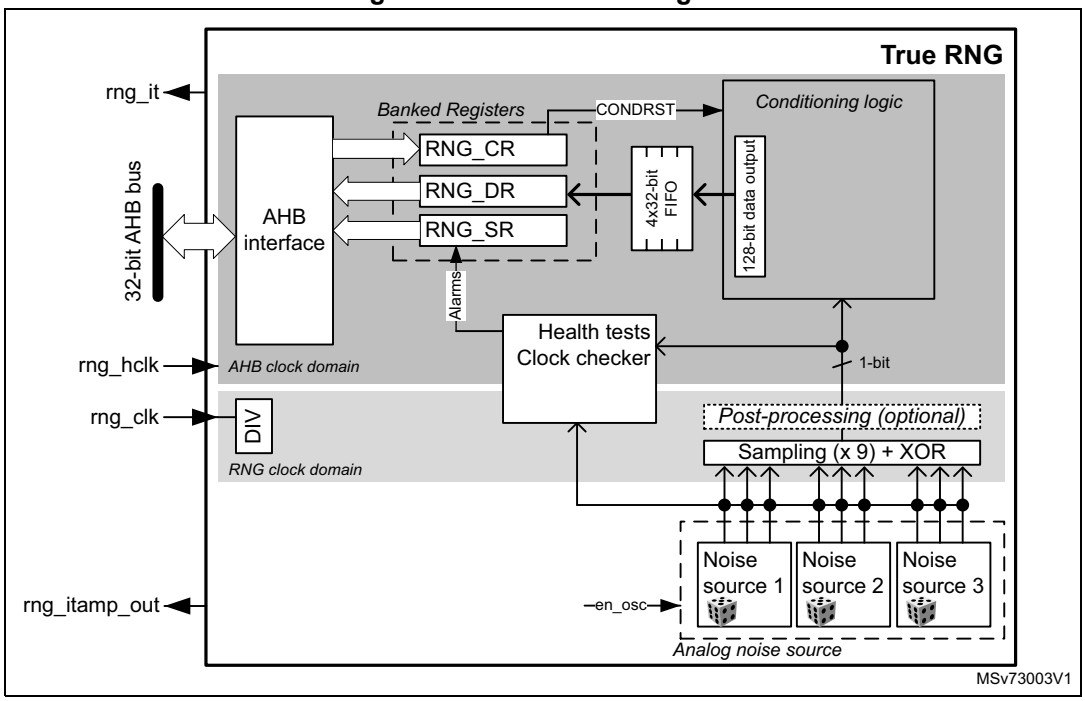
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by an NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the default configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- Embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- Can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- Has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

26.3 RNG functional description

26.3.1 RNG block diagram

Figure 173 shows the RNG block diagram.

Figure 173. RNG block diagram



26.3.2 RNG internal signals

Table 197 describes internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 197. RNG internal input/output signals

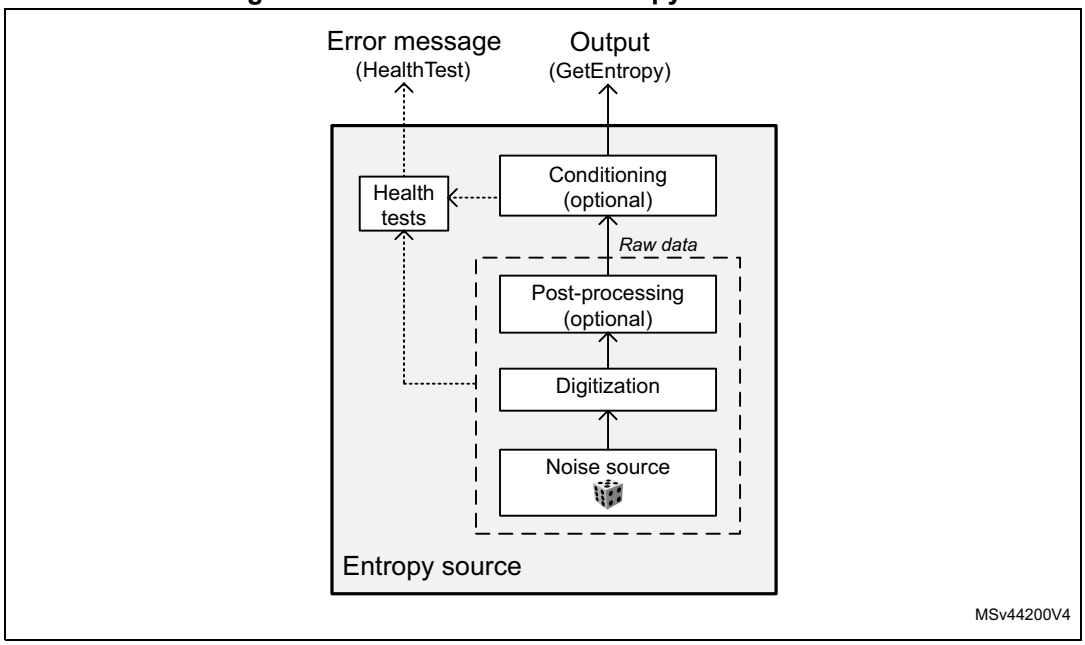
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	Digital output	RNG internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples. A reset and a new initialization are needed to use it again.

26.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary, it integrates all the required NIST components shown in *Figure 174*: an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block, and two interfaces used to interact with the entropy source (GetEntropy and HealthTest).

Figure 174. NIST SP800-90B entropy source model



Noise source

This component contains the nondeterministic, entropy-providing activity, ultimately responsible for the uncertainty associated with the bit string output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x3), each based on three XOR-ed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 26.3.8*. Multiple oscillators are also disabled for configuration A (see *Table 199*).
- The XOR-ing of all the noise sources into a single analog output.
- A sampling stage of this output is clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent from the AHB interface clock frequency (rng_hclk), with the possibility, for the software, to decrease the sampling frequency by using the integrated divider.

Note: The recommended clock frequencies and associated divider value are detailed in Section 26.6.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In the non-NIST configuration B (as defined in *Section 26.6.2*) a normalization debiasing is applied: half of the bits are taken from the sampled noise source, the other half from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bit string output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The time intervals required between two random number generations, and between the RNG initialization and availability of the first sample are detailed in *Section 26.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 26.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When the interrupts are enabled, an interrupt is generated when this flag transitions from 0 to 1. The interrupt is then cleared automatically by the RNG, as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts, then operates as expected, obtaining the assurance that failures are caught quickly, with high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the default RNG_HTCR0 configuration.

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 28 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 684 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source (XORed output and individual oscillator outputs, see *Figure 173: RNG block diagram*):
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous test:

  - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than the AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 26.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 26.3.7*.

Note: An interrupt can be generated when an error is detected.

The health test thresholds are modified by changing the value in the corresponding RNG_HTCRx registers. See Section 26.6 for details.

26.3.4 RNG initialization

The RNG simplified state machine is shown in *Figure 175*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering the startup behavior test on the raw data samples and known-answer tests.
3. Start-up health tests are completed, with the conditioning stage ready to use.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 26.5*.

Figure 175. RNG initialization overview

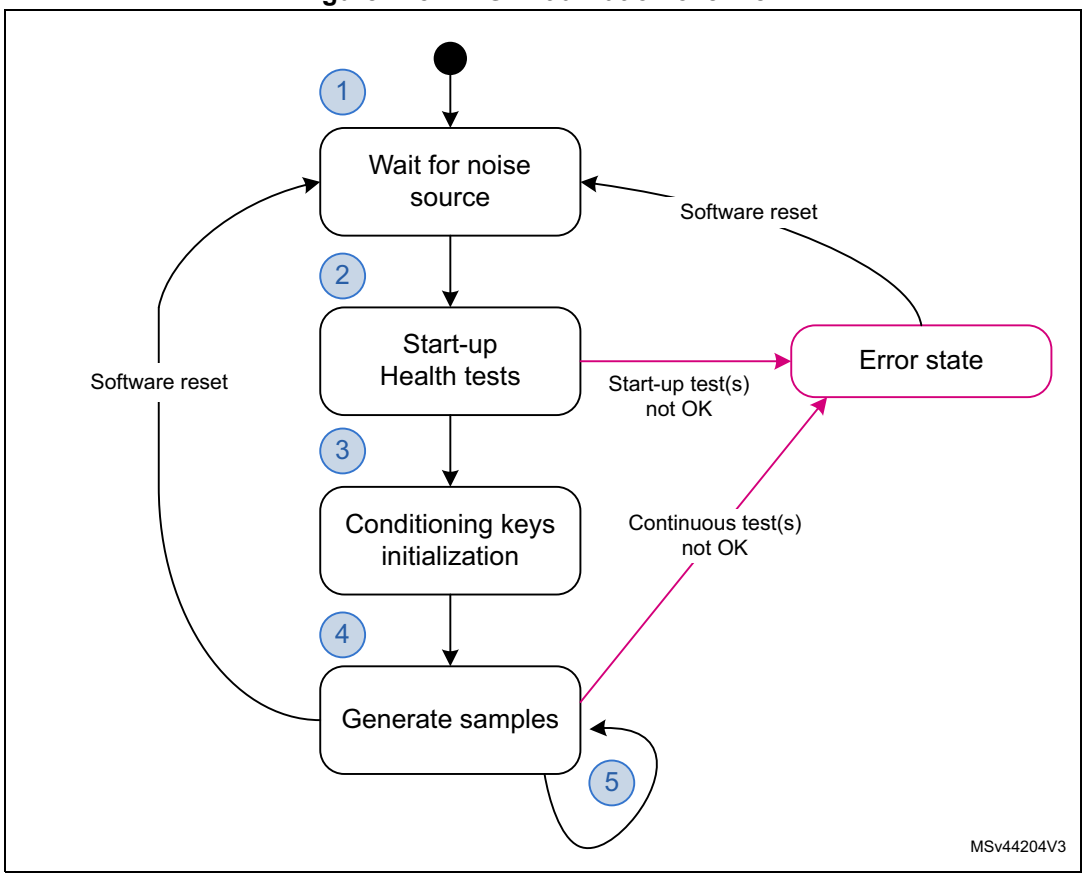


Figure 175 also highlights a possible software reset sequence, implemented by:

1. Write RNGEN = 0, then wait for BUSY = 0 in the RNG_SR register.
2. Write bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
3. Write RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
4. Wait for a random number to be ready, after initialization completes.

Note: When the RNG is reset by RCC (hardware reset), the configuration for optimal randomness is lost in the registers.

26.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check in *Section 26.6* if a specific RNG configuration is required for the application.
   – If this is the case, when bits RNGEN and BUSY=0, write in the RNG_CR register bit CONDRST = 1, together with the correct RNG configuration. Then perform a second write with bit CONDRST = 0, interrupt enable bit IE = 1, and RNG enable bit RNGEN = 1.

- If this is not the case, perform a write to the RNG_CR register with interrupt enable bit IE = 1 and RNG enable bit RNGEN = 1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 26.3.7* must be used.

To run the RNG in polling mode the following steps are recommended:

1. Check in *Section 26.6* if a specific RNG configuration is required for the application.
   - If this is the case, when bits RNGEN and BUSY = 0, write in the RNG_CR register bit CONDRST = 1, together with the correction RNG configuration. Then perform a second write to the RNG_CR register with bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If this is not the case, only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If the above conditions are true, read the content of the RNG_DR register up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, use the recovery sequence described in *Section 26.3.7*.

Note: When data are not ready (DRDY = 0) RNG_DR returns 0. It is recommended to verify that RNG_DR is different from 0. When this happens, a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required, it is possible to select a faster configuration by using the configuration B, described in *Section 26.6*. The gain in random number generation speed is summarized in *Section 26.5*.

Low-power operation

If power consumption is a concern, low-power strategies can be used, as described in *Section 26.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 26.3.3*.

26.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 26.6*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 26.3.1* for details (AHB and RNG clock domains).

26.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 26.3.6*), and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

When CECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 26.3.5*.

Note: CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred.

When SECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 26.3.5*.

To manage a seed error, follow the recommended sequence below. It requires the CONFIGLOCK bit to be cleared.

- When ARDIS is cleared (auto-reset enabled):
  - Clear the SEIS bit in the RNG_SR register
  - If SECS is cleared RNG can be used as usual. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case, executes following step 1)
- When ARDIS is set (auto-reset disabled) the following sequence must be used to recover from a seed error.

1. Read both RNG_HTSRx registers. Using the previous HTSR values saved in RAM, compute the accumulated number of errors per oscillator (can be RPERR or ADERR errors). In a *nsmr* variable, mask the oscillators with accumulated number of errors greater than 3. This variable is used in step 3). Save the updated HTSR values in RAM.
   - During the first iteration, initialize *nsmr* variable with the RNG_NSCR value.
2. If fewer than 6 oscillators are active in the *nsmr* variable, reinitialize it with the RNG_NSCR value.
   - When RNG configuration is Configuration B, and the third time you reach this loop, go to Step 3 changing the RNG configuration to Configuration C. Both configurations are defined in *Section 26.6.2*.
3. Write CONDRST at 1 in RNG_CR, then update RNG_NSMR with *nsmr* value. Refer to *Section 26.7.8* for details.
4. Write CONDRST at 0 to finalize the reset of the RNG conditioning logic.
5. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY bit= 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
6. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1). If SECS is cleared, restart the generation of random numbers by setting RNGEN in RNG_CR.

If application has set the CONFIGLOCK bit (not recommended), do the following:

- When ARDIS is cleared (auto-reset enabled) clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.
- When ARDIS is set (auto-reset disabled), the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY = 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

RNG tamper errors

When an unexpected error is found by the RNG, an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral, using the central reset management or the global SoC reset. A proper initialization of the RNG is required again.

Note: Seed errors do not trigger tamper errors.

26.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while the RNGEN = 0, the following features are available to the software:

- If there are valid words in the output buffer, four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register, four additional random numbers can be still be read from the RNG_DR register. If it is not the case, RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode), and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 26.5*.

When disabling the RNG, the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when the RNGEN bit is set to 1, unless the application resets the conditioning logic using the CONDRST bit in the RNG_CR register.

When the application wants to gate the RNG kernel clock, the software must issue the following sequence: first, disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR, and finally, disable the RNG kernel clock using the RCC.

Before entering a power mode where RNG is deactivated (like Stop mode), the software must first disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR.

Note: The power modes where RNG is deactivated (retained or not available) can be found in the PWR section.

26.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 26.3.7*
- Clock error, see *Section 26.3.7*

Dedicated interrupt enable control bits are available as shown in *Table 198*.

Table 198. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 26.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources globally by using the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

26.5 RNG processing time

In the recommended configuration described in *Table 199*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

If f_{RNG}= 48 MHz, $f_{threshold}$ = 1.6 x f_{RNG} = 77 MHz. The value of N is defined in *Section 26.6*.

Note: When CLKDIV is different from 0, f_{RNG} must take into account the internal divider ratio.

26.6 RNG entropy source validation

26.6.1 Introduction

To assess the amount of entropy available from the RNG, the peripheral has been tested using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

26.6.2 Validation conditions

The RNG true random number generator has been tested in the following conditions:

- RNG clock rng_clk = 48 MHz, with CED bit cleared in RNG_CR.
- Configurations are described in *Table 199* and *Table 200*. Only configuration A can be certified NIST SP800-90B. Refer to *Table 201* to select the best configuration for the application.
- Configuration B or C can be used when configuration A is not present in AN4230, or is not flagged as certified.

Table 199. RNG configurations

Configuration	RNG_CR bits					Loop number (N)	RNG_HTCR0 register	RNG_NSCR register
	NISTC bit	RNG_ CONFIG1 [7:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0]	RNG_ CONFIG3 [3:0](1)			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program.*							
B	1	0x83	0x0	0x0	0xF	3	0x0000 AAC7(2)	Default
C	1	0x84	0x5(3)	0x0	0xF	4		Default

1. RNG_CONFIG1[3:0] = i defines the loop number N. If i = 0, N =1. If NISTC = 0, N = 2 regardless of i.
2. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).
3. The noise source sampling must be 48 MHz or less. If the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 26.7.1* for details.

Table 200. Additional health test configurations

Configuration	RNG_HTCR1 register (oscillator set 1)	RNG_HTCR2 register (oscillator set 2)	RNG_HTCR3 register (oscillator set 3)
A	Refer to NIST compliant RNG configuration table in AN4230 available from www.st.com.		
B and C	default	default	default

Table 201. Configuration selection

Section criteria	Configuration A	Configuration B	Configuration C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Not certified	
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using the AIS-31 test suite (T0 to T8).
2. When the speed is not enough for the application, an NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*Introduction to random number generation validation using the NIST statistical test suite for STM32 MCUs and MPUs*", available on *www.st.com*.

26.7 RNG registers

The RNG is associated with a control register, a data register, and a status register.

26.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]			
rs	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG configuration lock

Set this bit to lock the RNG configuration. Since CONFIGLOCK prevents the implementation of robust seed error management described in *Section 26.3.7*, it is recommended not to use it.

0: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_NSMR, RNG_NSCR, RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that sets any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

Before setting this bit to 1 RNGEN must be set to 0, and BUSY flag must be cleared in RNG_SR.

When CONDRST is set to 0, wait for BUSY flag to be cleared in RNG_SR. CONDRST bit then reads as 0.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **RNG_CONFIG1[7:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 26.6*.

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration for 128-bit output, two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 26.6* for recommended configurations.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset is enabled

1: Auto-reset is disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 26.3.7*.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enabled.

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS flag is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

This bit enables/disables the true random number generator.

0: True random number generator disabled.

1: True random number generator enabled.

Clearing RNGEN powers off the analog noise sources and gates the logic clocked by the RNG clock.

When clearing this bit verify that BUSY flag is cleared in RNG_SR before updating this register again.

Bits 1:0 Reserved, must be kept at reset value.

26.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0	r		r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)

1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 4 **BUSY:** Busy

This flag indicates whether RNG is idle or busy.

0: Idle

1: Busy

RNG is flagged as busy when a conditioning soft reset (using CONDRST) is in progress. It is also flagged busy when RNG is activated by application setting RNGEN bit, or when a connected peripheral fetches randoms via the dedicated RNG hardware bus.

More information on BUSY usage can also be found in *Section 26.3.8*.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

– Start-up or continuous repetition count test on noise source failed.

– Start-up or continuous adaptive proportion test on noise source failed.

– Start-up post-processing/conditioning sanity check failed.

After enabling the RNG, SECS bit clears once startup health tests completes successfully. If SECS does not clear, run the software reset sequence with CONDRST described in *Section 26.3.4*.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

This flag is raised to indicate valid (non-0) random data available in the RNG_DR register. It is cleared by hardware upon reading the RNG_DR register (which empties the RNG buffer). The DRDY flag can also be raised upon disabling RNG (upon clearing the RNGEN bit).

0: No valid random data ready.

1: Valid random data ready.

Raising DRDY generates an interrupt if enabled with the IE bit of the RNG_CR register.

26.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

This register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, valid when DRDY = 1. When DRDY = 0, the RNDATA value is 0.

The 0 value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

26.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0000 01FF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

26.7.5 RNG health test control register x (RNG_HTCRx)

Address offset: 0x010 + 0x4 * x (x=0 to 3)

Reset value: 0x0000 72AC, 0x0003 FFFF, 0x0003 FFFF, 0x0003 FFFF

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTCFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests (repetition count test, adaptive proportion test) at position x. See *Section 26.6* for the recommended value. With its default value, health tests have a very small probability of triggering any error at position x.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

26.7.6 RNG health test status register 0 (RNG_HTSR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPERR3	RPERR2	RPERR1	RPERRX
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:1 **RPERR{i}:** Repetitive error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds a repetitive test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Repetition count test is defined in *Health checks* section.

0: No repetitive test error found in oscillator *i*.
1: Repetitive test error found in oscillator *i*.

Bit 0 **RPERRX:** Repetitive error after the XOR

This bit is set when the health test logic finds a repetitive test error after the XOR clock tree (see *Figure 173*). The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No repetitive test error found after the XOR clock tree.
1: Repetitive test error found after the XOR clock tree.

26.7.7 RNG health test status register 1 (RNG_HTSR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:1 **ADERR{i}:** Adaptative error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds an adaptative test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Adaptive proportion test is defined in *Health checks* section.

0: No adaptative test error found in oscillator *i*.
1: Adaptative test error found in oscillator *i*.

Bit 0 **ADERRX:** Adaptative error after the XOR

This bit is set when the health test logic finds an adaptative test error after the XOR clock tree (see *Figure 173*). The bit is cleared when application clears SEIS bit in RNG_SR If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No adaptative test error found after the XOR clock tree.
1: Adaptative test error found after the XOR clock tree.

26.7.8 RNG noise source mask register (RNG_NSMR)

Address offset: 0x030

Reset value: 0x0000 01FF

Writing to this register is ignored if CONFIGLOCK = 1. It is recommended to have at least 6 oscillators running when loop number N = 3 (or higher). See *Section 26.6.2* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOSC 9	MOSC 8	MOSC 7	MOSC 6	MOSC 5	MOSC 4	MOSC 3	MOSC 2	MOSC 1
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **MOSC{i}:** Mask oscillator i (i = 1 to 9)

Clear this bit to masks the oscillator *i*, and deactivate its related health tests. The bit has no effect if the oscillator *i* is deactivated with the bit RNG_NSCR[i].

Application can mask an oscillator when it is not contributing to the generation of entropy.

0: Oscillator *i* is masked, and related health tests are deactivated.
1: Oscillator *i* is not masked. Related health tests are running.

26.7.9 RNG register map

Table 202. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]				RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
	Reset value																										0	0	0		0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]		EN_OSC2[2:0]			EN_OSC1[2:0]			
	Reset value																								1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR0	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0
0x014	RNG_HTCR1	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	RNG_HTCR2	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x01F	RNG_HTCR3	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x020	RNG_HTSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPER3	RPERR2	RPERR1	RPERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	RNG_HTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	RNG_MSMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOSC9	MOSC8	MOSC7	MOSC6	MOSC5	MOSC4	MOSC3	MOSC2	MOSC1
	Reset value																								1	1	1	1	1	1	1	1	1

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

27 AES hardware accelerator (AES)

In this section, any references to SAES only apply to STM32C59x/5A3 products. The STM32C53x/542/55x/562 products do not have the SAES peripheral.

27.1 AES introduction

The AES hardware accelerator (AES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

AES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits.

AES has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

27.2 AES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data

- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if AES needs to process another message with a higher priority, then resume the original message

27.3 AES implementation

The devices have one AES peripheral, implemented as per the following table. It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 203. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

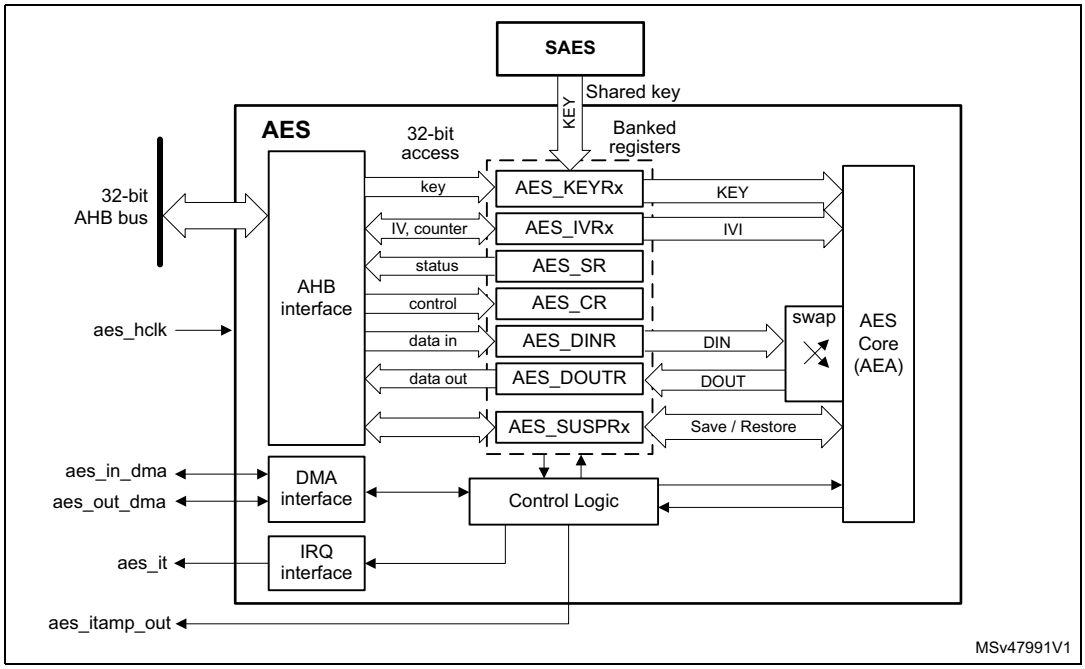
1. X = supported.

27.4 AES functional description

27.4.1 AES block diagram

Figure 176 shows the block diagram of AES.

Figure 176. AES block diagram



27.4.2 AES internal signals

Table 204 describes the user relevant internal signals interfacing the AES peripheral.

Table 204. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	AES incoming data DMA single request/acknowledge
aes_out_dma	Input/Output	AES processed data DMA single request/acknowledge
aes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, AES automatically clears key registers. A reset is required for AES to be usable again.

27.4.3 AES reset and clocks

The AES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit controlled through the RCC.

27.4.4 AES symmetric cipher implementation

The AES hardware accelerator (AES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 205*. Those functions can be certified NIST PUB 140-3.

Table 205. AES approved symmetric key functions

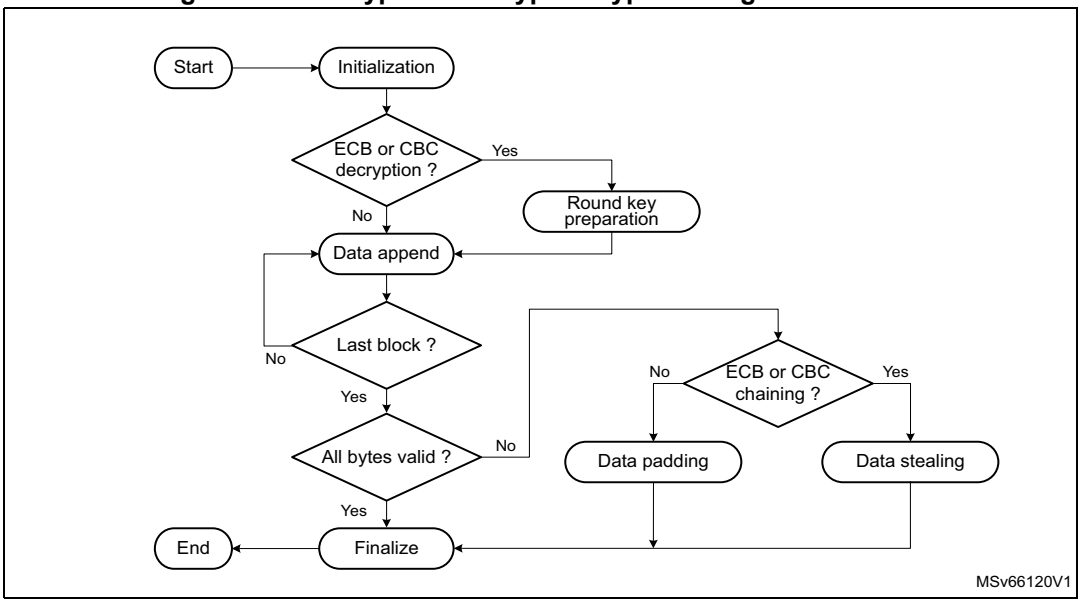
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

AES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any AES processing, following the sequence described in *Section 27.4.8*.

27.4.5 AES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 177. Encryption/ decryption typical usage



Initialization

The AES peripheral is initialized according to the chaining mode. Refer to *Section 27.4.9: AES basic chaining modes (ECB, CBC)* and *Section 27.4.10: AES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 27.4.7: AES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral when KEYVALID is set, by setting the EN bit of the AES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_ISR register, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).
3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the AES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES, by setting the CCFIE bit of the AES_IER register.
2. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the AES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the AES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and AES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in *Section 27.6: AES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 27.4.8* is not supported.

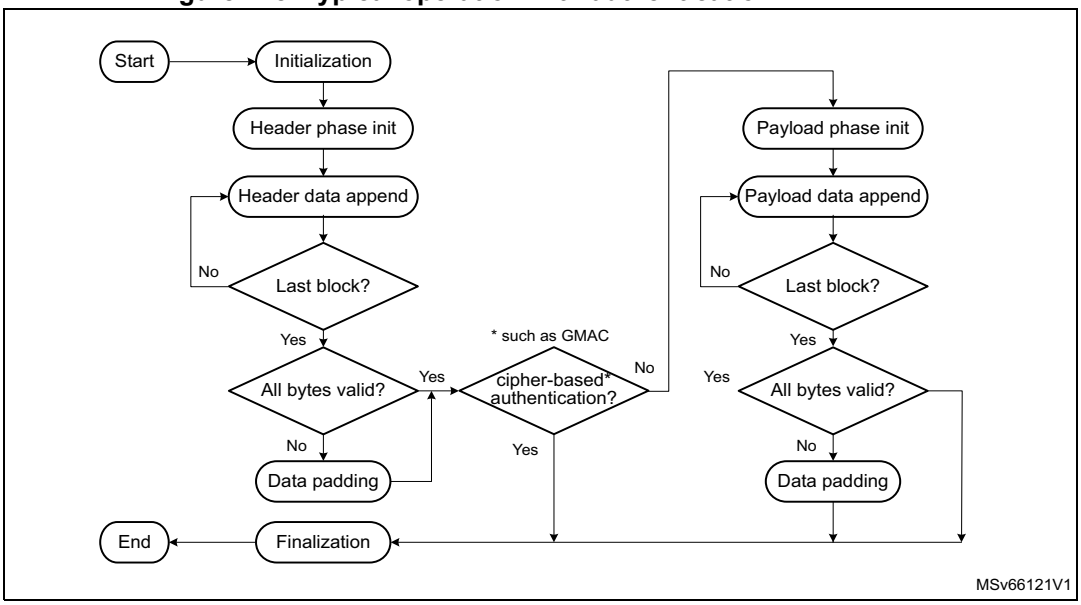
Note: The CCF flag has no use with this method because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

27.4.6 AES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 178. Typical operation with authentication



Section 27.4.11: AES Galois/counter mode (GCM) and *Section 27.4.13: AES counter with CBC-MAC (CCM)* describe detailed sequences supported by AES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by AES is described in *Section 27.4.12: AES Galois message authentication code (GMAC)*.

27.4.7 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since AES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since AES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

27.4.8 AES suspend and resume operations

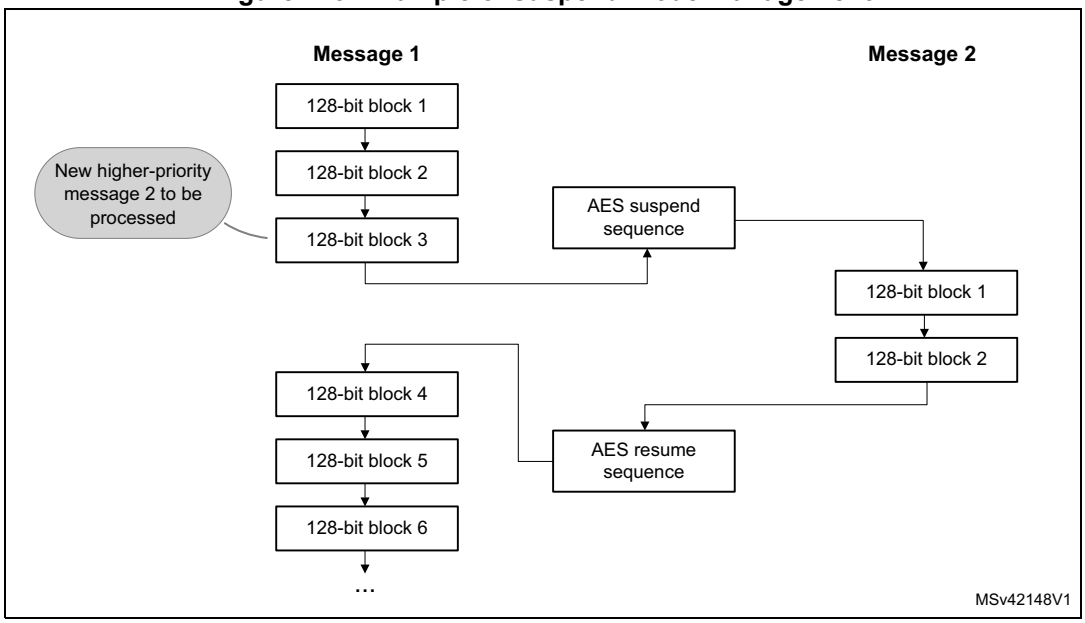
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as AES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

Figure 179 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 179. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

27.4.9 AES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 180 and *Figure 181* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 180. ECB encryption

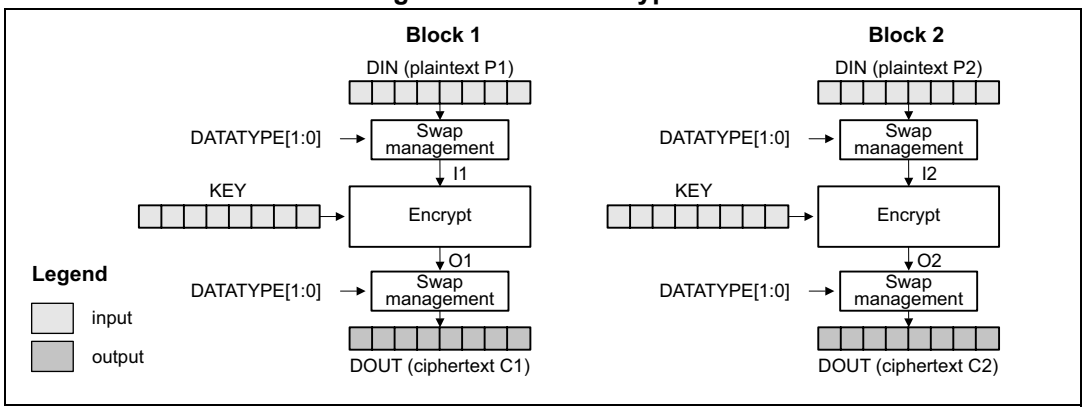
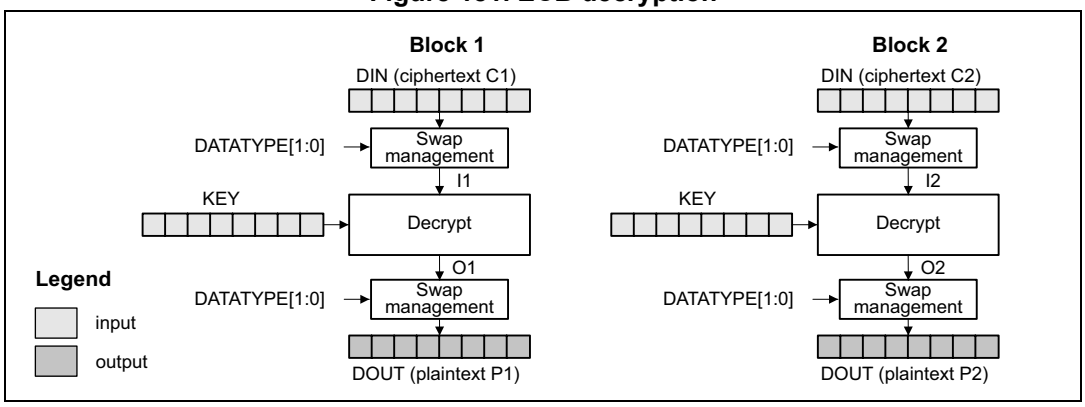


Figure 181. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 182 and *Figure 183* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 182. CBC encryption

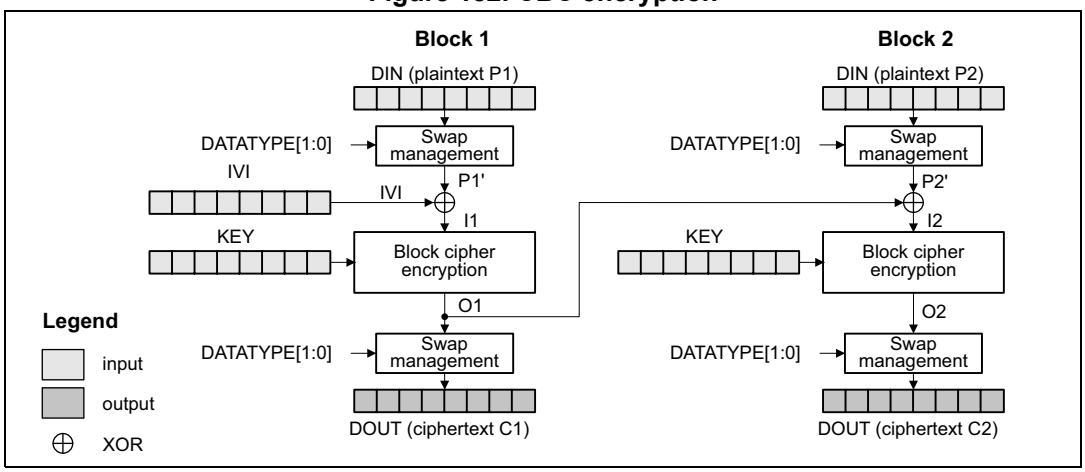
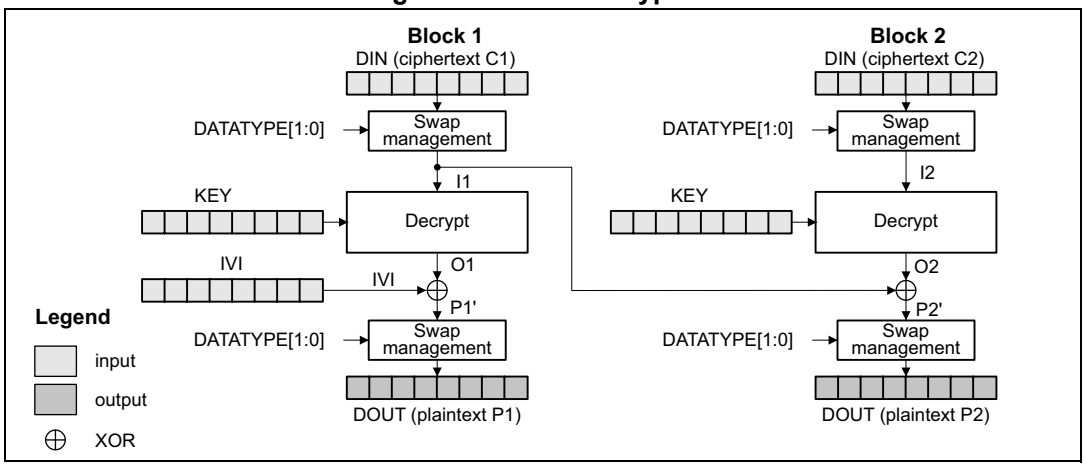


Figure 183. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 27.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 27.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 27.4.7*.
   b) Append the cleartext block into AES as described in *Section 27.4.5*, then read the AES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 27.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 27.4.14*).
4. Wait until KEYVALID is set (the key loading completed).
5. Enable the AES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
6. Wait until the CCF flag in the AES_ISR register is set.
7. Clear the CCF flag, by setting the CCF bit of the AES_ICR register. The decryption key is available in the AES core and AES is disabled automatically.
8. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
9. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
10. Enable the AES peripheral, by setting EN.
11. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 27.4.7*.
    b) Append the ciphertext block into AES as described in *Section 27.4.5*, then read the AES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
12. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. Read four times the AES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the AES_IVRx registers are altered during the data processing).

5. Disable the AES peripheral, by clearing EN.
6. Save the AES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Restore the AES_CR register (with correct KEYSIZE) then restore the AES_KEYRx registers. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 27.4.14*).
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the AES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the AES peripheral, by setting EN.

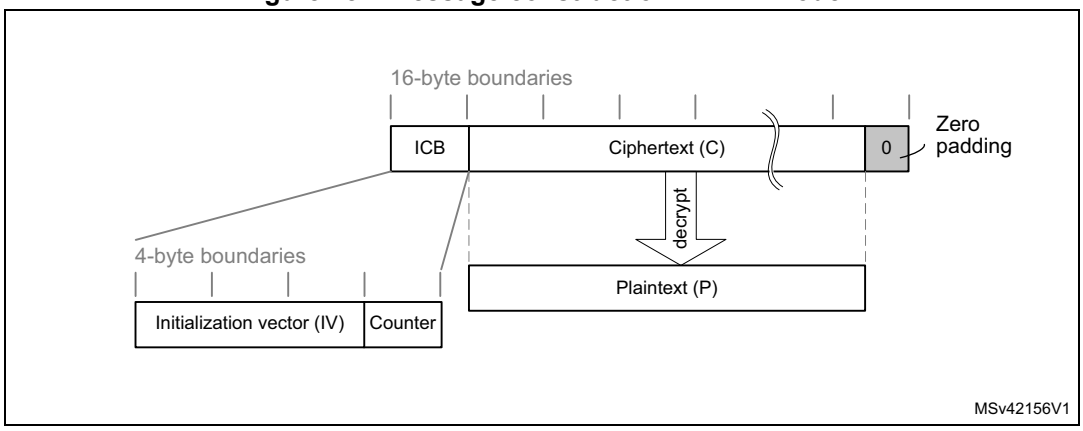
Note: It is not required to save the key registers as the application knows the original key.

27.4.10 AES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 184*.

Figure 184. Message construction in CTR mode



The structure of this message is:

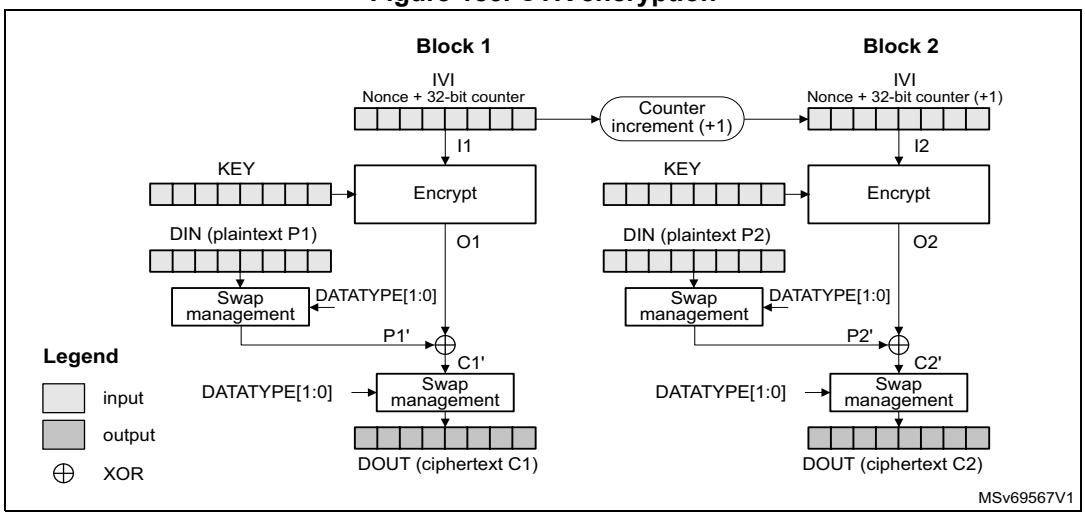
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 185 describes the counter (CTR) chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 185. CTR encryption



Initialization vectors in AES must be initialized as shown in *Table 206*.

Table 206. Counter mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 27.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers according to *Table 206*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 27.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.

7. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 27.4.5*, then read the AES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 27.4.9: AES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

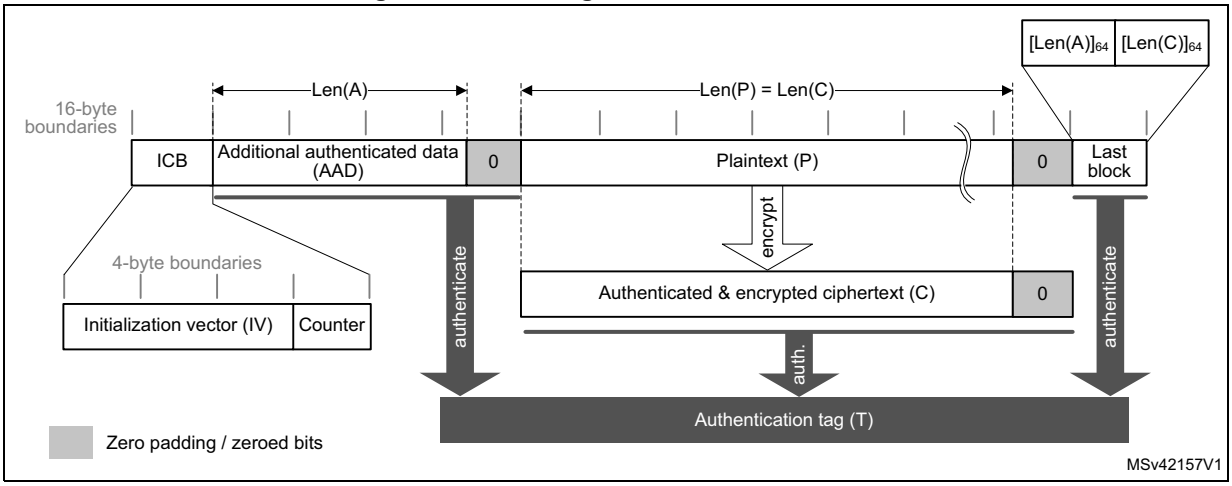
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

27.4.11 AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 186. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 208*.

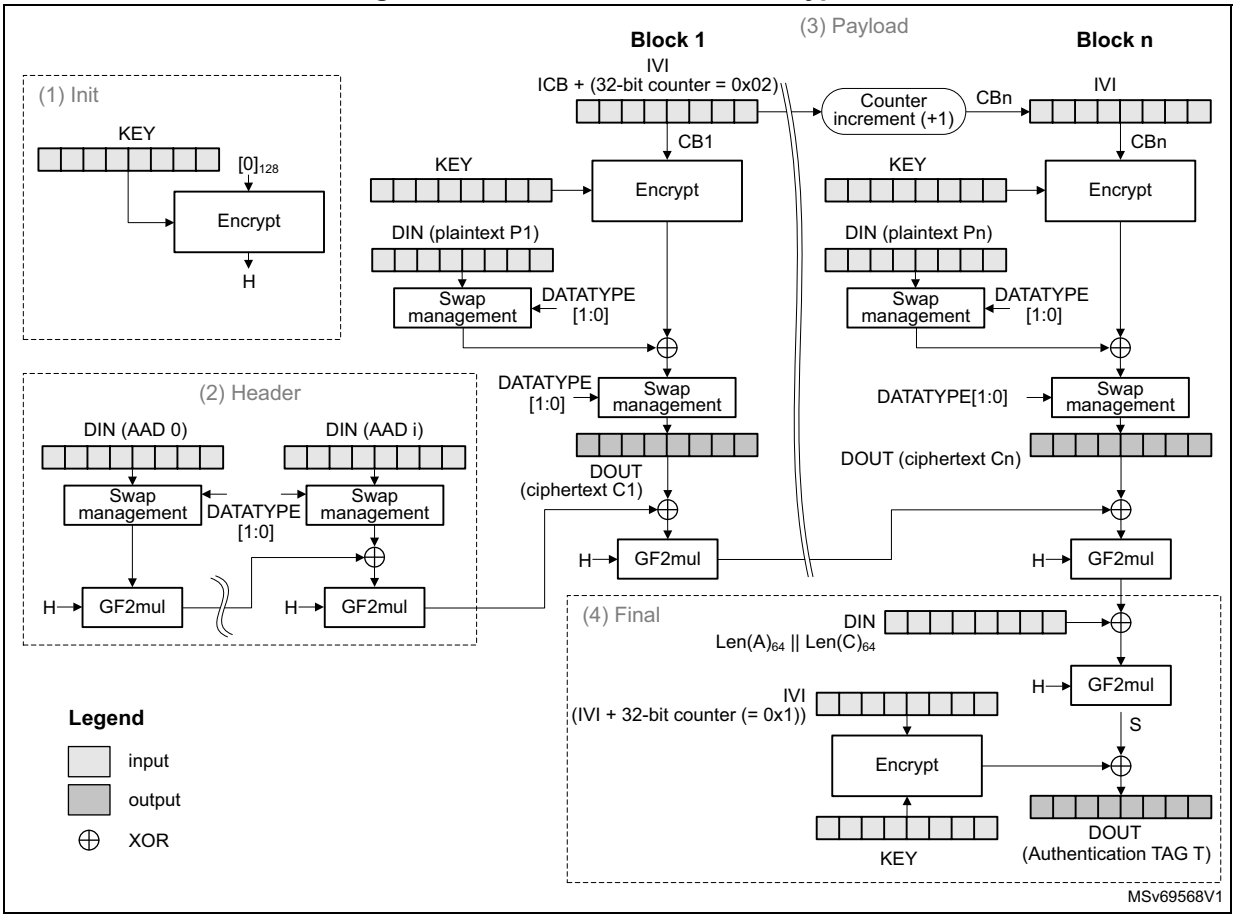
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 187 describes the GCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 187. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 207*.

Table 207. Initialization of IV registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 208*.

Table 208. GCM last block definition

Word order to AES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 27.4.6*, with the following sequence of events:

GCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the GCM initialization phase, by writing CPHASE[1:0] with 0x0.
3. Write the initialization vector in AES_IVRx registers according to *Table 207*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 27.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
7. Wait until the CCF flag is set in the AES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
8. Clear the CCF flag by setting the CCF bit of the AES_ICR register.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to CPHASE[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 27.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

11. Initialize payload phase:
    a) Select the GCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the AES peripheral by setting EN.

12. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 27.4.5*, then read the AES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

13. Select the GCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
14. Write the final GCM block into the AES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 208*.
15. Wait until the CCF flag in the AES_ISR register is set.
16. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
18. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag of the AES_ISR register, by setting the CCF bit of the AES_ICR register. When GCM encryption payload phase is selected, verify that BUSY is cleared in AES_SR, to ensure that GF2mul hash function is completed.
4. Save the AES_SUSPRx registers in the memory.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

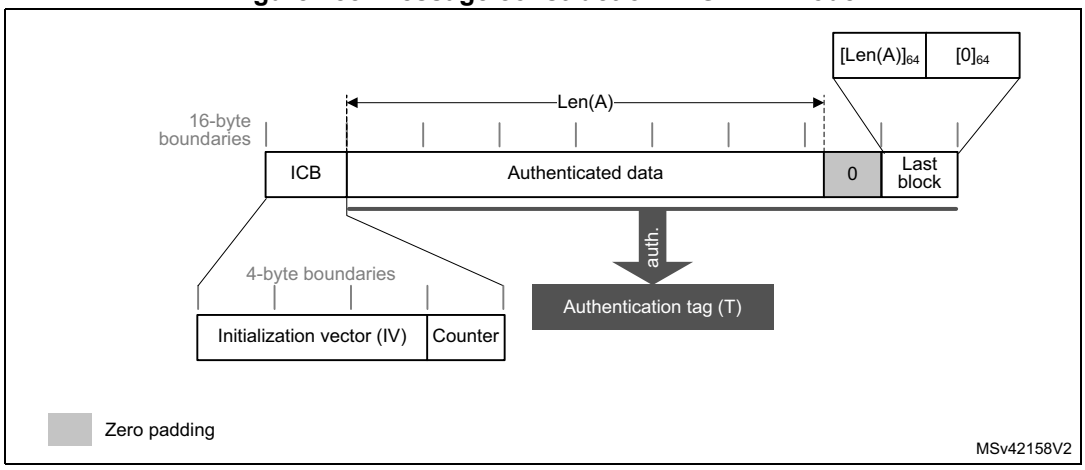
1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 27.4.14*).
5. Enable the AES peripheral, by setting EN.

27.4.12 AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

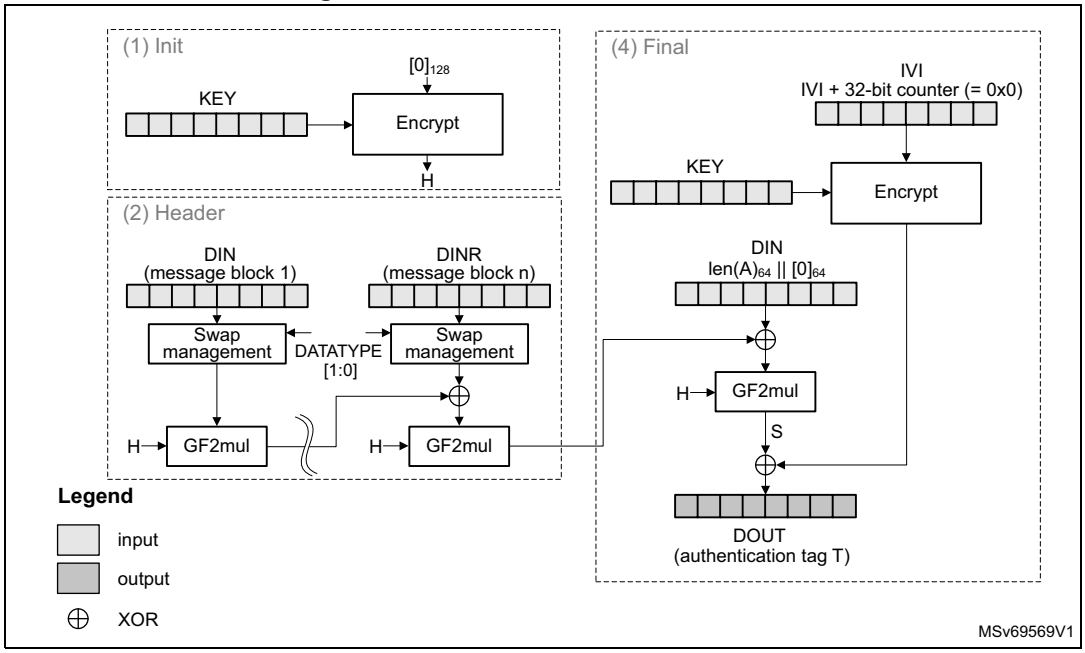
Figure 188. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 189 describes the GMAC chaining implementation in the AES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 189. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

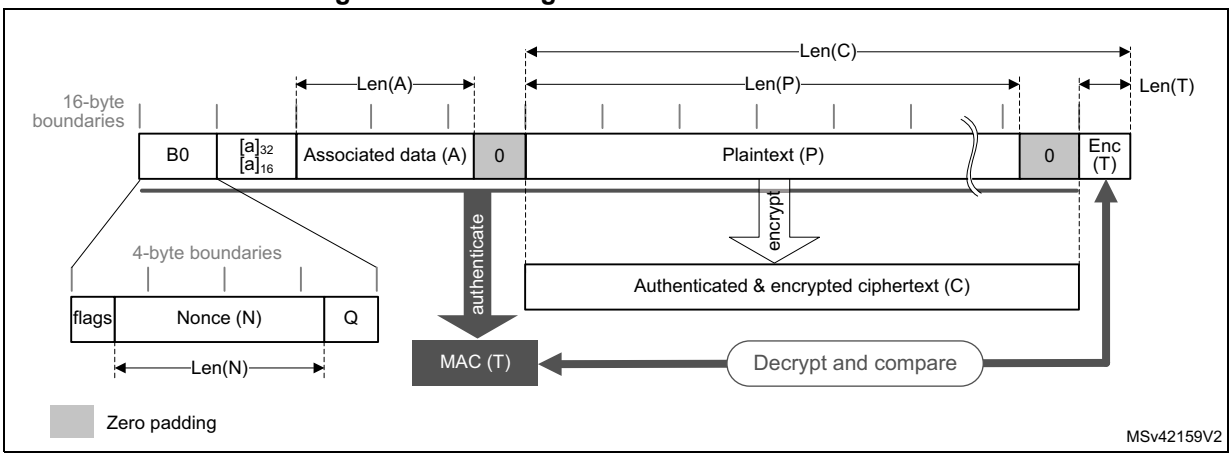
27.4.13 AES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 190. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 190*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 190*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

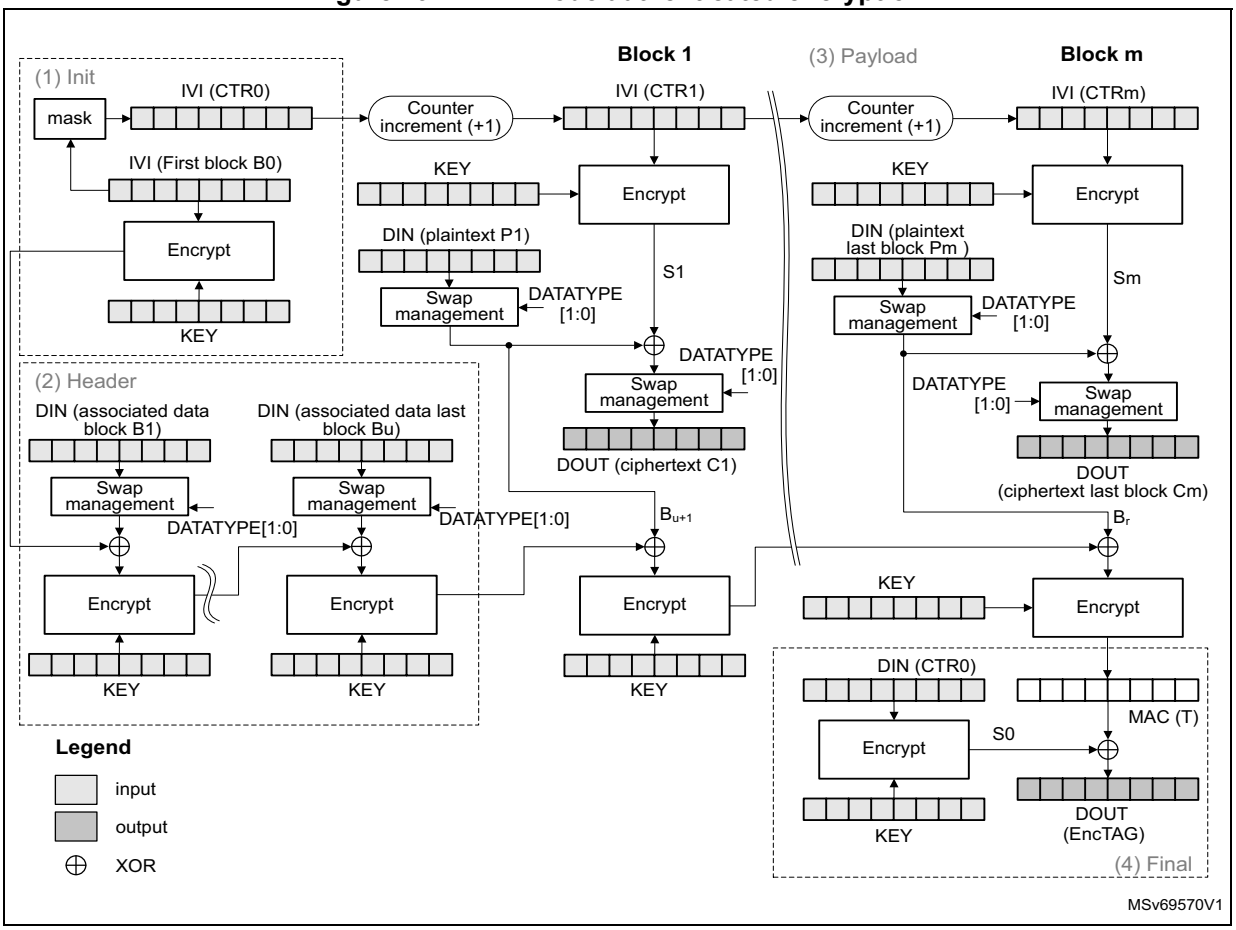
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 191 describes the CCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 191. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 209*.

Table 209. Initialization of IV registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

AES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 27.4.6*, with the following sequence of events:

CCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the CCM initialization phase, by writing CPHASE[1:0] with 0x0.
3. Write the B0 data in AES_IVRx registers according to *Table 209*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 27.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
7. Wait until the CCF flag in the AES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing CPHASE[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 27.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

11. Initialize payload phase:
    a) Select the CCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the AES peripheral, by setting EN.
12. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 27.4.5*, then read the AES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

13. Select the CCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
14. Wait until CCF flag in the AES_ISR register is set.
15. Get the CCM authentication tag, by reading the AES_DOUTR register four times.
16. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
17. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the AES_ISR register, by setting the CCF bit of the AES_ICR register.
4. Save the AES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the AES peripheral, by clearing EN.

7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. Restore AES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 27.4.14*).
5. Enable the AES peripheral, by setting EN.

27.4.14 AES key sharing with secure AES co-processor

The AES peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral and when it is ready, the AES application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in the section *SAES operations with shared keys* in the *SAES* section of this document.

1. In SAES peripheral, the application encrypts (wraps) the key to share in Shared-key mode (KMOD[1:0] at 0x2).
2. Each time the shared key is required in AES peripheral, the application decrypts it in the SAES peripheral in Shared-key mode (KMOD[1:0] at 0x2).
3. Once the shared key is decrypted (unwrapped) and loaded in SAES_KEYRx registers it can be shared with AES. To load the shared key in AES, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transfered by hardware into the AES_KEYRx registers and the BUSY flag in the AES_SR register set.
4. Once the key transfer is completed, the BUSY flag is cleared and the KEYVALID flag set in the AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KEIF flag is set in the AES_ISR register, either the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is completed, the AES is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in AES peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

27.4.15 AES data registers and data swapping

Data input and output

A 16-byte data block enters the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the AES peripheral with four successive 32-bit word reads of the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 210* “no swapping” option for details.

Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both AES input and output.

Note: The data in AES key registers (AES_KEYRx) and initialization vector registers (AES_IVRx) are not sensitive to the swap mode selection.

The AES data swapping feature is summarized in *Table 210* and *Figure 192*.

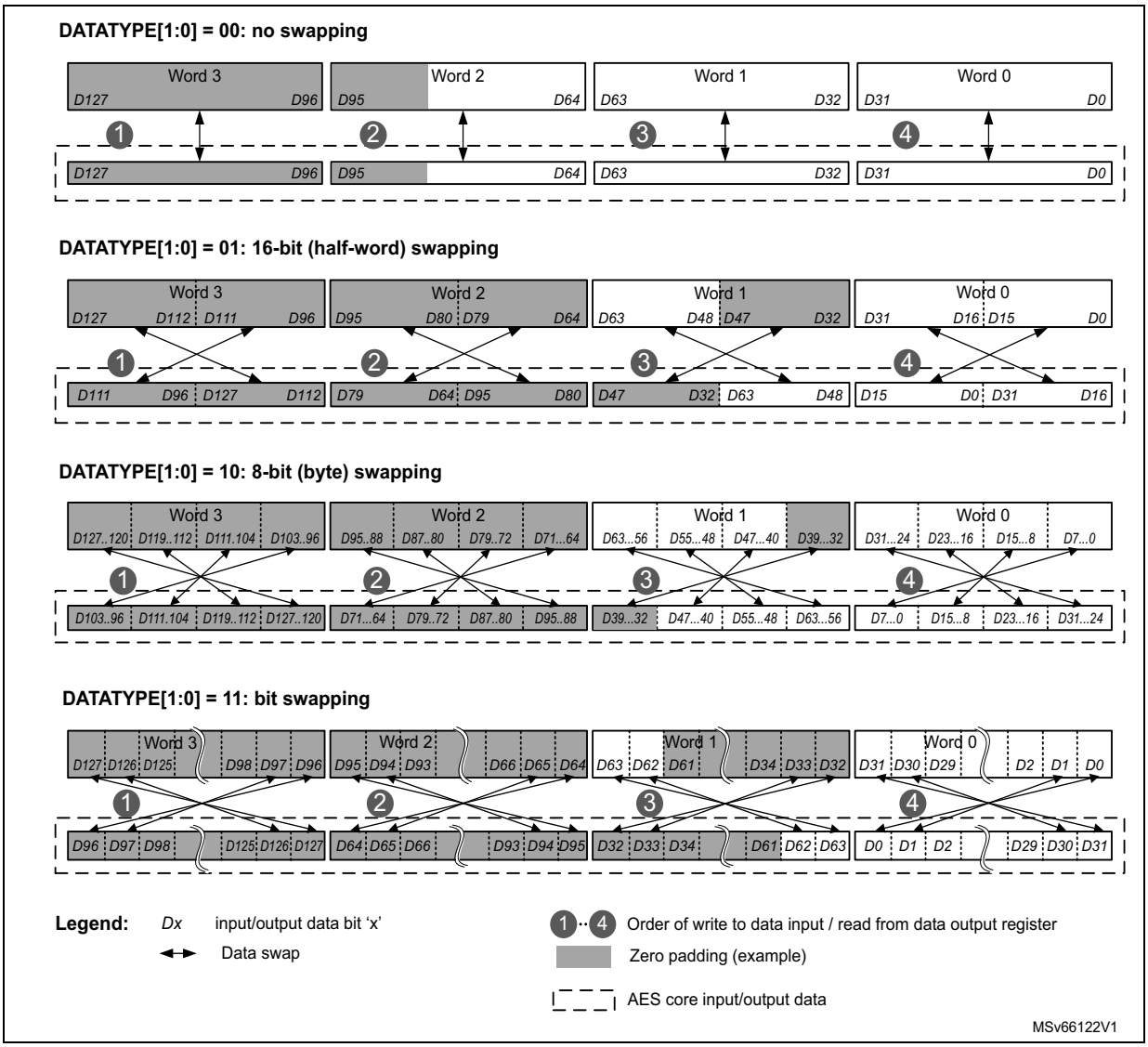
Table 210. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**

Table 210. AES data swapping example (continued)

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 192. 128-bit block construction according to the data type



Data padding

Figure 192 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

27.4.16 AES key registers

The eight AES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 211*. Reads are not allowed for security reason.

Note: In memory and in AES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 211. Key endianness in AES_KEYRx registers (128/256-bit keys)

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the AES_KEYRx registers are ignored when AES peripheral is enabled (EN bit set). The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes:

- AES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- AES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 27.4.18*).

27.4.17 AES initialization vector registers

The four AES_IVRx registers store the initialization vector (IV) information, as shown in *Table 212*. They can only be written if the AES peripheral is disabled (EN cleared).

Note: In memory and in AES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 212. IVI bitfield spread over AES_IVRx registers

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, AES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

27.4.18 AES error management

The AES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the AES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: AES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 27.5: AES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the AES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: AES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 27.5: AES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the AES_ISR register and clear the KEYVALID bit of the AES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 27.4.16: AES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.

- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 27.4.14: AES key sharing with secure AES co-processor* for details.

The KEIF flag is cleared with corresponding bit of the AES_ICR register. An interrupt is generated if the KEIE bit of the AES_IER register is set. For more details, refer to *Section 27.5: AES interrupts*.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring AES.

27.5 AES interrupts

There are multiple individual maskable interrupt sources generated by the AES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)

See *Section 27.4.18: AES error management* for details on AES errors.

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable AES interrupt sources individually by setting/clearing the corresponding enable bit of the AES_IER register.

The status of the individual maskable interrupt sources can be read from the AES_ISR register. They are cleared by setting the corresponding bit of the AES_ICR register.

Table 213 gives a summary of the available features.

Table 213. AES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)

1. Bit of the AES_ICR register.
2. Flag of the AES_SR register, mirrored by the flag RWEIF of the AES_ISR register.

27.6 AES DMA requests

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the AES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

AES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with AES can be found in *Appending data using DMA* subsection of *Section 27.4.5: AES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into AES. AES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the AES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 27.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from AES. AES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the AES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, AES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when AES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

27.7 AES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 214. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
128-bit	Encryption or decryption(1)	ECB, CBC, CTR	51
	Key preparation	-	59
256-bit	Encryption or decryption(1)	ECB, CBC, CTR	75
	Key preparation	-	82

1. Excluding key preparation time (ECB and CBC only).

Table 215. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Encryption/ Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Encryption/ Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

27.8 AES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

27.8.1 AES control register (AES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
rw						rw	rw	rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: AES peripheral software reset

Setting the bit resets the AES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the AES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the AES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable.

0x2: Shared key mode. If shared key mode is properly initialized in SAES peripheral, the AES peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in AES key registers when BUSY bit is cleared and KEYVALID is set in the AES_SR register. Key error flag KEIF is set otherwise in the AES_ISR register.

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection
This bitfield defines the key length in bits of the key used by AES.
0: 128-bit
1: 256-bit
Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **CPHASE[1:0]:** Chaining phase selection
This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable
This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable
This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode
This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode
This bitfield selects the AES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 27.4.15: AES data registers and data swapping*.

0x0: No swapping (32-bit data).

0x1: Half-word swapping (16-bit data)

0x2: Byte swapping (8-bit data)

0x3: Bit-level swapping

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the AES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the AES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

– The bit cannot be set as long as KEYVALID is cleared

27.8.2 AES status register (AES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in AES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KMOD[1:0] is different from zero, the BUSY flag is automatically set by AES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is usually set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 27.4.16: AES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether AES is idle or busy.
0: Idle
1: Busy
AES is flagged as idle when disabled (when EN is low) or when the last processing is completed.
AES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), or transferring a shared key from the SAES peripheral.
When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the AES_DINR register occurred. When set WRERRF bit has no impact on the AES operations.
0: No error
1: Unexpected write to AES_DINR register occurred during computation or data output phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the AES_DOUTR register occurred. When set RDERRF bit has no impact on the AES operations.
0: No error
1: Unexpected read to AES_DOUTR register occurred during computation or data input phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 0 Reserved, must be kept at reset value.

27.8.3 AES data input register (AES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.
Reads return zero.

27.8.4 AES data output register (AES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

27.8.5 AES key register 0 (AES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the AES_CR register.

Writes to AES_KEYRx registers are ignored when AES is enabled (EN bit set). When the key comes from the SAES peripheral (KMOD[1:0] at 0x2), writes to key registers are also ignored and they result in setting the KEIF bit of the AES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

27.8.6 AES key register 1 (AES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

27.8.7 AES key register 2 (AES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

27.8.8 AES key register 3 (AES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

27.8.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

AES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in AES_CR register

27.8.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

27.8.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

27.8.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

27.8.13 AES key register 4 (AES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

27.8.14 AES key register 5 (AES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

27.8.15 AES key register 6 (AES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

27.8.16 AES key register 7 (AES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

27.8.17 AES suspend registers (AES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: Suspend data

AES_SUSPRx registers contain the complete internal register states of the AES when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 27.4.8: AES suspend and resume operations* for more details.

Clearing EN bit of the AES_CR register clears this register to zero.

AES_SUSPRx registers are not used in other chaining modes than GCM, GMAC or CCM.

27.8.18 AES interrupt enable register (AES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

27.8.19 AES interrupt status register (AES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

The flag setting generates an interrupt if the KEIE bit of the AES_IER register is set. It also clears the key registers and the KEYVALID flag in the AES_SR register.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

KEIF is raised upon any of the following events:

–AES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE set, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 then AES_KEYR4 then AES_KEYR5 then AES_KEYR6 then AES_KEYR7, or reverse).

–AES fails to load the key shared by the SAES peripheral (KMOD[1:0] = 0x2).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the AES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the AES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

27.8.20 AES interrupt clear register (AES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear
Setting this bit clears the KEIF status bit of the AES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear
Setting this bit clears the RWEIF status bit of the AES_ISR register, and clears both RDERRF and WRERRF flags in the AES_SR register.

Bit 0 **CCF:** Computation complete flag clear
Setting this bit clears the CCF status bit of the AES_ISR register.

27.8.21 AES register map

Table 216. AES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	AES_CR	IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1]	KMOD[0]	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0						0	0	0	0	0	0		0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 216. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	AES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Table 216. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	AES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
	Reset value																														0	0	0
0x304	AES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x308	AES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x309-0x3FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

28 Secure AES coprocessor (SAES)

This section only applies to STM32C59x/5A3 products.

28.1 SAES introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/ decryption (wrapped-key mode) and key sharing with faster AES peripheral (shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the AES peripheral.

28.2 SAES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster AES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together

- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if SAES needs to process another message with a higher priority, then resume the original message

28.3 SAES implementation

The devices have one SAES peripheral, implemented as per the following table. It shares the key with the AES peripheral. For comparison, the AES peripheral is also included in the table.

Table 217. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

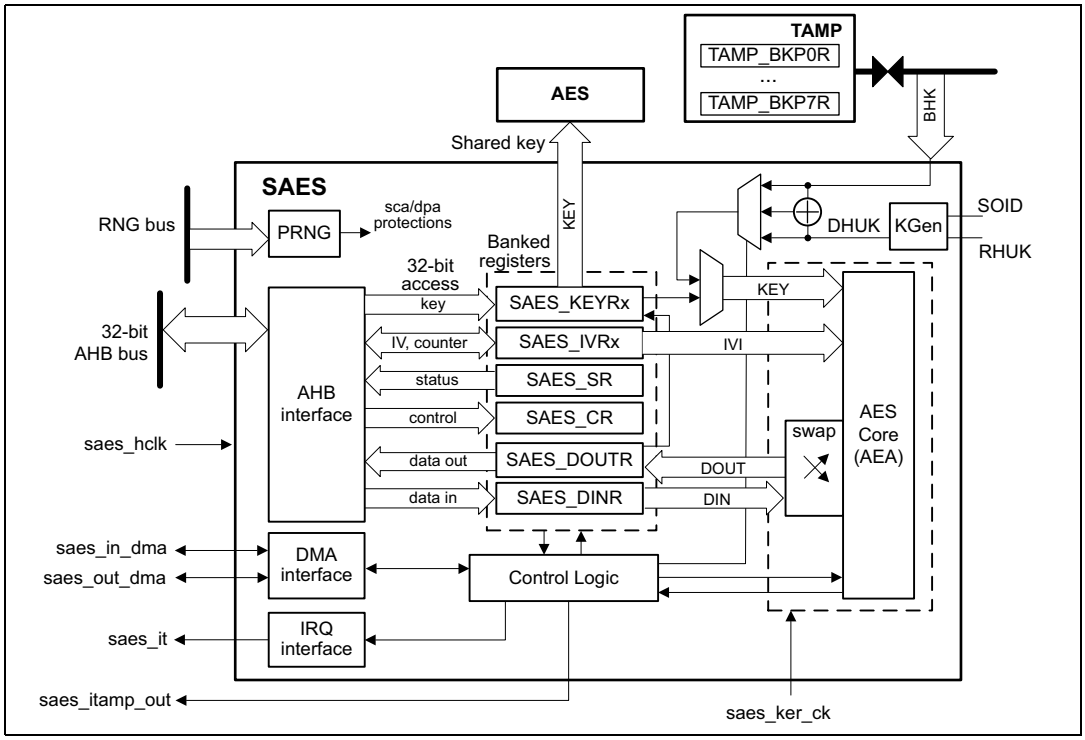
1. X = supported.

28.4 SAES functional description

28.4.1 SAES block diagram

Figure 193 shows the block diagram of SAES.

Figure 193. SAES block diagram



Note: AES represents the AES peripheral.

28.4.2 SAES internal signals

Table 218 describes the user relevant internal signals interfacing the SAES peripheral.

Table 218. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_ker_ck	Input	SAES kernel clock.
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	SAES incoming data DMA single request/acknowledge
saes_out_dma	Input/Output	SAES processed data DMA single request/acknowledge
saes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.
RHUK	Input	256-bit root hardware unique key (nonvolatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)

Table 218. SAES internal input/output signals (continued)

Signal name	Signal type	Description
BHK(1)	Input	256-bit boot hardware key (BHK) stored in tamper-resistant backup registers and written during boot. Once written, this key cannot be read nor written by any application until the next product reset.
SOID	Input	Static operating ID hardware input used during each DHUK computation. Its change causes the erasure of key registers and sets the KEIF flag.

1. Connected to a set of backup registers in TAMP peripheral that are written, then read/write locked, by the application software (see *Section 28.4.17* for details).

28.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit and a dedicated kernel clock, controlled through the RCC.

The SAES reset in RCC is mandatory when a hardware tamper event occurs in SAES. Refer to *Managing tamper errors* in *Section 28.4.19* for details.

After clocking then releasing the reset of the SAES peripheral in the RCC, SAES automatically draws random numbers from the RNG, setting the BUSY bit of the SAES_SR register. Refer to the *RNG error interrupt flag (RNGEIF)* in *Section 28.4.19* for details.

This background task must be completed before the device enters a low-power mode.

28.4.4 SAES symmetric cipher implementation

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 219*. Those functions can be certified NIST PUB 140-3.

Table 219. SAES approved symmetric key functions

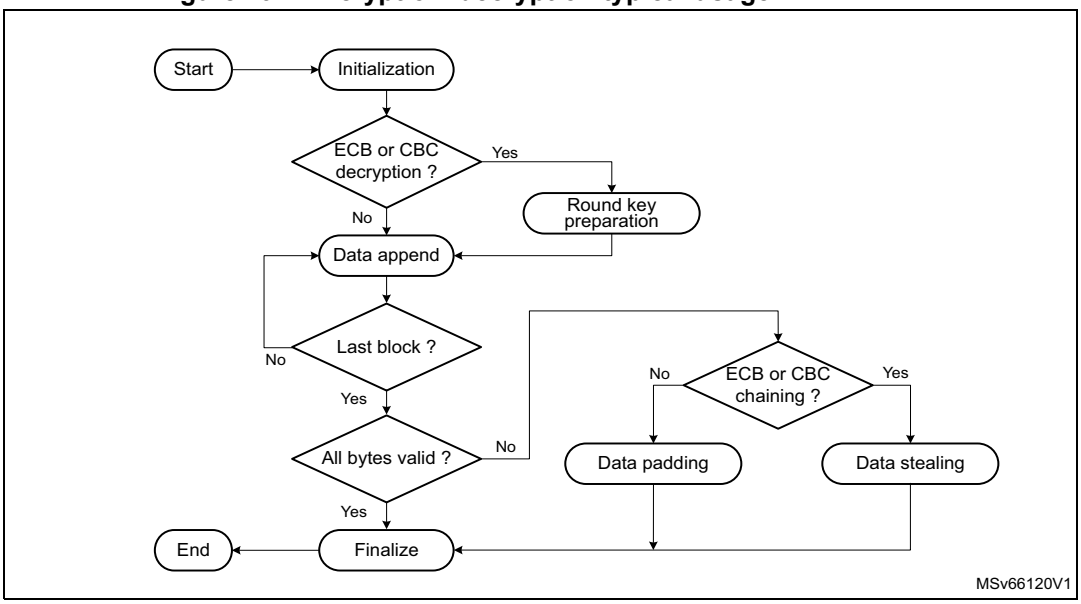
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any SAES processing, following the sequence described in *Section 28.4.8*.

28.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 194. Encryption/ decryption typical usage



Initialization

The SAES peripheral is initialized according to the chaining mode. Refer to *Section 28.4.9: SAES basic chaining modes (ECB, CBC)* and *Section 28.4.10: SAES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 28.4.7: SAES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral when KEYVALID is set, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 28.6: SAES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 28.4.8* is not supported.

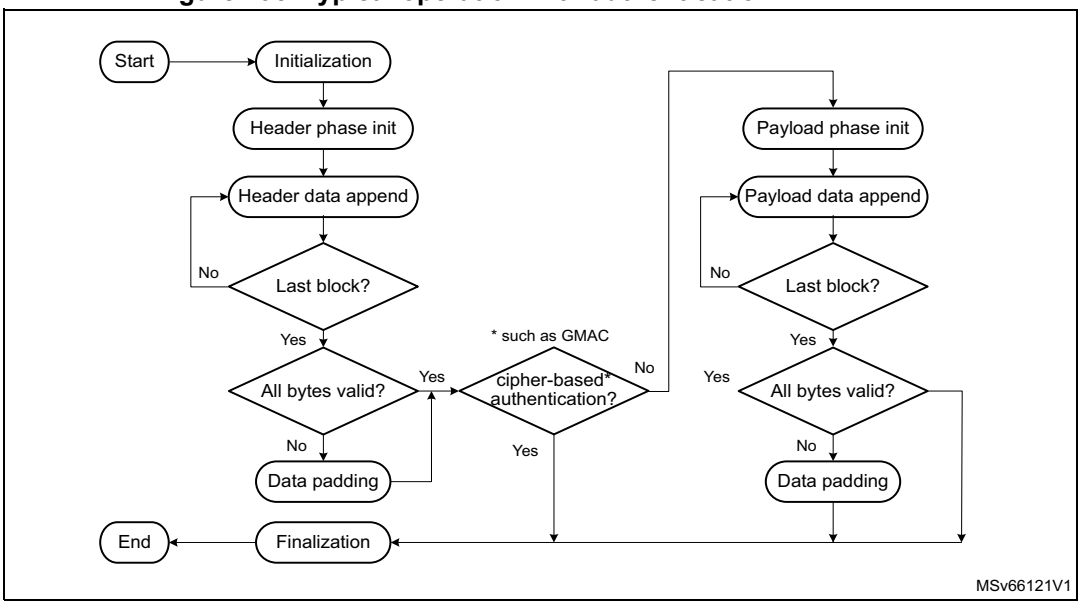
Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

28.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 195. Typical operation with authentication



Section 28.4.11: SAES Galois/counter mode (GCM) and *Section 28.4.13: SAES counter with CBC-MAC (CCM)* describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in *Section 28.4.12: SAES Galois message authentication code (GMAC)*.

28.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A*, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since SAES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since SAES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

28.4.8 SAES suspend and resume operations

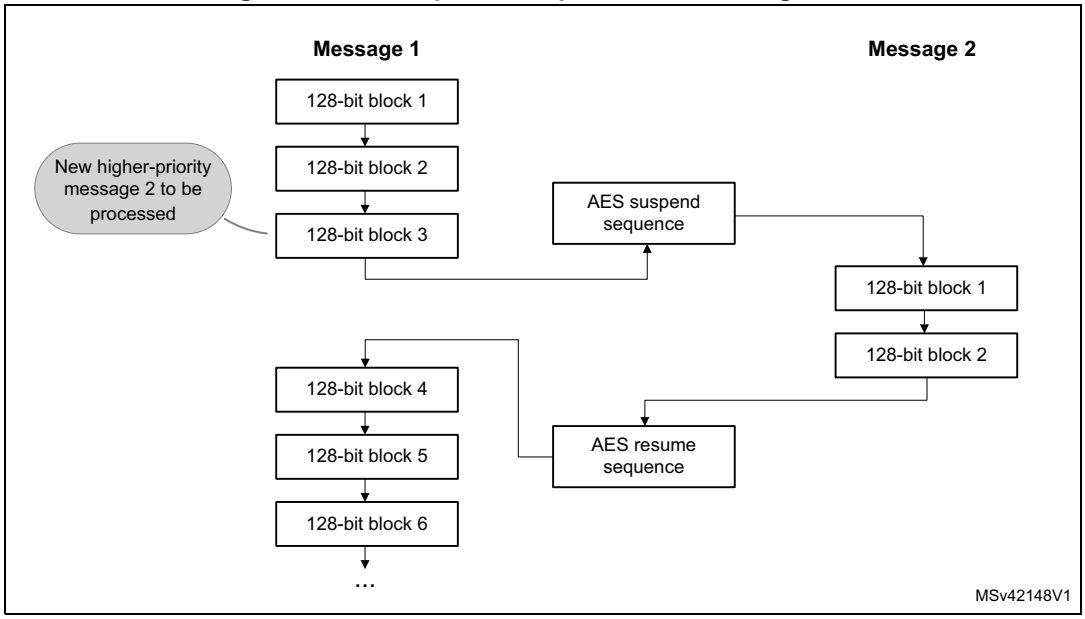
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

Figure 196 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 196. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

28.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 197 and *Figure 198* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 197. ECB encryption

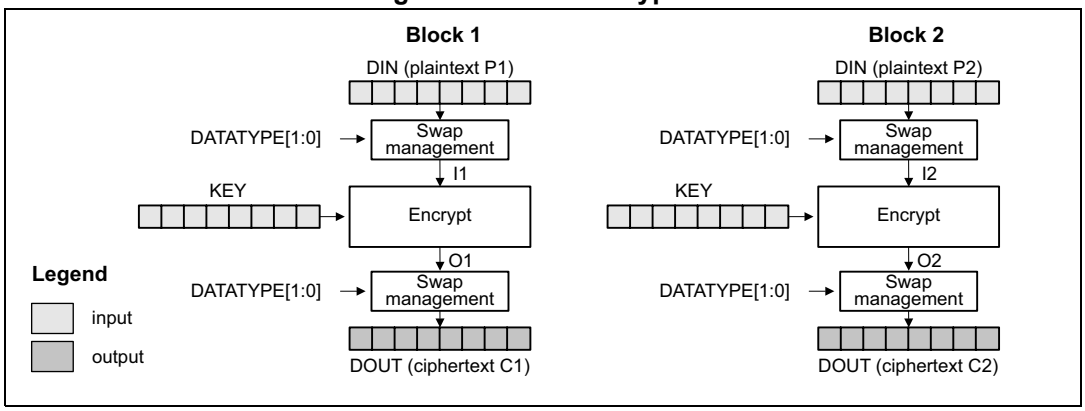
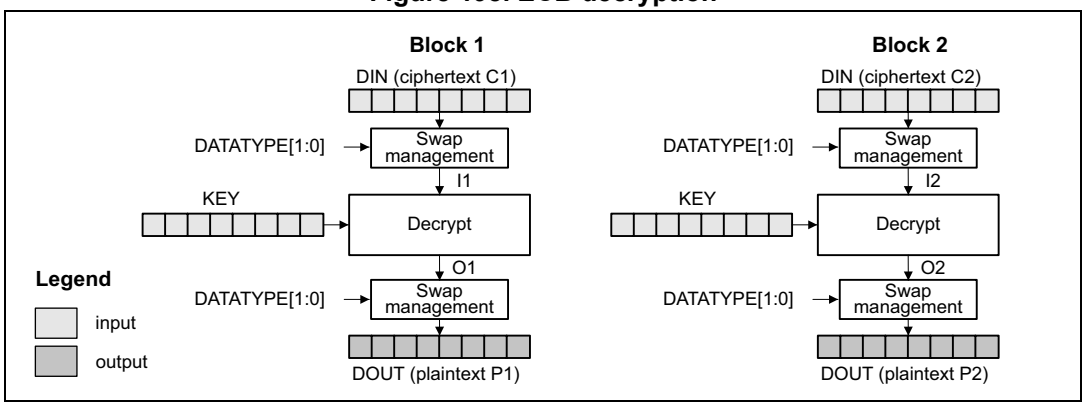


Figure 198. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 199 and *Figure 200* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 199. CBC encryption

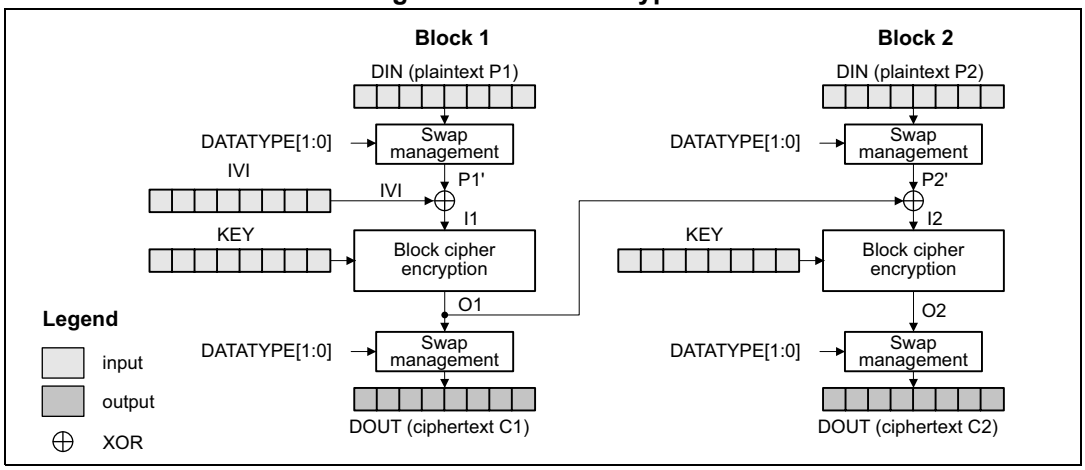
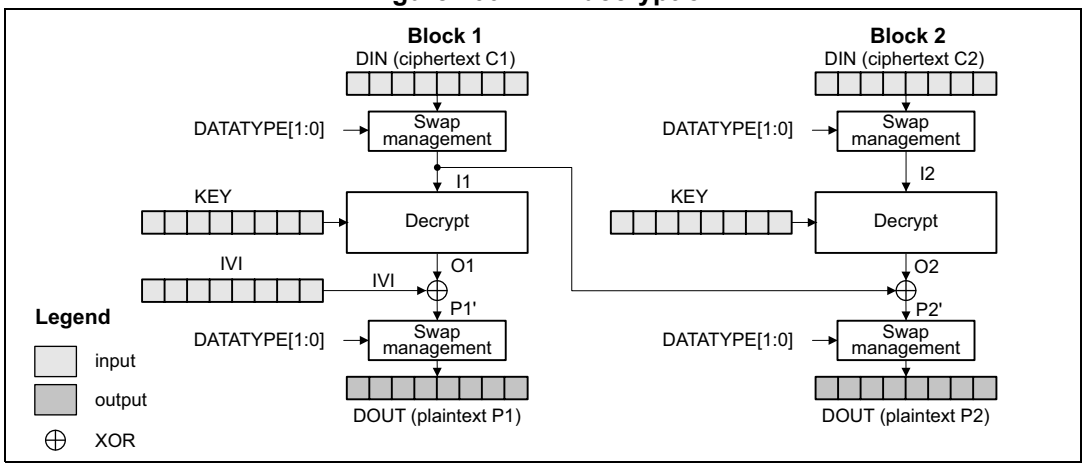


Figure 200. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 28.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 28.4.14* (wrapped keys) and *Section 28.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 28.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 28.4.7*.
   b) Append the cleartext block into SAES as described in *Section 28.4.5*, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 28.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 28.4.14* (wrapped keys) and *Section 28.4.15* (shared keys).
4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 28.4.17: SAES key registers* for details.
5. Wait until KEYVALID is set (the key loading completed).

6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag in the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 28.4.7*.
    b) Append the ciphertext block into SAES as described in *Section 28.4.5*, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the SAES_ISR register is set (computation completed).
2. Read four times the SAES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).
5. Disable the SAES peripheral, by clearing EN.
6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from key registers, refer to *Section 28.4.17: SAES key registers* for details.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the SAES peripheral, by setting EN.

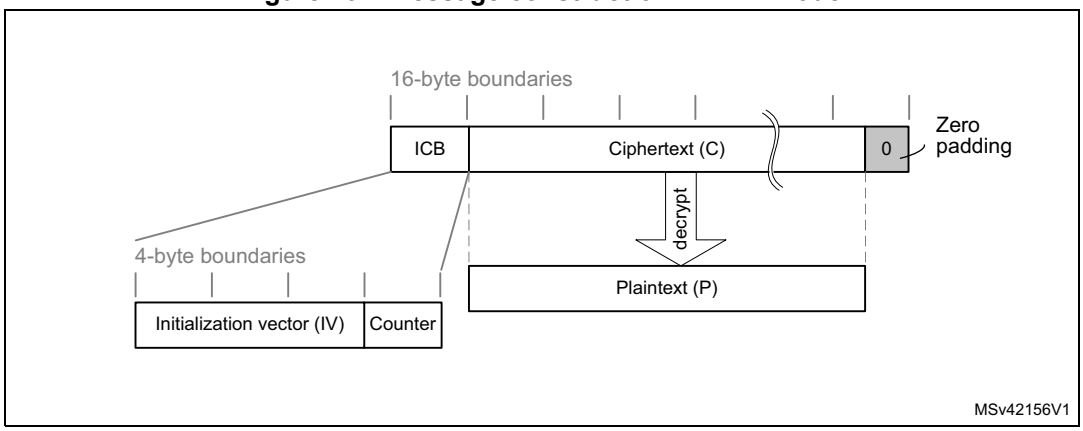
Note: It is not required to save the key registers as the application knows the original key.

28.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 201*.

Figure 201. Message construction in CTR mode



The structure of this message is:

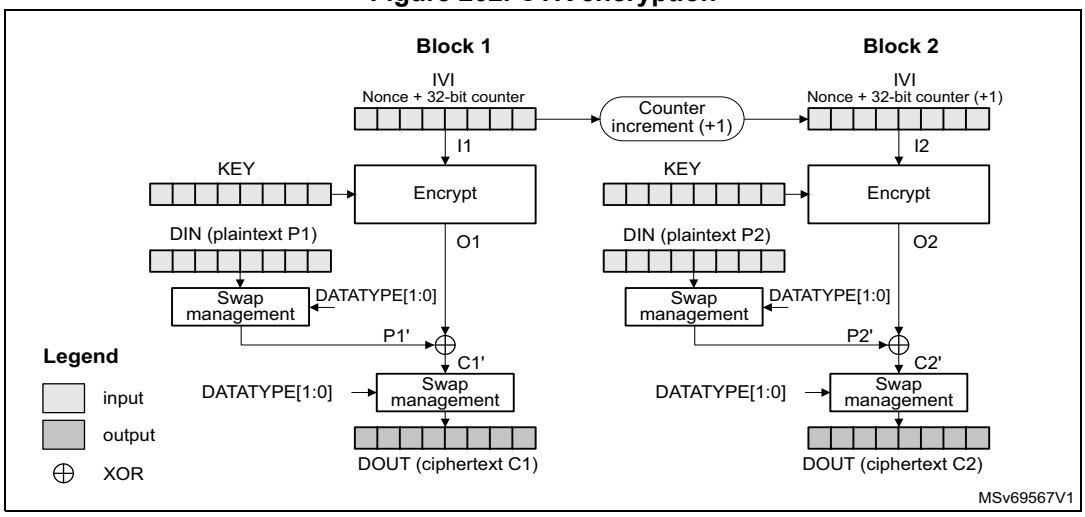
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 202 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 202. CTR encryption



Initialization vectors in SAES must be initialized as shown in *Table 220*.

Table 220. Counter mode initialization vector definition

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 28.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 28.4.14* (wrapped keys) and *Section 28.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to *Table 220*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 28.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.

8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in *Section 28.4.5*, then read the SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 28.4.9: SAES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

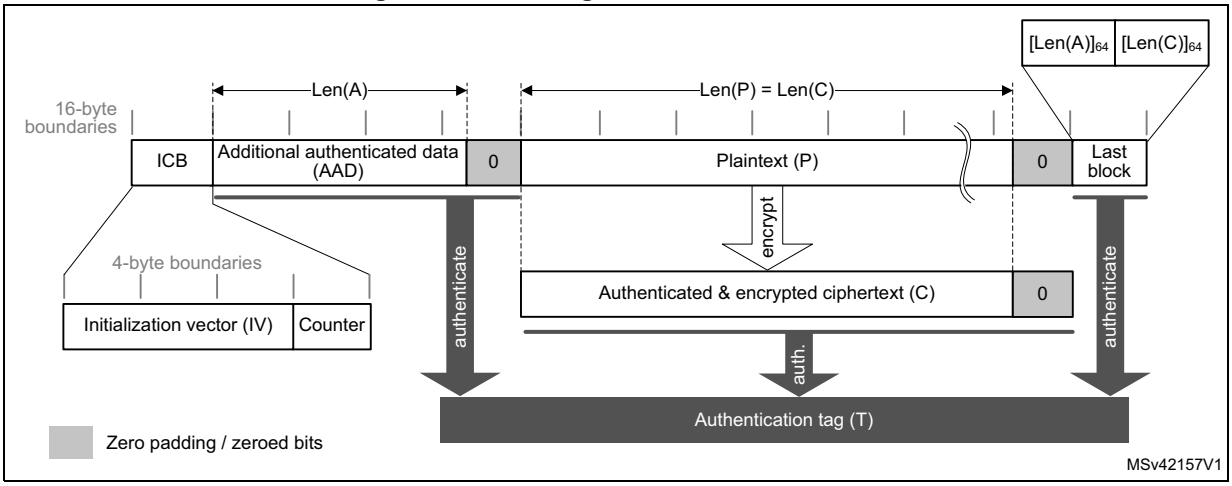
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

28.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 203. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 222*.

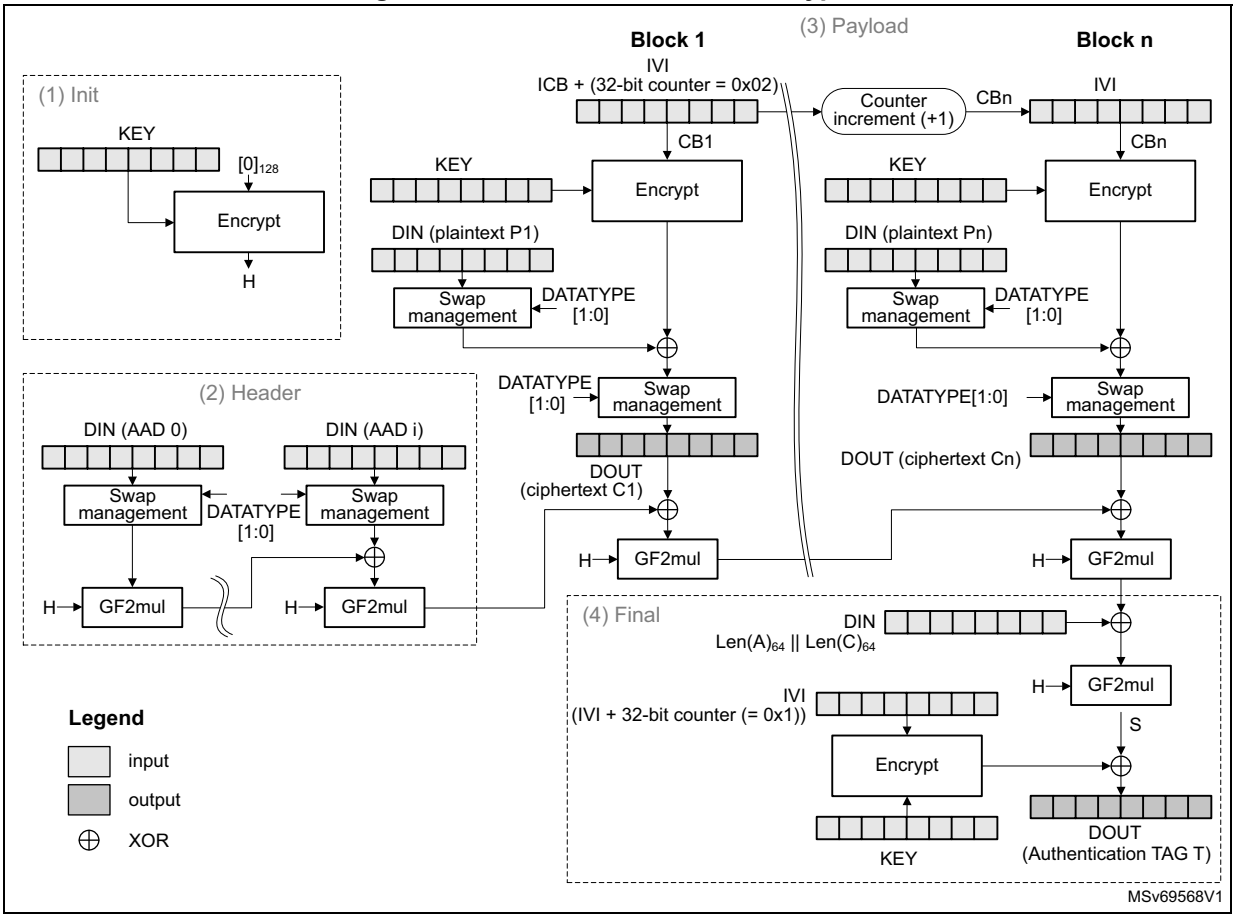
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 204 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 204. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 221*.

Table 221. Initialization of IV registers in GCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 222*.

Table 222. GCM last block definition

Word order to SAES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 28.4.6*, with the following sequence of events:

GCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 28.4.14* (wrapped keys) and *Section 28.4.15* (shared keys).
   - Select the GCM initialization phase, by writing CPHASE[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to *Table 221*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 28.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

GCM header phase

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to CPHASE[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 28.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.

13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 28.4.5*, then read the SAES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.
15. Select the GCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 222*.
17. Wait until the CCF flag in the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

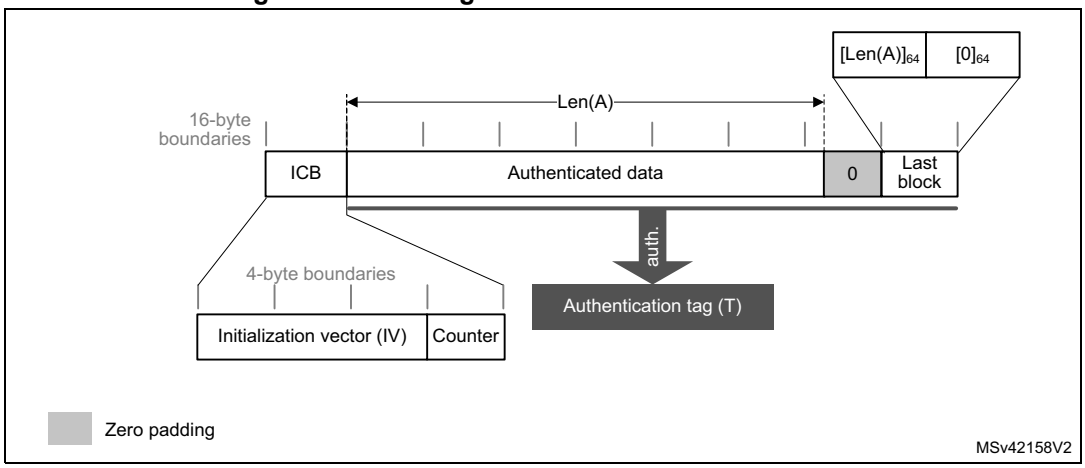
Suspend/resume operations are not supported in GCM mode.

28.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

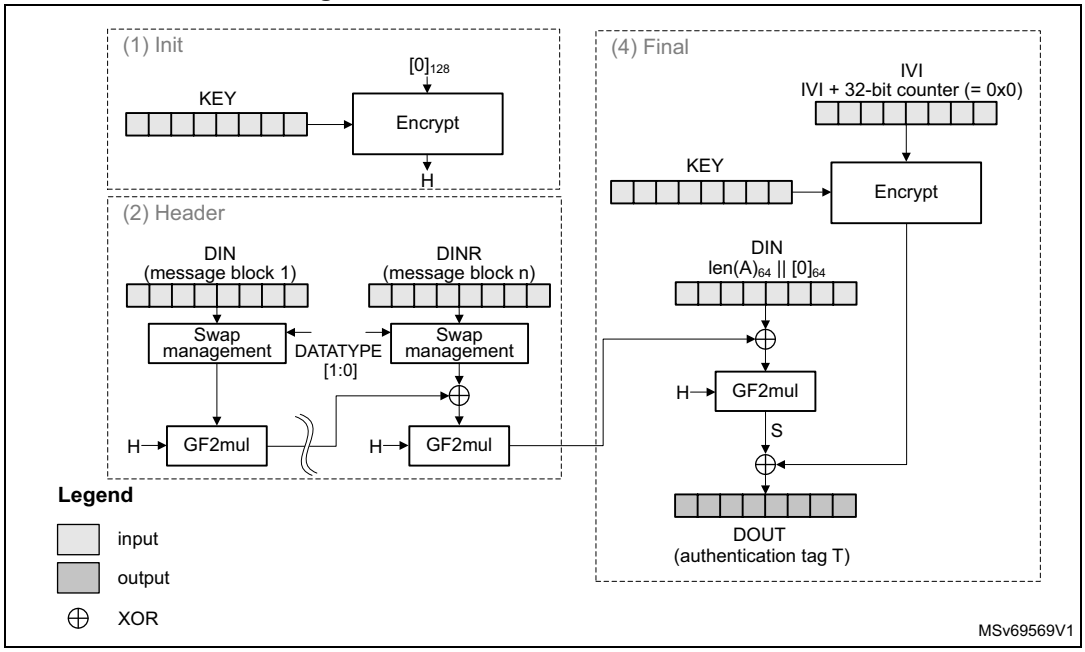
Figure 205. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 206 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 206. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

Suspend/resume operations are not supported in GMAC mode.

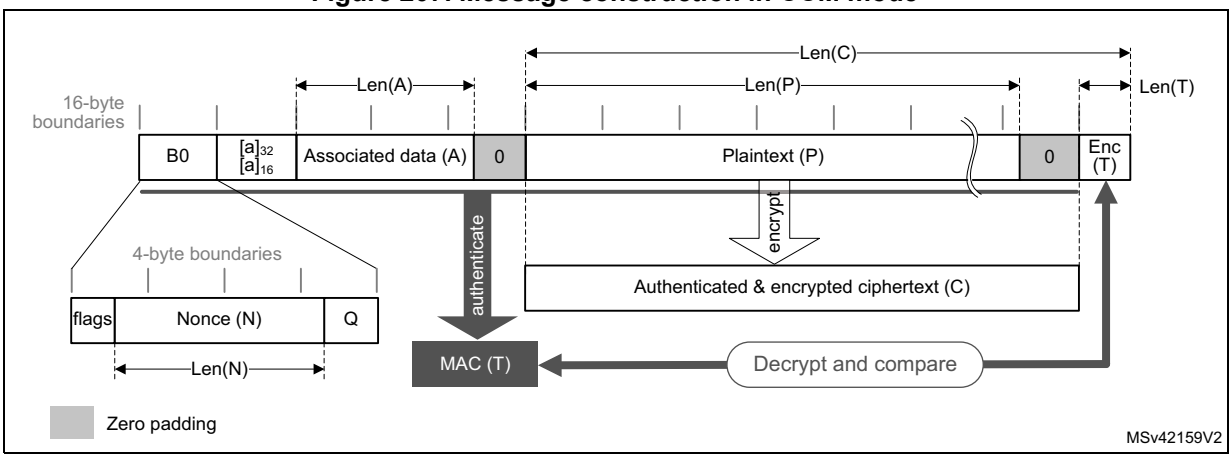
28.4.13 SAES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 207. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 207*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 207*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

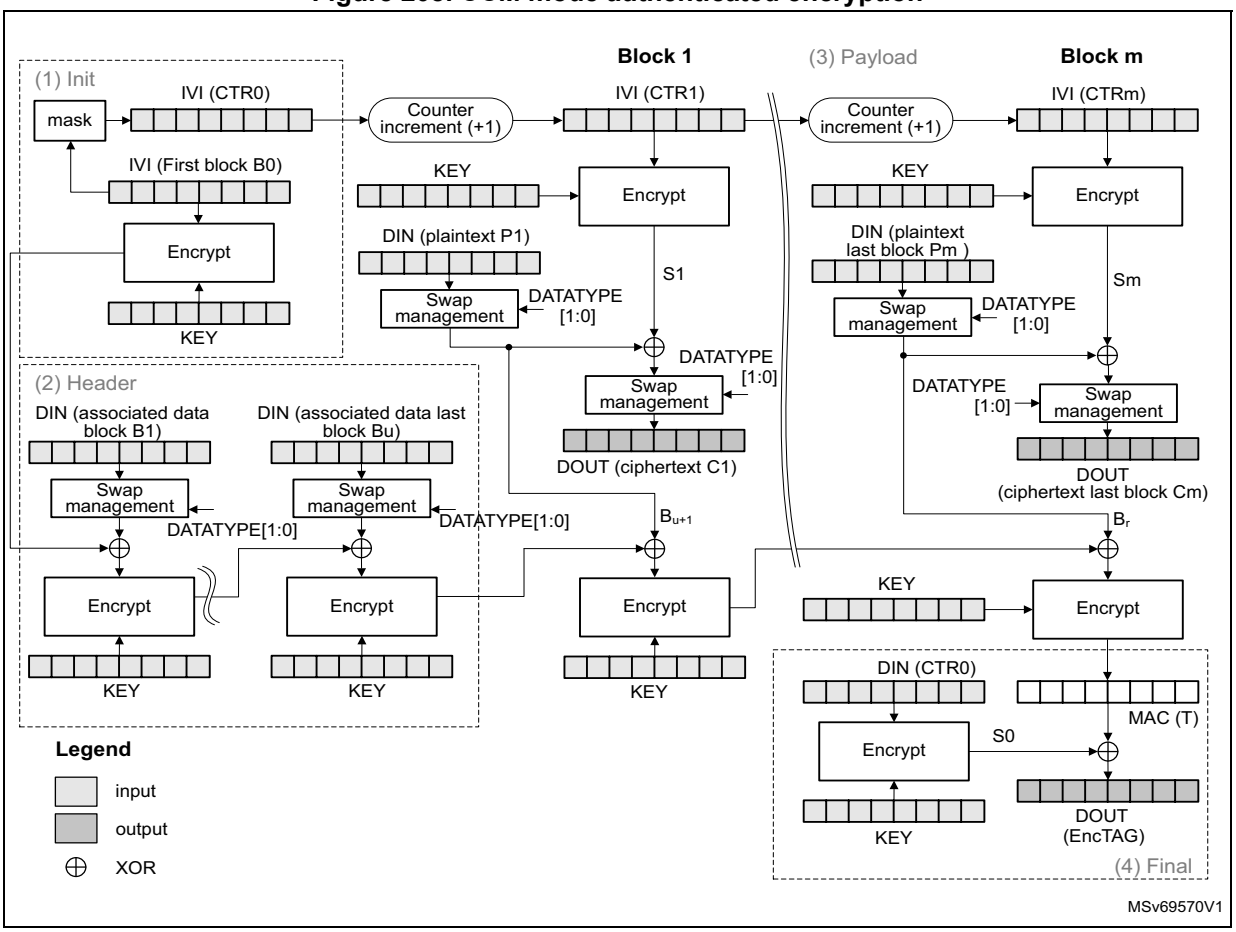
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C*, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 208 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 208. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 223*.

Table 223. Initialization of IV registers in CCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 28.4.6*, with the following sequence of events:

CCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 28.4.14* (wrapped keys) and *Section 28.4.15* (shared keys).
   - Select the CCM initialization phase, by writing CPHASE[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to *Table 223*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 28.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

CCM header phase

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing CPHASE[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 28.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Select the CCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral, by setting EN.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 28.4.5*, then read the SAES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
15. Wait until CCF flag in the SAES_ISR register is set.
16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation completed).
2. In the payload phase, read four times the SAES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.
4. Save the SAES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
3. Restore SAES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from the key registers, refer to *Section 28.4.17: SAES key registers* for details.
5. Enable the SAES peripheral, by setting EN.

28.4.14 SAES operation with wrapped keys

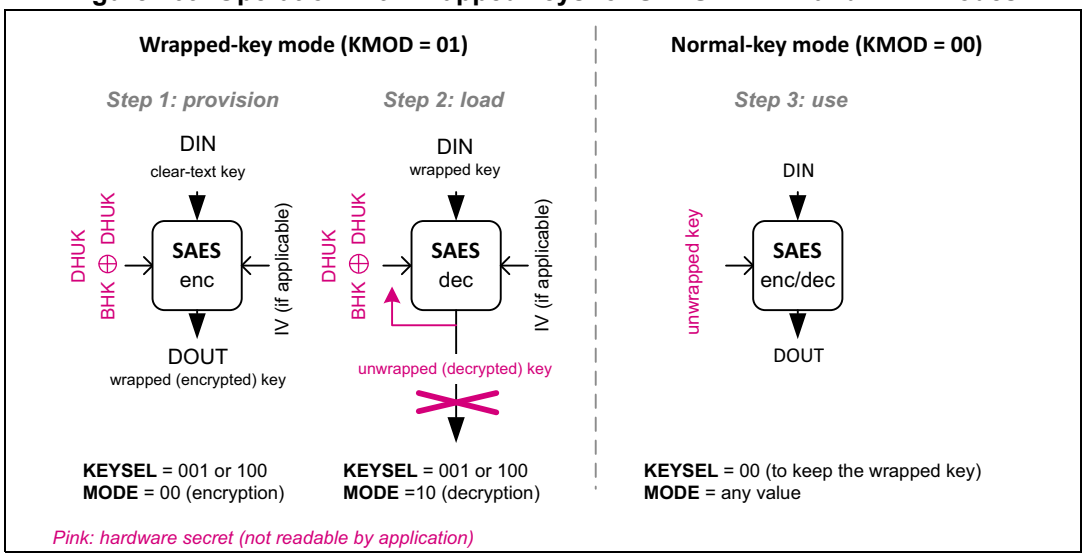
SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

Figure 209 summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 209. Operation with wrapped keys for SAES in ECB and CBC modes



Note: DHUK value depends on privilege, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

Key wrapping for SAES

The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 28.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see *Table 225*).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register.
11. Repeat steps *8.* to *10.* if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.

Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in *key derivation* mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. With KEYSEL[2:0], select the same key source as when the key was wrapped/encrypted.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the *decryption* mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see *Table 225*).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
14. Repeat steps *12.* and *13.* if KEYSIZE is set.
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

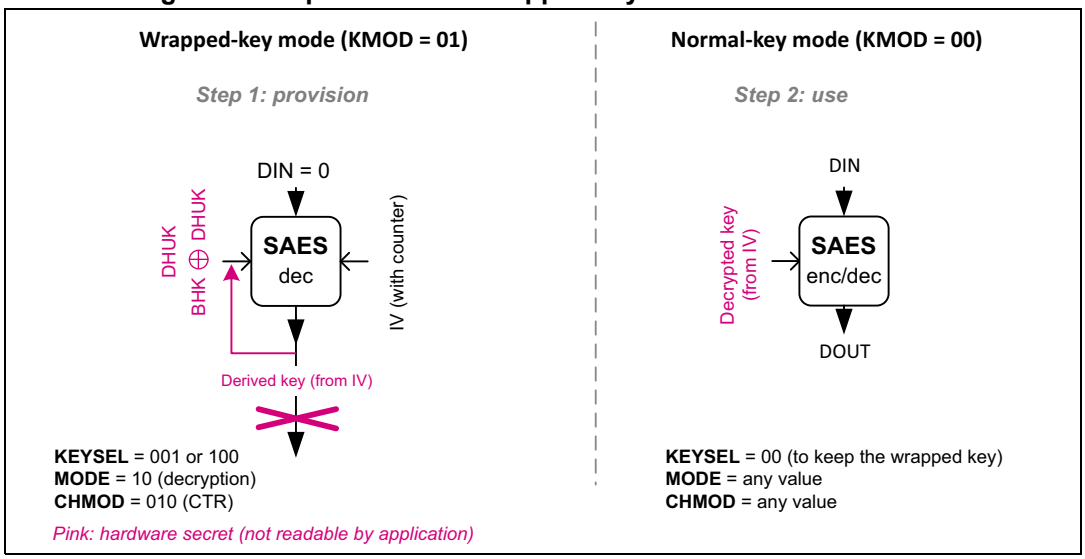
Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERRF).

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

Operation with wrapped keys for SAES in CTR mode

Figure 210 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 210. Operation with wrapped keys for SAES in CTR mode



Note: DHUK value depends on privilege, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

The recommended sequence for SAES wrapped key mode using CTR is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 28.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps *8.* and *9.* if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Note: The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_*CR register.*

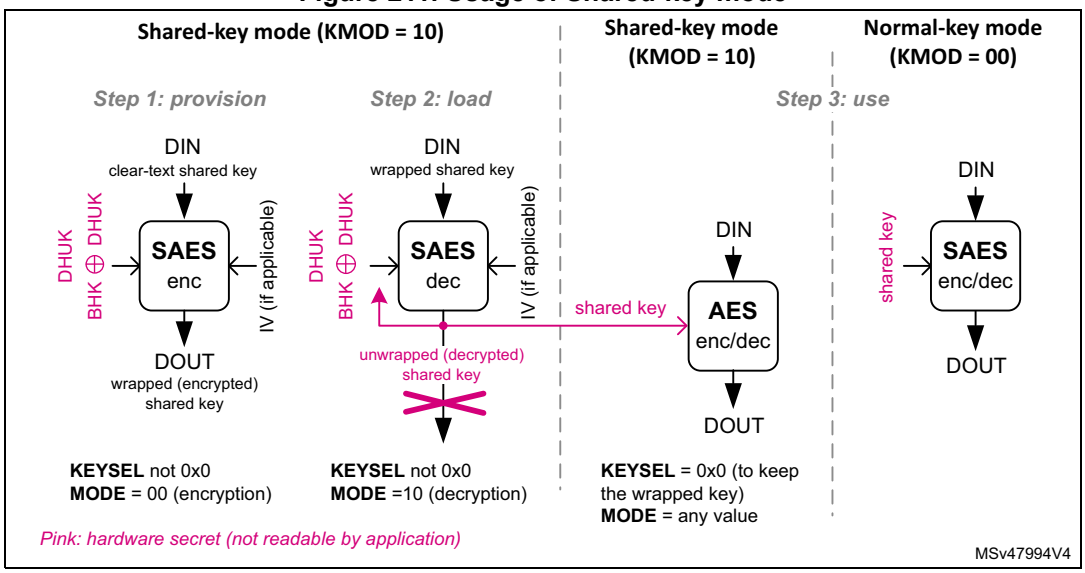
28.4.15 SAES operation with shared keys

SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, the application software can make the AES keys available to the AES peripheral, without exposing them in clear-text (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

Figure 211 summarizes how to wrap or unwrap keys to share with *AES* peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 211. Usage of Shared-key mode



Note: DHUK value depends on privilege, KMOD[1:0], KSHAREID, KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

In the step 3, AES represents the AES peripheral.

Key wrapping for AES peripheral

Before SAES can share a key with the AES peripheral, the key must be encrypted (wrapped) once. The encryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 211*. See *Key wrapping for SAES* for details.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected.

Key unwrapping for AES peripheral (shared key)

Each time SAES needs to share a key with the AES peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by AES. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 211*. See *Key unwrapping for SAES* for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERRF).

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be written with 0x0 (normal key mode).

Sequence in the AES peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the AES peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] at 0x2 and KEYVALID set. The sequence in the AES key share target peripheral is described in *AES key sharing with secure AES co-processor* of the corresponding section in this document. It can be run multiple times (for example, to manage a suspend/resume situation) as long as SAES is unused and duly remains in key sharing state.

Note: When KMOD[1:0] is at 0x2 and BUSY set in the AES peripheral, and KEYSIZE value of AES and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

28.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 224* “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output.

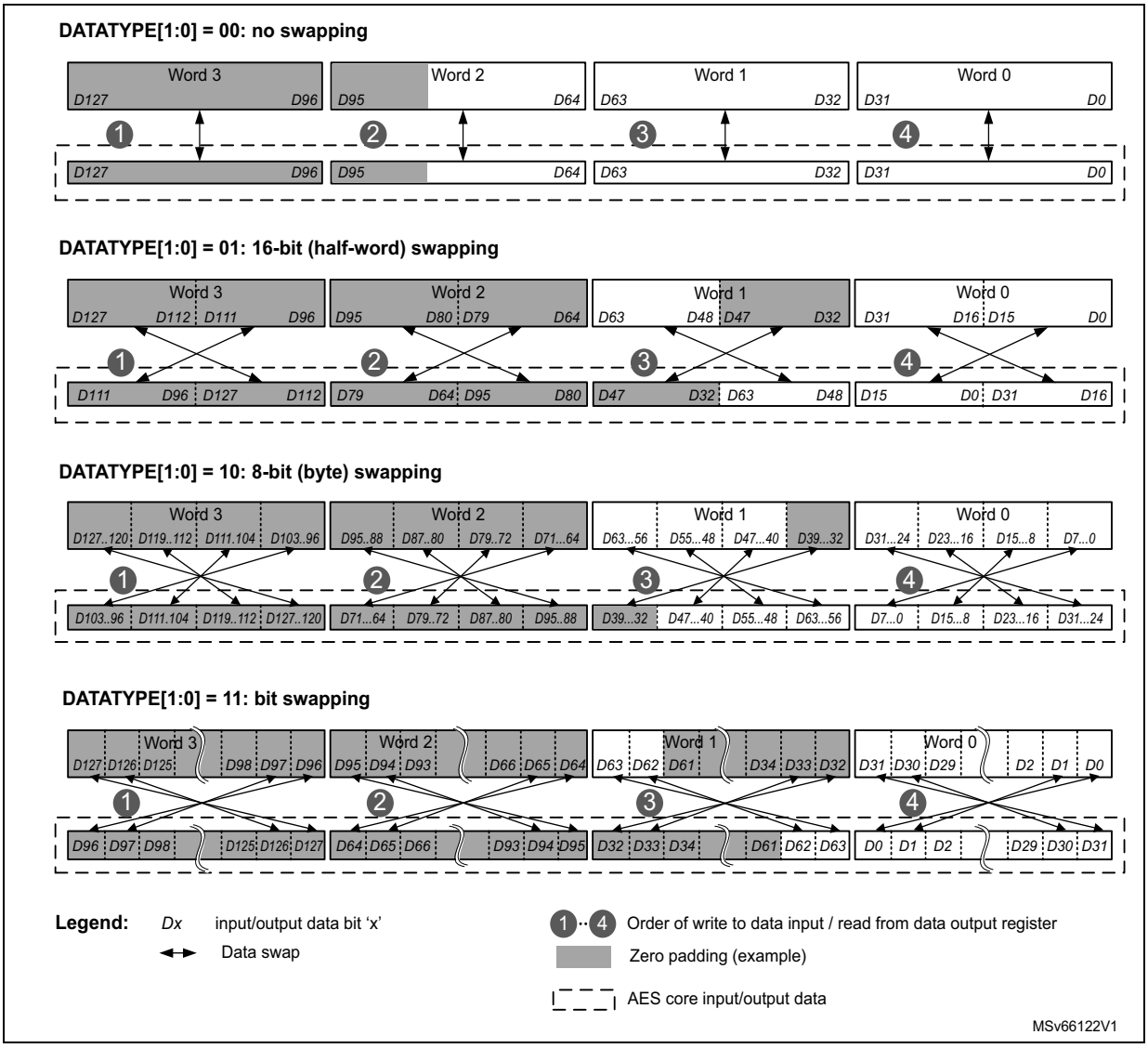
Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in *Table 224* and *Figure 212*.

Table 224. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 212. 128-bit block construction according to the data type



Data padding

Figure 212 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

28.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 225*. Reads are not allowed for security reason.

Note: In memory and in SAES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 225. Key endianness in SAES_KEYRx registers (128/256-bit keys)

SAES_KEYR7 [31:0]	SAES_KEYR6 [31:0]	SAES_KEYR5 [31:0]	SAES_KEYR4 [31:0]	SAES_KEYR3 [31:0]	SAES_KEYR2 [31:0]	SAES_KEYR1 [31:0]	SAES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
TAMP_BKP7R [31:0]	TAMP_BKP6R [31:0]	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 28.4.19*).

Key selection

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a nonvolatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information.

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 225*.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 225*.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID is cleared. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transfered to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see *Section 28.4.19*).

Note: DHUK, BHK and their XOR combination are not readable by software.

28.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in *Table 226*. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 226. IVI bitfield spread over SAES_IVRx registers

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

28.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 28.5: SAES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 28.5: SAES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 28.4.17: SAES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.
- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 28.4.15: SAES operation with shared keys* for details.
- **Hardware secret key loading error:** triggered upon failing load of DHUK or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK) or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 28.5: SAES interrupts*.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring SAES.

RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the RNGEIE bit of the SAES_IER register is set. For more details, refer to *Section 28.5: SAES interrupts*.

Upon an RNG error:

- Make sure that the RNG peripheral is properly configured.
- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

Note: To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

Managing tamper errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, reset the SAES peripheral through RCC or global reset.

28.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See *Section 28.4.19: SAES error management* for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 227 gives a summary of the available features.

Table 227. SAES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

28.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with SAES can be found in *Appending data using DMA* subsection of *Section 28.4.5: SAES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 28.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

28.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 228. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
128-bit	Encryption or decryption(2)	ECB, CBC, CTR	**480**
	Key preparation	-	**145**
256-bit	Encryption or decryption(2)	ECB, CBC, CTR	**680**
	Key preparation	-	**230**

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

Table 229. Processing latency for GCM and CCM (in SAES kernel clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	490	72(2)	480(3)	490
		CCM	490	490	800	490
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	650	72(2)	690(3)	650
		CCM	650	680	1350	650

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.

28.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

28.8.1 SAES control register (SAES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:28 **KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x4: XOR of DHUK and BHK

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID is cleared.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 27:26 **KSHAREID[1:0]**: Key share identification

This bitfield defines, at the end of a decryption process with KMOD[1:0] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.

0x0: AES peripheral

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the SAES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.

0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the key length in bits of the key used by SAES.

0: 128-bit

1: 256-bit

When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt.

Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **CPHASE[1:0]:** Chaining phase selection

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.

0x0: Initialization phase

0x1: Header phase

0x2: Payload phase

0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.

0: Disable

1: Enable

Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.

0: Disable

1: Enable

Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode

This bitfield selects the AES chaining mode:

0x0: Electronic codebook (ECB)

0x1: Cipher-block chaining (CBC)

0x2: Counter mode (CTR)

0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)

0x4: Counter with CBC-MAC (CCM)

others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the SAES operating mode:

0x0: Encryption

0x1: Key derivation (or key preparation), for ECB/CBC decryption only

0x2: Decryption

0x3: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 28.4.16: SAES data registers and data swapping*.

0x0: No swapping (32-bit data).

0x1: Half-word swapping (16-bit data)

0x2: Byte swapping (8-bit data)

0x3: Bit-level swapping

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the SAES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the SAES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:

- KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)
- KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

28.8.2 SAES status register (SAES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 28.4.17: SAES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether SAES is idle or busy.

0: Idle

1: Busy

SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral.

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected write to SAES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 0 Reserved, must be kept at reset value.

28.8.3 SAES data input register (SAES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.

Reads return zero.

28.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set triggers a read error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

28.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.

Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

28.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

28.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

28.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

28.8.9 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]
SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable).
Write to this register is ignored when EN bit is set in SAES_CR register

28.8.10 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

28.8.11 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

28.8.12 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

28.8.13 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

28.8.14 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

28.8.15 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

28.8.16 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

28.8.17 SAES suspend registers (SAES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: Suspend data

SAES_SUSPRx registers contain the complete internal register states of the SAES when the CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 28.4.8: SAES suspend and resume operations* for more details.

Clearing EN bit of the SAES_CR register clears this register to zero.

SAES_SUSPRx registers are not used in other chaining modes than CCM.

28.8.18 SAES interrupt enable register (SAES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIE:** RNG error interrupt enable
This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 2 **KEIE:** Key error interrupt enable
This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable
This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable
This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

28.8.19 SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected

1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in the SAES_SR register.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:

–SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4).

–SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4).

–AES fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2).

–SAES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the SAES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

28.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag clear
Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.

Bit 2 **KEIF:** Key error interrupt flag clear
Setting this bit clears the KEIF status bit of the SAES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear
Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 0 **CCF:** Computation complete flag clear
Setting this bit clears the CCF status bit of the SAES_ISR register.

28.8.21 SAES register map

Table 230. SAES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SAES_CR	IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1]	CHMOD[0]	MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	SAES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	SAES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SAES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SAES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 230. SAES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	SAES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SAES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SAES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SAES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SAES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SAES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SAES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SAES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SAES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SAES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SAES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	SAES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SAES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SAES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SAES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SAES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	SAES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	SAES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	SAES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 230. SAES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060-0x2FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	SAES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
	Reset value																													0	0	0	0
0x304	SAES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x308	SAES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x309-0x3FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

29 Hash processor (HASH)

29.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512).

29.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO that can receive a single data block of size depending on the selected algorithm
- Automatic padding to complete the input bit string to fit digest minimum block size
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- 8 × 32-bit words (H0 to H7) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers

– Hashing computation suspend/resume mechanism, including DMA

29.3 HASH implementation

The devices have a single instance of HASH peripheral.

The following table provides information on the hash algorithm implementation on different devices.

Table 231. HASH implementation on different devices

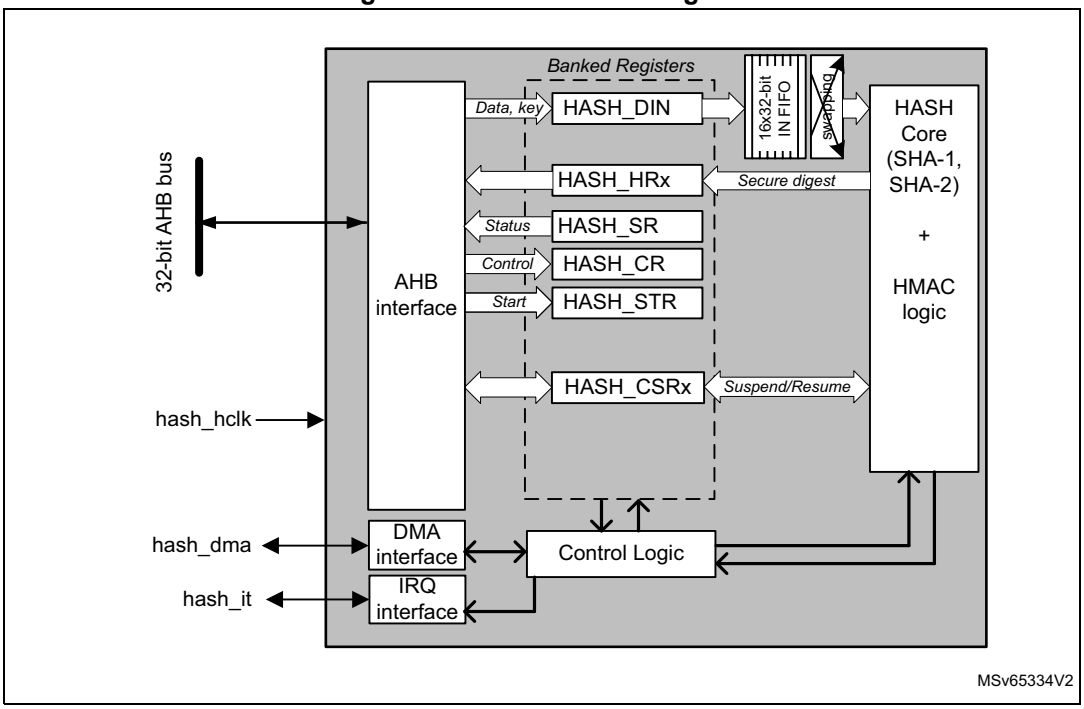
Hash algorithm	STM32C53x/C54x	STM32C55x/C56x	STM32C59x/C5Ax
SHA-1	X	X	X
SHA2-224	X	X	X
SHA2-256	X	X	X
SHA2-384	-	-	X
SHA2-512	-	-	X

29.4 HASH functional description

29.4.1 HASH block diagram

Figure 213 shows the block diagram of the hash processor.

Figure 213. HASH block diagram



29.4.2 HASH internal signals

Table 232 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 232. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

29.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 233*).

Table 233. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)(1)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA2-224	224			
SHA2-256	256			
SHA2-384	384	128	$< 2^{128}$ bits	Yes
SHA2-512	512(2)			

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

29.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with

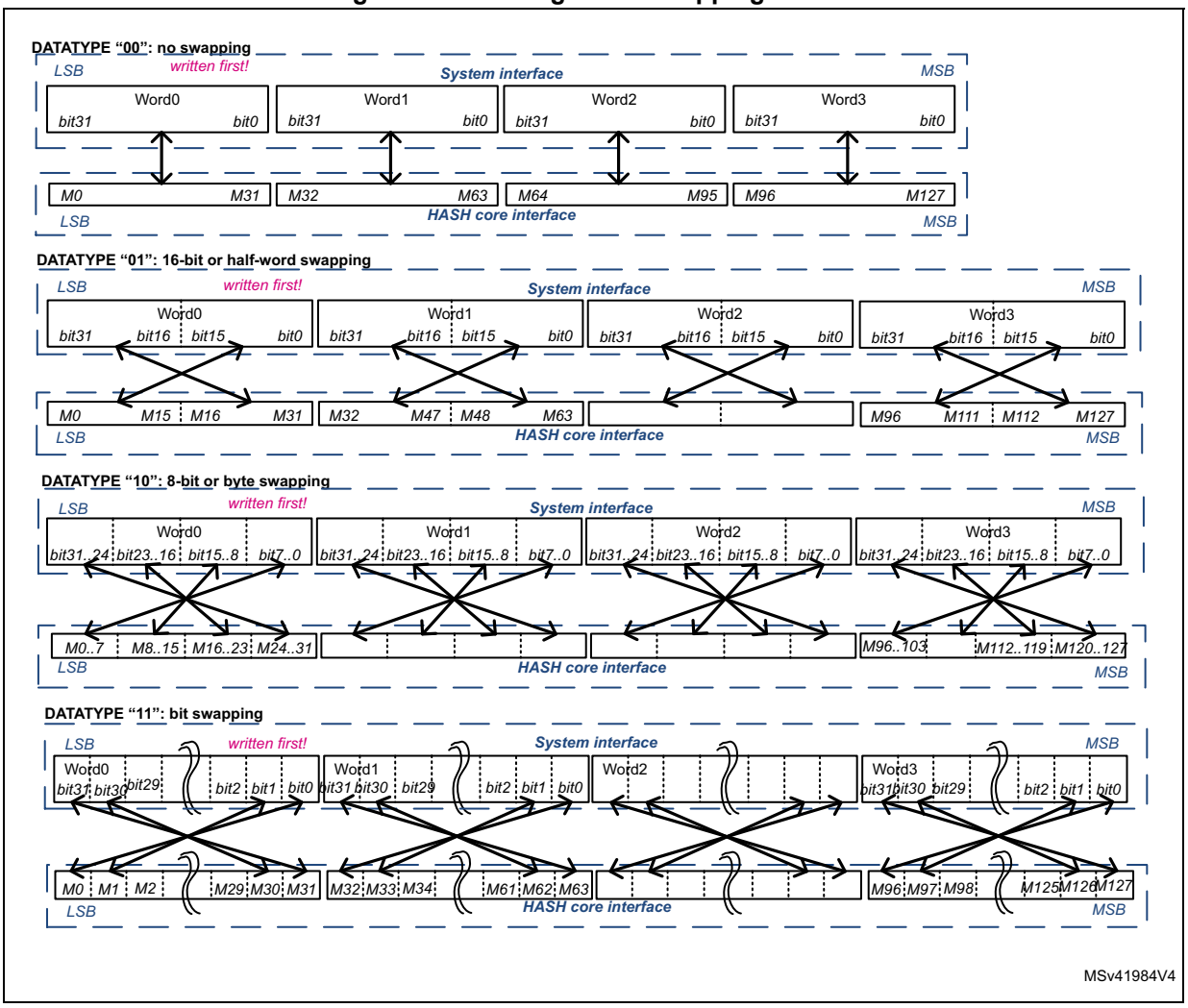
hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 214* shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 214. Message data swapping feature



29.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 233: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 29.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals **(NBWE-1) * 4**. This step is not required if the block size is already known (see *Table 233* for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 29.4.7: HMAC operation* for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 29.4.6* for details).
   - Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 29.4.6* for details).
   - Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA

transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 234*. When set, the DMAEN bit is automatically cleared by hardware.

Table 234. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256
SHA2-384	HASH_H0 to HASH_H11	HASH_H0[31]	384
SHA2-512	HASH_H0 to HASH_H15		512(1)

1. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to *Section 29.4.7: HMAC operation*.

Aborting HASH operation

HASH can be disabled at any time by writing 0x4 to HASH_CR, after stopping any running DMA transfer to HASH (if applicable).

29.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed hashing sequences with padding are described in *Section 29.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “**1**” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where ‘**U**’ means don’t care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

29.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
               Hash((Key | pad) XOR ipad | message))
```

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.
3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 29.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 29.4.11: HASH processing time.

Endianness management details can be found in Section 29.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   ```
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F
   ```

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

   ```
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU
   ```

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

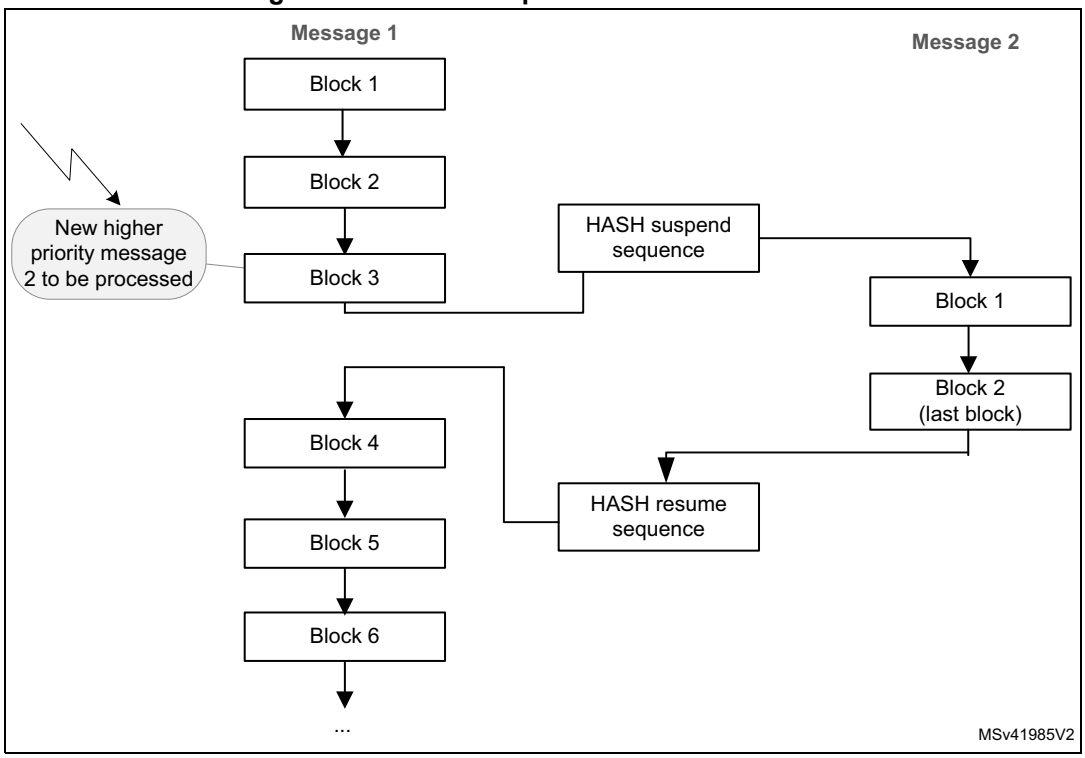
   ```
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
   ```

29.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 215*.

Figure 215. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 215*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

29.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 29.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

29.4.10 HASH error management

No error flags are generated by the hash processor.

29.4.11 HASH processing time

Table 235 summarizes the time required to process an intermediate block for each mode of operation.

Table 235. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA2-224		16	50	**66**
SHA2-256				
SHA2-384	128	32	66	**98**
SHA2-512(2)				

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

29.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 236* gives a summary of the available features.

Table 236. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

29.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

29.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:0]				LKEY
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	w		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

0000: SHA-1

0001: reserved

0010: SHA2-224

0011: SHA2-256

1100: SHA2-384

1101: SHA2-512/224

1110: SHA2-512/256

1111: SHA2-512

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see *Table 233: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bitfield is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit resets the hash processor core, so that HASH is ready to compute the message digest of a new message. Clearing this bit has no effect.

Bits 1:0 Reserved, must be kept at reset value.

29.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see *Table 233: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.

Do not write to this register when BUSY is set.

Reading this register returns zeros.

29.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							w				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

Note:

29.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 29.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 29.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 29.6.4: HASH digest registers* introduction.

29.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

29.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

29.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 53)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (others)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSx[31:0]:** Context swap x

Refer to *Section 29.6.7: HASH context swap registers* introduction.

29.6.8 HASH register map

Table 237. HASH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:2]		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NBWE[4:0]					DINNE		NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												1	0	0	0	1	0		0	0	0	0	0						0	0	0	1
Reserved																																	
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 237. HASH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x290	HASH_CSR102	CSR102[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 7) Last address: 0x32C	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

30 Public key accelerator (PKA)

This section applies entirely to the STM32C5A3xx devices. It only applies partly (functionality limited to ECDSA signature verification) to the STM32C59xxx devices.

30.1 PKA introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, the accelerator performs all computations with no further hardware or software elaboration required to process the inputs or the outputs.

For manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.

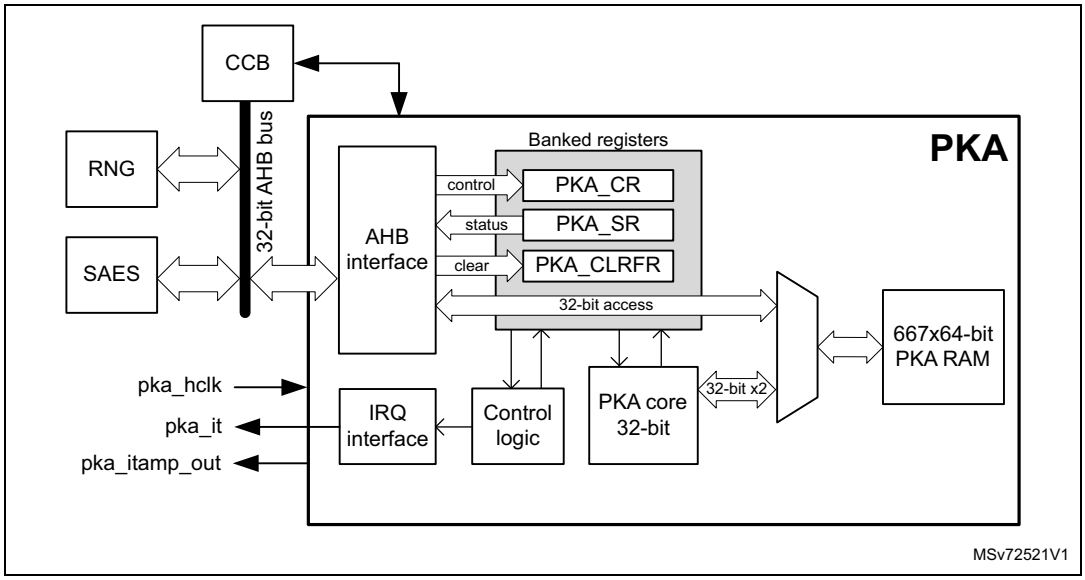
30.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Support for CCB chaining operations required to protect private key used in PKA protected operations
- Hardware protections to monitor usage of private keys during protected operation initialization

30.3 PKA functional description

30.3.1 PKA block diagram

Figure 216. PKA block diagram



30.3.2 PKA internal signals

Table 238 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 238. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). A tamper is also generated if application does not respect the maximum supported size of operands, or when PKA finds a fault in critical configuration registers(1). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

1. When tamper is active, the AHB interface is blocked: RAM and register reads are 0-ed and writes are ignored. There is no PKA RAM erase.

30.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 2000 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 30.3.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

30.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 30.5.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 239* and *Table 240*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 239. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R^2 mod *n*	*Section 30.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 30.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 30.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 30.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 30.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 30.4.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 30.4.7*
0x0D	001101	Modular reduction A mod n	*Section 30.4.8*
0x09	001001	Arithmetic addition A+B	*Section 30.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 30.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 30.4.11*

Table 239. PKA integer arithmetic functions list (continued)

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 30.4.12*
0x07	000111	RSA CRT exponentiation	*Section 30.4.13*

Table 240. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 30.4.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 30.4.15*
0x23	100011	ECC complete addition	*Section 30.4.18*
0x24	100100	ECDSA sign (protected)	*Section 30.4.16*
0x26	100110	ECDSA verification	*Section 30.4.17*
0x27	100111	ECC double base ladder	*Section 30.4.19*
0x2F	101111	ECC projective to affine	*Section 30.4.20*

Each of these operating modes has an associated code that has to be written to the MODE[5:0] bitfield in the PKA_CR register. If the application selects any value that is not documented below the write to MODE[5:0] bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in *Table 239* and *Table 240* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE[5:0] = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reasons, the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming, the application can decide to use a faster mode of operation during which the precomputed Montgomery parameter is supplied before starting the operation. The performance improvement is detailed in *Section 30.5.2: Computation times*.

The only operation using fast mode is the modular exponentiation (MODE[5:0] = 0x02).

30.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions such as:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

The following publications specify the listed functions:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

Refer to these documents for full details. This document limits itself to describing principles.

RSA key pair

For the following RSA operations, a public key and a private key information are defined as follows:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively, this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information, refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*), must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) × (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
4. Convert the integer message representative *m* to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

To accelerate these operations with the PKA, use *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process, the protected version of modular exponentiation (MODE[5:0] = 0x3) is strongly recommended for security reason. For the encryption process, the MODE[5:0] = 0x3 setting cannot be used as it requires the knowledge of the private key.

Elliptic curve selection

For the following ECC operations, the curve parameters are defined as follows:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 30.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, $n \times G$ = identity element O).

ECDSA message signature generation

This section describes the ECDSA (elliptic curve digital signature algorithm) signature generation function principle.

Alice, to sign a message *m* using her private key integer d_A, goes through the following steps:

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

The PKA accelerates the steps 4. to 7. by using:

- *ECDSA sign*, or
- All of the following operations:
  - *ECC Fp scalar multiplication* $k \times P$
  - *Modular reduction* A mod n
  - *Modular inversion* A^{-1} mod n
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

This section describes the ECDSA (elliptic curve digital signature algorithm) signature verification function principle.

Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A. Bob can verify that Q_A is a valid curve point by going through the following steps:

1. Check that Q_A is not equal to the identity element O
2. Check that Q_A is on the agreed curve
3. Check that $n \times Q_A = O$.

Then Bob executes the following procedure:

1. Verify that r and s are integer in [1, n-1]
2. Calculate e = HASH(m), where HASH is the agreed cryptographic hash function
3. Let z be the L_n leftmost bits of e
4. Calculate $w = s^{-1}$ mod n
5. Calculate $u_1 = zw$ mod n and $u_2 = rw$ mod n
6. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. The signature is valid if $r = x_1$ (mod n), it is invalid otherwise.

The PKA accelerates the steps 4. to 7. by using *ECDSA verification*.

30.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit in the PKA_CR register enables the PKA peripheral. The PKA becomes available when the INITOK bit is set in the PKA_SR register. When EN = 0, the PKA peripheral is kept under reset, with its memory still accessible by the application through the AHB interface.

Note: When the PKA is in the process of clearing its memory, the EN bit cannot be set.

When setting the EN bit in PKA_CR, make sure that the MODE[5:0] bitfield value corresponds to an authorized PKA operation (see OPERRF in Section 30.3.8).

Clearing the EN bit while a calculation is in progress aborts the operation. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only the PKA registers can be accessed, with writes to EN bits ignored.

If the INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable the PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 30.4*.

Executing a PKA operation

To execute any of the supported PKA operations, proceed as follows:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE[5:0] bitfield of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1), any access by the application to PKA RAM is ignored, and the RAMERRF flag is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error and prevents the PROCENDF flag from being set and the procedure step *3* from being completed. See *Section 30.3.8* for details.

Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 30.3.4*, when computing many operations with the same modulus, it can be beneficial for the application to compute the corresponding Montgomery parameter only once (see, for example, *Section 30.4.5*). This is referred to as *fast mode*.

To use the PKA peripheral in fast mode, follow this recommended procedure:

1. Load the modulus size and value information into the PKA RAM. Refer to *Section 30.5.1* for detail.
2. Configure the PKA_CR register to *Montgomery parameter computation* mode (MODE[5:0] = 0x1), then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set. Then, read the corresponding Montgomery parameter from the PKA memory and clear the PROCENDF bit by setting PROCENDFC in PKA_CLRFR.
4. Continue with the required PKA operation, loading the Montgomery information R2 mod *m* on top of the regular input data. Relevant addresses are provided in *Section 30.4*.

30.3.7 Private keys protection using CCB

Chaining modes

As shown in *Figure 217*, setting CCOP to a value different from 0 in the CCB peripheral automatically activates a chaining mechanism between PKA, RNG, and SAES, depending on the OPSTEP value (see *Table 241*), then sets the CCEN bit in the PKA_SR register.

For the effect of different OPSTEP values in each protected operation (ECDSA signature, scalar multiplication, modular exponentiation), refer to *Section 30.3.7: Private keys protection using CCB*.

Figure 217. Chaining of PKA with SAES or RNG

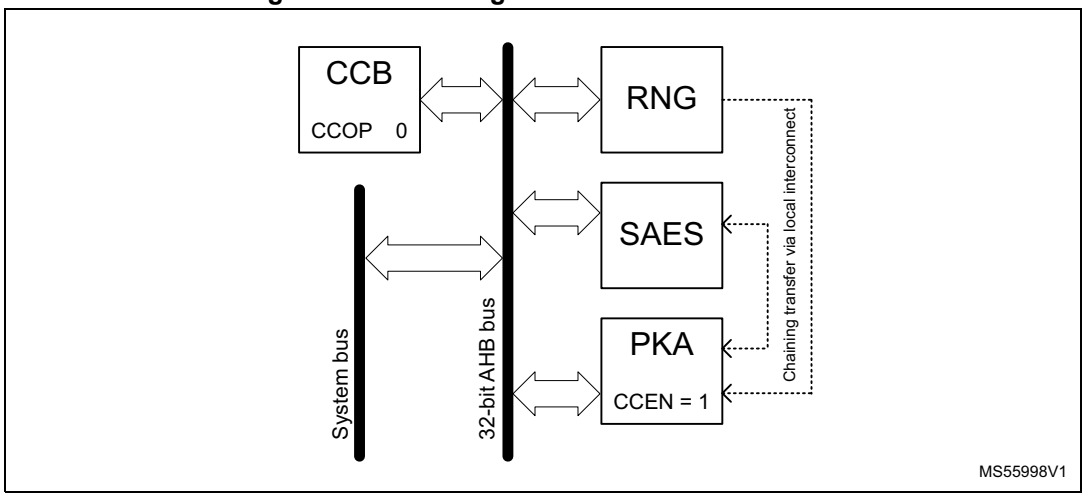


Table 241. Chaining mode to/from PKA

CCOP		OP STEP	Expected MODE[5:0] (1)	Chaining traffic(2)	Protected parameter written to PKA RAM	Expected flags(3)
x	0xC3	0x1	0x24	N/A	Hash (unprotected)	N/A
ECDSA signature	0xC0 or 0xC2	0x3	0x24	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0xC3	0x13				
		0x14		SAES → PKA	Decrypted private key d	DATAOKF or CMF error
		0x16		RNG → PKA	Random integer k	RNGOKF or CMF error
	0xC0	0x4		N/A	Clear text private key d	DATAOKF or CMF error
	0xC2	0x6		RNG → PKA	Random private key d	RNGOKF or CMF error
	0xC0 or 0xC2	0x8		PKA → SAES	Clear-text private key d	DATAOKF or CMF error
Scalar multiplication	0x80 or 0x82	0x3	0x20	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0x81	0x13				
		0x14		SAES → PKA	Decrypted k	DATAOKF or CMF error
	0x80	0x04		N/A	Clear text scalar mult k	
	0x82	0x6		RNG → PKA	Random scalar mult k	RNGOKF or CMF error
	0x80 or 0x82	0x8		PKA → SAES	Clear-text scalar mult k	DATAOKF or CMF error

Table 241. Chaining mode to/from PKA (continued)

CCOP		OP STEP	Expected MODE[5:0] (1)	Chaining traffic(2)	Protected parameter written to PKA RAM	Expected flags(3)
Modular exponentiation	0x44	0x3	0x3	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0x45	0x13				
		0x14		SAES → PKA	Decrypted exponent e	DATAOKF or CMF error
		0x15			Decrypted phi	
	0x44	0x4		N/A	Clear text exponent e	
		0x5			Clear text phi	
		0x8		PKA → SAES	Clear text exponent e	
		0x9			Clear text phi	

1. This mode must be written in PKA_CR before the START bit is set in PKA_CR.
2. Chaining traffic triggered by the CPU access to PKA RAM. “N/A” means no chaining traffic, only CPU traffic.
3. In PKA_SR bits 8 to 15 (“CMF error” refers to error flag bits 8 to 14).
4. In this mode chaining traffic includes the 32-bit data word plus the 32-bit physical address to PKA RAM.
5. Defined in *Table 242*.

When the application uses a PKA protected operation (with its private key), it must load the related blob information in the PKA RAM and in the SAES_DINR register. If the information exactly matches the conditions of the blob generation (the choice and order of writing protected parameters in PKA RAM, the selection of blob encryption key, and the usage of the original encrypted secrets), the CCB allows it to set the START bit in PKA_CR. When the PKA ignores the setting of the START bit, the application must reset the CCB using the IPRST bit in CCB_CR, then start the protected operation sequence again.

A protected application typically uses the PKA in the following sequence composed of three phases.

Step 1: Public phase

Before using a protected operation, the application may need to write unprotected parameters to the PKA RAM, as defined in *Table 241*.

Step 2: Integrity protected phase

To use a PKA operation, the application must load specific parameters in PKA RAM. When CCEN bit is set in PKA_SR, the integrity protected parameters defined in *Table 242* must always be written in the same way. Integrity protection is enforced in specific OPSTEPs, as defined in *Table 241*.

Typical issues with integrity of protected operation parameters are:

- The blob encryption key loaded in SAES registers is not correct.
- At least one confidentiality protected parameter defined in *Table 243* has an incorrect value, or was not properly written (DATAOKF = 0 or RNGOKF = 0, see *PKA status register (PKA_SR)*).
- At least one integrity protected parameter defined in *Table 242* has an incorrect value, is written at an incorrect address, or its write sequence to PKA RAM does not match

the sequence used during the blob creation. On this specific point, the application must follow the sequence detailed in *Table 242* when writing each item described in the column *Parameters*.

CCB feeds integrity protected parameters to SAES during the header phase of AES-GCM (GCMPH = 0x1).

Table 242. Integrity protected parameters list

Protected operation blob(1)	Parameters
ECC key blob	Curve prime order *n* length, curve modulus *p* length, curve coefficient *a* sign, curve coefficient |*a*|, curve coefficient *b*, curve modulus value *p*, curve prime order *n*
ECC signature blob	Same as scalar multiplication operation, adding curve base point G coordinates *x* and *y*
RSA key blob	Exponent *e* length, operand length, modulus value *n*

1. See *Table 243* for more details.

Step 3: Confidentiality phase

To use any PKA protected operation, the application must load confidentiality protected parameters in the PKA RAM. When the CCEN bit is set in PKA_SR, the parameters defined in *Table 243* are protected in confidentiality by:

- Making PKA RAM write-only
- Triggering a MODE[5:0] error (MDERRF) in PKA_SR when MODE[5:0] bitfiled in PKA_CR does not match the expected value when application sets START bit in PKA_CR
- Monitoring the correct management of protected parameters through DATAOKF and RNGOKF flags (see *PKA status register (PKA_SR)*)
- Optionally, generating ECC secrets from the on-chip TRNG when CCOP = 0xC2 or 0x82 and OPSTEP = 0x6, and when CCOP = 0xC3 and OPSTEP = 0x16. The PKA performs a sanity check before accepting this random value (see the RNGERRF flag).

Confidentiality protected parameters are described in *Table 243*. CCB feeds them to SAES during the payload phase of the AES-GCM (GCMPH = 0x2).

Table 243. Confidentiality protected parameter list

PKA operation	Secrets	Protected operation blob(1)	Start offset in PKA RAM	Size(2)
Modular exponentiation (protected, MODE[5:0] = 0x3)	Exponent e	RSA key blob	@0x14B8	ROS
	Phi value		@0xC68	
Scalar multiplication (MODE[5:0] = 0x20)	Scalar k	ECC key blob	@0x12A0	EOS
ECDSA signature (MODE[5:0] = 0x24)	Integer k	Not available	@0x12A0	EOS
	Private key d	ECC signature blob	@0xF28	

1. Information blob where confidentiality and integrity protected parameters are stored (see also *Table 242*)
2. See *Section 30.4.1* for details

Step 4: Running PKA protected operation

Set the START bit in PKA_CR. When the whole sequence executes correctly, the PKA RAM becomes inaccessible for reading and writing. Input parameters are automatically erased when the PROCENDF flag in PKA_SR is set, or if the application aborts the current operation by clearing the EN bit in PKA_CR.

30.3.8 PKA error management

The following error flags in the PKA_SR register are set to indicate errors that may occur when using the PKA:

- **ADDRERRF** flags that the access to PKA RAM falls outside the expected range.
- **RAMERRF** flags an AHB access to the PKA RAM occurring while the PKA core uses the PKA RAM. While the flag is set, reads form the PKA RAM return 0 and writes are ignored.
- **OPERRF** flags that the selected operating mode using MODE[5:0] bitfield is not listed in PKA operating modes (see the bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). While the flag is set, writes to the MODE[5:0] bitfield are ignored.
- An ongoing chaining mode sequence error raises the chaining mode flag **CMF** and the corresponding error flag:
  - **INCRERRF** flags increment error
  - **DATAZF** flags data zeroed error
  - **TRZERRF** flags trailing zeros error
  - **MDERRF** flags mode error
  - **RNGERRF** flags RNG error

  For details on each flag and on enabling the chaining mode sequence, see *Section 30.7.2*.

For each error flag, the PKA generates an interrupt if enabled with the corresponding bit in the PKA_CR register (see *Section 30.6* for details).

The ADDRERRF, OPERRF, and RAMERRF error flags are cleared by setting the ADDRERRFC, OPERRFC, and RAMERRFC bit in PKA_CLRFR, respectively. The OPERRF error flag must be cleared before a new operation is initiated via the PKA_CR register.

Upon an ongoing chaining mode sequence error, the application must set the CMFC bit in the PKA_CLRFR register, then restart the ongoing protected operation sequence in the CCB peripheral.

Note: The PKA can be re-initialized at any moment by clearing the EN bit in the PKA_CR register.

30.4 PKA operating modes

30.4.1 Introduction

The operations supported by the PKA are detailed in the following subsections, which define the format of both input data and results stored in the PKA RAM.

Warning: Before starting any operation, ensure that all input parameters to the PKA are valid and consistent, as the PKA assumes their correctness. Additionally, input parameters must not exceed the operand sizes specified in the operation tables.

The following information applies to all PKA operations:

- The PKA core processes 64-bit words in its RAM. Accordingly, in the PKA section of this document, 'word' denotes a 64-bit unit.
- When writing an element as input in the PKA RAM, append an additional 64-bit word of zeros after the most significant word, except when the operand size is fixed at one word.
- All RAM storage addresses refer to the least significant word of the data. To obtain the actual address, the application must add the base address of the PKA to the indicated offset.
- Supported operand sizes are:
  - **ROS (RSA operand size)**: Data size is calculated as *(rsa_size / 64) + 1* words, where *rsa_size* is the modulus length in bits. For example, a 1024-bit RSA operand corresponds to 17 words (1088 bits).
  - **EOS (ECC operand size)**: Data size is *(ecc_size / 64) + 1* words, where *ecc_size* is the prime modulus length in bits. For example, a 192-bit ECC operand corresponds to four words (256 bits).

  Note: Both ROS and EOS include the additional zero word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Any fractional results in the formulas must be rounded to the nearest integer, as the PKA core processes data in 64-bit words.

Note: The maximum ROS is 66 words (4160-bit maximum exponent size), while the maximum EOS is 11 words (640-bit max operand size).

Example 1: Understanding endianness in PKA memory during ECC Fp scalar multiplication preparation

Write the *x*-coordinate of point P for an ECC P256 curve (EOS = 5 words) by placing the least significant bit at bit 0 of the word at address offset 0x578, and the most significant bit at bit 63 of the word at address offset 0x590. Then, write a zero word (0x0000000000000000) at address offset 0x598.

Example 2: Writing the curve coefficient '*a*' for ECC Fp scalar multiplication

To write the value *a* = -3 for a curve with a modulus length of 224 bits (rounded up to five 64-bit words), write the PKA memory with data as in the following table:

Table 244. Example of *'a'* curve coefficient for ECC Fp scalar

RAM offset	Value	Note
0x410	0x0000000000000001	Curve coefficient *a* sign without extra word
0x418	0x0000000000000011	Value of |*a*| LSB

Table 244. Example of *'a'* curve coefficient for ECC Fp scalar

RAM offset	Value	Note
0x420	0x0000000000000000	-
0x428	0x0000000000000000	-
0x430	0x0000000000000000	Value of |*a*| MSB
0x438	0x0000000000000000	Additional zero word

30.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod n) used by the PKA to convert operands into the Montgomery residue system representation. This operation can be very useful for fast mode operation because passing the Montgomery parameter as input saves the time for its computation.

Note: This operation also supports ECC curves when specifying the prime modulus length and EOS size.

The following table summarizes operation instructions for Montgomery parameter computation.

Table 245. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	ROS
OUT	Result: R^2 mod *n*	-	RAM@0x620	

30.4.3 Modular addition

The following table summarizes operation instructions for modular addition operation that calculates *A* + *B* mod *n*.

Table 246. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	(0 ≤ *A* < *n*)	RAM@0xA50	ROS
	Operand *B*	(0 ≤ *B* < *n*)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: *A* + *B* mod *n*	(0 ≤ result < *n*)	RAM@0xE78	

30.4.4 Modular subtraction

Modular subtraction operation calculates:

- ***A* - *B* mod *n***, when $A \geq B$
- ***A* + *n* - *B* mod *n***, when $A < B$

The following table summarizes the operation instructions.

Table 247. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \leq A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \leq B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: *A* - *B* mod *n*	($0 \leq$ result $< n$)	RAM@0xE78	

30.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications, the PKA accelerates a multiplication that has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication $A \times B \bmod n$

The method to perform these operations is described next. The *x* function is this operation, and the *A*, *B*, and *C* operands are in the natural domain.

1. Inward or outward conversion to or from the Montgomery domain
   a) Assuming that *A* is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – The result of $AR = A \times r2modn \bmod n$ is *A* in the Montgomery domain.
   b) Assuming that *BR* is an integer in the Montgomery domain:
      – The result of $B = BR \times 1 \bmod n$ is *B* in the natural domain.
      – Similarly, the *AR* value computed in *a)* can be converted into the natural domain by computing $A = AR \times 1 \bmod n$.
2. Simple modular multiplication $A \times B \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $AB = AR \times B \bmod n$. The output is in the natural domain.
3. Multiple modular multiplication $A \times B \times C \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $BR = B \times r2modn \bmod n$. The output is in the Montgomery domain.
   d) Compute $ABR = AR \times BR \bmod n$. The output is in the Montgomery domain.
   e) Compute $CR = C \times r2modn \bmod n$. The output is in the Montgomery domain.
   f) Compute $ABCR = ABR \times CR \bmod n$. The output is in the Montgomery domain.
   g) (optional) Repeat the two previous steps if multiplying with additional operands is required.
   h) Compute $ABC = ABCR \times 1 \bmod n$ to retrieve the result in the natural domain.

The following table summarizes the operation instructions.

Table 248. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A \times B \bmod n$(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

30.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It calculates of $A^e \bmod n$.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 249* and *Table 250*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 251*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE[5:0] = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 249* (normal mode), *Table 250* (fast mode), and *Table 251* (protected mode). Fast mode usage is explained in *Section 30.3.6*.

Table 249. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 250. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
IN/OUT	Montgomery parameter R^2 mod *n*	(mandatory)	RAM@0x620	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 251. Modular exponentiation (protected mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x03	PKA_CR	6 bits
	Exponent *e* length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0x16C8	ROS
	Exponent *e*	($0 \le e < n$)	RAM@0x14B8	
	Modulus value *n*	(odd integer only, $n < 2^{4096}$)	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	
ERROR	Error A^e mod *n*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n[1] with phi = (p - 1) × (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (p, q, e, d) instead of (N, e, d, *phi*), in this case, to derive N and *phi* using PKA arithmetic multiplier: $N = p \times q$, and *phi* = (p - 1) × (q - 1).

30.4.7 Modular inversion

The modular inversion operation calculates multiplicative inverse A^{-1} mod n. If the modulus n is prime, for all values of A ($1 \le A < n$) modular inversion output is valid. If the modulus n is not prime, A has an inverse only if the greatest common divisor between A and n is 1.

If the operand A is a divisor of the modulus n the result is a multiple of a factor of n.

The following table summarizes the operation instructions.

Table 252. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result: A^{-1} mod *n*	0 < result < *n*	RAM@0xE78	

30.4.8 Modular reduction

The modular reduction operation calculates the remainder of A divided by n.

The following table summarizes the operation instructions.

Table 253. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand *A*	($0 \le A < 2n < 2^{4160}$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result *A* mod *n*	($0 <$ result $< n$)	RAM@0xE78	

30.4.9 Arithmetic addition

The arithmetic addition operation calculates *A* + *B*.

The following table summarizes the operation instructions.

Table 254. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x09	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* + *B*	($0 \le$ result $< 2^{M+1}$)	RAM@0xE78	ROS + 1

30.4.10 Arithmetic subtraction

The arithmetic subtraction operation calculates:

- ***A* - *B***, when $A \ge B$
- $\boldsymbol{A + 2^{int(M / 32) \times 32 + 1} - B}$, when $A < B$ and the residue of $M / 32$ is greater than zero
- $\boldsymbol{A + 2^{int(M / 32) \times 32} - B}$, when $A < B$ and the residue of $M / 32$ is zero

For the last two list items, the 32-bit word following the most significant word of the output is 0xFFFF FFFF, as the result is negative.

The following table summarizes the operation instructions.

Table 255. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0A	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le < A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le < B < 2^M$)	RAM@0xC68	
OUT	Result: *A* - *B*	($0 \le$ result $< 2^M$)	RAM@0xE78	

30.4.11 Arithmetic multiplication

The arithmetic multiplication operation calculates A × B.

The following table summarizes the operation instructions.

Table 256. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0B	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* × *B*	($0 \le$ result $< 2^{2M}$)	RAM@0xE78	2xROS

30.4.12 Arithmetic comparison

The arithmetic comparison operation yields the following results:

- **0xED2C**, when *A* = *B*
- **0x7AF8**, when *A* > *B*
- **0x916A**, when *A* < *B*

The following table summarizes the operation instructions.

Table 257. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0C	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	If result *A* = *B*	0xED2C, 0x7AF8, or 0x916A	RAM@0xE78	64 bits

30.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation m = A^d (mod pq) more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 258*.

Table 258. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	($0 < d_P < 2^{M/2}$)	RAM@0x730	ROS / 2
	Operand d_Q	($0 < d_Q < 2^{M/2}$)	RAM@0xE78	
	Operand q_{inv}	($0 < q_{inv} < 2^{M/2}$)	RAM@0x948	
	Prime p(1)	($0 < p < 2^{M/2}$)	RAM@0xB60	
	Prime q(1)	($0 < q < 2^{M/2}$)	RAM@0x1088	
IN	Operand A	($0 < A < 2^M$)	RAM@0x12A0	ROS
OUT	Result: A^d mod pq	($0 \leq$ result < pq)	RAM@0x838	

1. Must be different from 2.

30.4.14 Point on elliptic curve Fp check

This operation checks whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

The following table summarizes the operation instructions.

Table 259. Point on elliptic curve *Fp* check

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(|*b*| < *p*)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < *p* < 2640)	RAM@0x470	
	Point P coordinate *x*	(*x* < p)	RAM@0x578	
	Point P coordinate *y*	(*y* < *p*)	RAM@0x5D0	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Result: point *P* on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: *x* or *y* coordinate is not smaller than modulus *p*	RAM@0x680	64 bits

30.4.15 ECC Fp scalar multiplication

This operation calculates $k \times P\,(x_P, y_P)$, where *P* is a point on a curve over prime fields and *x* is the elliptic curve scalar point multiplication. The result is a point that belongs to the same curve or a point at infinity.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation *point on elliptic curve* can be used to avoid this.

The following table summarizes the operation instructions.

Table 260. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 260. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier *k*	($0 \le k < n$)	RAM@0x12A0	
	Point P coordinate x_P	($x_P < p$)	RAM@0x578	
	Point P coordinate y_P	($y_P < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: *k* × *P* coordinate *x'*	(result < *p*)	RAM@0x578	EOS
	Result: *k* × *P* coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error *k* × *P*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For *k* = 0 this function returns a point at infinity (0, 0) if curve parameter *b* is nonzero, (0, 1) otherwise. For *k* different from 0 it might happen that a point at infinity is returned. When the application detects this behavior, a new computation must be carried out.
- For *k* < 0 (that is, a negative scalar multiplication is required) the multiplier absolute value
  k = |-*k*| must be provided to the PKA. After the computation completion, the formula -*P* = (*x*, -*y*) can be used to compute the *y* coordinate of the effective final result (the *x* coordinate remains the same).

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

30.4.16 ECDSA sign

Table 261 (input parameters) and *Table 262* (output parameters) summarize the ECDSA signing operation.

Upon an output error different from 0xD60D, the application must generate a new *k* and repeat the ECDSA sign operation.

Note: As long as this operation is ongoing (BUSY = 1), the PKA control register and the PKA memory are not accessible. The access is restored when the PKA clears the BUSY bit.

The operation completion with errors due to unexpected hardware events triggers a PKA tamper event in the TAMP peripheral and blocks access to the PKA RAM until the hardware erases the event. Missing programmed input point on the input ECC curve also triggers a PKA tamper event. The use of the *point on elliptic curve* PKA operation prevents this.

Table 261. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x24	PKA_CR	6 bits
	Curve prime order *n* length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer *k*(1)	($0 \leq k <$ n)	RAM@0x12A0	
	Curve base point *G* coordinate *x*	($x < p$)	RAM@0x578	
	Curve base point *G* coordinate *y*	($y < p$)	RAM@0x470	
	Hash of message *z*	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key *d*	($0 < d < n$)	RAM@0xF28	
	Curve prime order *n*	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases *k* can be deterministically generated.
2. To match the hash parameter size to the length of the curve prime order *n,* pad the result with zeros or use the hash truncation.

Table 262. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	($0 < r < n$)	RAM@0x730	EOS
	Signature part *s*	($0 < s < n$)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: The error code is not affected when the PKA automatically clears the PKA RAM at the end of this protected operation. The error output 0xD60D does not affect the result output either.

Extended ECDSA support

The PKA also supports the extended ECDSA signature, which accepts the same inputs as the standard ECDSA signature (see *Table 261*) and produces similar outputs (see *Table 262*), with the addition of the point *kG* coordinates. The following table defines these additional outputs.

Table 263. Extended ECDSA sign - additional outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point *kG* coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point *kG* coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

30.4.17 ECDSA verification

Table 264 (input parameters) and *Table 265* (output parameters) summarize the ECDSA verification operation.

Upon an output error different from 0xD60D, the signature is not verified.

Table 264. ECDSA verification - inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x26	PKA_CR	6 bits
	Curve prime order *n* length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient |*a*|	(absolute value, $\|a\| < p$)	RAM@0x470	EOS
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x4D0	
	Curve base point *G* coordinate *x*	($x < p$)	RAM@0x678	
	Curve base point *G* coordinate *y*	($y < p$)	RAM@0x6D0	
	Public-key curve point *Q* coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point *Q* coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part *r*	($0 < r < n$)	RAM@0x10E0	
	Signature part *s*	($0 < s < n$)	RAM@0xC68	
	Hash of message *z*	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order *n*	(integer prime)	RAM@0x1088	

1. To match the hash parameter size to the length of the curve prime order *n,* pad the result with zeros or use the hash truncation.

Table 265. ECDSA verification - outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– ($0 < r < n$)	RAM@0x578	EOS

30.4.18 ECC complete addition

The ECC complete addition operation calculates the sum of two points on an elliptic curve.

Operation instructions are summarized in *Table 266*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. To convert resulting point to Jacobian coordinates conversion (x, y)= (X / Z^2, Y / Z^3) can be used.

Table 266. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient |*a*|	(absolute value, $\|a\| < p$)	RAM@0x418	
	First point P coordinate *X*	($x < p$)	RAM@0x628	
	First point P coordinate *Y*	($y < p$)	RAM@0x680	
	First point P coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point Q coordinate *X*	($x < p$)	RAM@0x730	
	Second point Q coordinate *Y*	($y < p$)	RAM@0x788	
	Second point Q coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *X*	($x < p$)	RAM@0xD60	
	Result coordinate *Y*	($y < p$)	RAM@0xDB8	
	Result coordinate *Z*	($z < p$)	RAM@0xE10	

30.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of $k \times P + m \times Q$, where (P, Q) are two points on an elliptic curve and (k, m) are two scalars. Operation instructions are summarized in *Table 267*.

If the resulting point is the point at infinity (error code 0xA3B7), the resulting coordinate equals to (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 267. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer *k*	($0 < k < 2^{640}$)	RAM@0x520	
	Integer *m*	($0 < m < 2^{640}$)	RAM@0x578	
	First point *P* coordinate *X*	($x < p$)	RAM@0x628	
	First point *P* coordinate *Y*	($y < p$)	RAM@0x680	
	First point *P* coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point *Q* coordinate *X*	($x < p$)	RAM@0x730	
	Second point *Q* coordinate *Y*	($y < p$)	RAM@0x788	
	Second point *Q* coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *x*	($x < p$)	RAM@0x578	EOS
	Result coordinate *y*	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

30.4.20 ECC projective to affine

ECC projective to affine operation calculates the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (*X*, *Y*, *Z*), it calculates the affine coordinates (*x*, *y*) = (*X* / *Z*, *Y* / *Z*).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 268*.

Table 268. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate *X* (projective)	($x < p$)	RAM@0xD60	
	Point P coordinate *Y* (projective)	($y < p$)	RAM@0xDB8	
	Point P coordinate *Z* (projective)	($z < p$)	RAM@0xE10	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Point P coordinate *x* (affine)	($x < p$)	RAM@0x578	
	Point P coordinate *y* (affine)	($y < p$)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

30.5 Example of configurations and processing times

30.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b$ (mod p).

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field *GF(p)*
- the (usually prime) integer *n*, the order of the group generated by *G*, defined below
- the base point of the curve *G*, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field *GF(p)*, for example ***p*-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 269 summarizes the family of curves supported by PKA for ECC operations.

Table 269. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

30.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 270. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
2^{16} + 1	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Protected	9958000	-	-	-
	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	CRT(1)	1775000	-	-	-
2048	Protected	-	63886000	-	-
	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	CRT(1)	-	11670000	-	-
3072	Protected	-	-	199403000	-
	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	CRT(1)	-	-	36886000	-
4096	Protected	-	-	-	454318000
	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	CRT(1)	-	-	-	84577000

1. CRT stands for chinese remainder theorem optimization (MODE[5:0] bitfield= 0x07).

Table 271. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	1590000	3083000	5339000	8518000	17818000	21053000	31826000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 272. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	1500000	2744000	4579000	7184000	14455000	16685000	24965000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 273. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 274. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
967000	1419000	2768000	4784000	7547000	15854000	18257000	28257000

Table 275. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
47600	78000	148300	253000	419000	838400	1049300

Table 276. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
10000	12000	18000	26000	39000	63000	89000

Table 277. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
3400	4200	6100	8300	10900	17200	-	-

Table 278. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
8600	8710	11870	17000	102000	410000	506000	822000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

30.6 PKA interrupts

There are five individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 30.3.8*
2. Access to unmapped address (ADDRERRF), see *Section 30.3.8*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 30.3.8*
4. PKA end of operation (PROCENDF)
5. Chaining mode flags (CMF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

CMF interrupt is triggered by the end of an operation (DATAOKF, RNGOKF) or by a chaining mode error (see *Section 30.3.8*). Chaining modes are described in *Section 30.3.7*.

Table 279 gives a summary of the available features.

Table 279. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit
	Chaining mode flag	CMF	CMFIE	Set CMFC bit

30.7 PKA registers

30.7.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFIE	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
									rw	rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMFIE**: Chaining mode flags interrupt enable
0: No interrupt is generated when any flag in bits 8 to 15 is set in PKA_SR.
1: An interrupt is generated when any flag in bits 8 to 15 is set in PKA_SR.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication (protected)
100100: ECDSA sign (protected)
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE[5:0] bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE[5:0] different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE[5:0] bit is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA

1: Enable PKA; PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE[5:0] bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

30.7.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMF	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
									r	r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RNGERRF	MDERRF	TRZERRF	DATAZF	INCRERRF	DATAOKF	RNGOKF	Res.	Res.	Res.	Res.	Res.	CCEN	LMF	INITOK
	r	r	r	r	r	r	r						r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMF**: Chaining mode flags

This status bit is only used when CCEN status bit is set in this register.

0: No chaining mode events.

1: Ongoing chaining mode sequence triggered at least one flag status in bits 8 to 14 of this register.

This bit is cleared using CMFC bit in PKA_CLRFR.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

When CCEN = 1 PKA sets BUSY bit when it runs a sanity check on a random number that CCB wrote to the PKA RAM during a chaining sequence (see *Section 30.3.7*).

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RNGERRF**: RNG error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No random number drawn from RNG, or generated ECC private key is NIST compliant.

1: Random number drawn from RNG did not pass NIST “key pair generation by testing candidates” equivalent algorithm (see FIPS 186-4 annex B.4.2). The last chaining operation must be restarted, otherwise the selected PKA operation remains unusable.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 13 **MDERRF**: Mode error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No mode error

1: The value written in MODE[5:0] of PKA_CR registers is different from the expected value corresponding to the CCOP mode selected in the CCB peripheral. Refer to *Section 30.3.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 12 **TRZERRF**: Trailing zeroes error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No trailing 0s error

1: Clear-text coming from CPU, or decrypted data coming from SAES does not have the required trailing 0s.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 11 **DATAZF**: Data zeroed error flag

This read-only chaining mode error flag is active only when CCEN is set in this register. DATAZF is set when PKA finds the complete data written to PKA RAM equal to 0, with all the expected 32-bit words null. This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 10 **INCRERRF**: Increment error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No increment error

1: During the reading or writing of protected parameters from or to the PKA RAM, the application did not properly incremented the address. Refer to *Section 30.3.7* for details.

During the read or write of an operand, and before DATAOKF or RNGOKF is set, the INCRERRF flag stays cleared if the application reads or writes again this operand (starting from the lowest address). This bit is cleared by setting the CMFC bit in PKA_CLRFR.

Bit 9 **DATAOKF**: Data OK flag

This read-only chaining mode status flag is active only when CCEN is set in this register.

0: Data not OK (if applicable), otherwise no event.

1: Sequential writes or reads to or from the PKA RAM successfully transfered the expected amount of data from or to SAES. Alternatively, the CPU wrote the correct amount of data in the PKA RAM. Refer to *Section 30.3.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 8 **RNGOKF**: RNG OK flag

This read-only chaining mode status flag is active only when CCEN is set in this register.

0: Random number not OK (if applicable). No event otherwise.

1: Sequential writes to PKA RAM successfully transfered the expected amount of random data from the RNG. Refer to *Section 30.3.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CCEN**: Coupling and chaining mode enable

This bit is set when the application configures the CCB to run coupling and chaining sequences that perform protected PKA operation using the SAES and (optionally) RNG peripherals.

0: Access to PKA RAM and programming of PKA_CR has no extra limitation.

1: The PKA RAM is write-only until START is set (it is no access then), and the MODE[5:0] value written in PKA_CR must correspond to the CCOP mode selected in the CCB peripheral. The application must also follow the coupling and chaining sequences described in *Section 30.3.7*.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE[5:0] bitfield can be used.

1: Only ECDSA verification (MODE[5:0] = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK

This bit is asserted when PKA initialization is complete.

0: PKA is not initialized correctly. START bit cannot be set.

1: PKA is initialized correctly and can be used normally.

When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

30.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFC	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.
									w	w	w	w		w	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMFC**: Clear chaining mode flag

0: No action

1: Clear the CMF flag and the chaining mode status bits in 8 to 15 in PKA_SR

Bit 21 **OPERRFC**: Clear operation error flag

0: No action

1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag

0: No action

1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag

0: No action

1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

30.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 30.3.3: PKA reset and clocks.

30.7.5 PKA register map

Table 280. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFIE	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value										0	0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMF	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	RNGERRF	MDERRF	TRZERRF	DATAZF	INCRERRF	DATAOKF	RNGOKF		Res.	Res.	Res.	Res.	Res.	CCEN	LMF	INITOK
	Reset value										0	0	0	0		0	0		0	0	0	0	0	0	0						0	0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFC	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0	0		0																	

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

31 Advanced-control timers (TIM1/TIM8)

31.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 31.3.35: Timer synchronization*.

31.2 TIM1/TIM8 main features

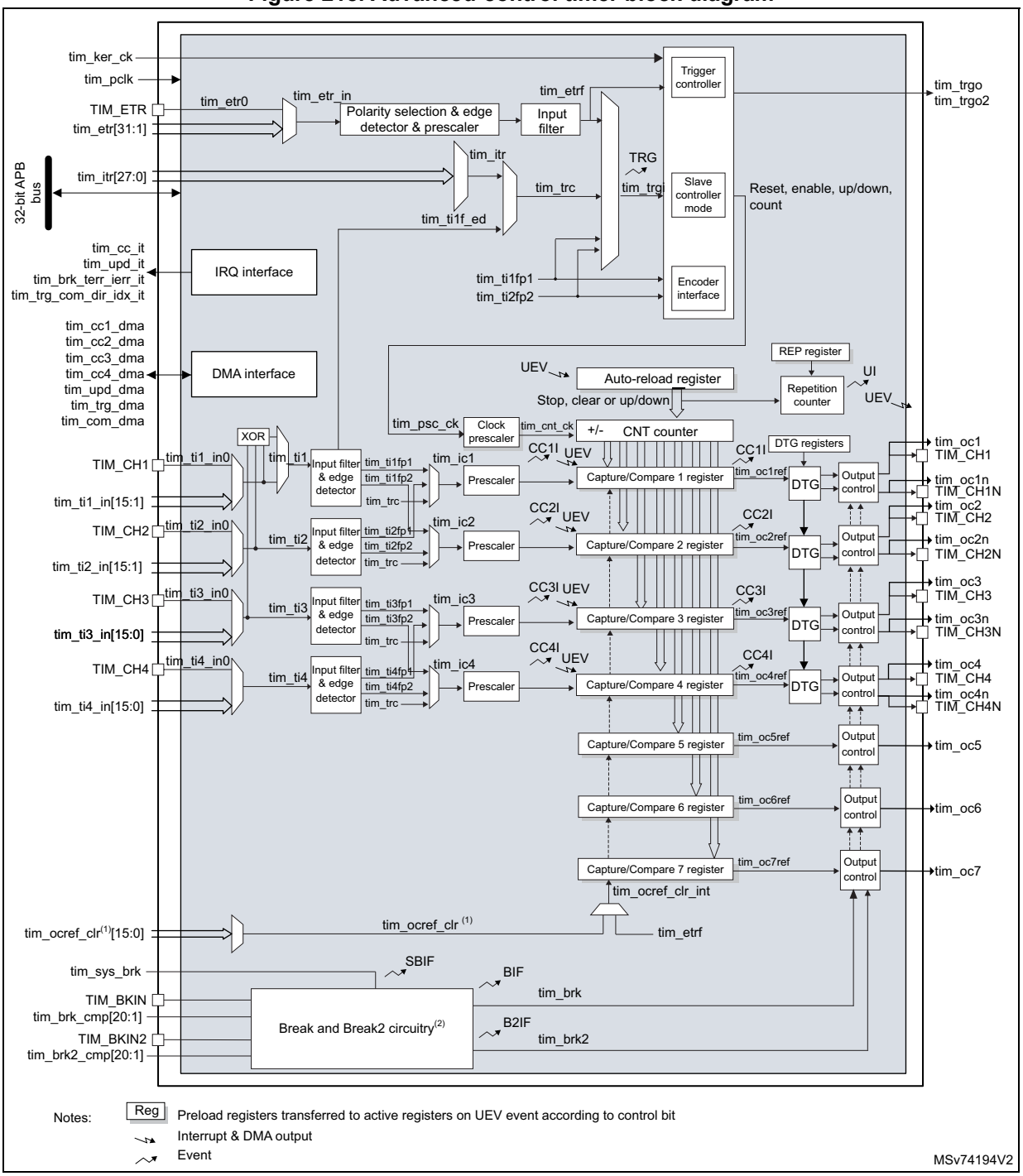
TIM1/TIM8 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Seven independent channels for:
  - Input capture (but channels 5, 6 and 7)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Incremental encoders, quadrature encoders and hall-sensors support
- Trigger input for external clock or cycle-by-cycle current management
- ADC synchronization for jitter-free sampling points

31.3 TIM1/TIM8 functional description

31.3.1 Block diagram

Figure 218. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals*.
2. See *Figure 269: Break and Break2 circuitry overview* for details.

31.3.2 TIM1/TIM8 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 281. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input / Output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 282. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[31:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 282. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocx (x = 1 to 4)	Output	Internal output compare signals of channels 1 to 4. These outputs can be used for triggering/synchronizing other internal peripherals.
tim_ocxn (x = 1 to 4)	Output	Internal timer complementary outputs, derived from tim_ocx internal outputs with the possibility to have deadtime insertion.
tim_ocx (x = 5 to 7)	Output	Internal output compare signals of channels 5 to 7. These outputs can be used for triggering/synchronizing other internal peripherals.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[20:1]	Input	Break input for internal signals
tim_brk2_cmp[20:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 283, *Table 284*, *Table 285*, and *Table 286* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 283. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM1	TIM8
tim_ti1_in0	TIM1_CH1	TIM8_CH1
tim_ti1_in1	comp1_out	
tim_ti1_in2	comp2_out	
tim_ti1_in[15:2]	Reserved	

Table 284. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM1	TIM8
tim_ti2_in0	TIM1_CH2	TIM8_CH2
tim_ti2_in[15:1]	Reserved	

Table 285. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources	
	TIM1	TIM8
tim_ti3_in0	TIM1_CH3	TIM8_CH3
tim_ti3_in[15:1]	Reserved	

Table 286. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources	
	TIM1	TIM8
tim_ti4_in0	TIM1_CH4	TIM8_CH4
tim_ti4_in[15:1]	Reserved	

Table 287 lists the internal sources connected to the tim_itr input multiplexer.

Table 287. Internal trigger connection

Timer internal trigger input signal	TIM1	TIM8
tim_itr0	Reserved	tim1_trgo
tim_itr1		tim1_trgo2
tim_itr2	tim2_trgo	
tim_itr3	tim3_trgo	
tim_itr4	tim4_trgo	

Table 287. Internal trigger connection (continued)

Timer internal trigger input signal	TIM1	TIM8
tim_itr5	tim5_trgo	
tim_itr6	tim8_trgo	Reserved
tim_itr7	tim8_trgo2	
tim_itr8	tim12_trgo	
tim_itr9	tim15_trgo	
tim_itr10	tim16_oc1	
tim_itr11	tim17_oc1	
tim_itr[27:12]	Reserved	

Table 288 lists the internal sources connected to the tim_etr input multiplexer.

Table 288. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM1	TIM8
tim_etr0	TIM1_ETR	TIM8_ETR
tim_etr1	comp1_out	
tim_etr2	Reserved	
tim_etr3	adc1_awd1	adc2_awd1
tim_etr4	adc1_awd2	adc2_awd2
tim_etr5	adc1_awd3	adc2_awd3
tim_etr6	Reserved	adc3_awd1
tim_etr7	Reserved	adc3_awd2
tim_etr8	Reserved	adc3_awd3
tim_etr[15:9]	Reserved	

Table 289. Interconnect to the tim_bkr_cmp multiplexer

tim_brk inputs	TIM1	TIM8
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin
tim_brk_cmp1	comp1_out	
tim_brk_cmp2	comp2_out	
tim_brk_cmp[4:3]	Reserved	
tim_brk_cmp5	Reserved	TIM1_BKIN
tim_brk_cmp6	TIM8_BKIN	Reserved
tim_brk_cmp7	Reserved	

Table 289. Interconnect to the tim_bkr_cmp multiplexer (continued)

tim_brk inputs	TIM1	TIM8
tim_brk_cmp8	TIM15_BKIN	
tim_brk_cmp9	TIM16_BKIN	
tim_brk_cmp10	TIM17_BKIN	

Table 290. Timer break2 interconnect

tim_brk2 inputs	TIM1	TIM8
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin
tim_brk2_cmp1	comp1_out	
tim_brk2_cmp2	comp2_out	
tim_brk2_cmp[4:3]	Reserved	
tim_brk2_cmp5	Reserved	TIM1_BKIN2
tim_brk2_cmp6	TIM8_BKIN2	Reserved
tim_brk2_cmp[10:7]	Reserved	

Table 291 list the sources connected to the tim_ocref_clr mutiplexer.

Table 291. Interconnect to tim_ocref_clr multiplexer

tim_ocref_clr inputs	TIM1	TIM8
tim_ocref_clr0	comp1_out	
tim_ocref_clr1	comp2_out	
tim_ocref_clr[7:2]	Reserved	

Table 292. System break interconnect

tim_sys_brk inputs	TIM1/TIM8	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	SRAM double ECC error	SEL
tim_sys_brk2	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk3	Non-Volatile memory double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)

31.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)
- Repetition counter register (TIM_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 219 and *Figure 220* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 219. Counter timing diagram with prescaler division change from 1 to 2

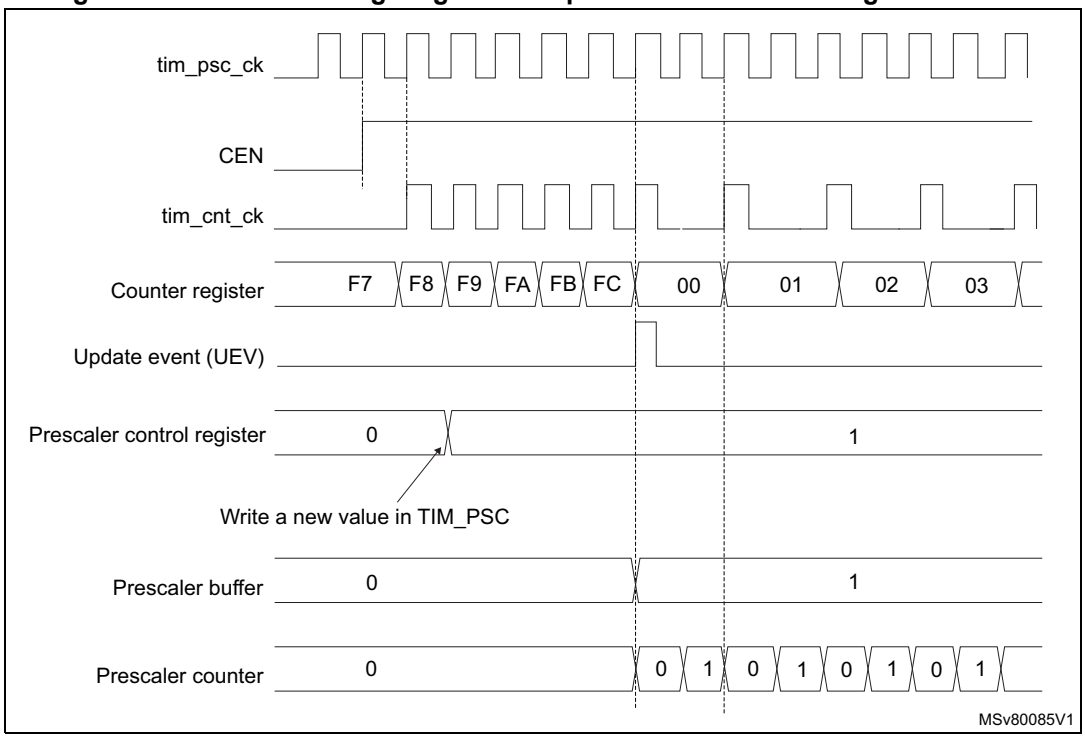
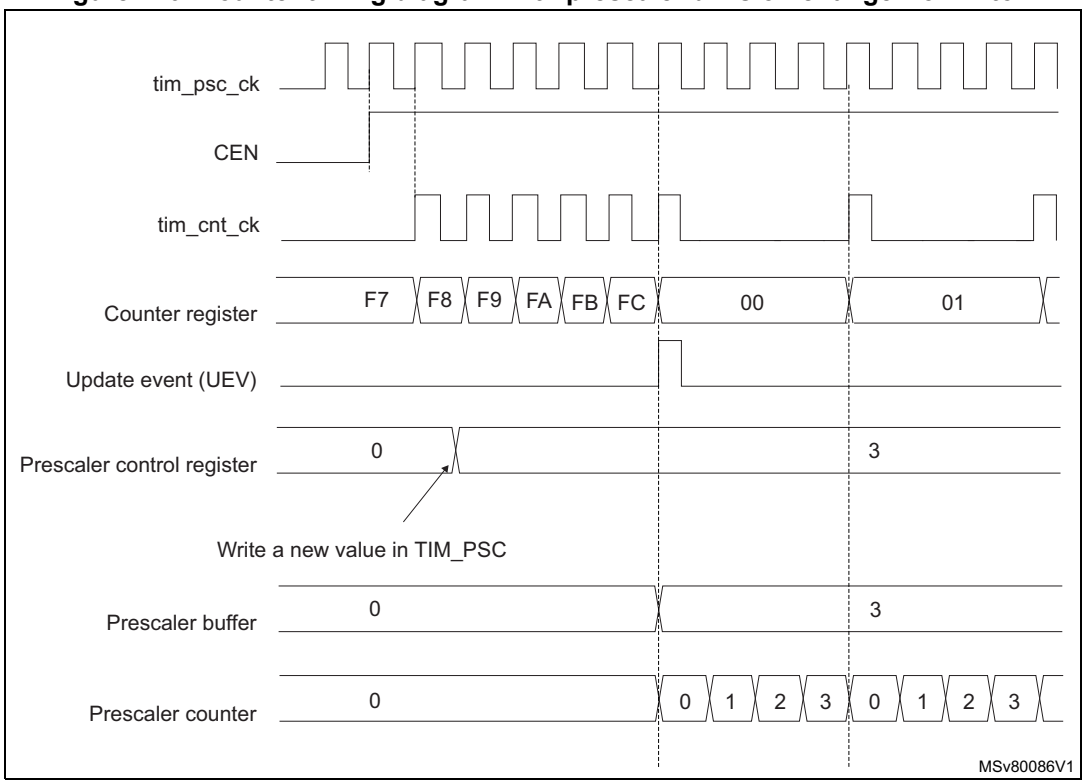


Figure 220. Counter timing diagram with prescaler division change from 1 to 4



31.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register,
- The autoreload shadow register is updated with the preload value (TIM_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 221. Counter timing diagram, internal clock divided by 1

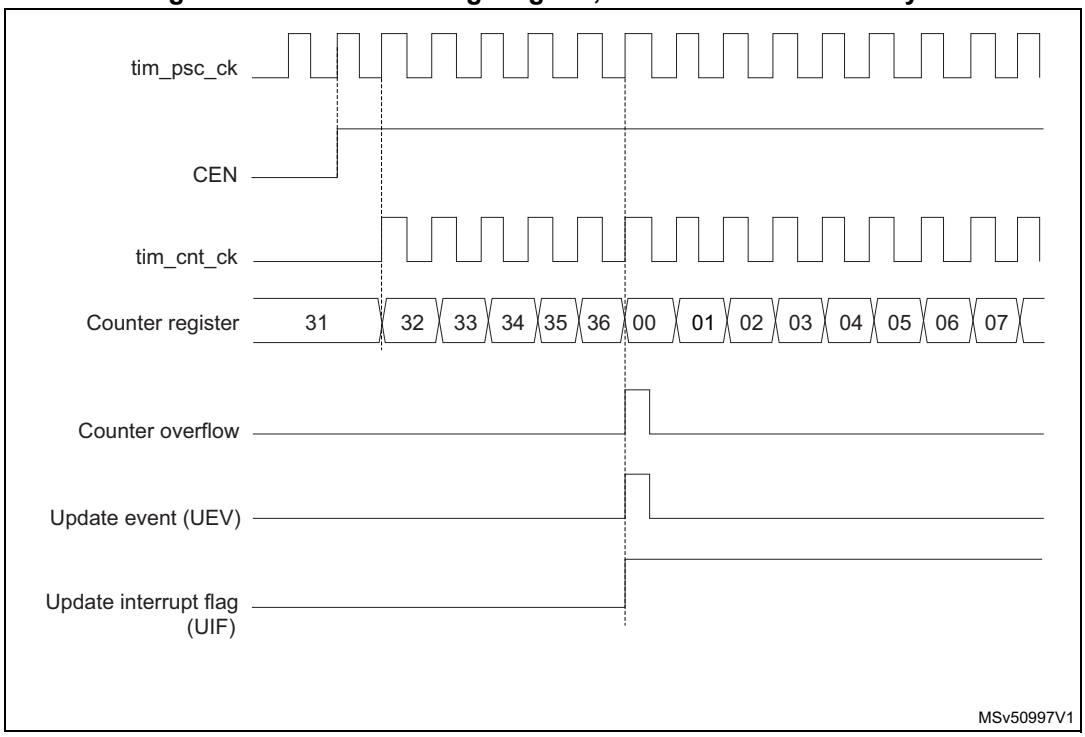


Figure 222. Counter timing diagram, internal clock divided by 2

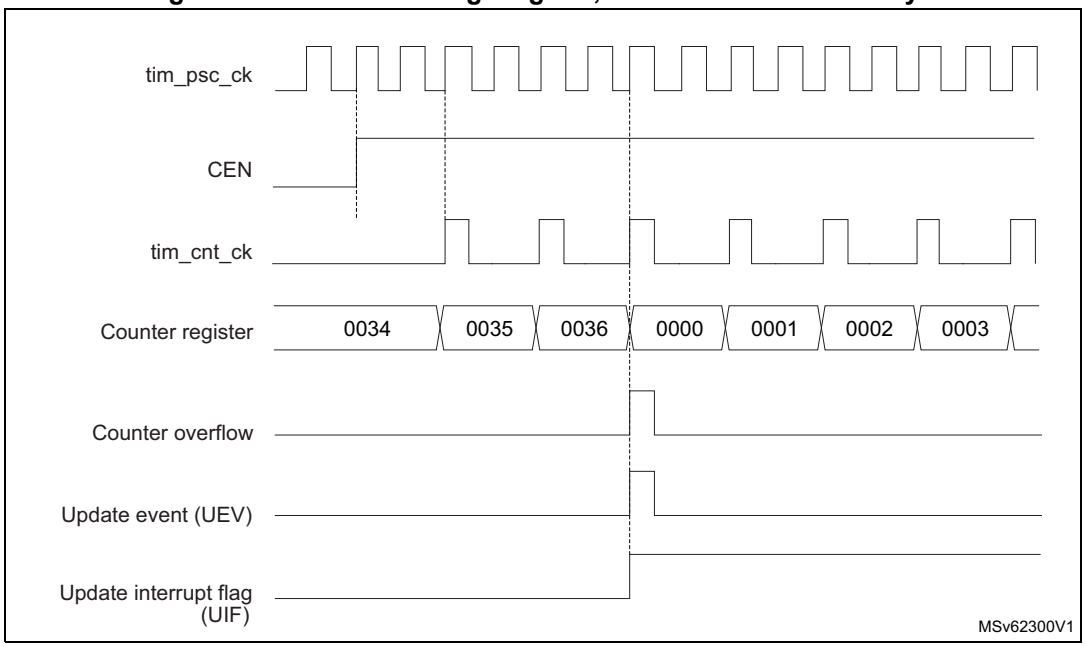


Figure 223. Counter timing diagram, internal clock divided by 4

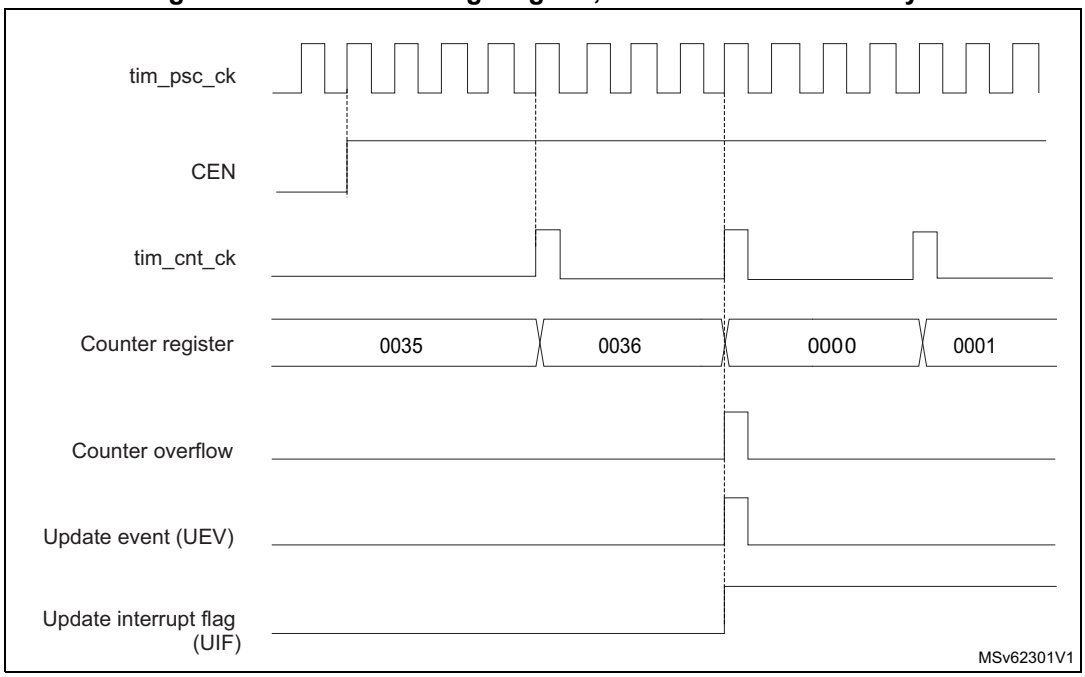


Figure 224. Counter timing diagram, internal clock divided by N

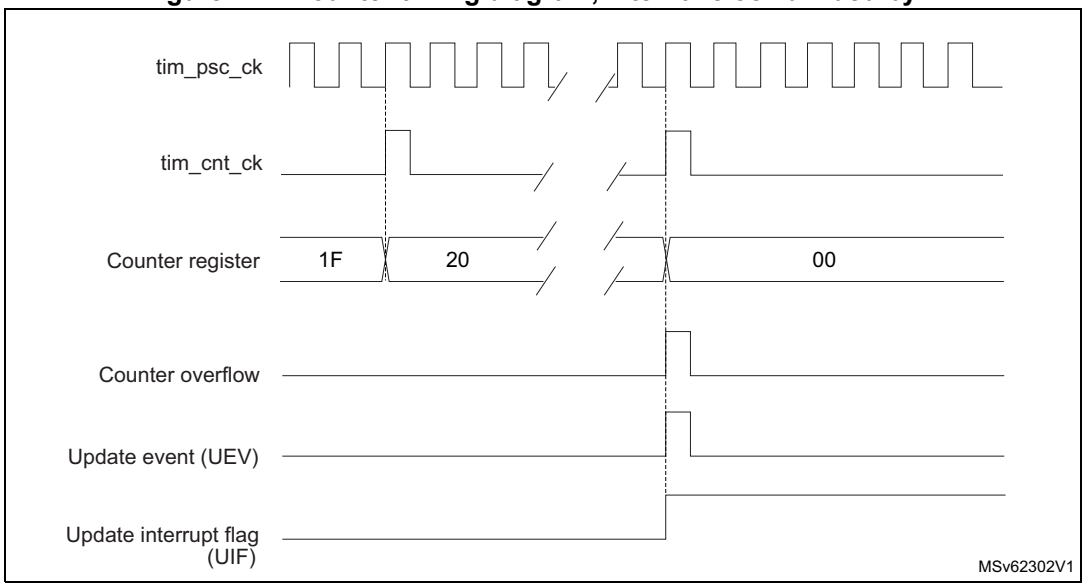


Figure 225. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

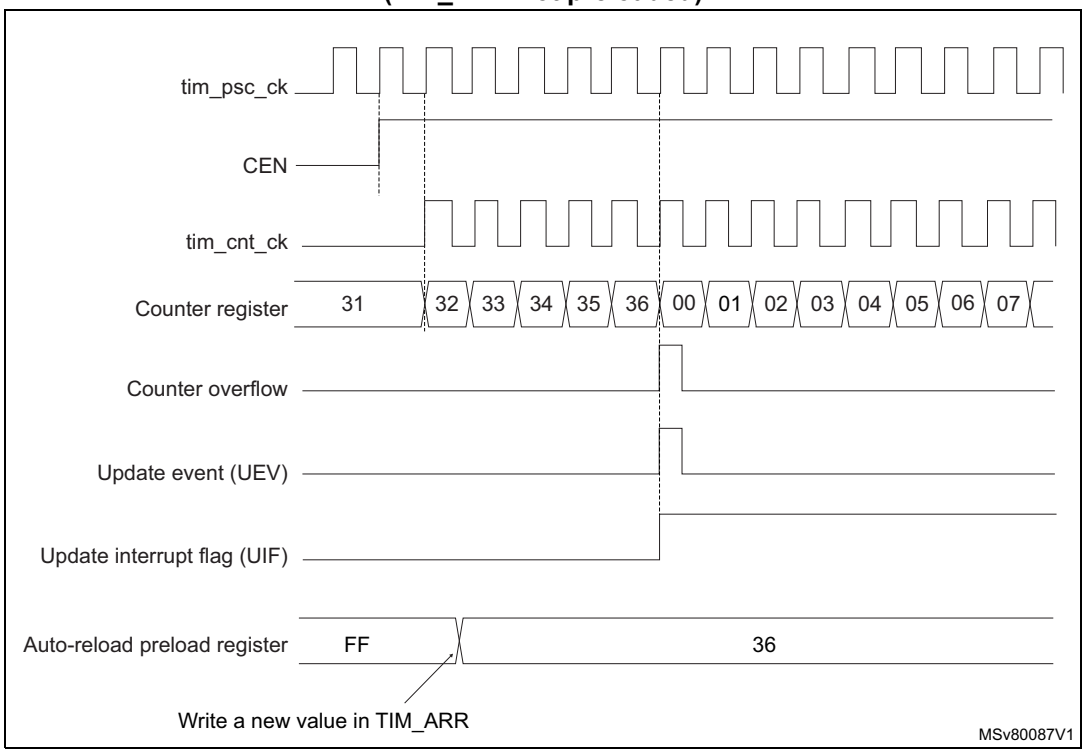
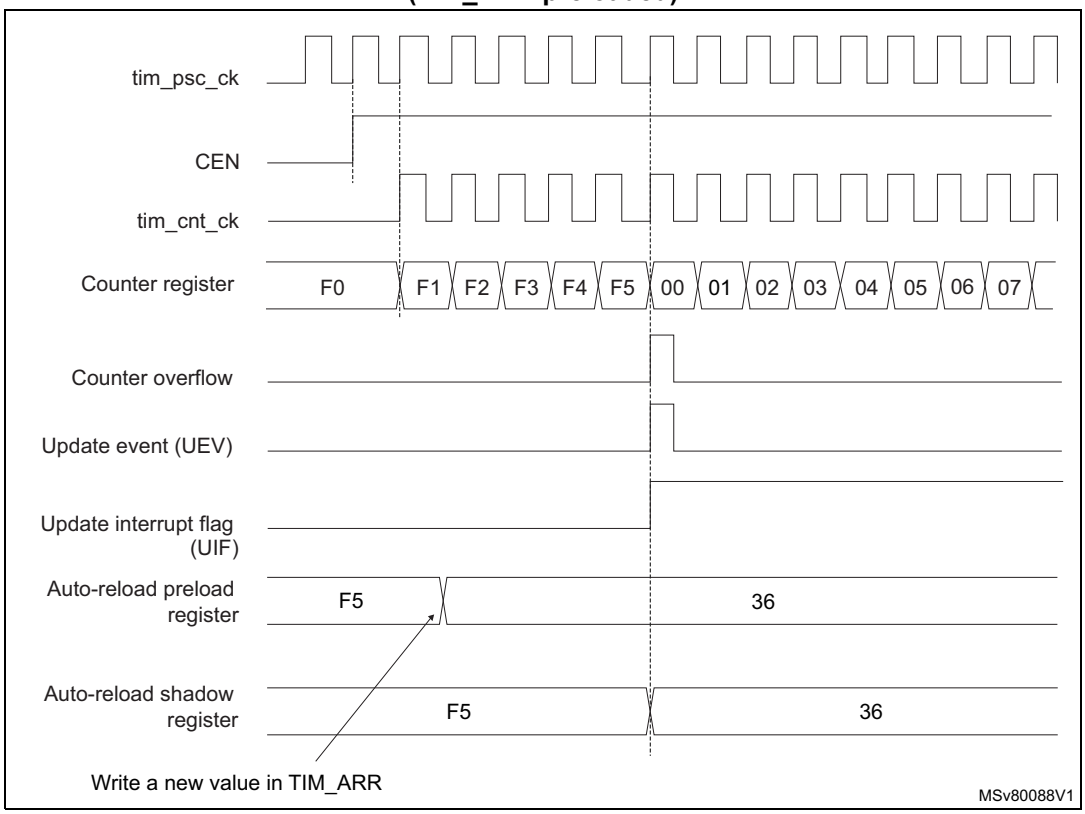


Figure 226. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 227. Counter timing diagram, internal clock divided by 1

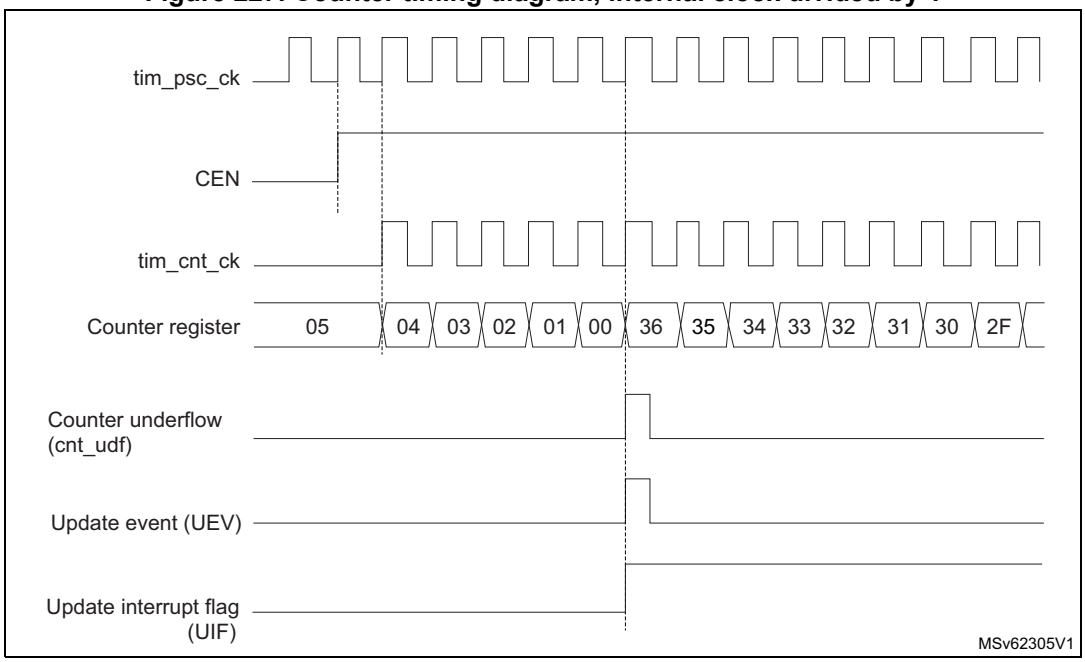


Figure 228. Counter timing diagram, internal clock divided by 2

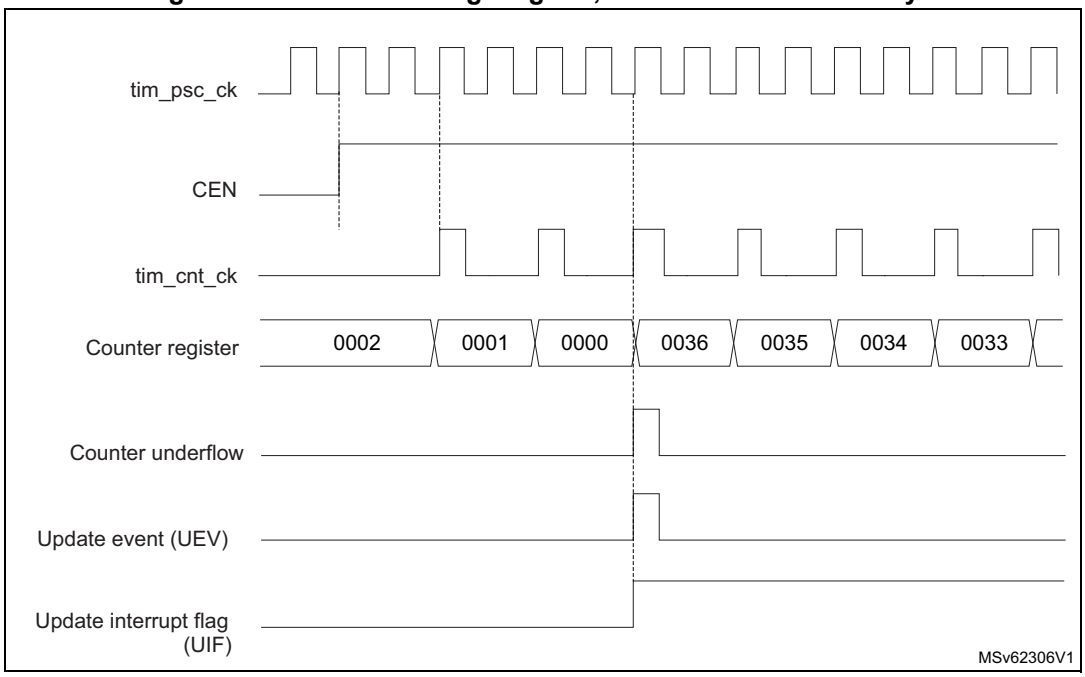


Figure 229. Counter timing diagram, internal clock divided by 4

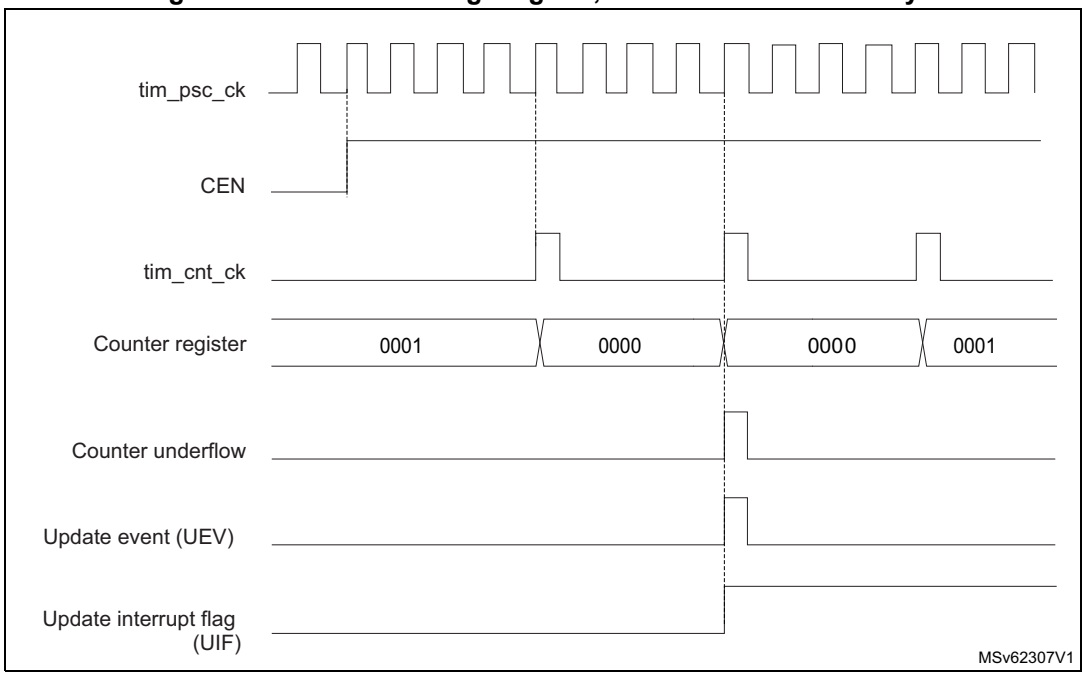


Figure 230. Counter timing diagram, internal clock divided by N

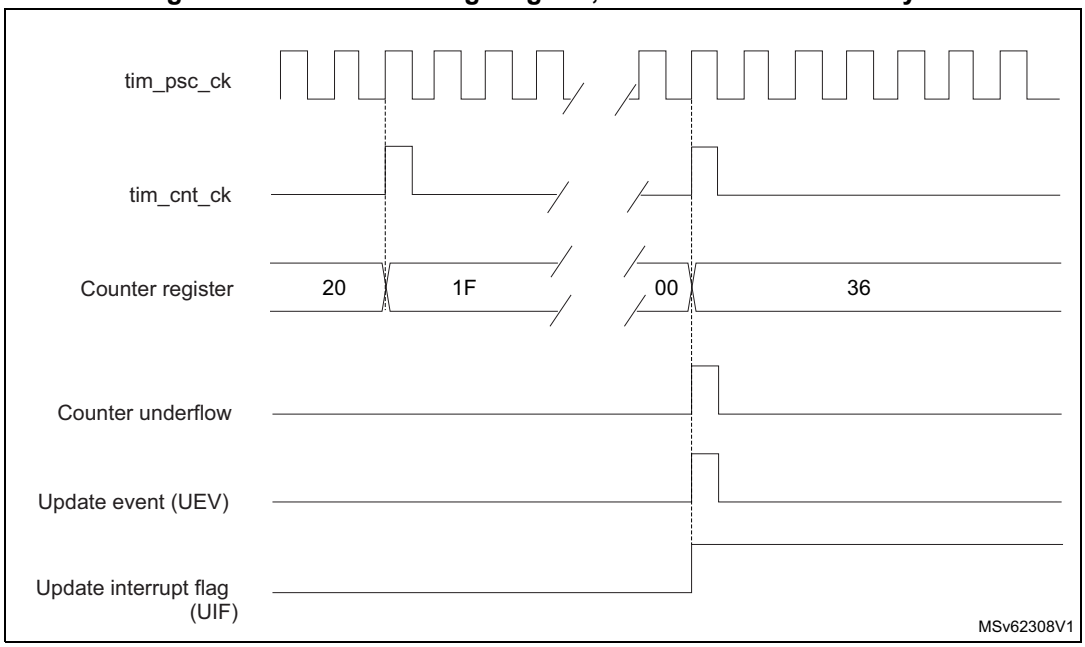
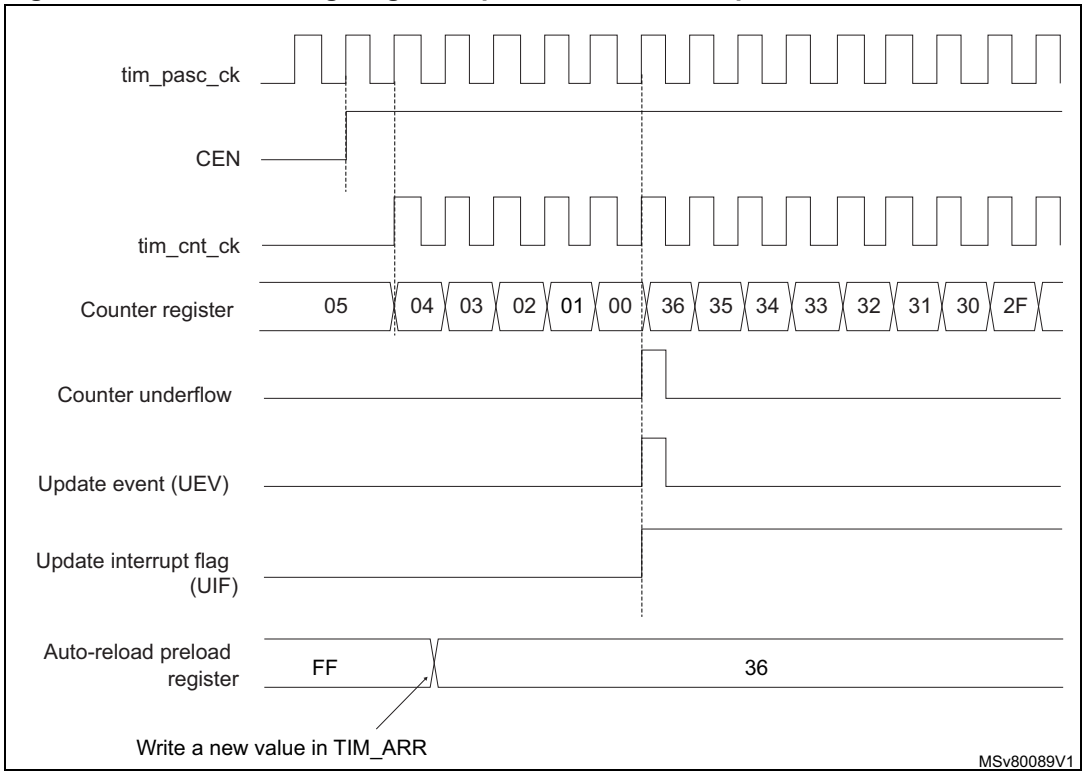


Figure 231. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIM_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

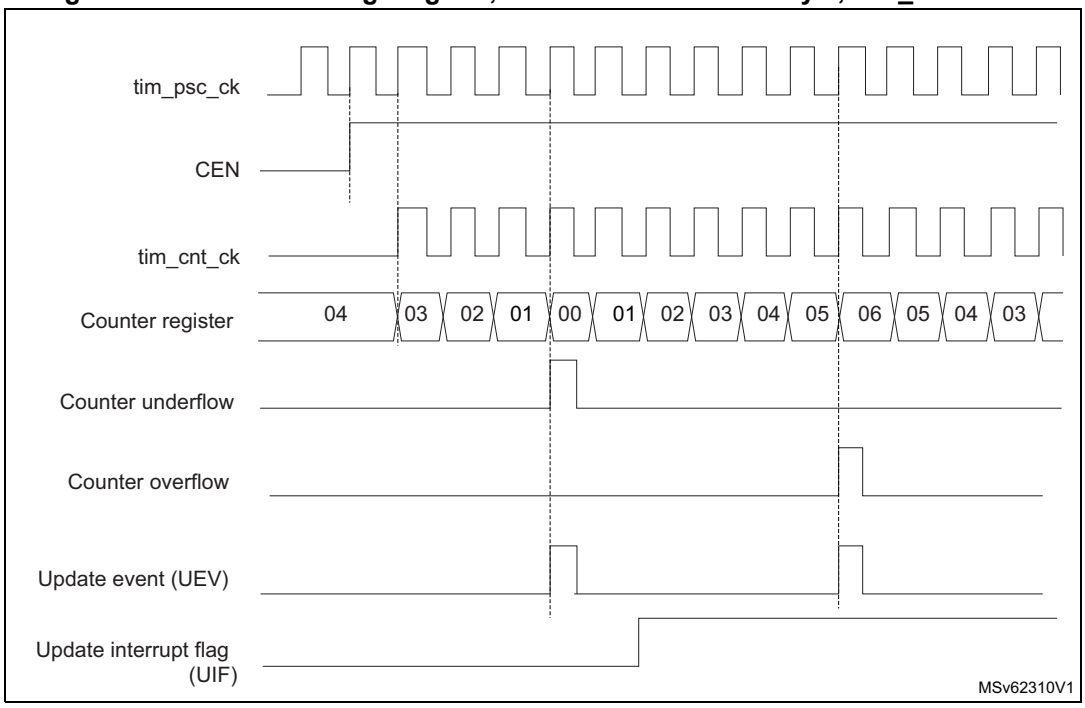
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register)
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 232. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 31.6: TIM1/TIM8 registers*).

Figure 233. Counter timing diagram, internal clock divided by 2

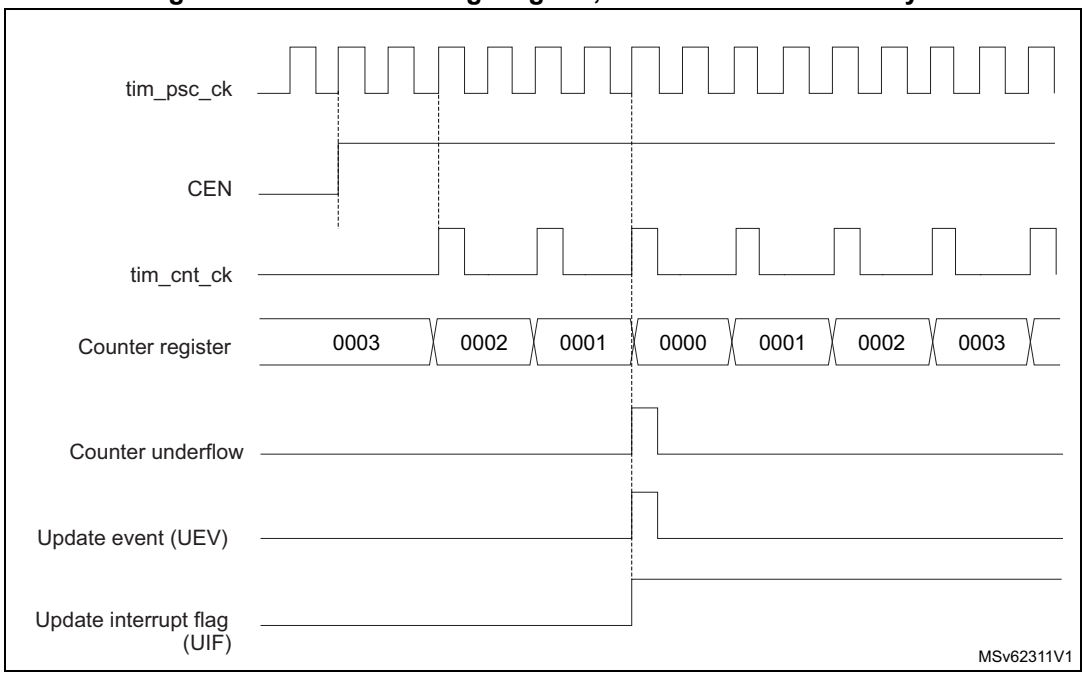


Figure 234. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36

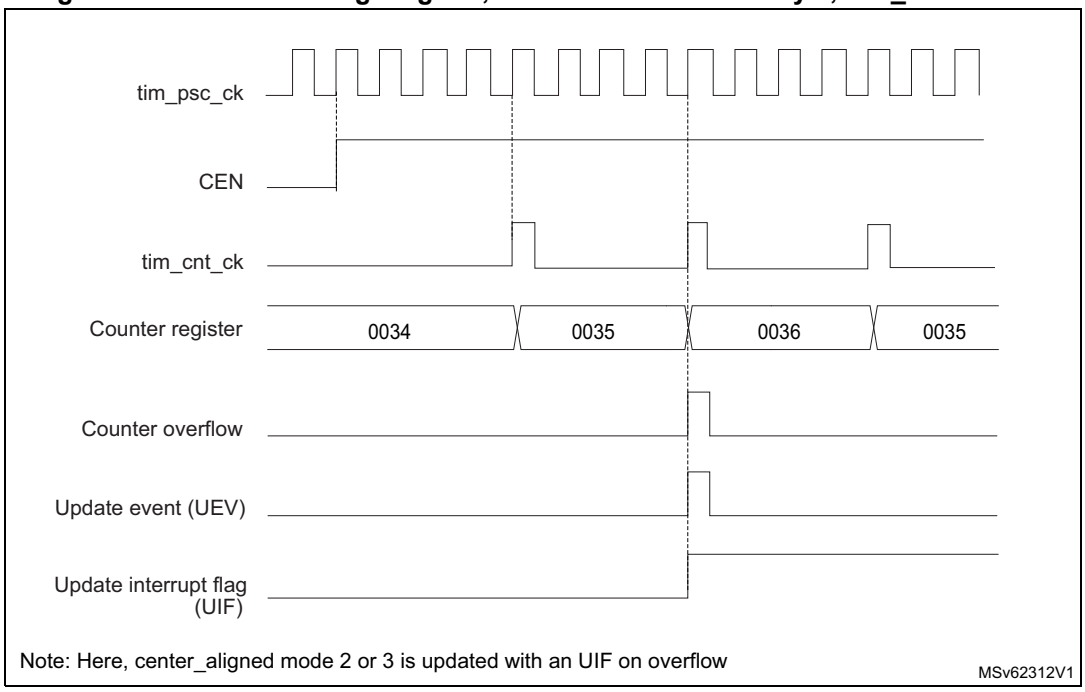


Figure 235. Counter timing diagram, internal clock divided by N

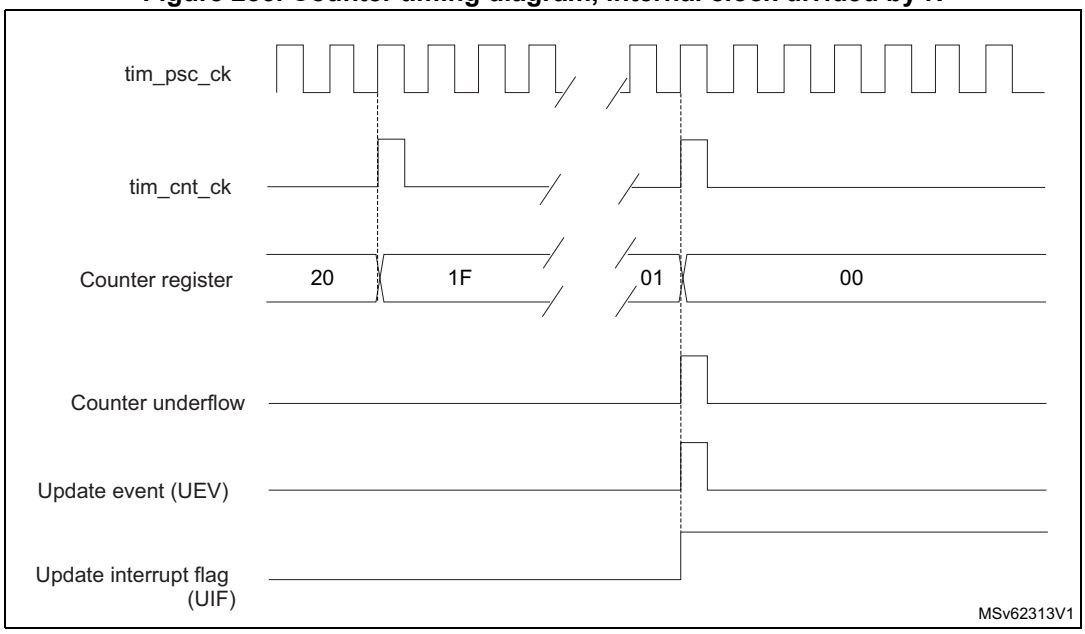


Figure 236. Counter timing diagram, update event with ARPE = 1 (counter underflow)

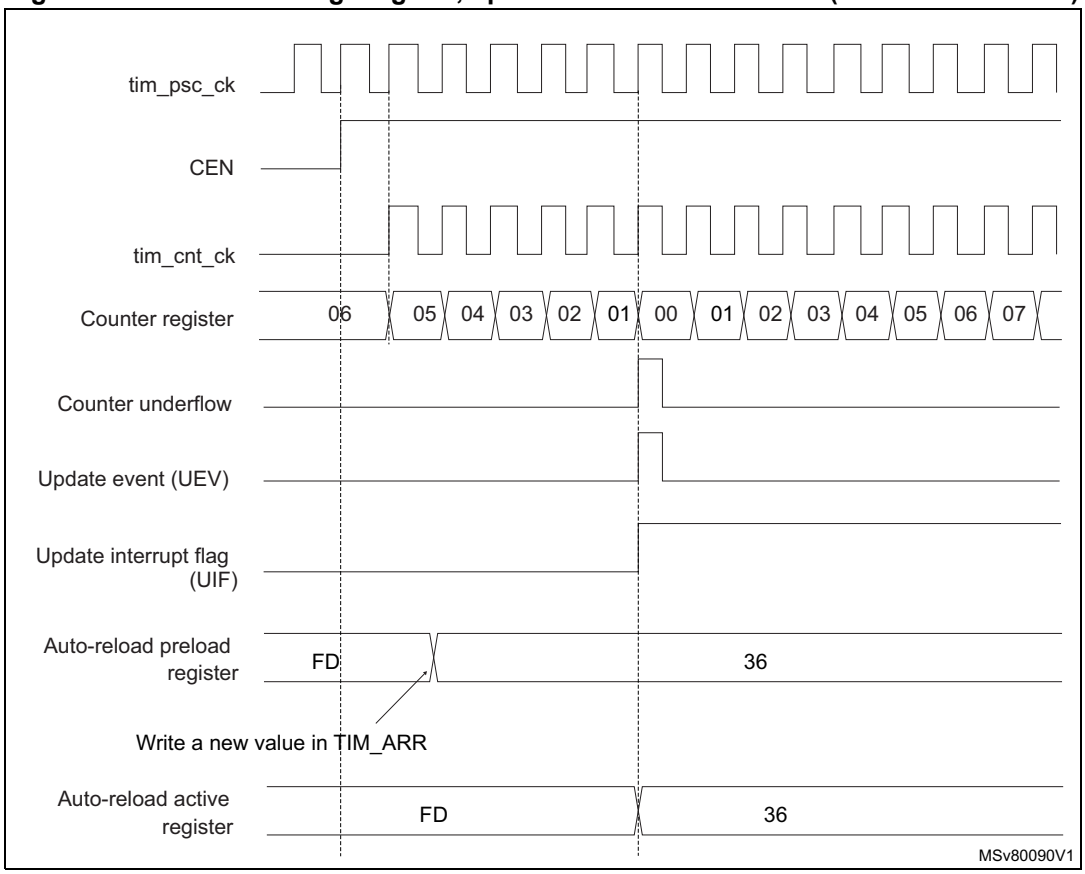
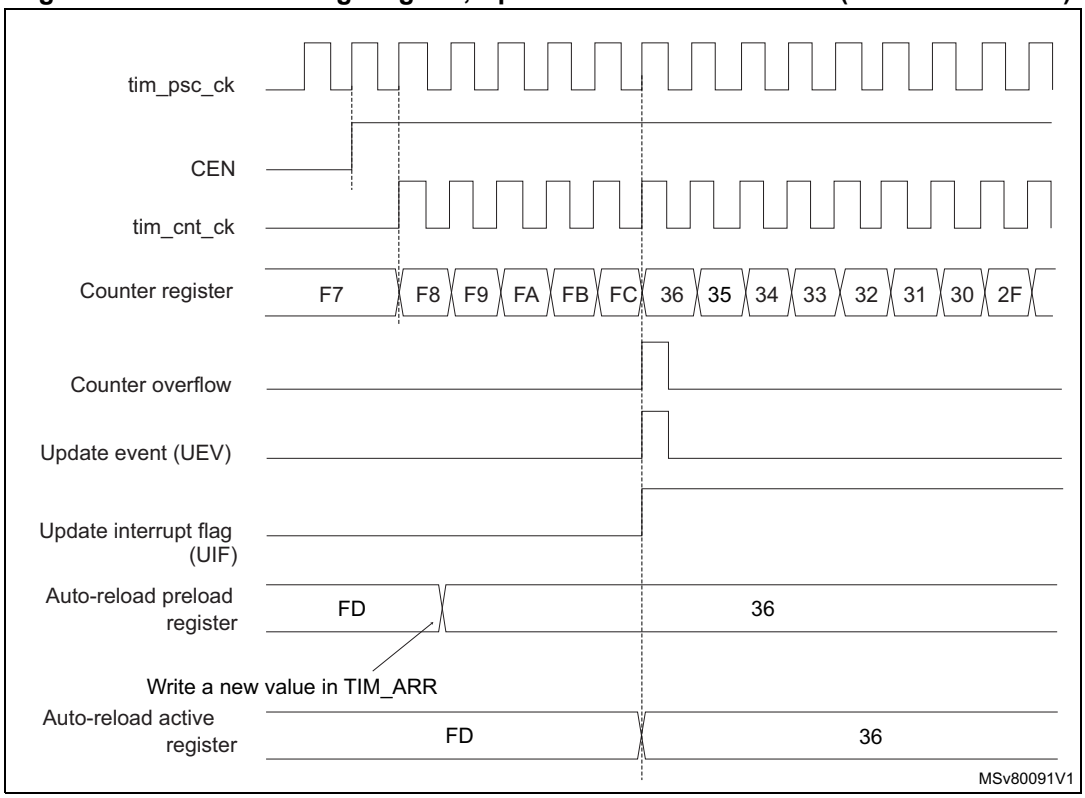


Figure 237. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



31.3.5 Repetition counter

Section 31.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIM_ARR autoreload register, TIM_PSC prescaler register, but also TIM_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIM_RCR repetition counter register.

The repetition counter is decremented:

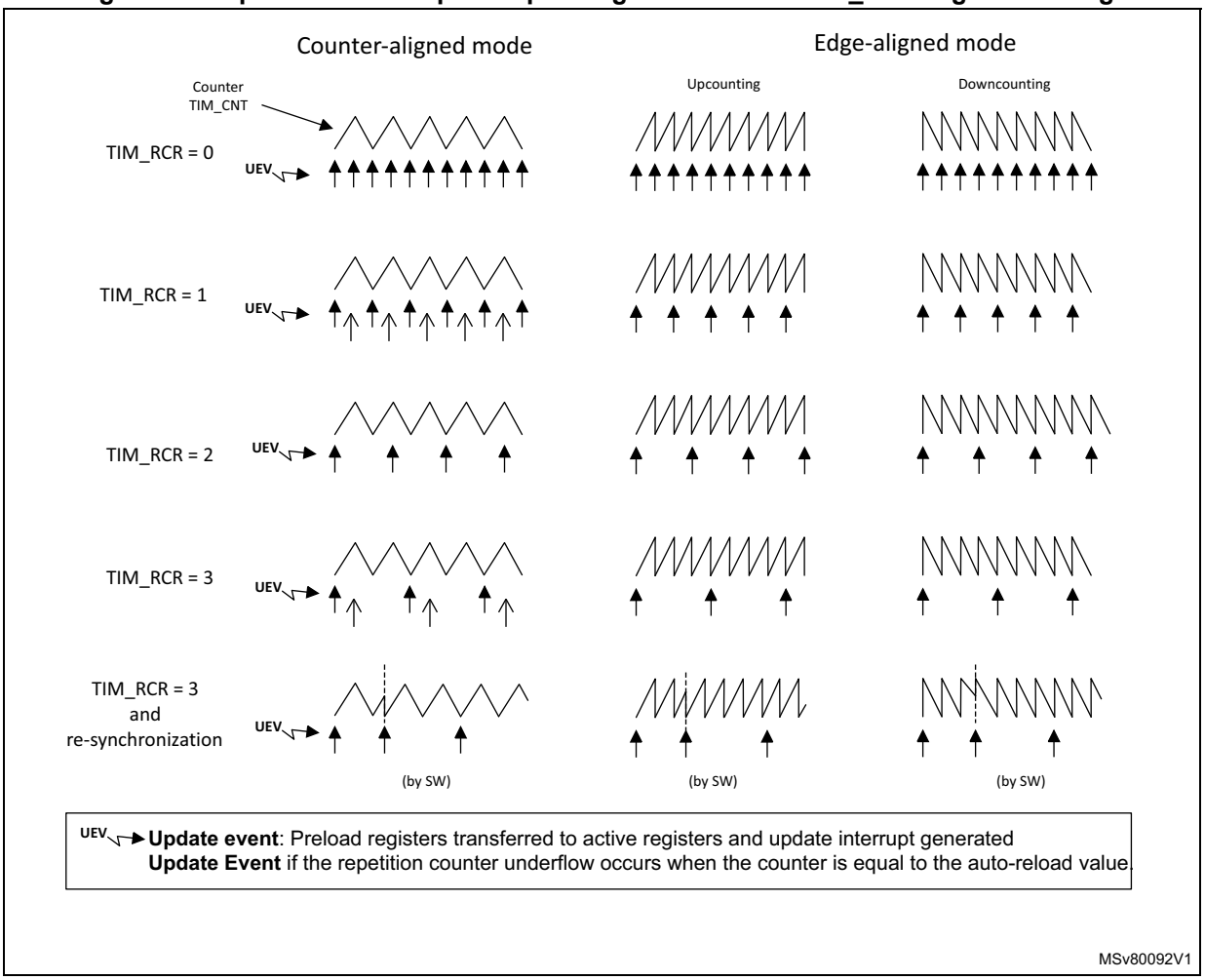
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIM_RCR register value (refer to *Figure 238*). When the update event is generated by software (by setting the UG bit in TIM_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIM_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 238. Update rate examples depending on mode and TIM_RCR register settings



31.3.6 External trigger input

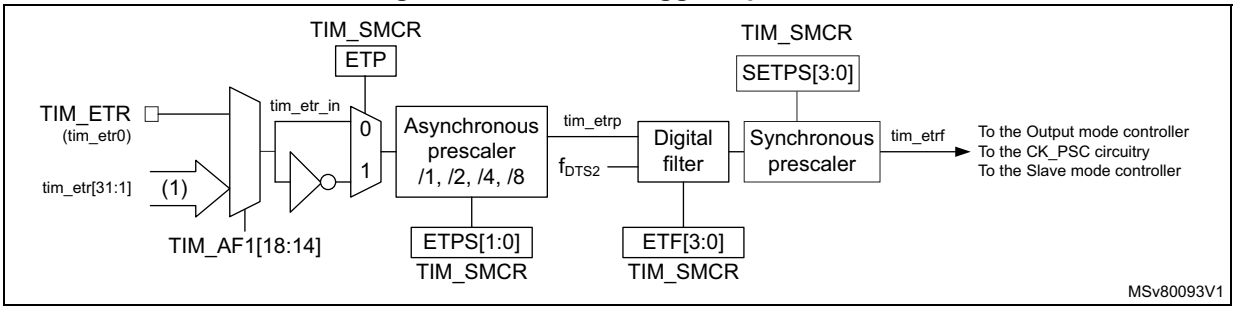
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 31.3.7*)
- trigger for the slave mode (see *Section 31.3.35*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 31.3.9*)

Figure 239 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIM_SMCR register. The trigger frequency can be decreased with the prescalers programmed by the ETPS[1:0] and SETPS[3:0] bitfields and digitally filtered as per the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an

external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 239. External trigger input block



The external trigger input block features 2 prescalers with different functionalities:

- An asynchronous prescaler (division by 1, 2, 4 or 8), able to process tim_etr_in signals with frequencies above tim_ker_ck
- A linear synchronous prescaler (division by 1,2,3,..15), only able to operate with tim_etrp signal frequencies below tim_ker_ck/3

Note: When tim_etrf is used as PWM reset input for cycle-by-cycle current regulation (see Section 31.3.23), SETPS[3:0] must be set to 0 (no prescaling).

Note: The SETPS[3:0] bitfield is always preloaded: a new value will become active only once an update event has occured.

The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[4:0] bitfield in the TIM_AF1 register.

Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for the list of sources connected to the etr_in input in the product.

31.3.7 Clock selection

The counter clock can be provided by the following clock sources:

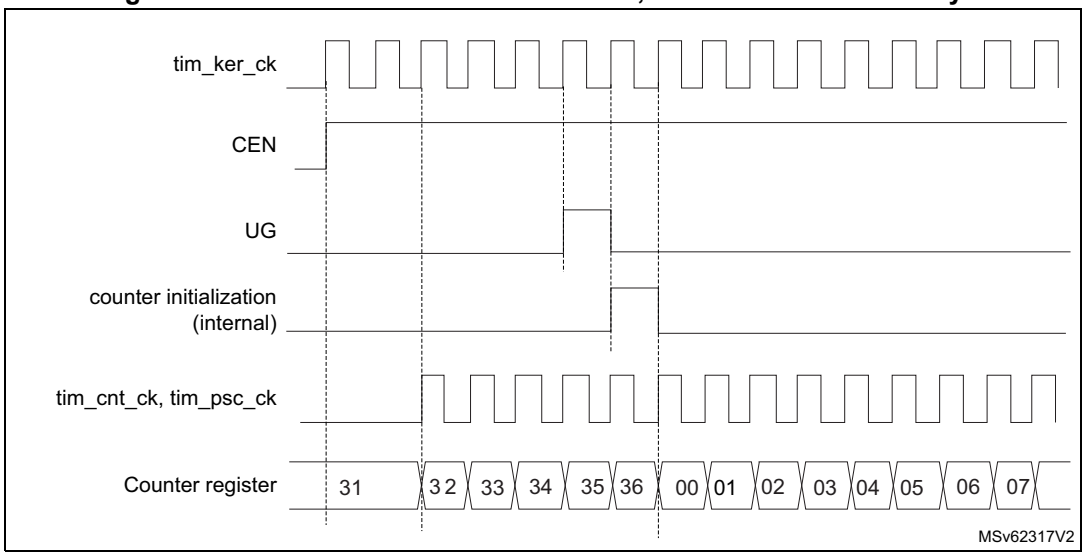
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 240 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

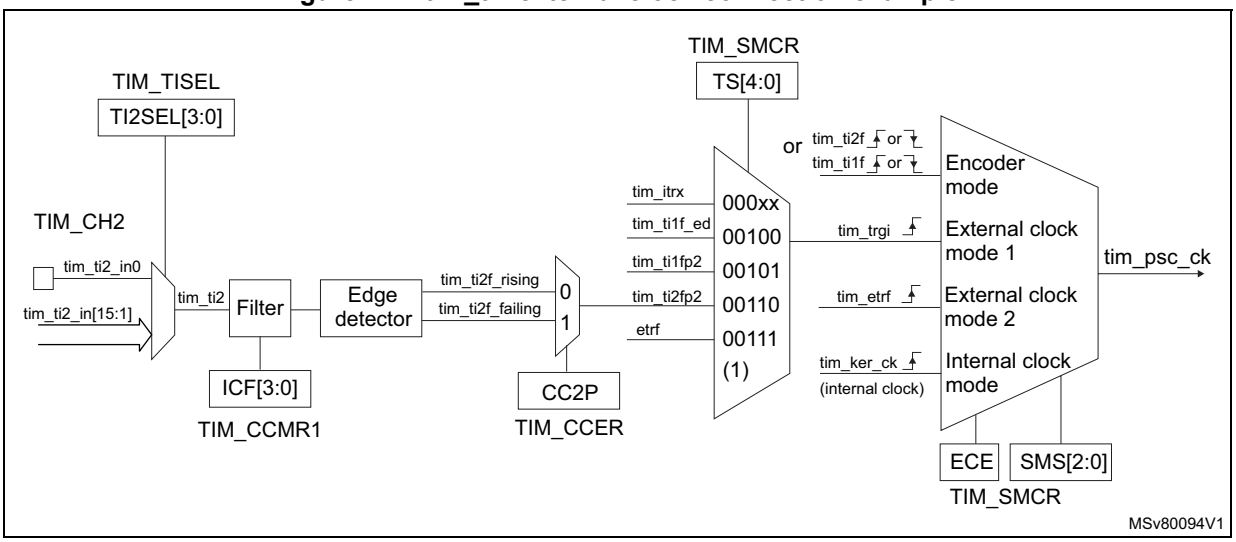
Figure 240. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 00111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 241. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

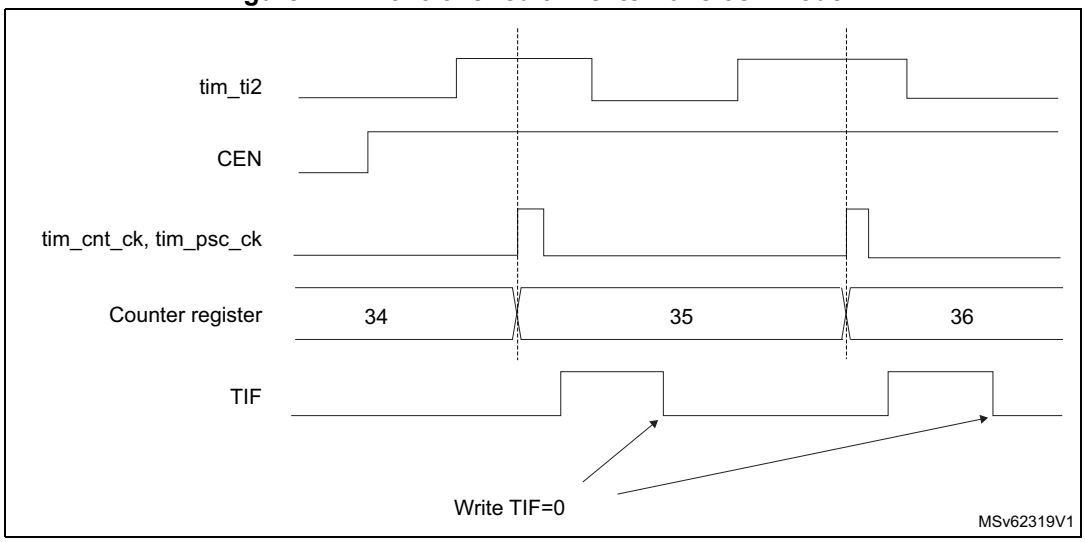
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIM_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 242. Control circuit in external clock mode 1



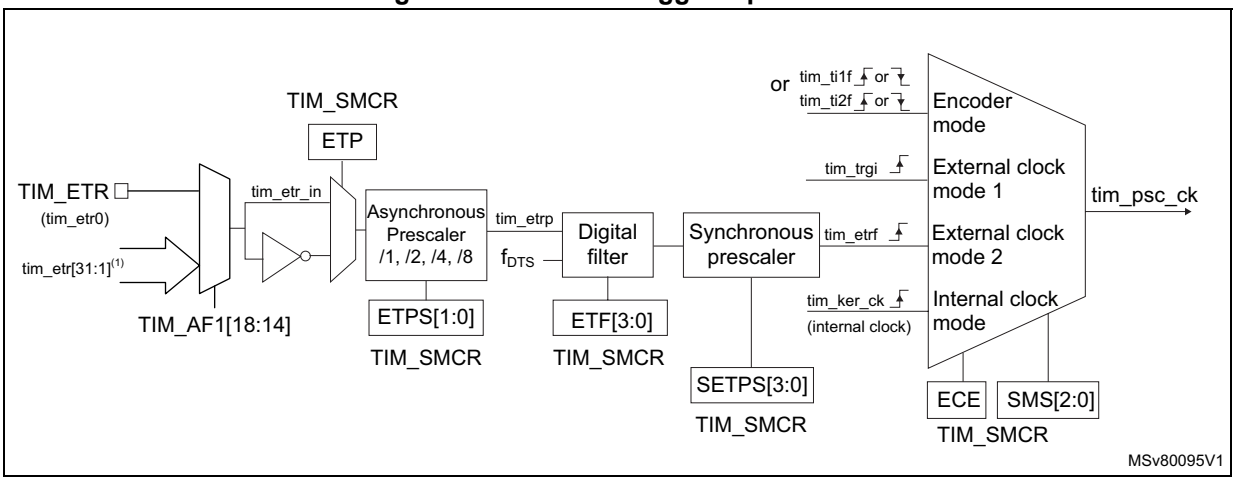
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIM_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 243* gives an overview of the external trigger input block.

Figure 243. External trigger input block



1. Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals*.

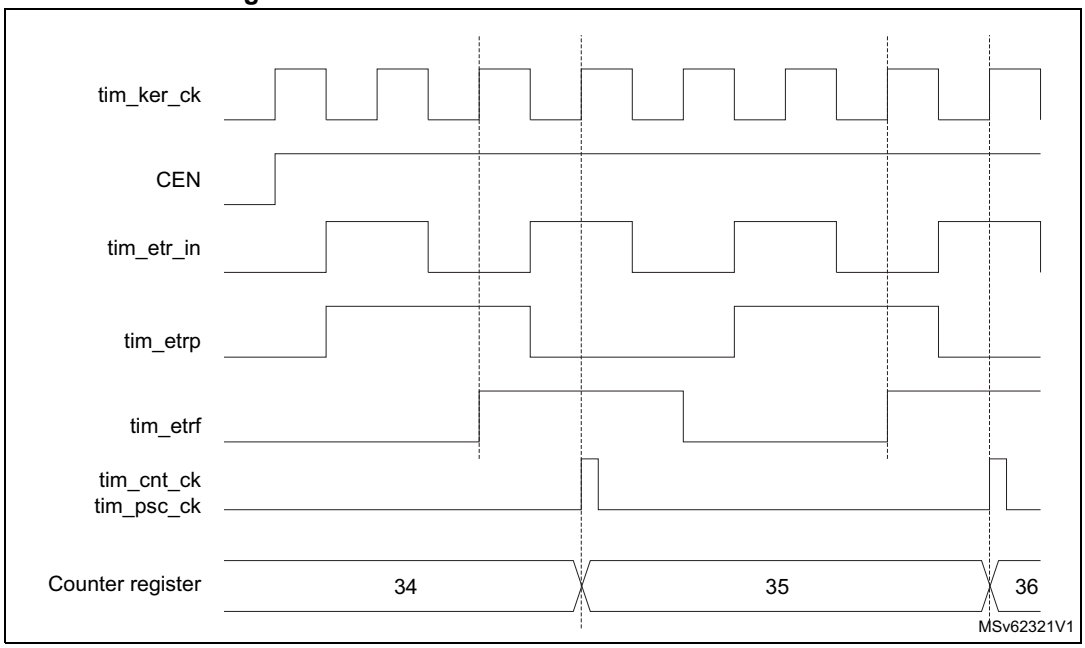
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIM_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 and SETPS[3:0] = 0 in the TIM_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIM_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIM_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 244. Control circuit in external clock mode 2



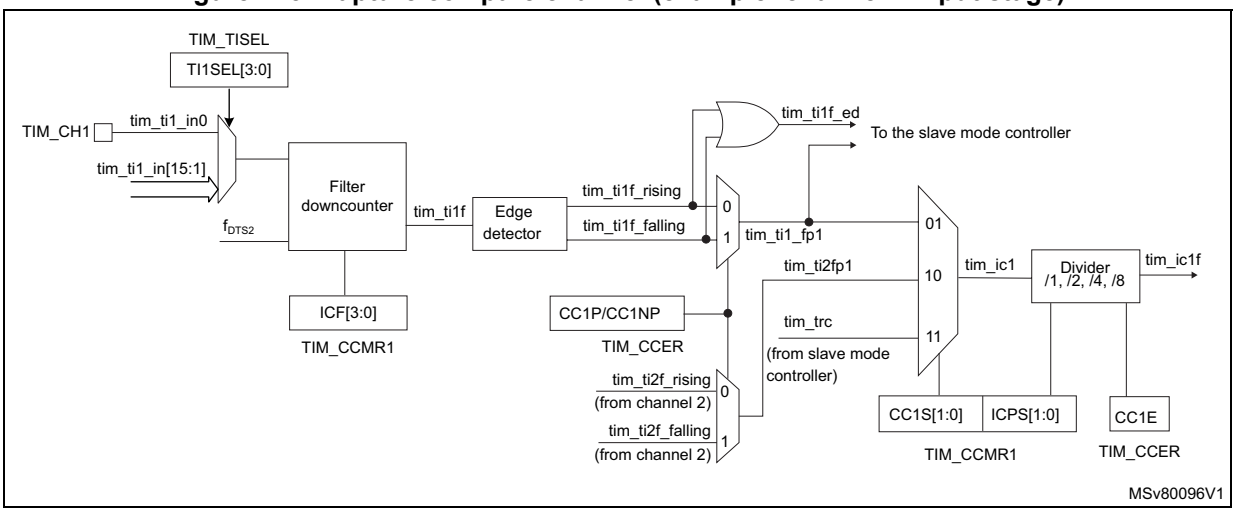
31.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 245 to *Figure 248* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 245. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 246. Capture/compare channel 1 main circuit

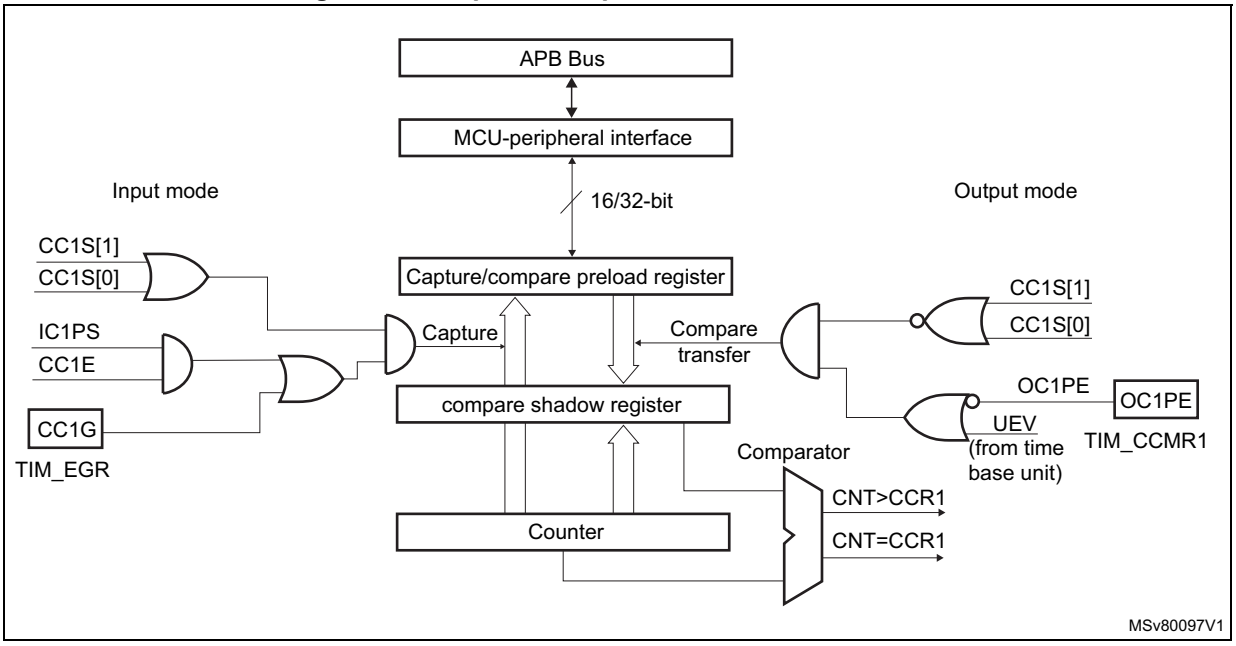
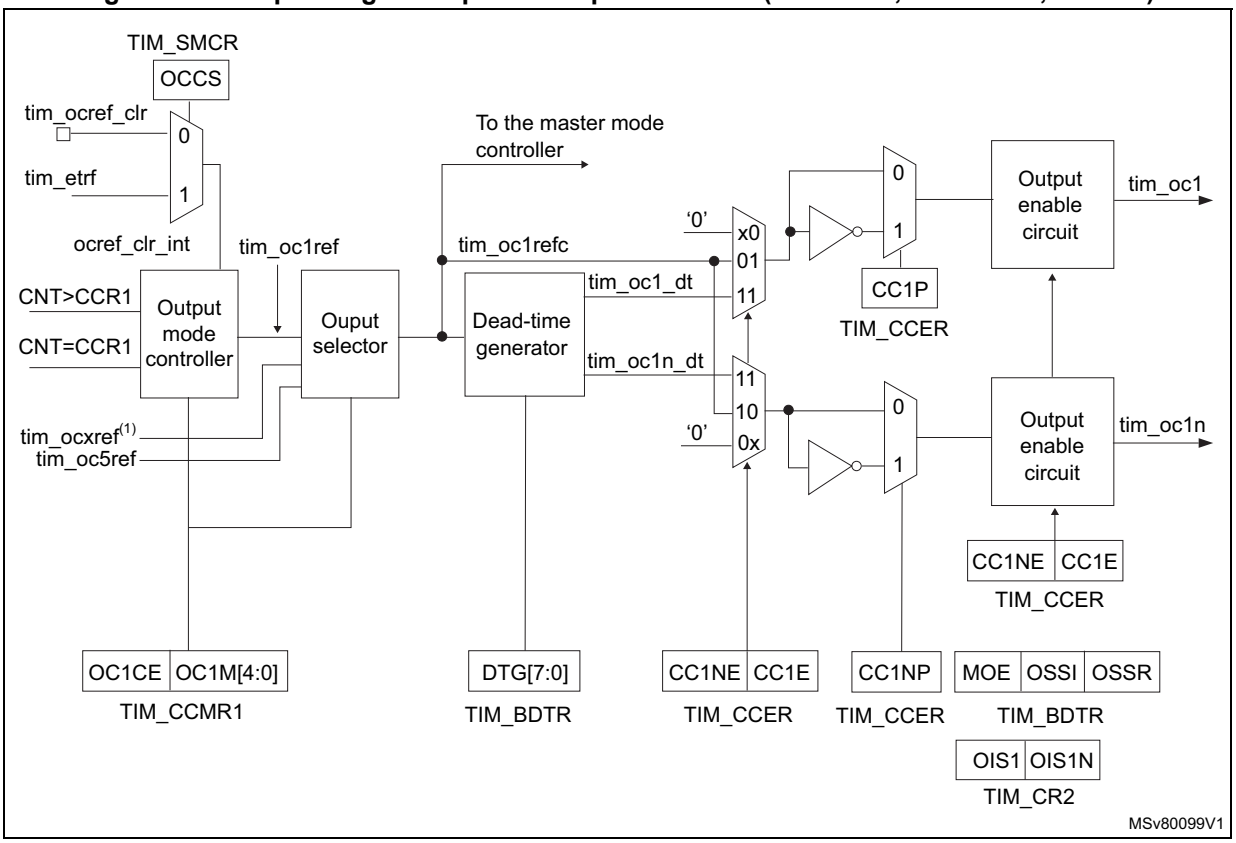
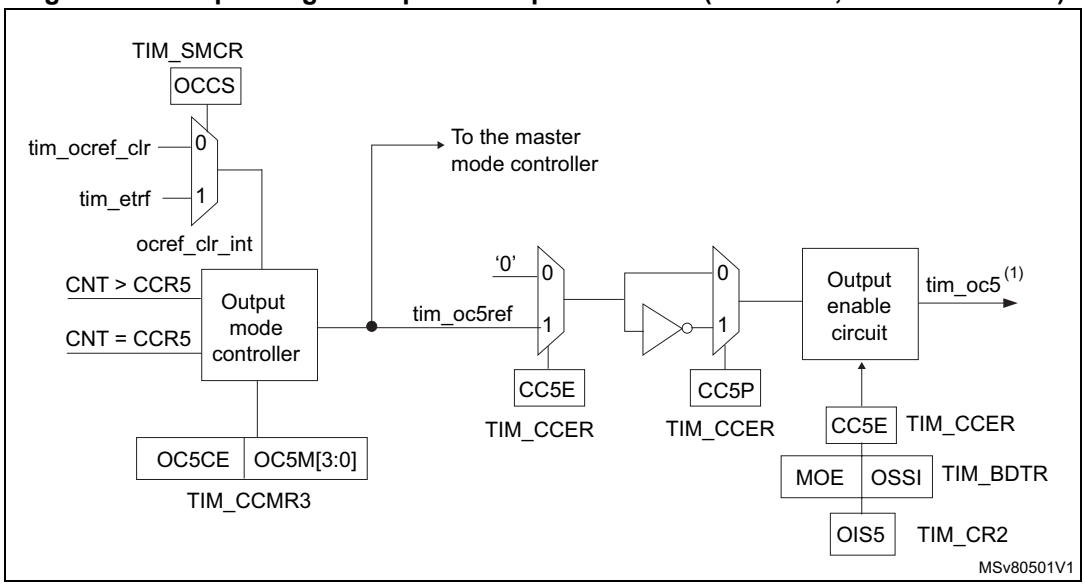


Figure 247. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 248. Output stage of capture/compare channel (channel 5, idem ch. 6 and 7)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

31.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIM_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS2} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIM_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

31.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

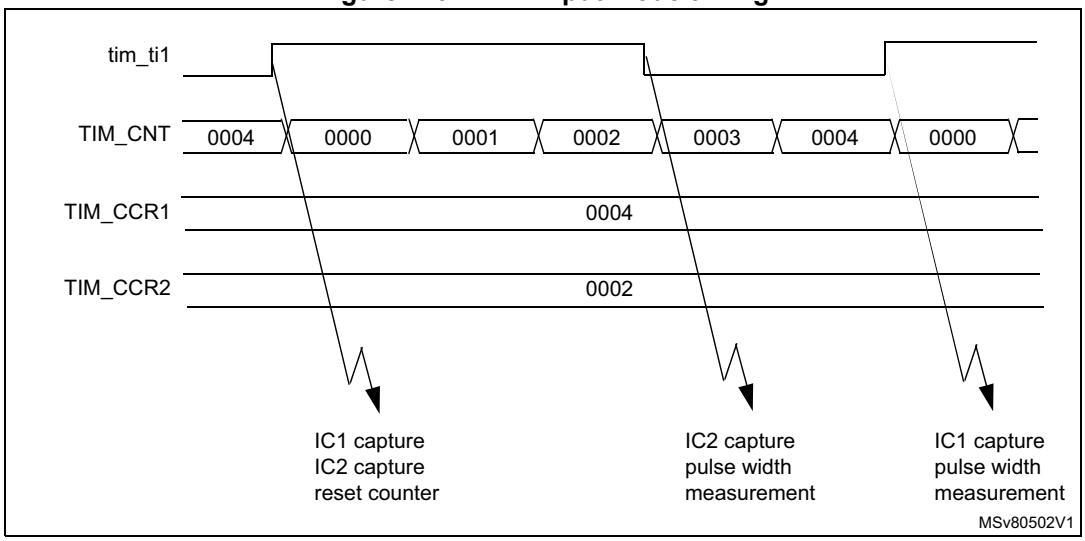
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIM_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 249. PWM input mode timing



31.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

31.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 to 7 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 00000), be set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

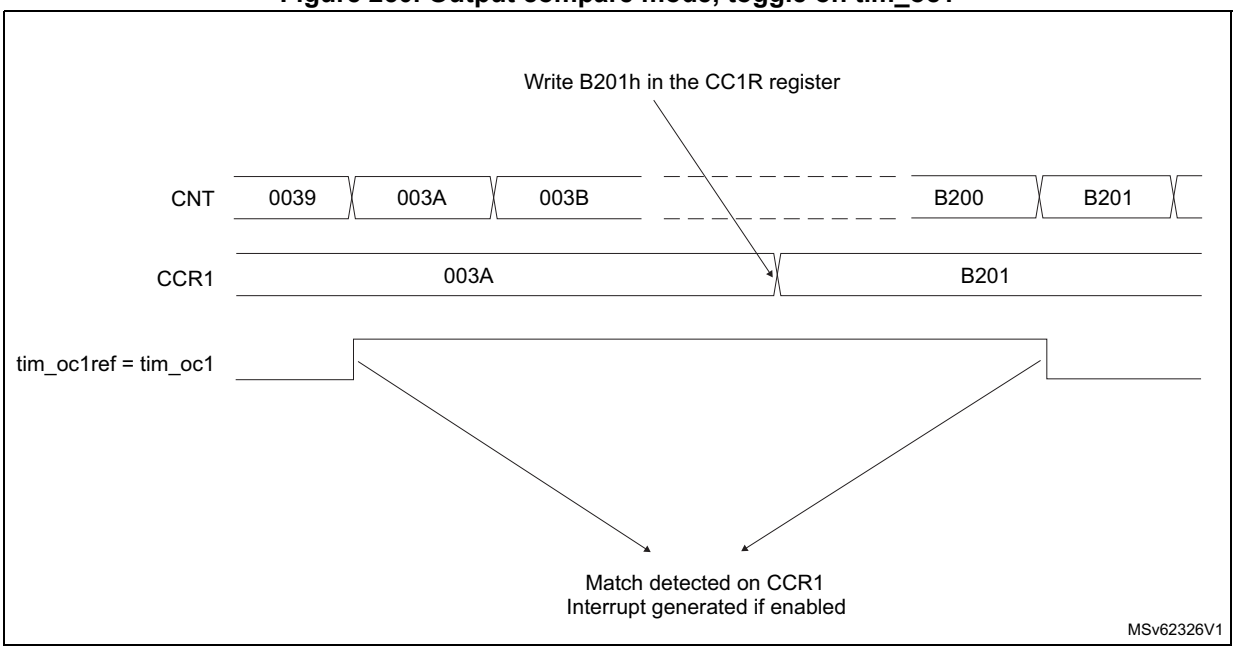
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 250*.

Figure 250. Output compare mode, toggle on tim_oc1



31.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIM_CCER and TIM_BDTR registers). Refer to the TIM_CCER register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

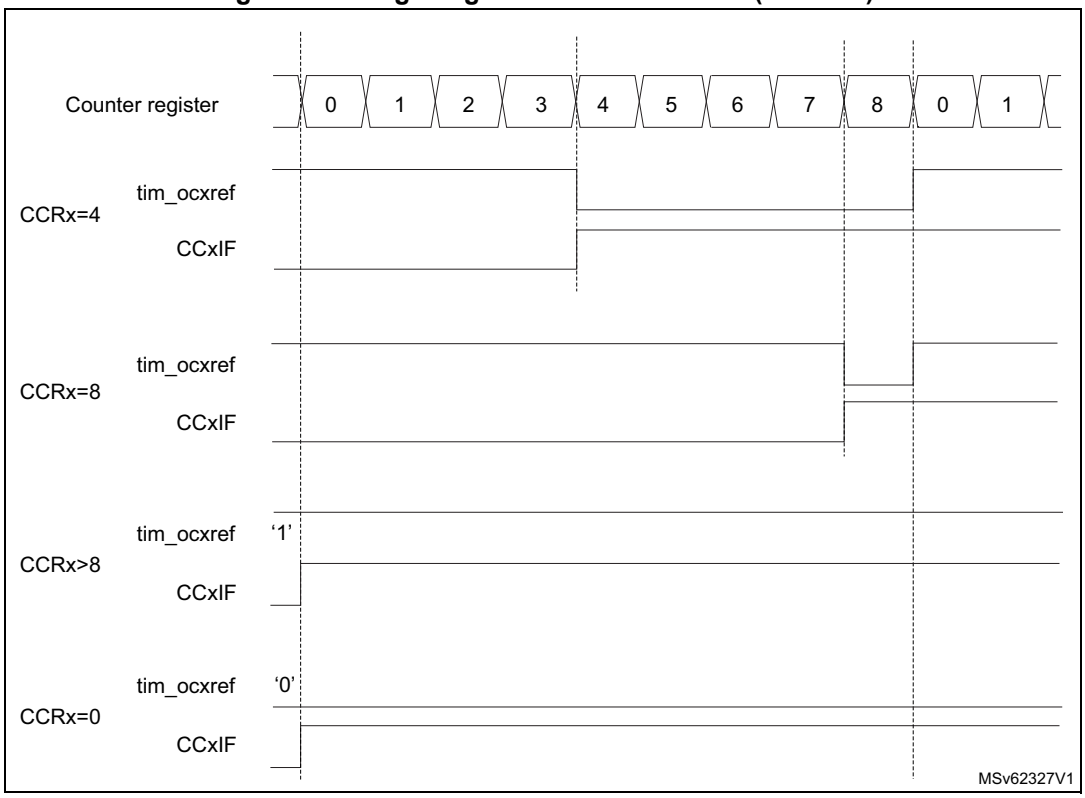
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 251* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 251. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIM_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT > TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

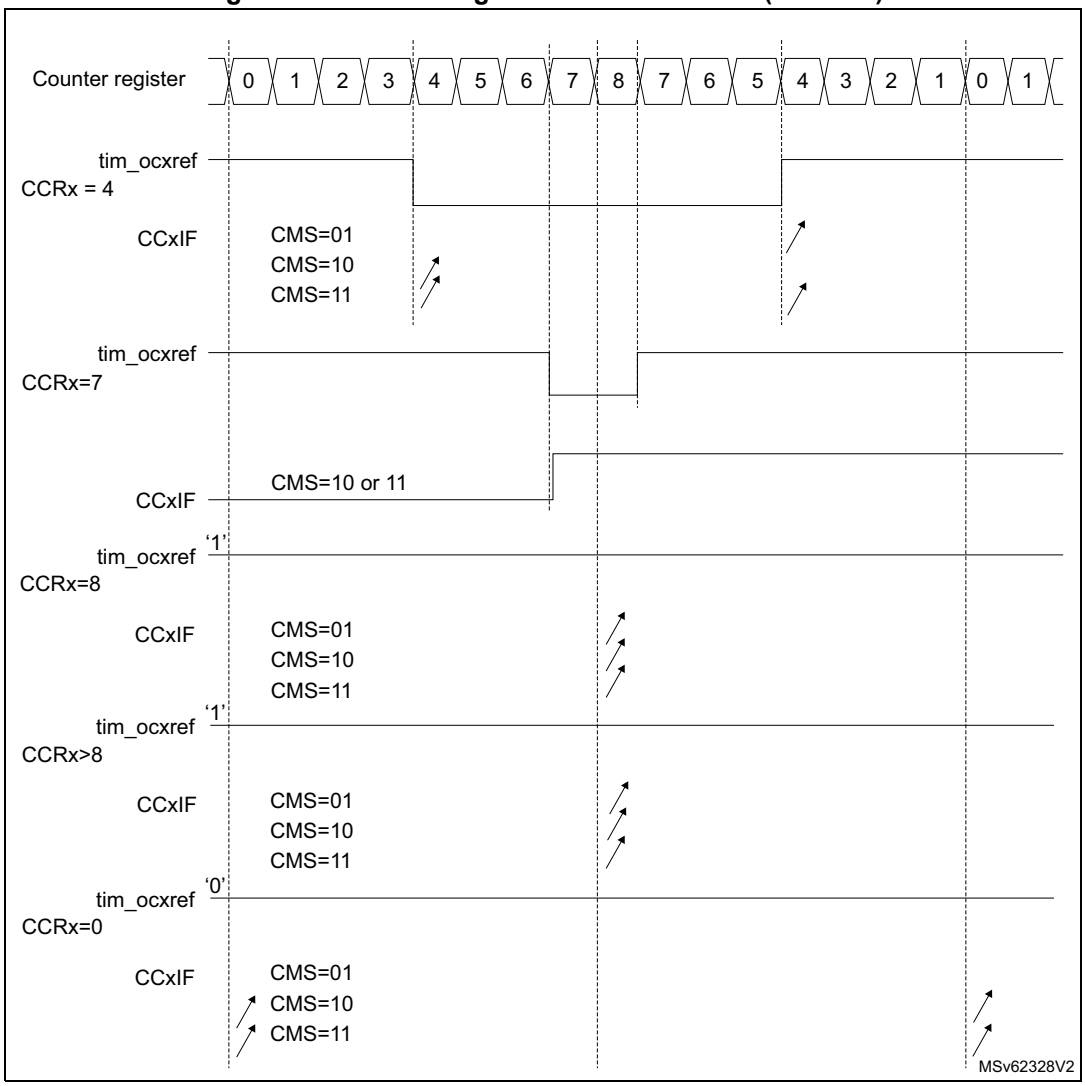
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 252 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 252. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

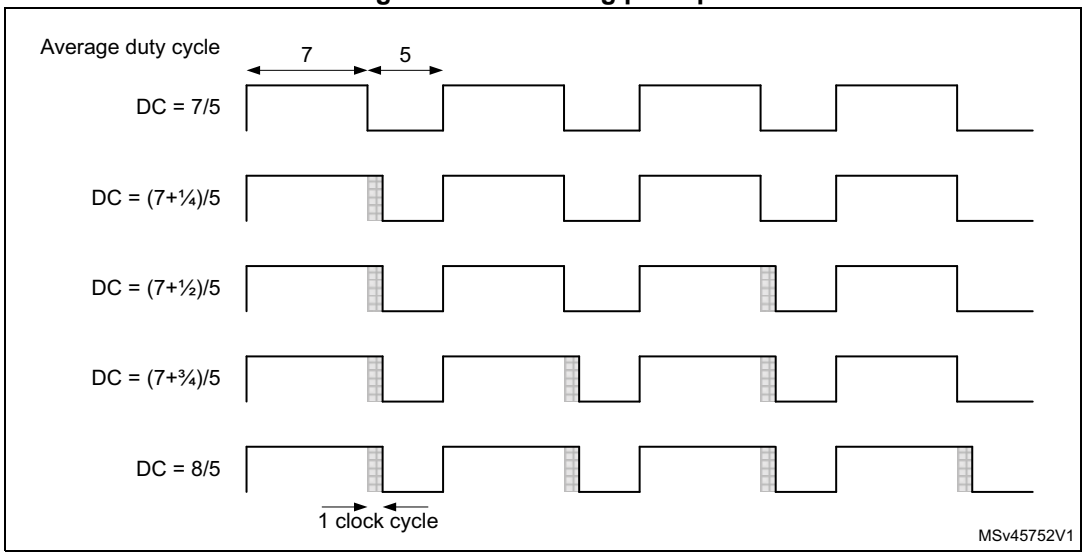
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT > TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 253* presents the dithering principle applied to four consecutive PWM cycles.

Figure 253. Dithering principle



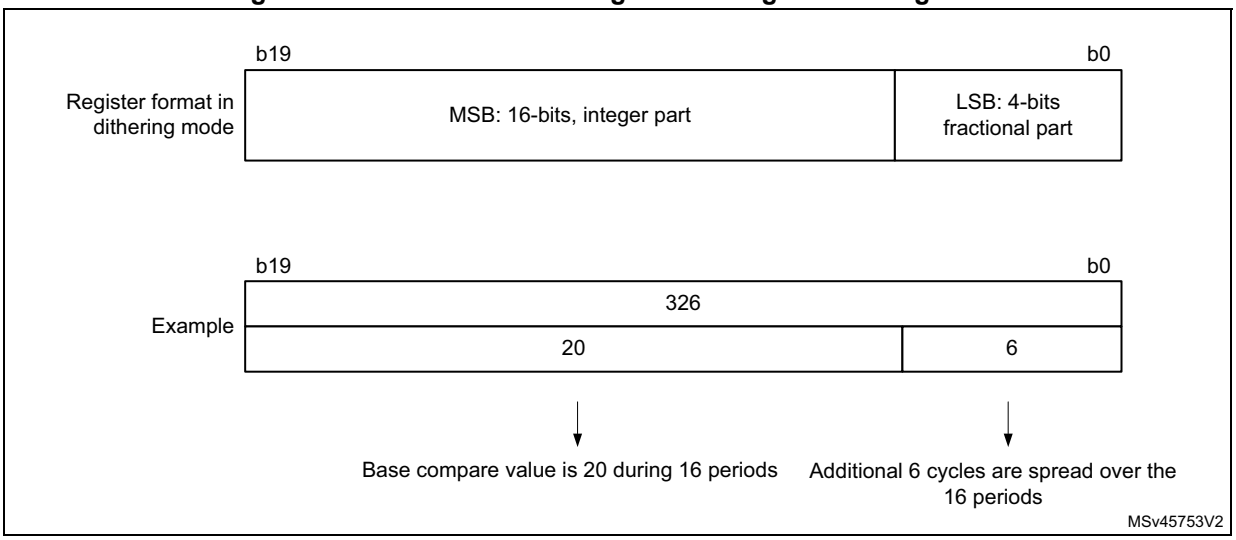
When the dithering mode is enabled, the register coding is changed as follows (see *Figure 254* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 254. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

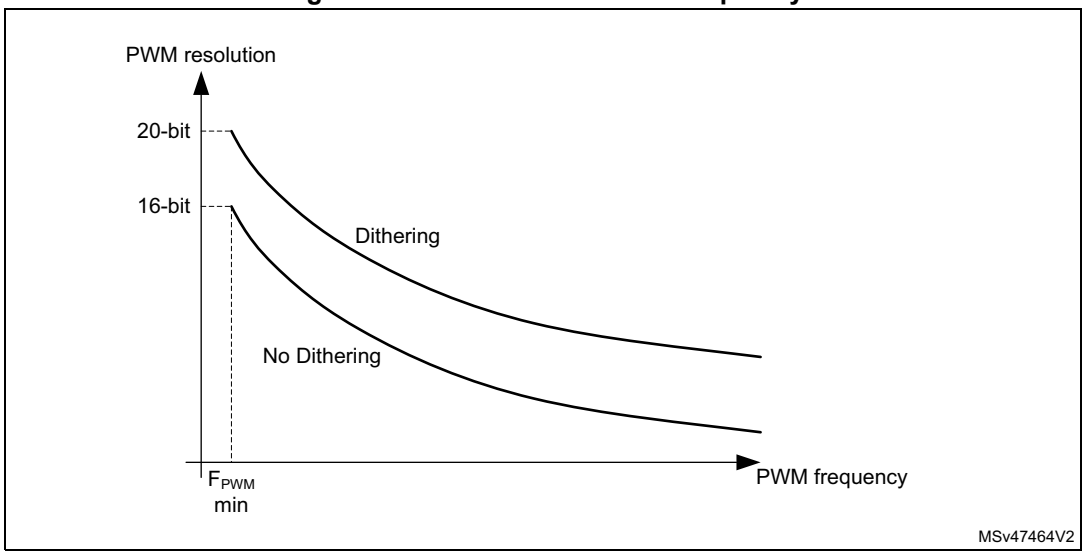
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

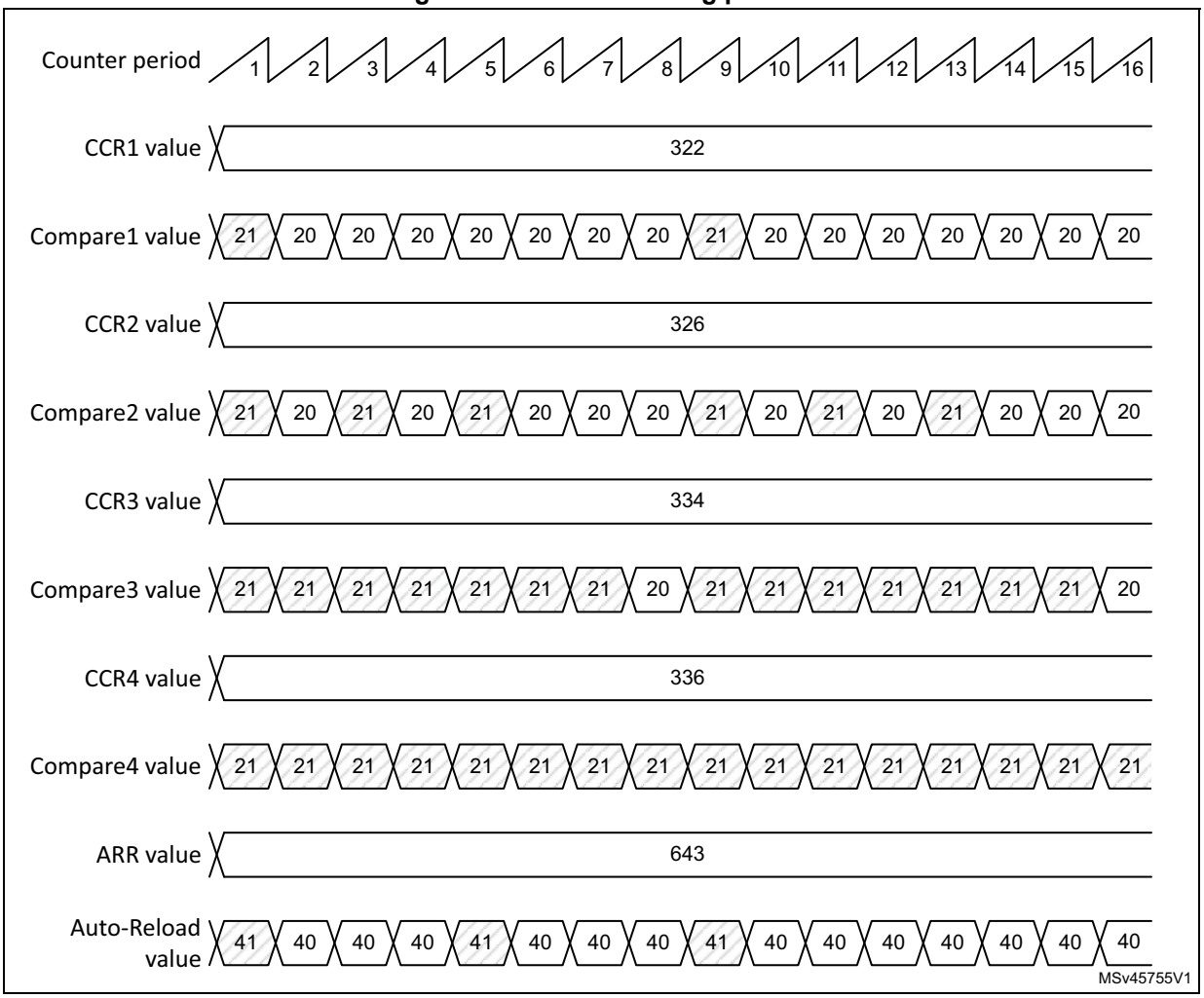
As shown on *Figure 255*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 255. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 256*.

Figure 256. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 293*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 293. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 293. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 257*.

Figure 257. Dithering effect on duty cycle in center-aligned PWM mode

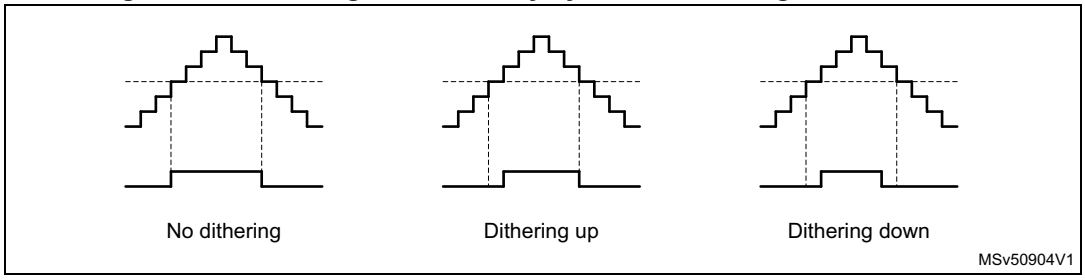


Table 294 shows how the dithering pattern is added in center-aligned PWM mode.

Table 294. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 294. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

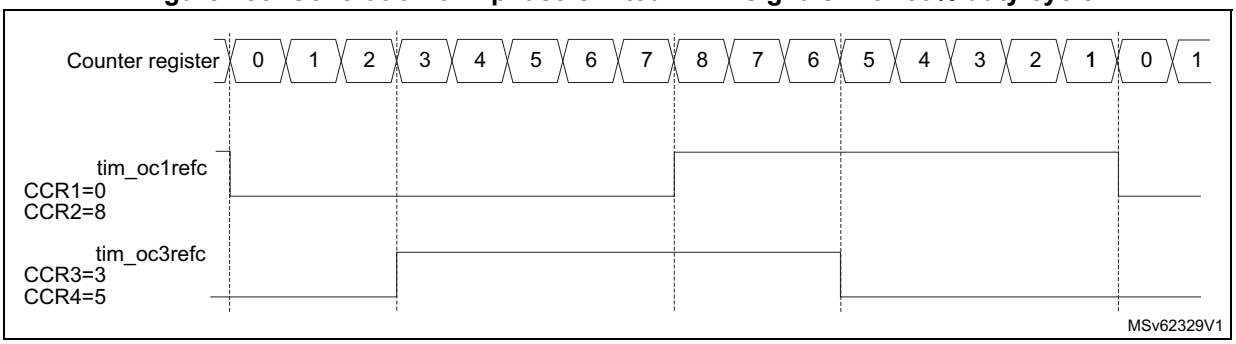
31.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle, for instance:

- tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2
- tim_oc3refc is controlled by TIM_CCR3 and TIM_CCR4
- ...

Figure 258 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 258. Generation of 2 phase-shifted PWM signals with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 295*. The selection is done

with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (e.g. Asymmetric PWM mode 3 and 4), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 to 6, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 00110 for the asymmetric PWM mode 1 on oc1refc.

For the asymmetric PWM modes 7 to 10, it is mandatory to have opposite PWM mode for both channels (one being in PWM mode 1 and the other in PWM mode 2). Refer to the *Table 295* for configuration.

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2 or a tim_oc2refc signal resulting from asymmetric PWM mode 1.

Table 295. Asymmetric PWM modes

<table>
<tr><th>Output compare mode</th><th>OCxM[4:0]</th><th>oc1refc</th><th>oc2refc</th><th>oc3refc</th><th>oc4refc</th><th>oc5refc</th><th>oc6refc</th><th>oc7refc</th><th>PWM Mode (1)</th></tr>
<tr><td>Asymmetric PWM mode 1</td><td>01110</td><td colspan="2" rowspan="2">oc1ref (up-counting)
oc2ref (down-counting)</td><td colspan="2" rowspan="2">oc3ref (up-counting)
oc4ref (down-counting)</td><td colspan="2" rowspan="2">oc5ref (up-counting)
oc6ref (down-counting)</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 2</td><td>01111</td><td>2</td></tr>
<tr><td>Asymmetric PWM mode 3</td><td>10010</td><td>oc1ref (up-counting)
oc5ref (down-counting)</td><td>oc2ref (up-counting)
oc5ref (down-counting)</td><td>oc3ref (up-counting)
oc5ref (down-counting)</td><td>oc4ref (up-counting)
oc5ref (down-counting)</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 4</td><td>10011</td><td>2</td></tr>
<tr><td>Asymmetric PWM mode 5</td><td>10100</td><td>oc1ref (up-counting)
oc6ref (down-counting)</td><td>oc2ref (up-counting)
oc6ref (down-counting)</td><td>oc3ref (up-counting)
oc6ref (down-counting)</td><td>oc4ref (up-counting)
oc6ref (down-counting)</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 6</td><td>10101</td><td>2</td></tr>
</table>

Table 295. Asymmetric PWM modes

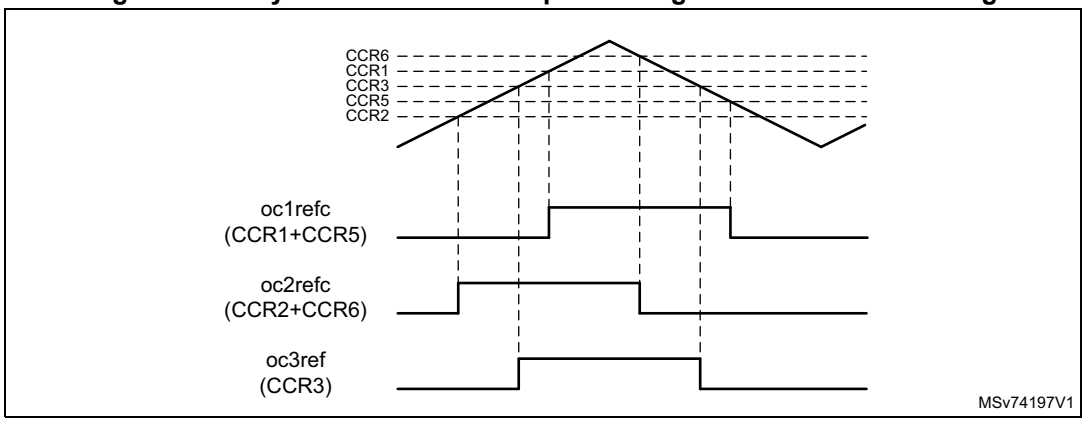
Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	oc5refc	oc6refc	oc7refc	PWM Mode (1)
Asymmetric PWM mode 7	10110	oc1ref OR oc5ref (up-counting) oc1refc=1 (down-counting)	oc2ref OR oc6ref (up-counting) oc2refc=1 (down-counting)	oc3ref OR oc4ref (up-counting) oc3refc=1 (down-counting)	oc4ref OR oc7ref (up-counting) oc4refc=1 (down-counting)	-	-	-	1(2)
Asymmetric PWM mode 8	10111	oc1ref AND oc5ref (up-counting) oc1refc=0 (down-counting)	oc2ref AND oc6ref (up-counting) oc2refc=0 (down-counting)	oc3ref AND oc4ref (up-counting) oc3refc=0 (down-counting)	oc4ref AND oc7ref (up-counting) oc4refc=0 (down-counting)				2(3)
Asymmetric PWM mode 9	11000	oc1refc=1 (up-counting) oc1ref OR oc5ref (down-counting)	oc2refc=1 (up-counting) oc2ref OR oc6ref (down-counting)	oc3refc=1 (up-counting) oc3ref OR oc4ref (down-counting)	oc4refc=1 (up-counting) oc4ref OR oc7ref (down-counting)	-	-	-	1(2)
Asymmetric PWM mode 10	11001	oc1refc=0 (up-counting) oc1ref AND oc5ref (down-counting)	oc2refc=0 (up-counting) oc2ref AND oc6ref (down-counting)	oc3refc=0 (up-counting) oc3ref AND oc4ref (down-counting)	oc4refc=0 (up-counting) oc7ref AND oc4ref (down-counting)				2(3)

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the 2 channels used for the asymmetric mode
2. oc5ref, oc6ref, oc4ref and oc7ref must be programmed in PWM mode 2
3. oc5ref, oc6ref, oc4ref and oc7ref must be programmed in PWM mode 1

The *Figure 259* represents an example suitable for single-shunt current sensing techniques requiring phase-shifted PWMs, for 3-phase motor control. This is done using asymmetric PWM mode 3 and 5:

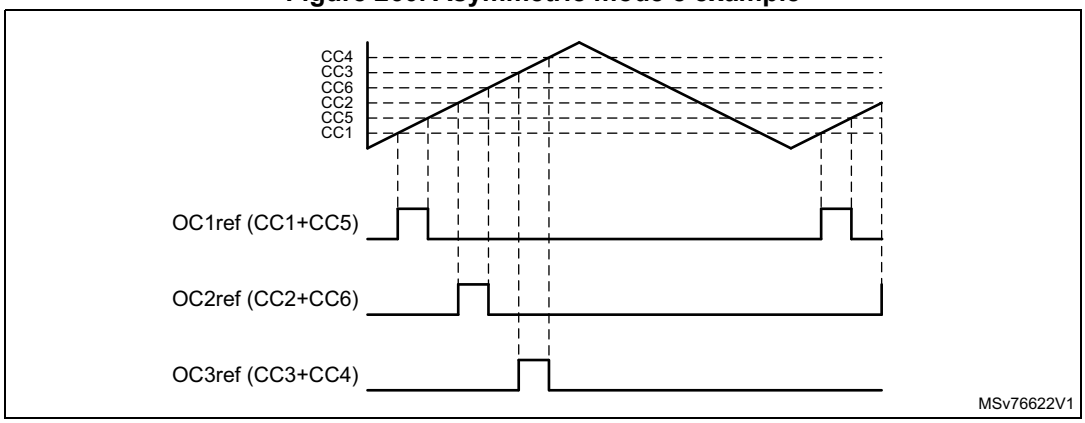
- Channel 1 is configured in asymmetric PWM mode 4
- Channel 2 is configured in asymmetric PWM mode 5
- Channel 3 is configured in regular PWM mode 2
- Compare 4 and 7 are available for ADC triggering purpose

Figure 259. Asymmetric mode example for single-shunt current sensing



The *Figure 260* represents an example of the waveforms that can be generated with asymmetric modes 7 to 10.

Figure 260. Asymmetric mode 8 example



31.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs, as summarized in the below:

Table 296. Combined PWM mode overview

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	oc5refc	oc6refc	oc7refc	PWM mode(1)
Combined PWM mode 1	01100	oc1ref OR oc2ref		oc3ref OR oc4ref		oc5ref OR oc6ref		-	1
Combined PWM mode 2	01101	oc1ref AND oc2ref		oc3ref AND oc4ref		oc5ref AND oc6ref		-	2
Combined PWM mode 3	10000	oc1ref OR oc5ref	oc2ref OR oc6ref	oc3ref OR oc5ref	oc4ref OR oc6ref	-	-	-	1
Combined PWM mode 4	10001	oc1ref AND oc5ref	oc2ref AND oc6ref	oc3ref AND oc5ref	oc4ref AND oc6ref	-	-	-	2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value.

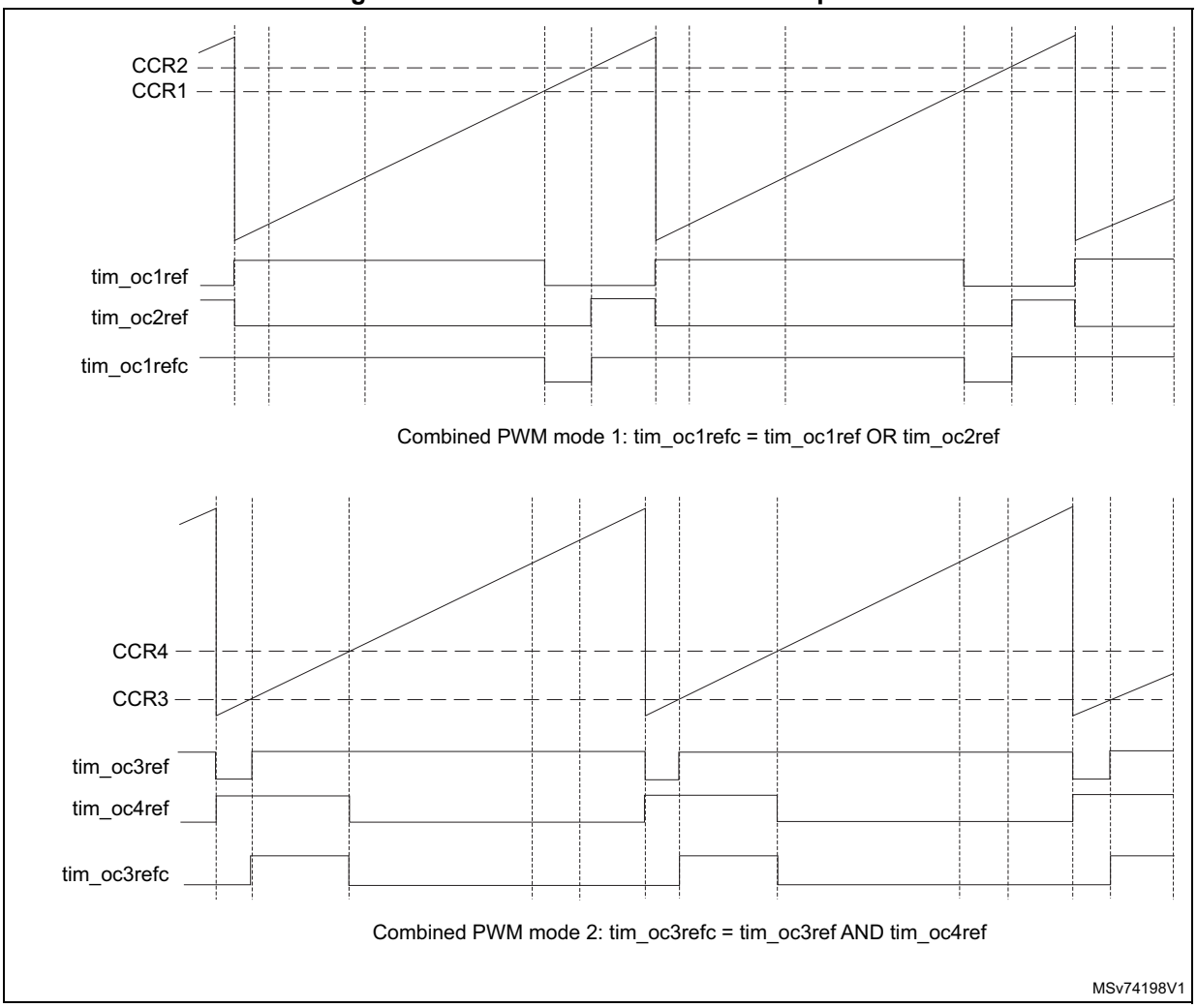
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in PWM mode 1 and the other in PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

Figure 261 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

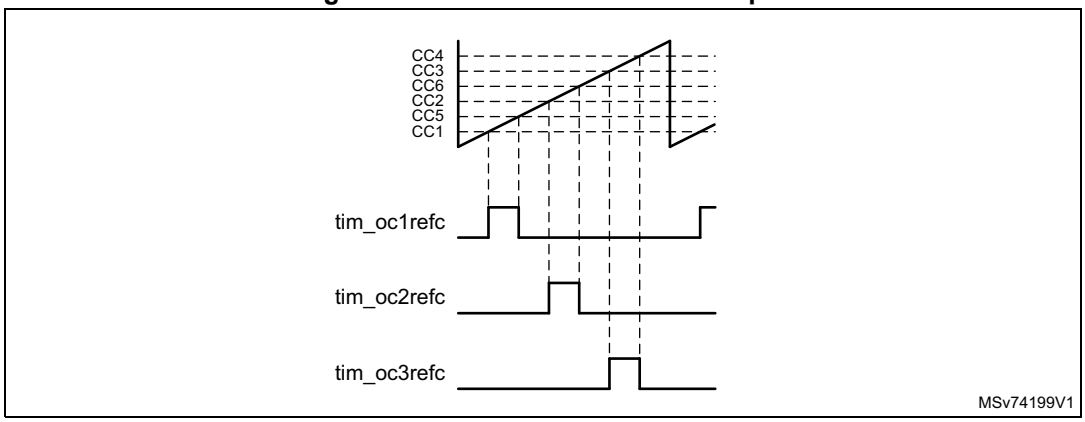
Figure 261. Combined PWM mode examples



The *Figure 262* represents an example suitable for a 3-phase interleaved converter requiring independent control of duty cycle and phase-shift. This is done using Combined PWM mode 2 and 4:

- Channel 1 is configured in Combined PWM mode 4
- Channel 2 is configured in Combined PWM mode 4
- Channel 3 is configured in Combined PWM mode 2

Figure 262. Combined PWM 3 example



31.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal.

The 4 bits GC5C[4:1] in the TIM_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is the logical AND of tim_oc1ref and tim_oc5ref
- If GC5C2 is set, tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
- If GC5C3 is set, tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
- If GC5C4 is set, tim_oc4refc is the logical AND of tim_oc4ref and tim_oc5ref

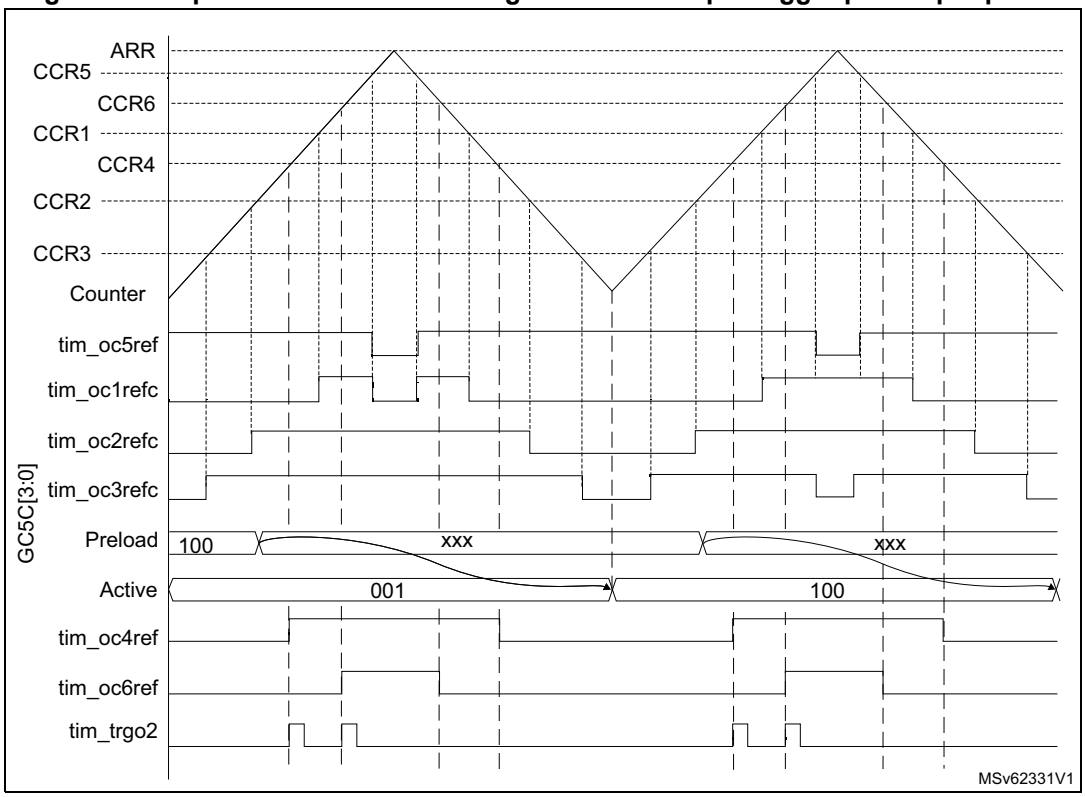
It is possible also to select an OR fonction with GC5C1O, GC5C2O, GC5C3O and GC5C4O bits, as following:

- If GC5C1O is set, tim_oc1refc is the logical OR of tim_oc1ref and tim_oc5ref
- If GC5C2O is set, tim_oc2refc is the logical OR of tim_oc2ref and tim_oc5ref
- If GC5C3O is set, tim_oc3refc is the logical OR of tim_oc3ref and tim_oc5ref
- If GC5C4O is set, tim_oc4refc is the logical OR of tim_oc4ref and tim_oc5ref

If both GC5Cx and GC5CxO bits are set, GC5Cx bit overrides GC5CxO programming and tim_ocxrefc is equal to tim_ocxref AND tim_oc5ref.

Combined 3-phase PWM mode can be selected independently on channels 1 to 4 by setting at least one of the bits GC5C[4:1] or GC5CO[4:1].

Figure 263. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 31.3.36: ADC triggers* for more details.

31.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIM_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIM_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIM_BDTR and TIM_CR2 registers. Refer to *Table 304: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

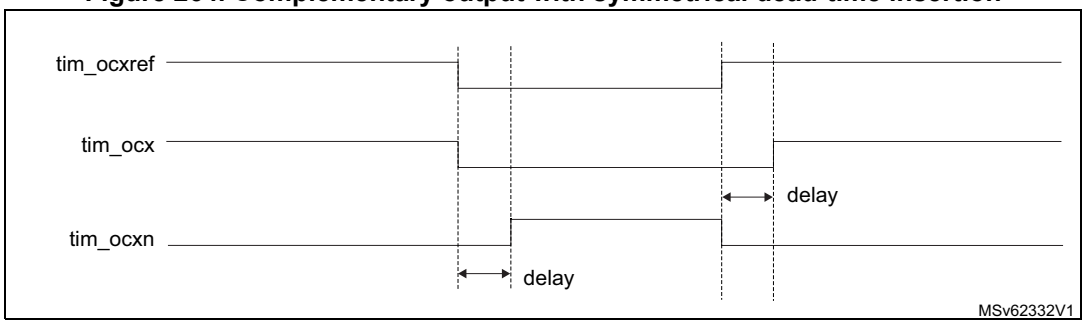
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 264. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIM_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 265*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIM_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIM_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIM_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 265. Asymmetrical deadtime

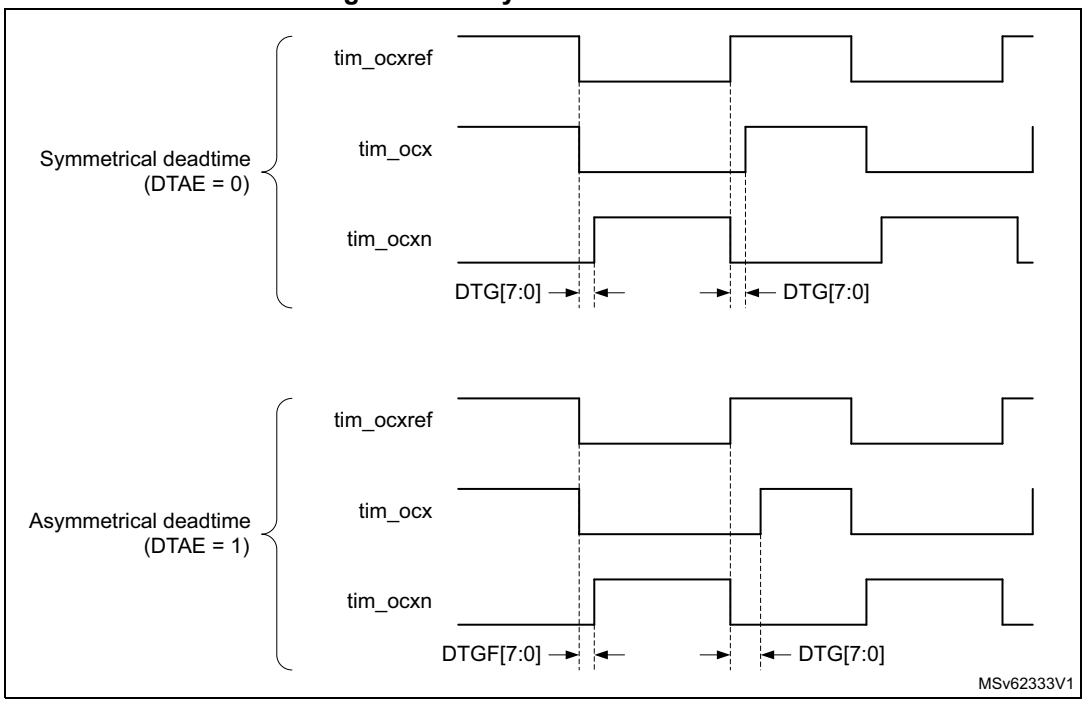


Figure 266. Dead-time waveforms with delay greater than the negative pulse

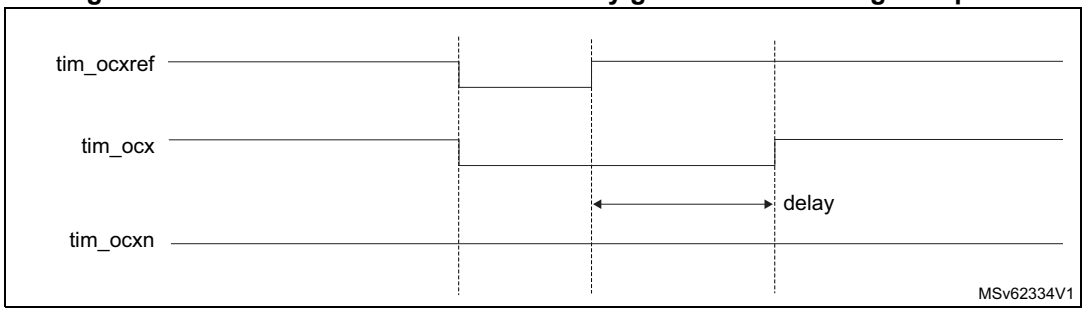
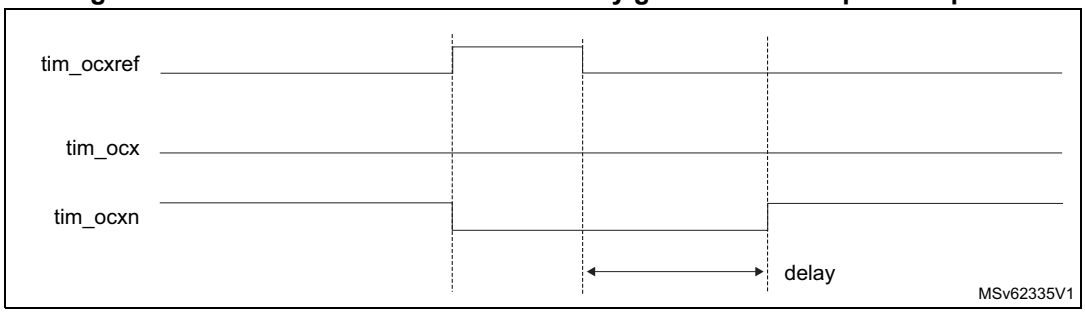


Figure 267. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIM_BDTR register. Refer to *Section 31.6.20: TIM break and dead-time register (TIM_BDTR)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIM_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

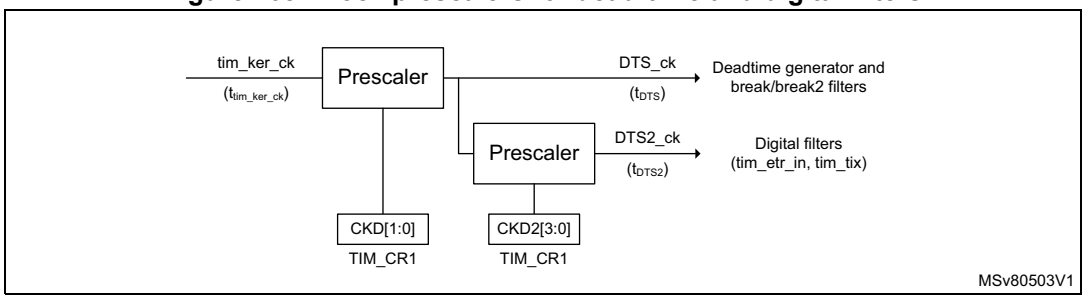
Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

31.3.18 Clock dividers for dead-time generators and digital filters

The dead-time generators and the digital filters (for tim_etr_in, tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the deadtime period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend solely the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 268. Clock prescalers for dead-time and digital filters



31.3.19 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIM_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIM_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIM_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 304: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIM_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIM_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

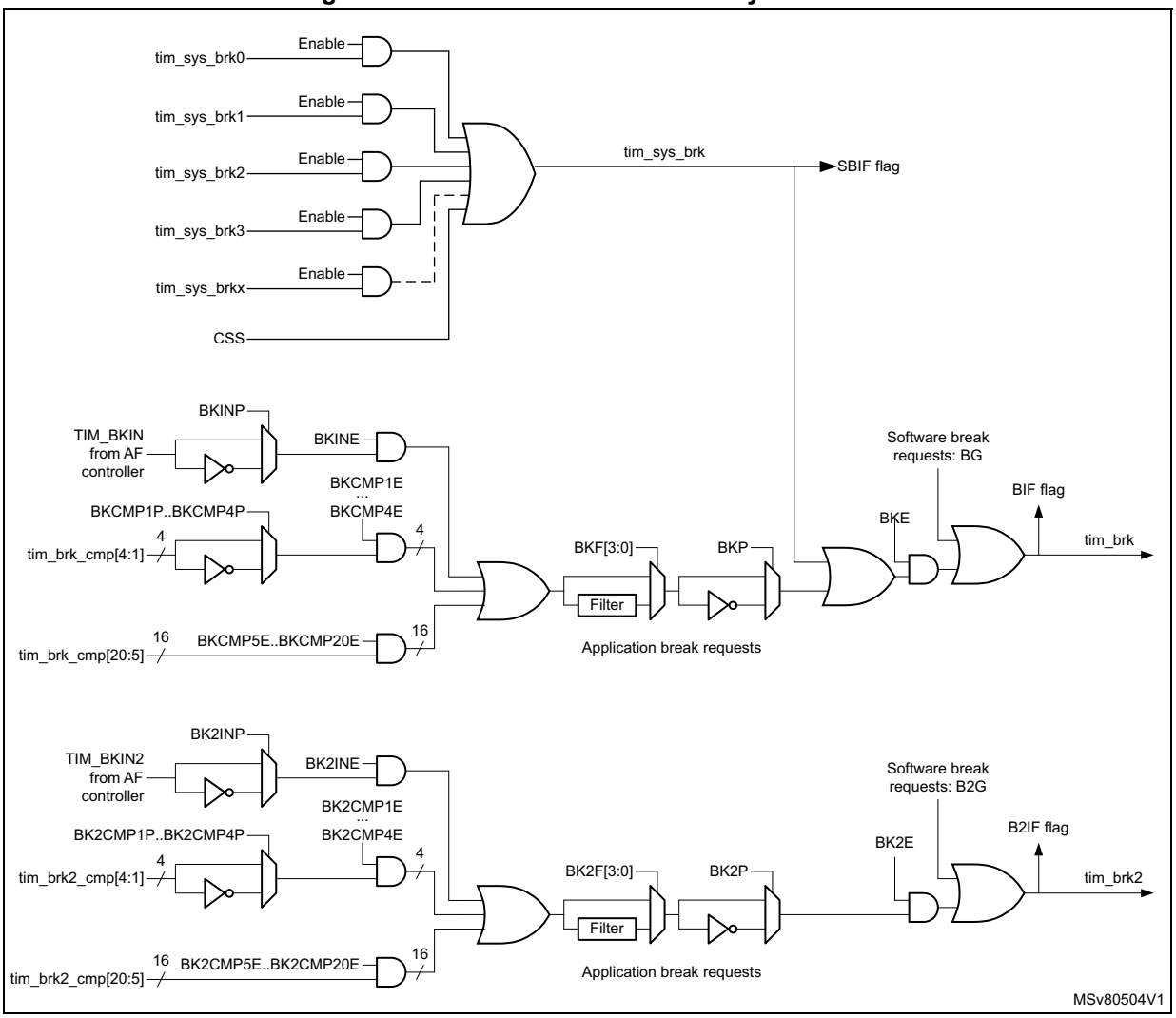
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIM_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIM_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 269* below.

Figure 269. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIM_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flags (B2GF, BGF, SBIF, BIF, and B2IF bits in the TIM_SR register) are set. An interrupt is generated if the BIE bit in the TIM_DIER register is set.
- An output disable status flag (ODS in the TIM_SR register) indicates the status of the outputs:
  - ODS = 0: Idle state. Break was triggered (or MOE was written to 0) and OIS states are forced on tim_ocx and tim_ocxn outputs
  - ODS = 1: Disabled state. Break2 was triggered, tim_ocx and tim_ocxn outputs are inactive
- If the AOE bit in the TIM_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

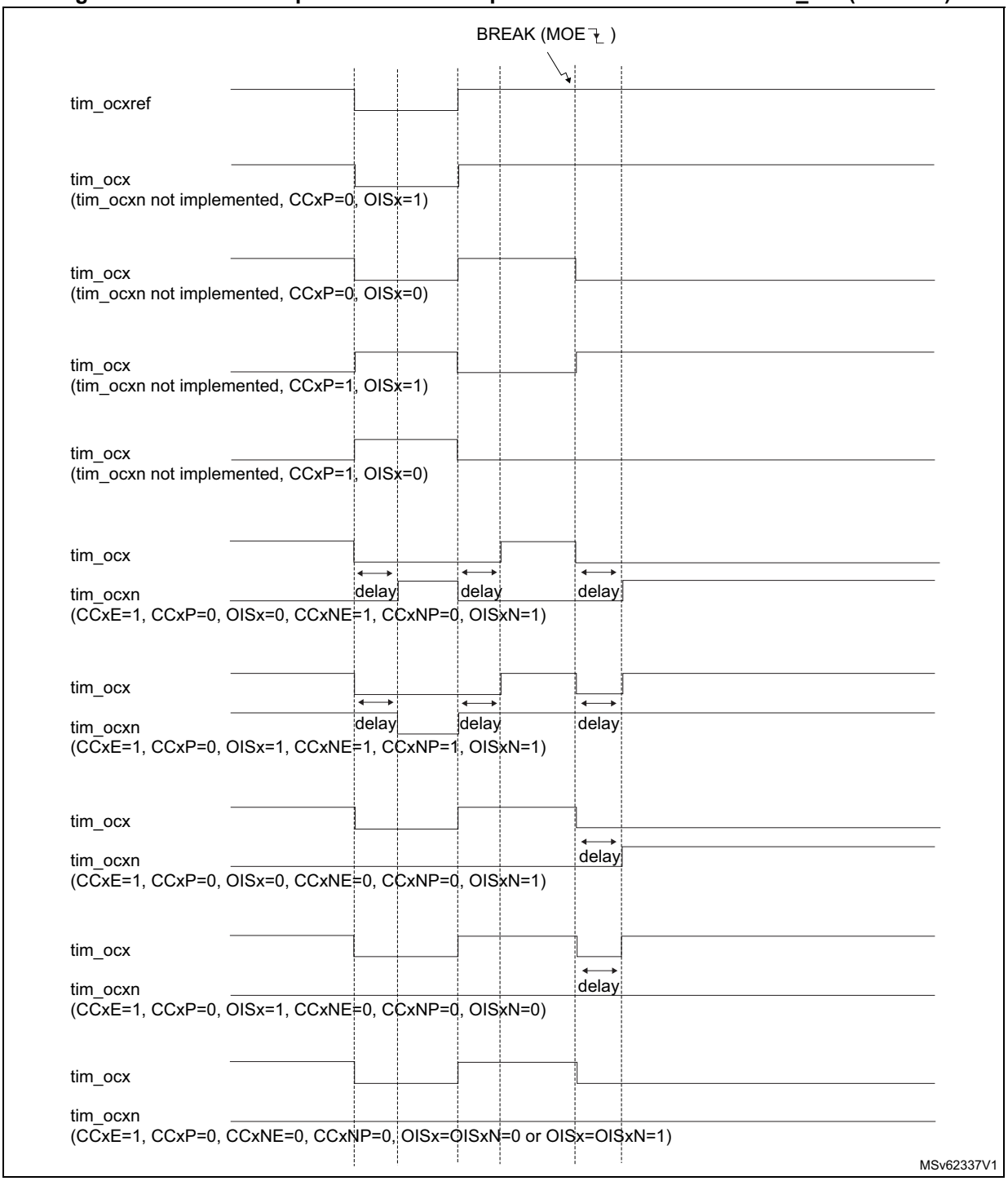
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIM_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIM_BDTR register. Refer to *Section 31.6.20: TIM break and dead-time register (TIM_BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 270 shows an example of behavior of the outputs in response to a break.

Figure 270. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 297*.

The output disable status flag (ODS in the TIM_SR register) indicates the status of the outputs after a break event and is useful to distinguish break and break2 safe states.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 297. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 271 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIM_CCER register).

Figure 271. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

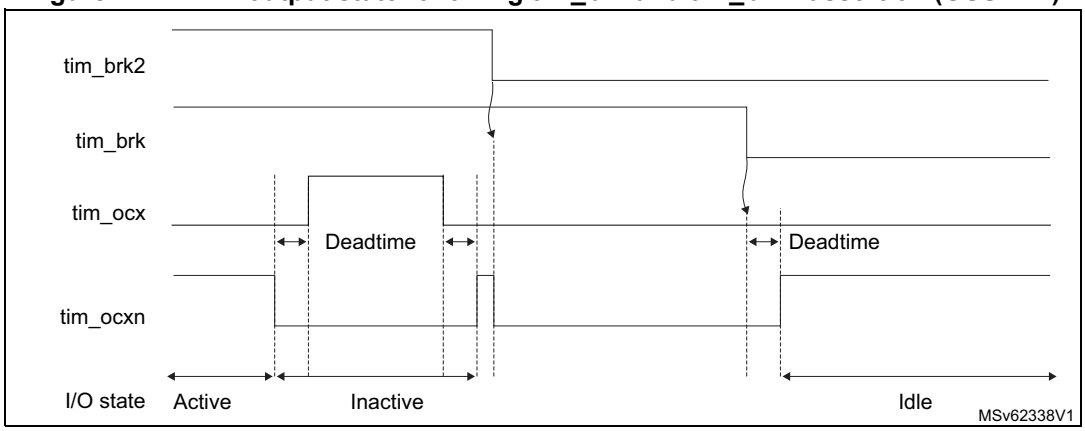
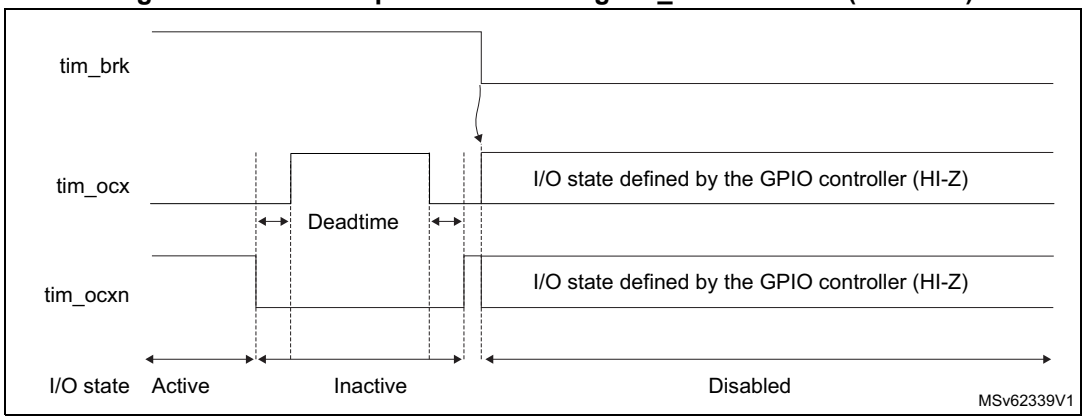


Figure 272. PWM output state following tim_brk assertion (OSSI = 0)



31.3.20 Delayed break

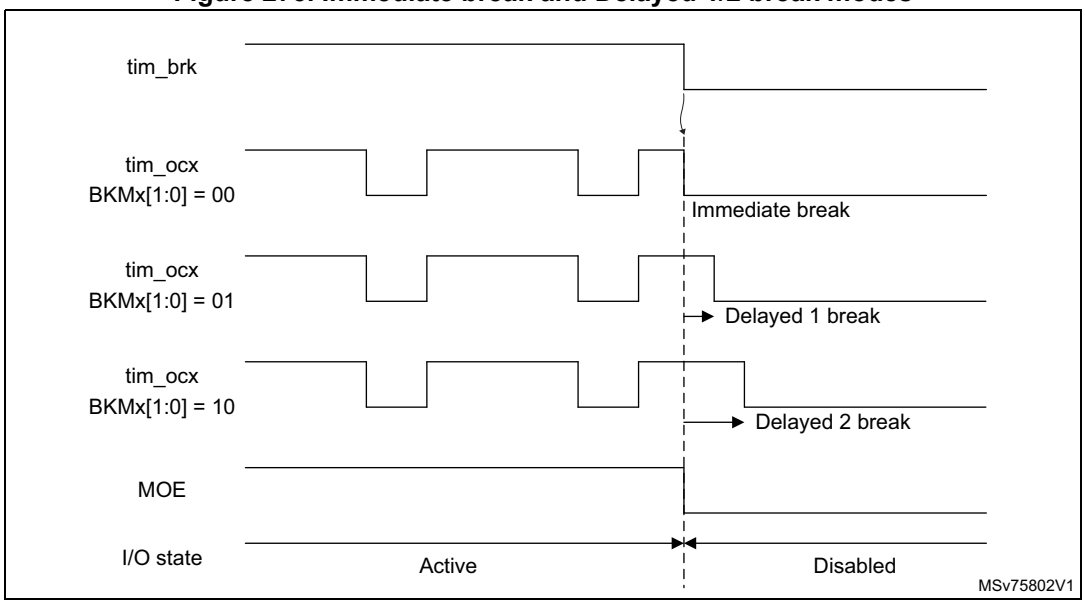
In Neutral-Point Clamped (NPC) multilevel converter topology, a break strategy with immediate outputs shut-down is not suitable, as it may damage the power switches with excessive voltage levels.

The delayed break features provide 2 options delaying the response to a break event, and implement a safe shut-down sequence.

BKMx[1:0] and BKMxN[1:0] bitfields in TIM_MPR1 provides 3 options for each complementary outputs (tim_oc1/tim_oc1n to tim_oc4/tim_oc4n), as show on *Figure 273* below:

- BKM[1:0] = 00: the break response is immediate
- BKM[1:0] = 01: Delayed 1 break. The break response is delayed with a delay defined in the DBK1[7:0] bitfield of the TIM_MPR2 register
- BKM[1:0] = 10: Delayed 2 break. The break response is delayed with a delay defined in the DBK2[7:0] bitfield of the TIM_MPR2 register

Figure 273. Immediate break and Delayed 1/2 break modes



The delayed break mode is triggered when:

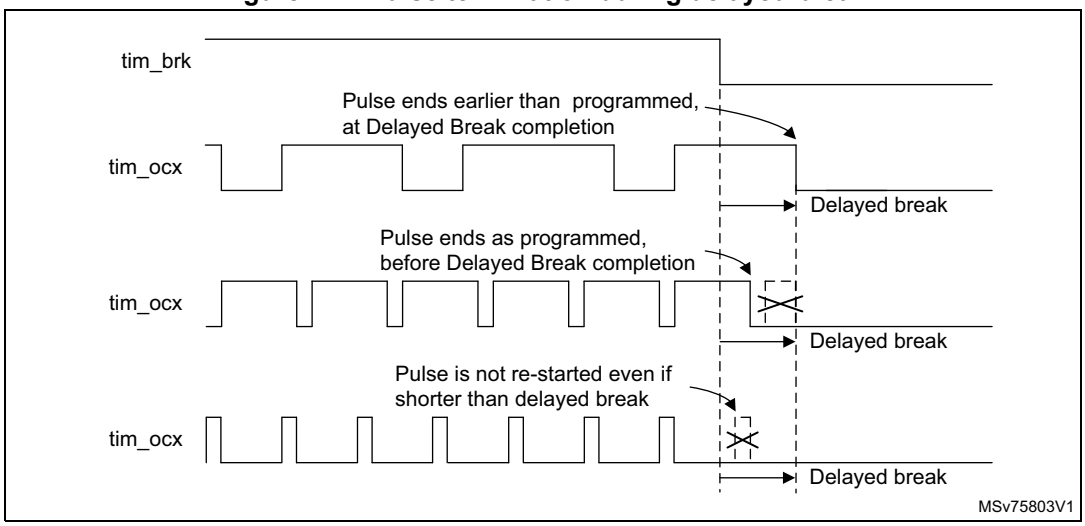
- A break event occurs
- The MOE bit is reset
- When the microcontroller enters debug mode with DBG_TIM_STOP=1

Warning: The break2 feature must be disabled when the delayed mode is enabled.

When delayed break mode is triggered, the MOE bit is immediately cleared, and cannot be set during the delayed break duration. In bidirectional break mode, the break output is reset immediately to indicate the delayed break start to external components. Once the longest of the 2 delays is over, the deadtime generator is enabled to prevent any deadtime violation if OISx/OISxN bits are set.

If the tim_ocx/tim_ocxn signal is reset before the end of the delayed break (due to a compare match in PWM mode, typically), the signal is not maintained during the full delayed break period, to maintain the duty cycle, as shown on *Figure 274* below. Still, the Delayed break circuitry guarantees that the tim_ocx/tim_ocxn signals cannot be set again once the break event occurred.

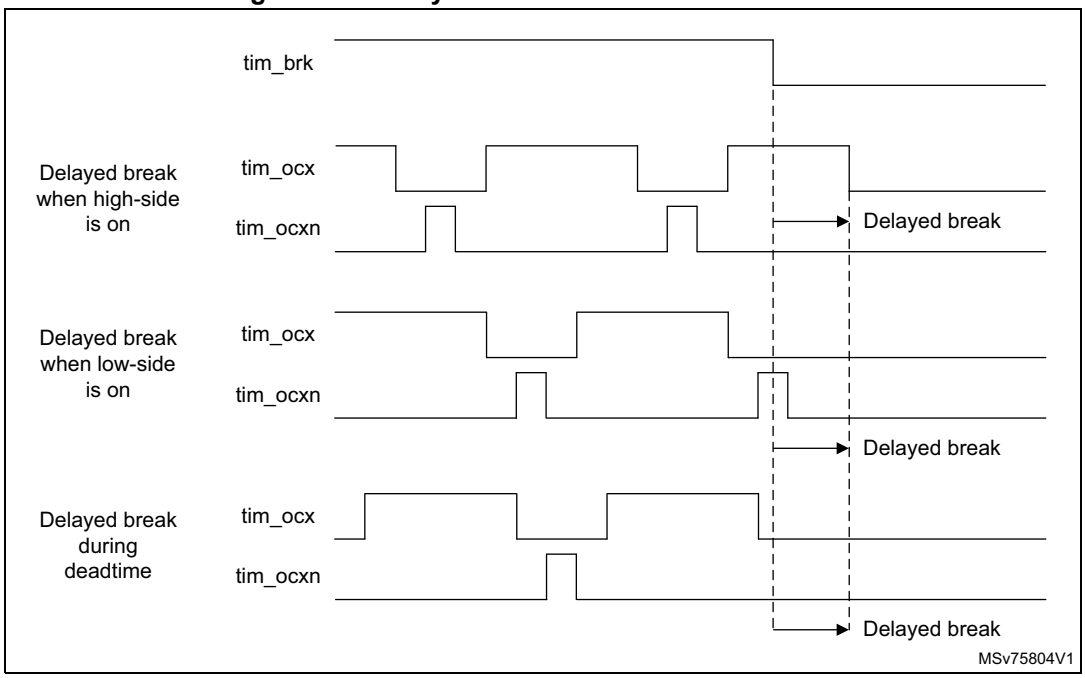
Figure 274. Pulse termination during delayed break



The Figure 563 depicts the behavior when the deadtime generator is enabled, for the 3 cases:

– Delayed break triggered when tim_ocx is set
– Delayed break triggered when tim_ocxn is set
– Delayed break triggered during deadtime interval

Figure 275. Delayed break with deadtime insertion



The delay programming is identical to deadtime generator's, with 2 user-programmable durations in the TIM_MPR2 register: DBK1[7:0] bitfield for the Delayed 1 break and DBK2[7:0] bitfield for the Delayed 2 break.

DBK1[7:0] and DBK2[7:0] bitfields are preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

DBK1[7:0] and DBK2[7:0] bitfields are partially preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

The bits coding for t_{DTG} selection are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBKx[6:0] if DBKx[7] = 0
- DBKx[5:0] if DBKx[7:6] = 10
- DBKx[4:0] if DBKx[7:5] = 110 or 111

The BKMx[1:0] and BKMxN[1:0] bits are static, they can only be modified when the counter is stopped (CEN=0 in TIM_CR1).

The content of the TIM_MPR1 and TIM_MPR2 register can be write protected using the LOCK[1:0] bits in the TIM_BDTR register.

31.3.21 Bidirectional break inputs

The TIM1/TIM8 feature bidirectional break I/Os, as represented on *Figure 276*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIM_BDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIM_BDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN and TIM_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 298*).

Table 298. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

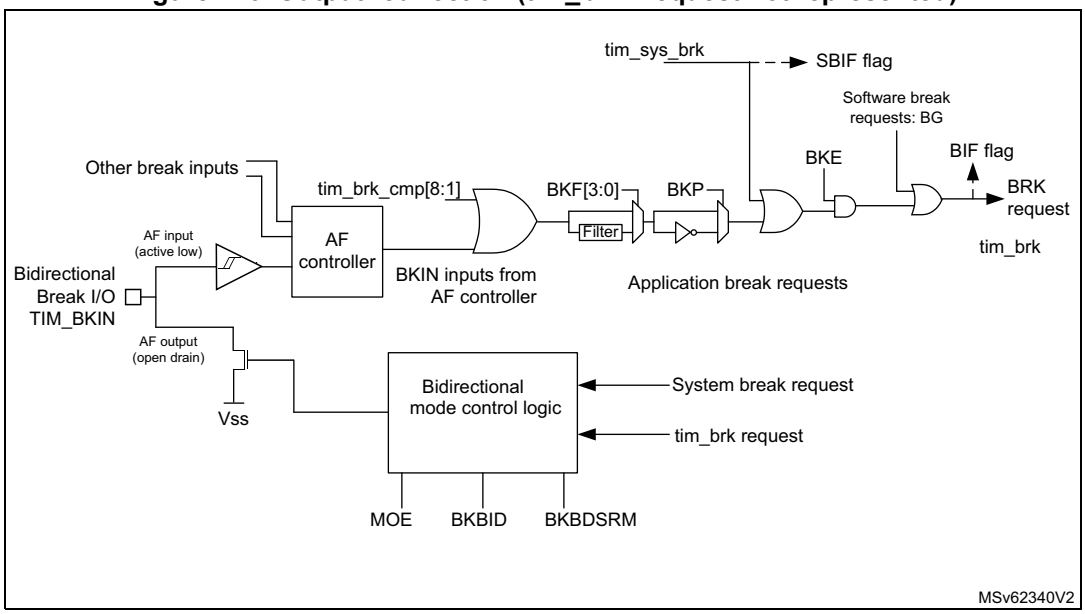
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 276. Output redirection (tim_brk2 request not represented)



31.3.22 Output idle state override

It is possible to override the outputs states when they are disabled (ie MOE bit reset in TIM_BDTR). This feature can be used to apply specific patterns prior to start PWM operations, for instance to refresh bootstrap capacitors of power switch gate drivers.

The OOC control bit in the TIM_OOR register acts as following:

- OOC= 0: 0: tim_ocx and tim_ocxn outputs are in the default idle state
- OOC=1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits in the TIM_OOR register

The OOC bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared by hardware when the MOE bit is set, and the deadtime generator ensures there's no deadtime violation on complementary outputs.

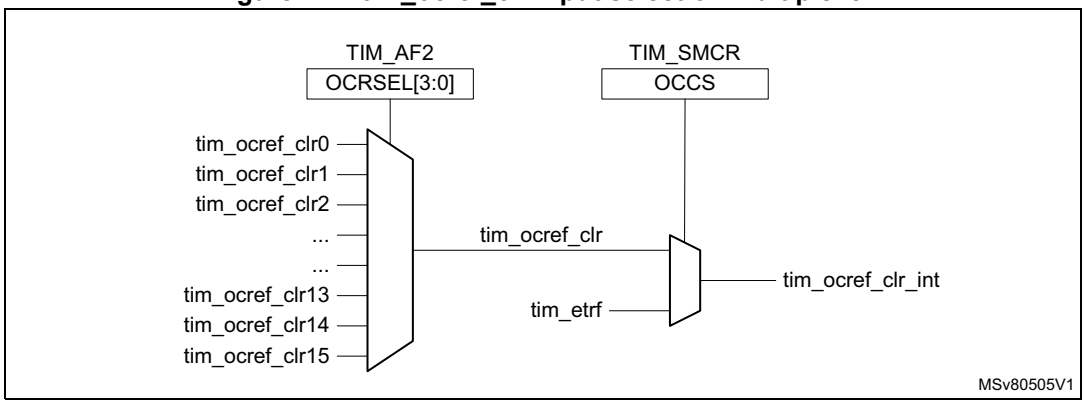
The OOSx/OOSxN bits in the TIM_OOR register cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

31.3.23 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It must not be used in Forced mode. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIM_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[3:0] bitfield in the TIM_AF2 register, as shown on the *Figure 277* below. Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for a list of sources available in the product.

Figure 277. tim_ocref_clr input selection multiplexer

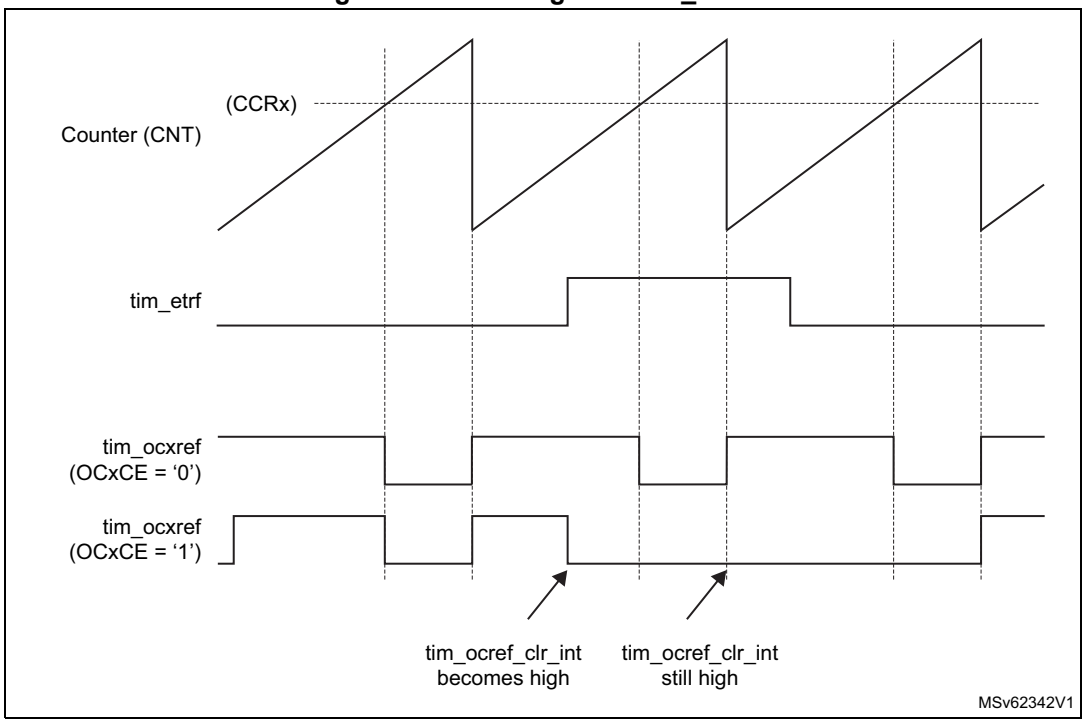


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIM_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIM_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 278 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 278. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to have the OCxCE bit enabled for the channel, the OCxCE bit of the complementary channel (e.g. OC2CE = X for oc2ref in *Figure 279*) is ignored in this case.

Figure 279 to *Figure 282* are describing the output waveforms for some use cases, when the ocref_clr_int signal is asserted.

Figure 279. Asymmetric PWM mode 1

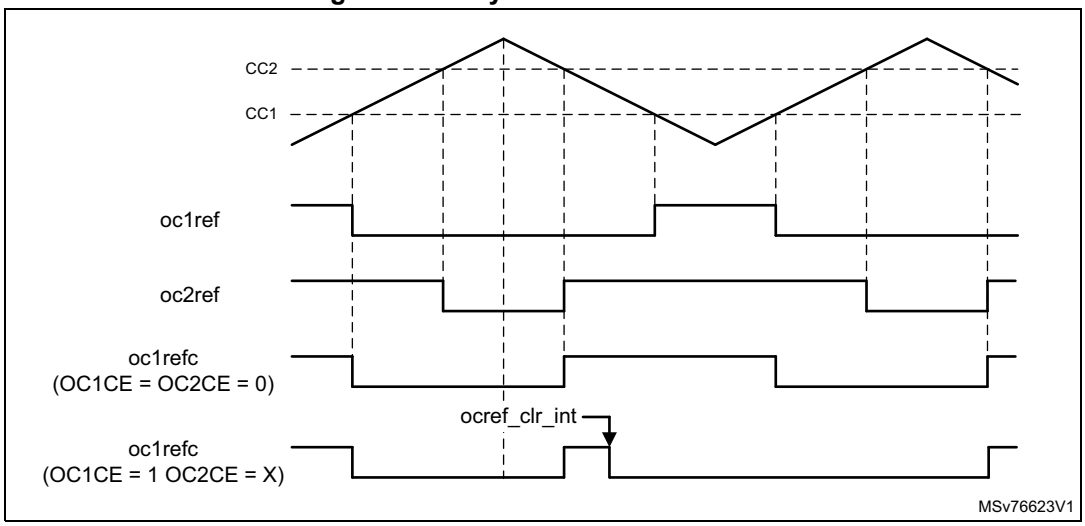


Figure 280. Asymmetric PWM mode 2

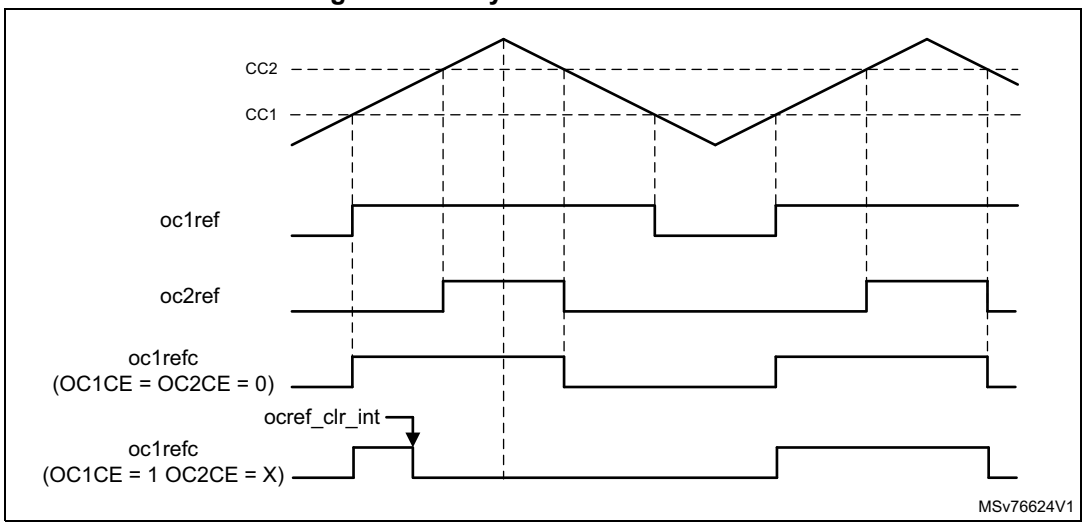


Figure 281. Asymmetric PWM mode 7 (OC1refc = oc1ref OR oc5ref)

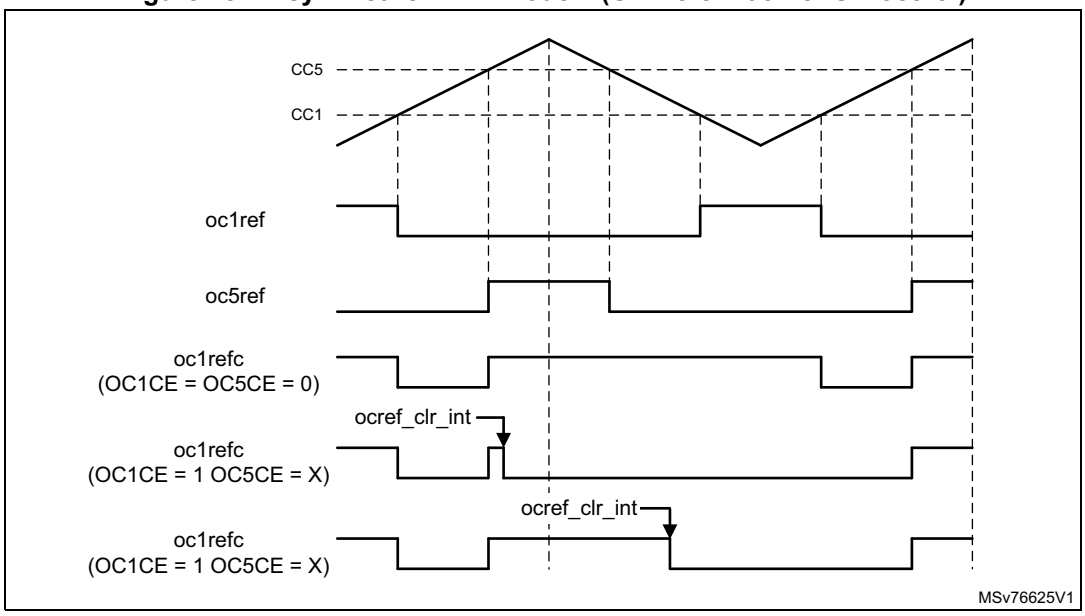
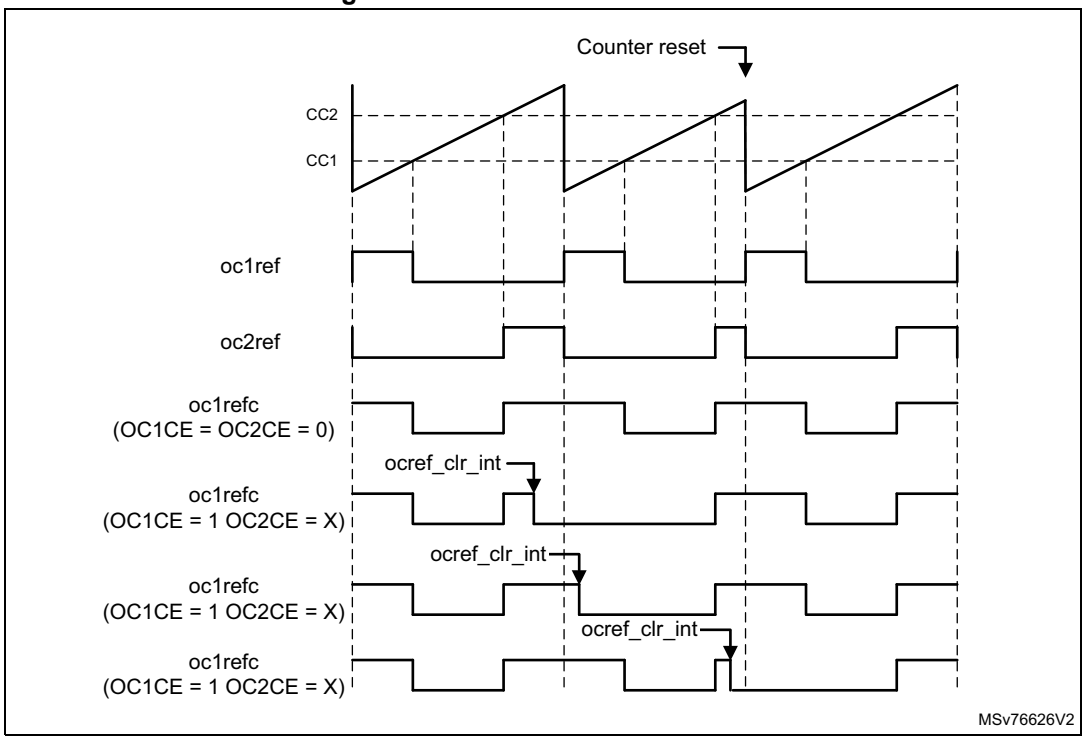


Figure 282. Combined PWM mode 1



Conditions of use

This feature is intended to support cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used for the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One pulse mode with PWM active on match (OPM=1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E, 0x10, 0x12, 0x14, 0x16, 0x18)
- Combined 3-phase modes (GC5Cx or GC5CxO = 1)
- Combined PWM mode 1 or 3 in center aligned mode (CMS[1:0] = 1,2,3 and OCxM[4:0] = 0x0C or 0x10)

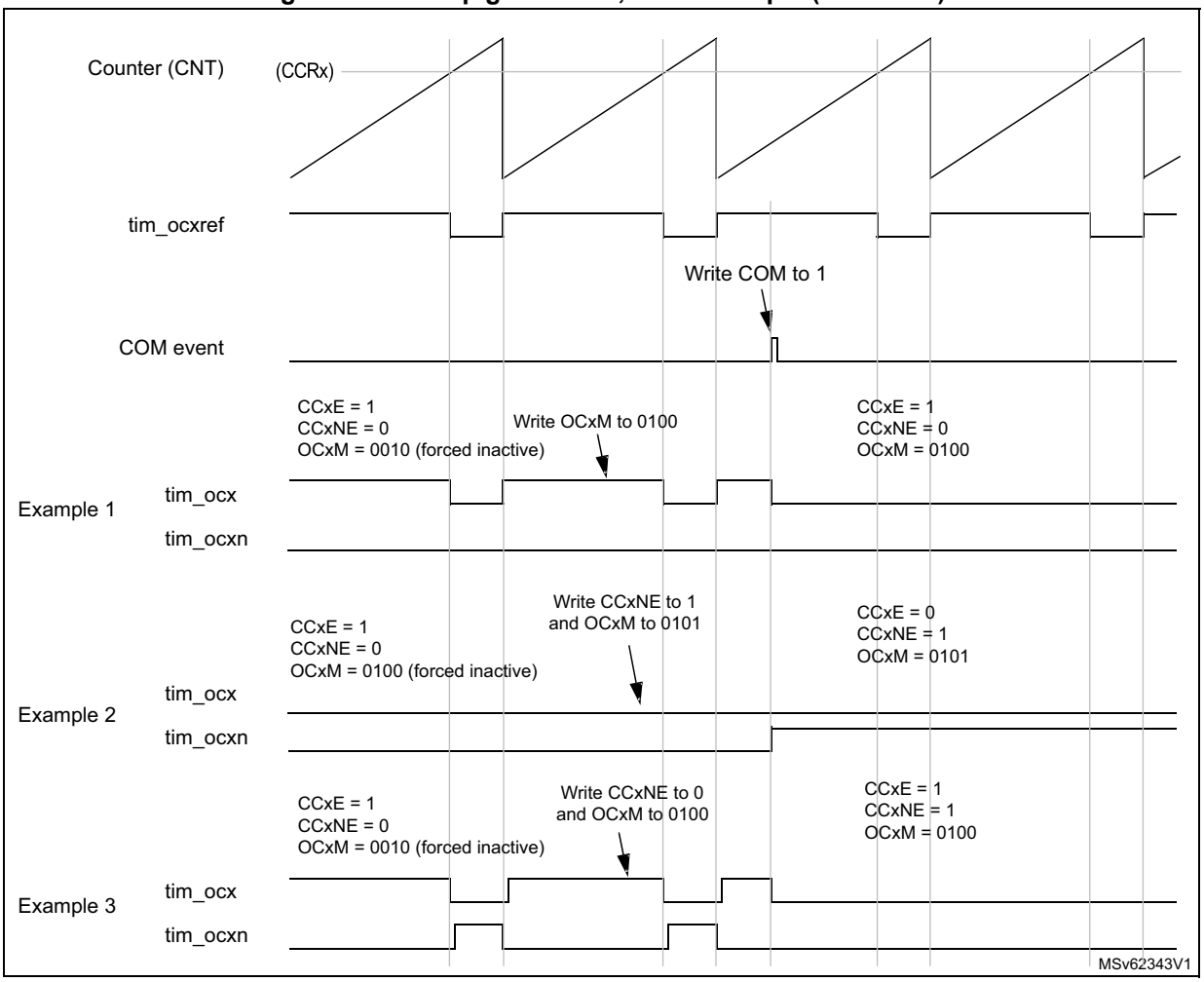
31.3.24 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIM_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIM_SR register), which can generate an interrupt (if the COMIE bit is set in the TIM_DIER register) or a DMA request (if the COMDE bit is set in the TIM_DIER register).

Figure 283 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 283. 6-step generation, COM example (OSSR = 1)



31.3.25 One-pulse mode

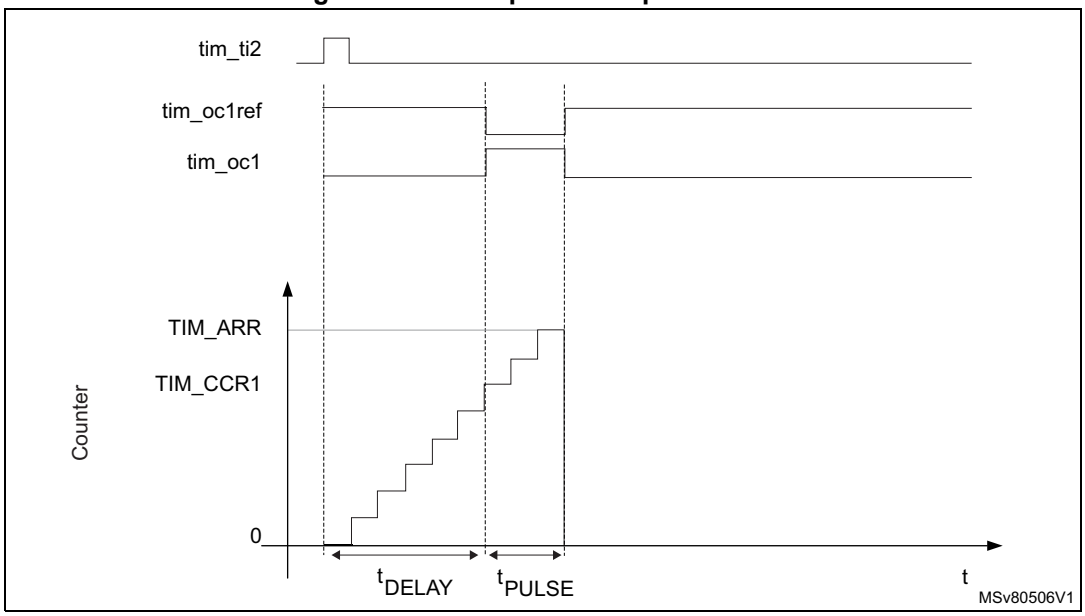
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 284. Example of one pulse mode.



In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 00111 in TIM_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIM_CR1 register must be low.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

31.3.26 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in *Section 31.3.25*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

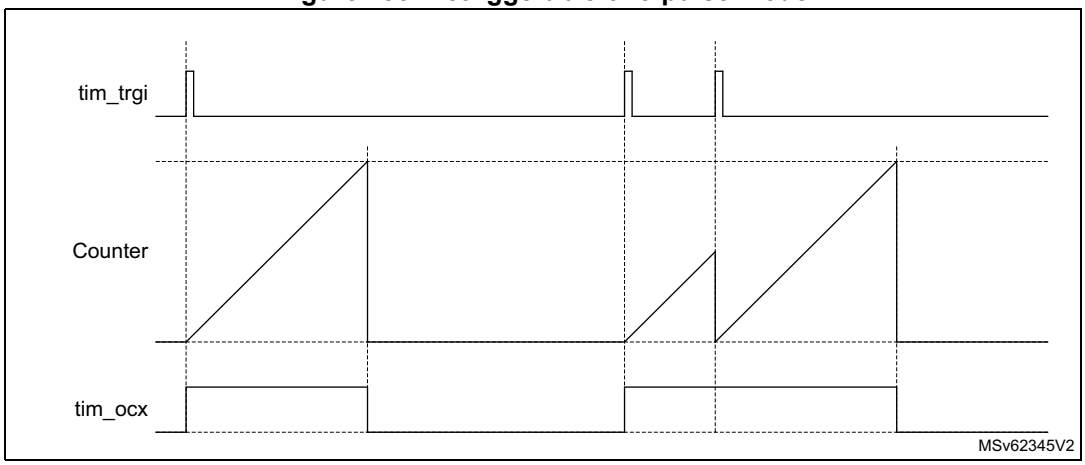
The timer must be in Slave mode, with the bits SMS[4:0] = 01000 (Combined Reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[4:0] and SMS[4:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 285. Retriggerable one-pulse mode



31.3.27 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 286*.

Figure 286. Pulse generator circuitry

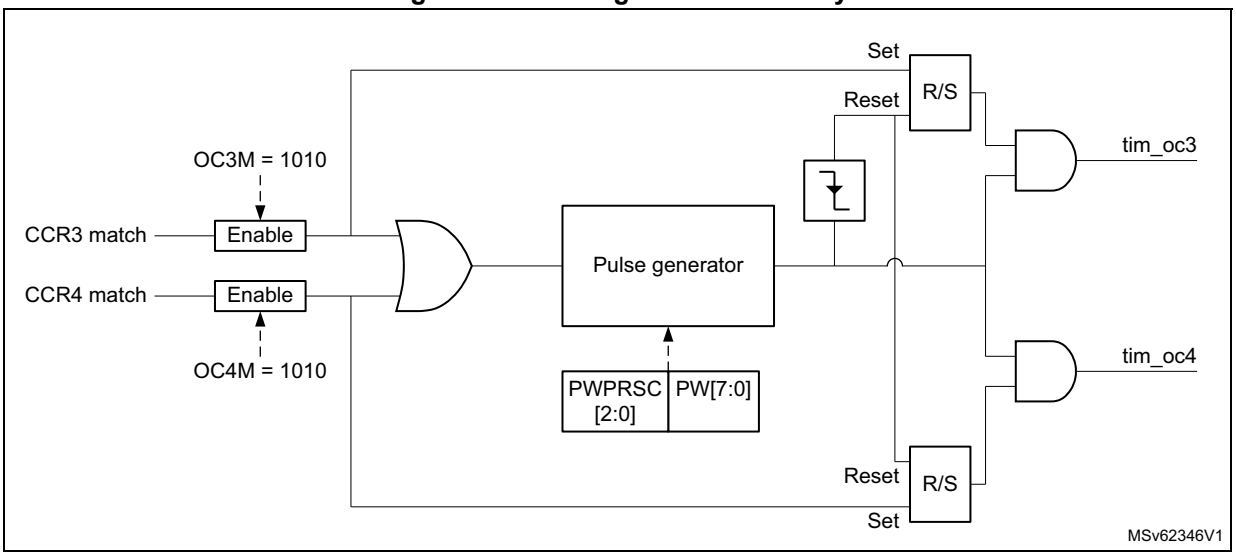
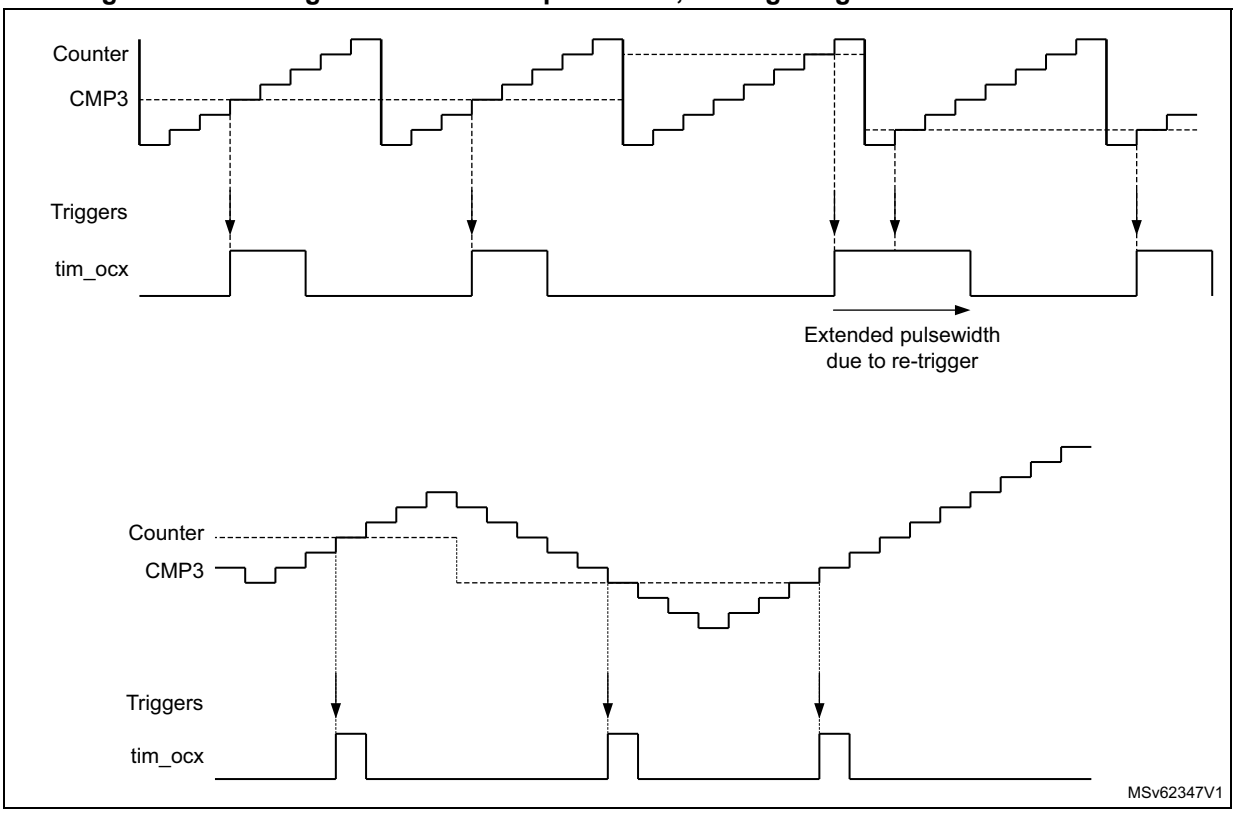


Figure 287 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 287. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIM_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

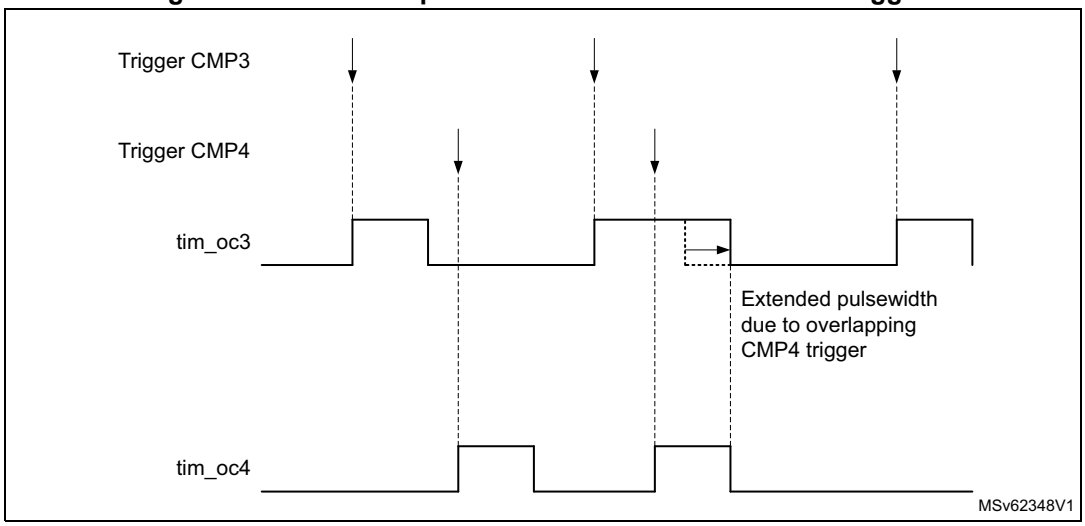
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 288).

Figure 288. Extended pulsewidth in case of concurrent triggers



31.3.28 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 0001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 299*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 299. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 289 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIM_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, Counter enabled).

Figure 289. Example of counter operation in encoder interface mode.

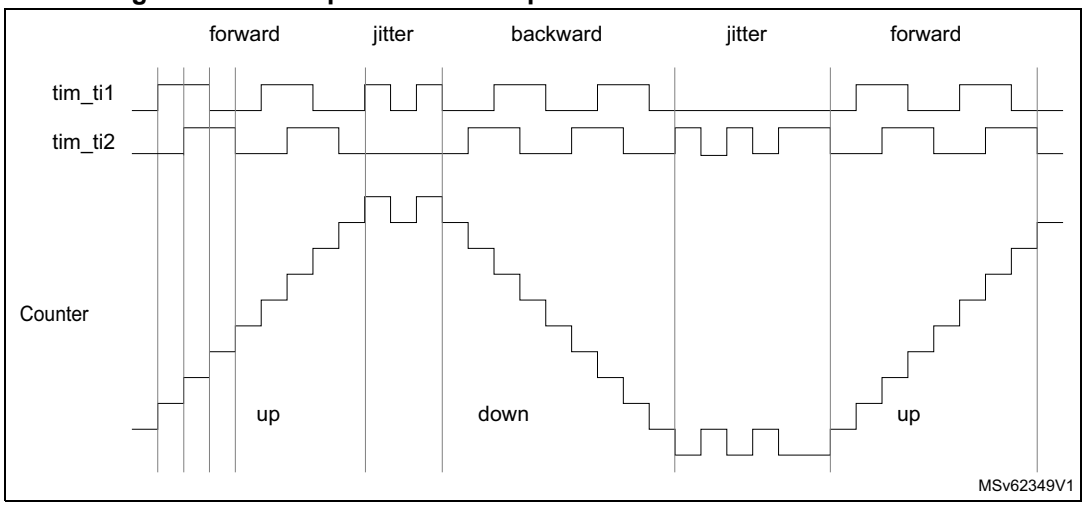


Figure 290 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 290. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

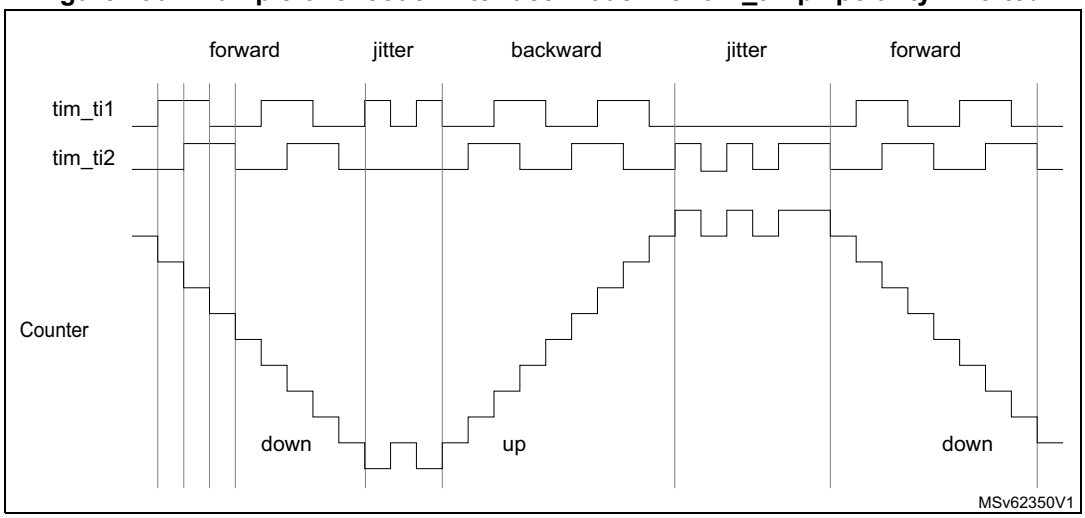
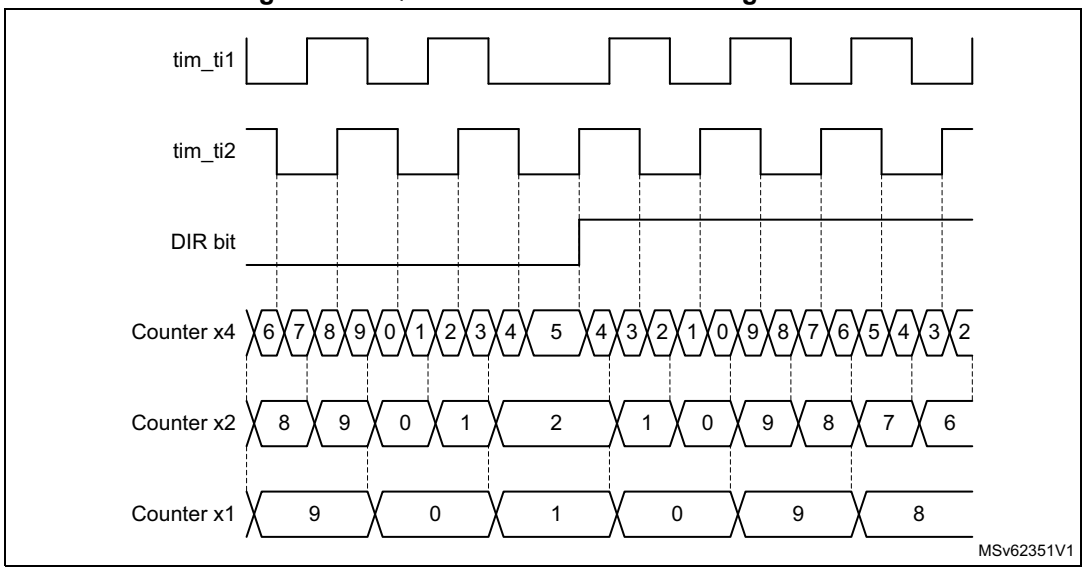


Figure 291 shows the timer counter value during a speed reversal, for various counting modes.

Figure 291. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

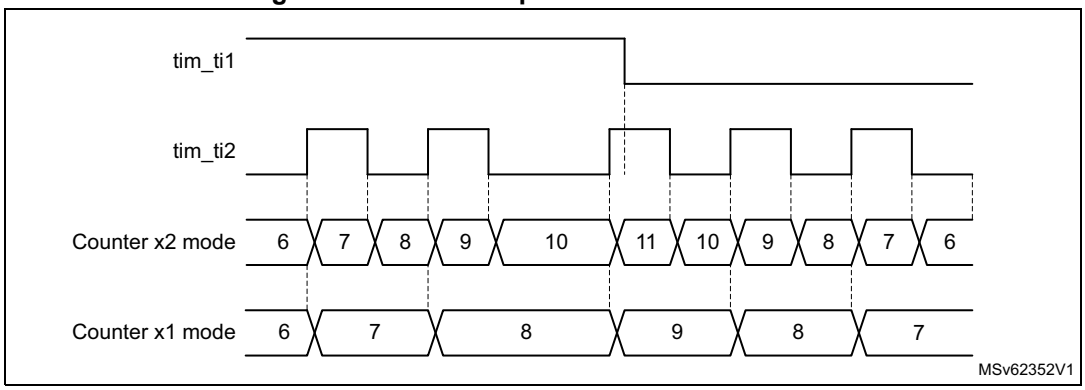
In the clock plus direction mode shown on *Figure 292*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 292. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 293*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 293*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 294*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 293*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 294*).

Figure 293. Directional clock encoder mode (CC1P = CC2P = 0)

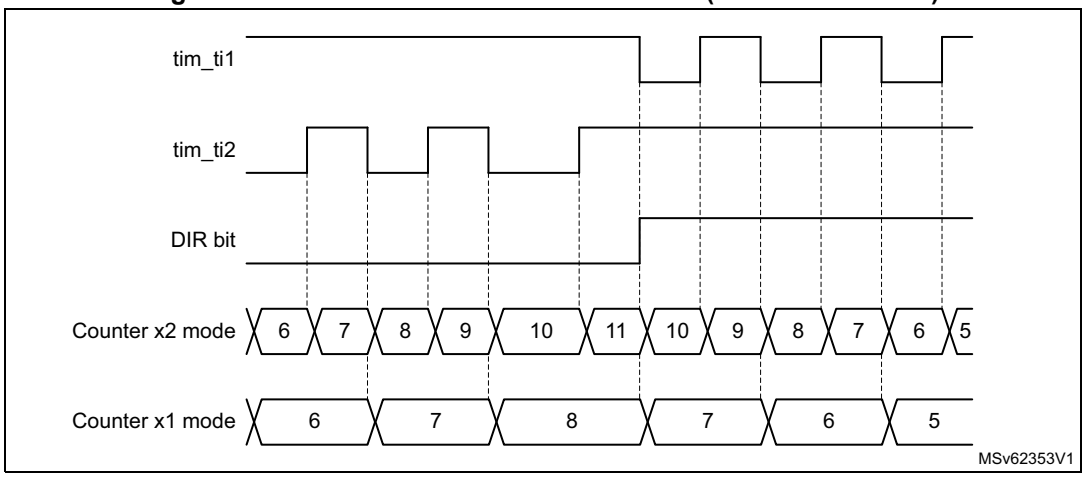


Figure 294. Directional clock encoder mode (CC1P = CC2P = 1)

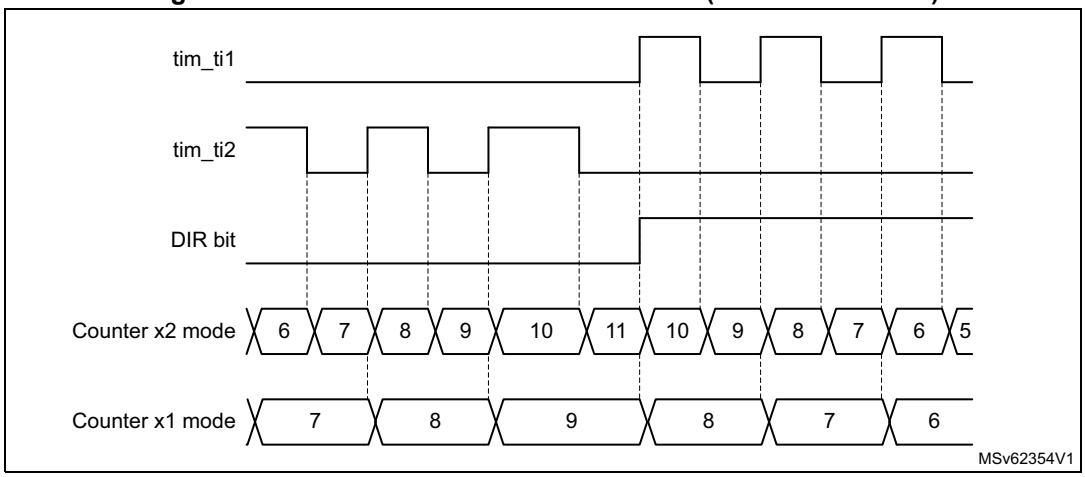


Table 300 here-below details how the directional clock mode operates, for any input transition.

Table 300. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

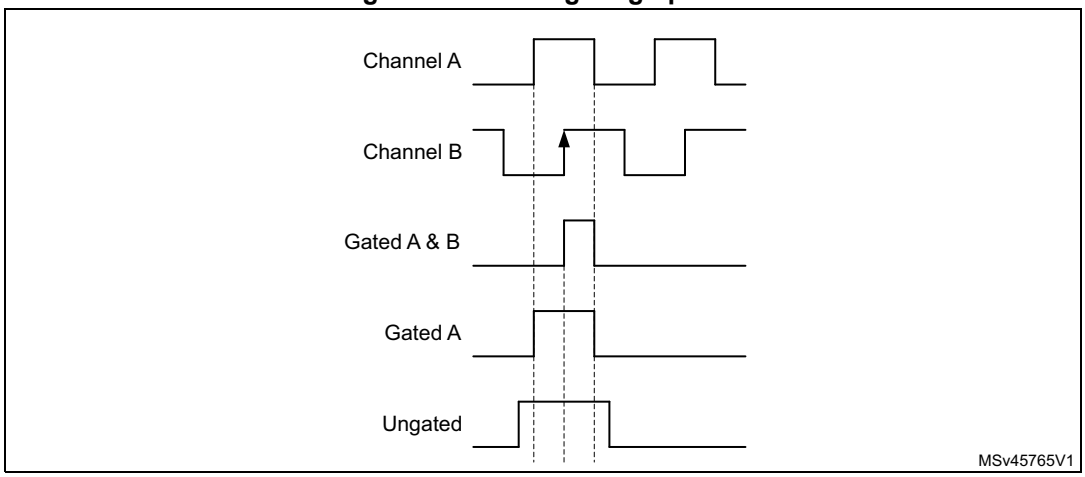
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIM_ECR register. The IE bit must be set only in encoder mode, when the SMS[4:0] bitfield has the following values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 295*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

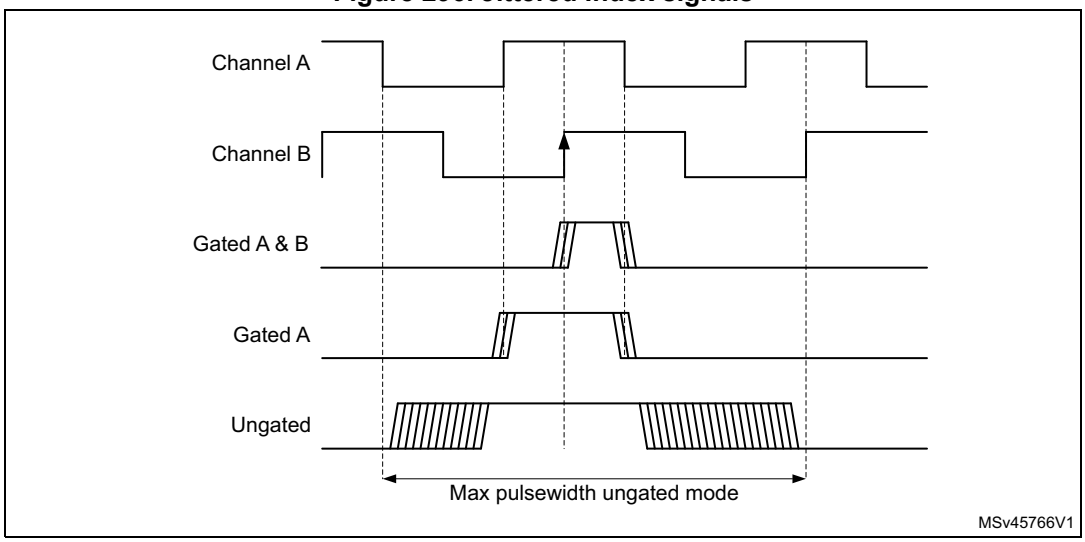
Figure 295. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 296*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 296. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIM_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIM_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 297* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 297. Index generation for IPOS[1:0] = 11

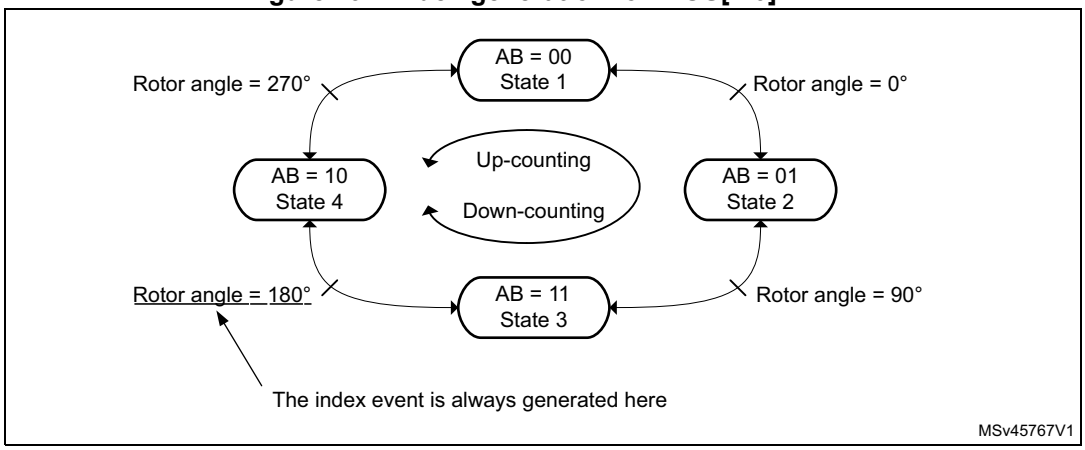


Figure 298 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIM_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 298. Counter reading with index gated on channel A (IPOS[1:0] = 11)

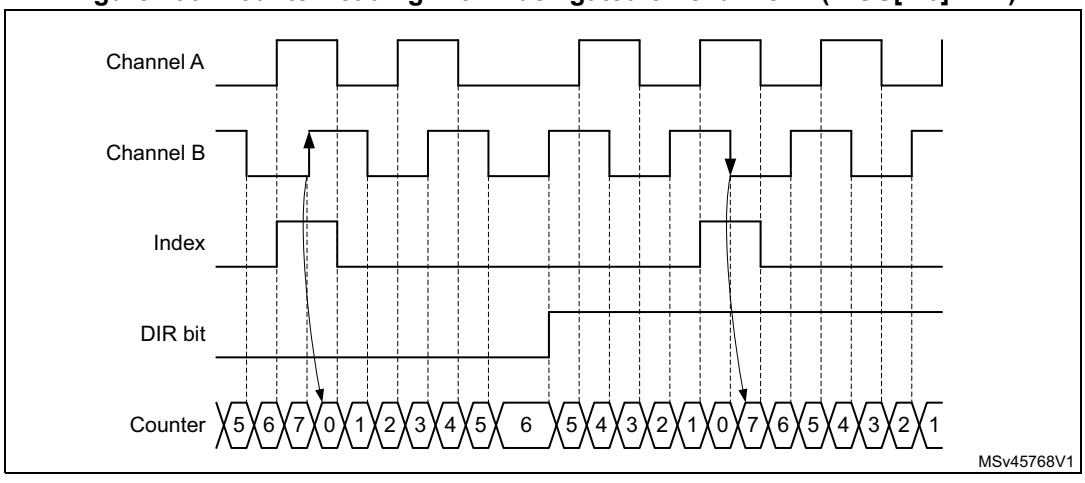


Figure 299. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 299. Counter reading with index ungated (IPOS[1:0] = 00)

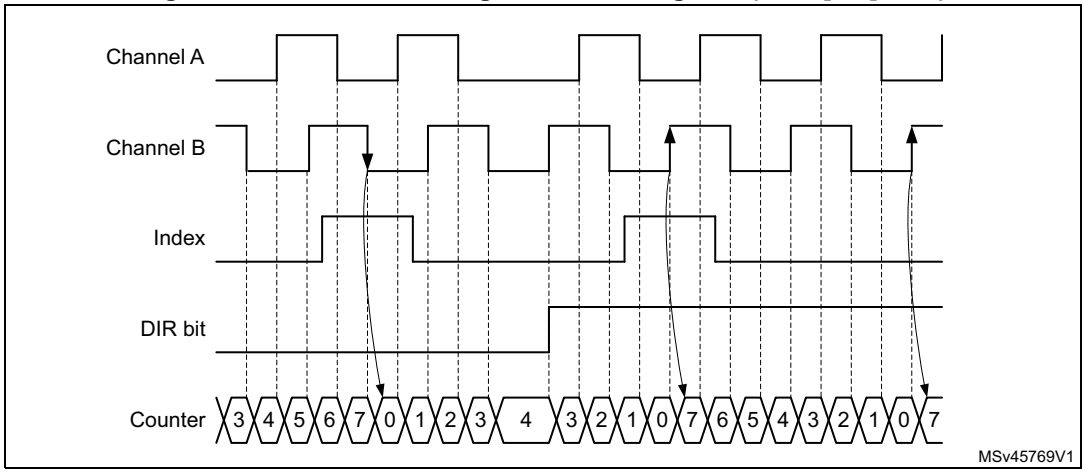


Figure 300. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 300. Counter reading with index gated on channel A and B

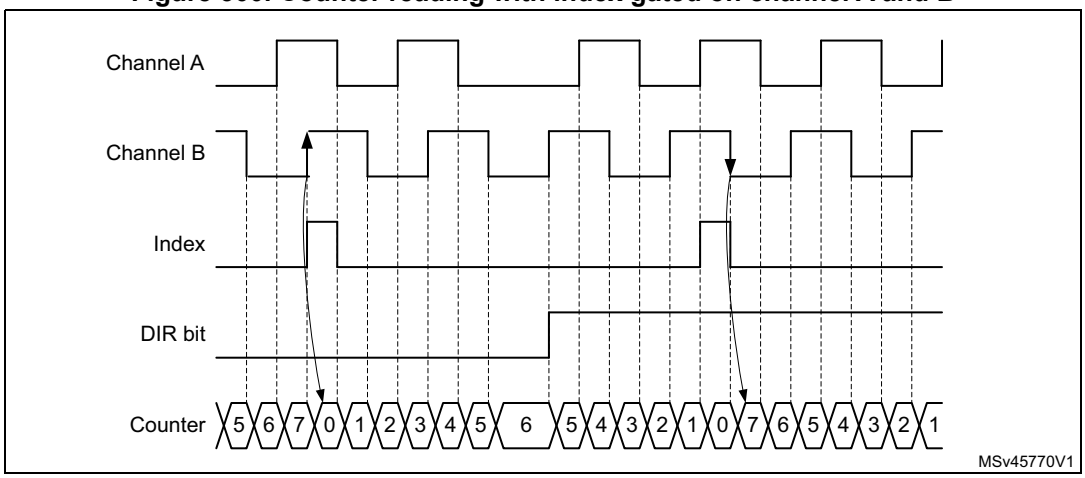


Figure 301 and *Figure 302* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 301. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

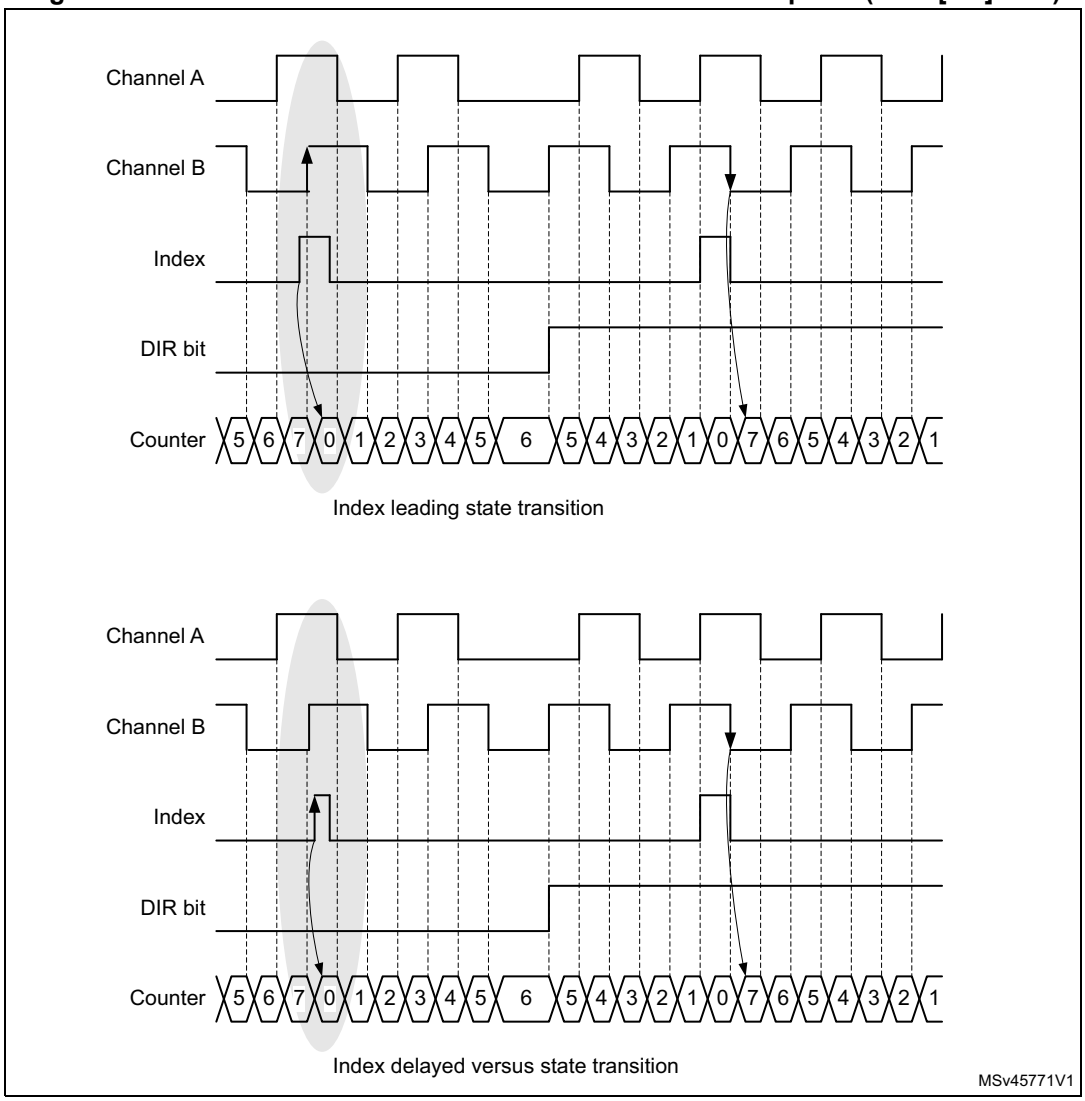


Figure 302. Counter reset Narrow index pulse (closer view, ARR = 0x07)

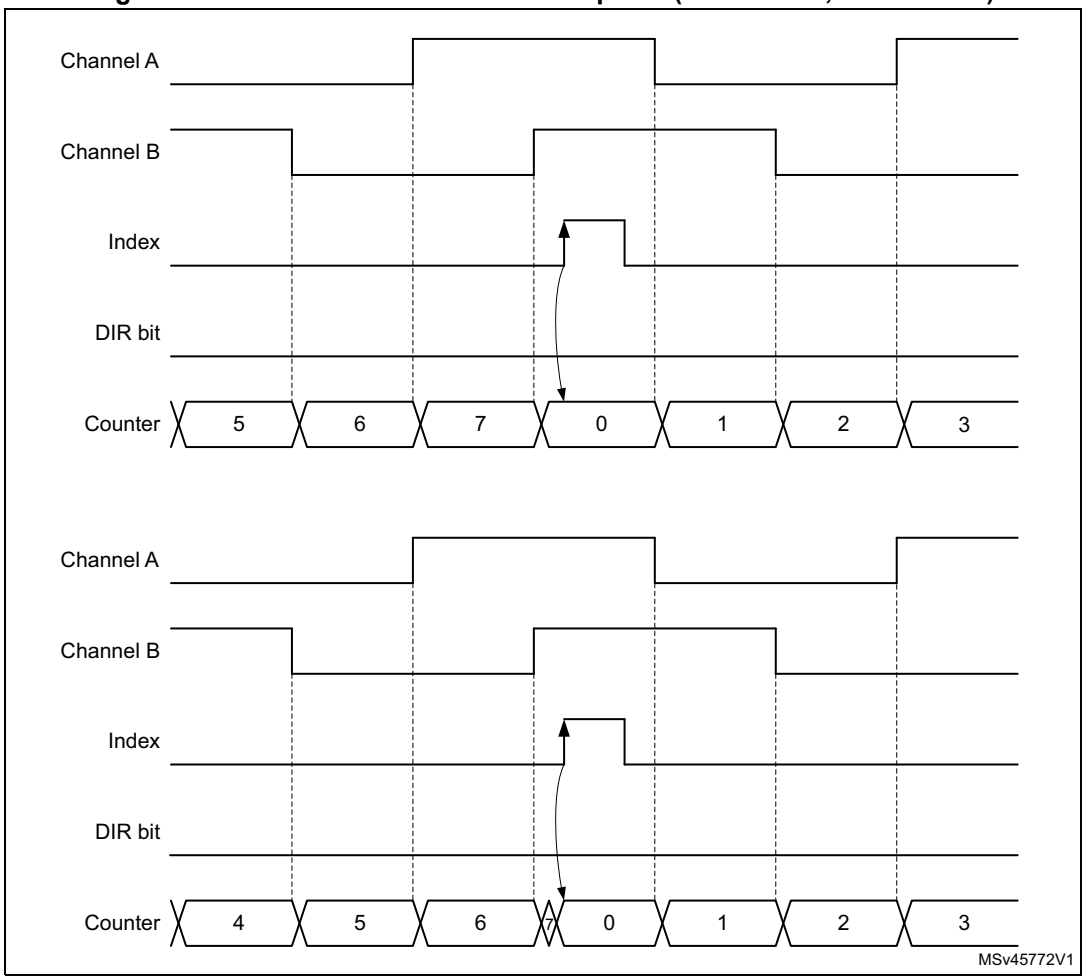
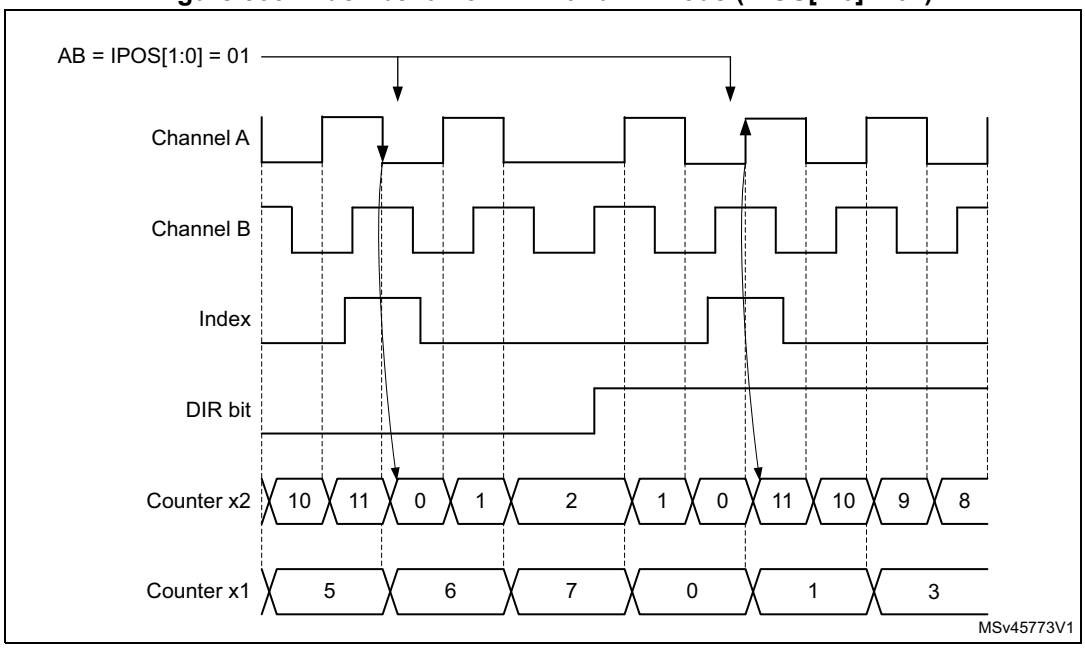


Figure 303 shows how the index is managed in x1 and x2 modes.

Figure 303. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

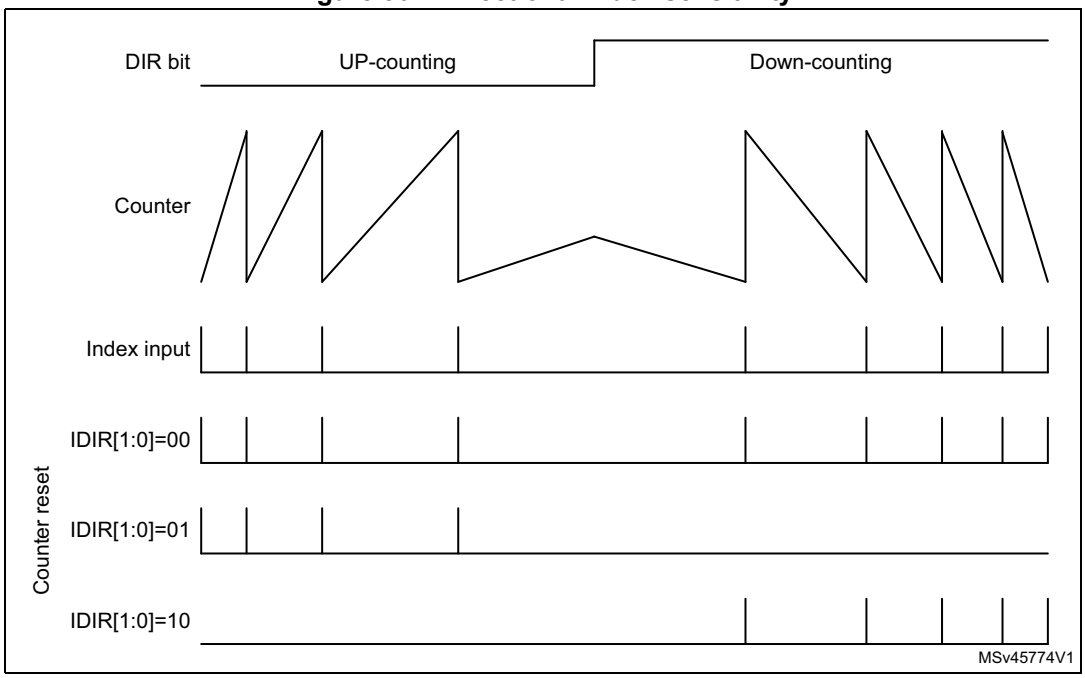


Directional index sensitivity

The IDIR[1:0] bitfield in the TIM_ECR register allows the index to be active only in a selected counting direction.

Figure 304 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

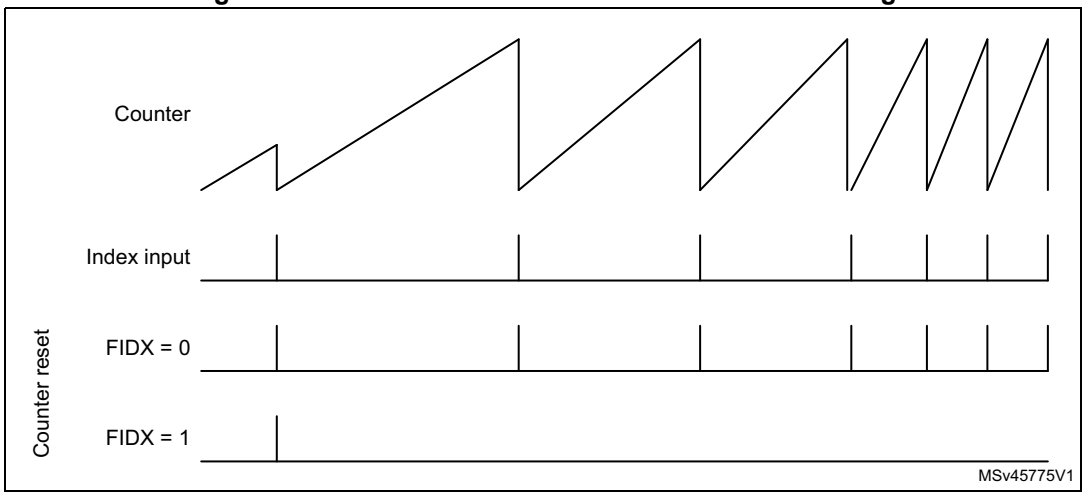
Figure 304. Directional index sensitivity



Special first index event management

The FIDX bit in the TIM_ECR register allows the index to be taken only once, as shown on *Figure 305*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 305. Counter reset as function of FIDX bit setting



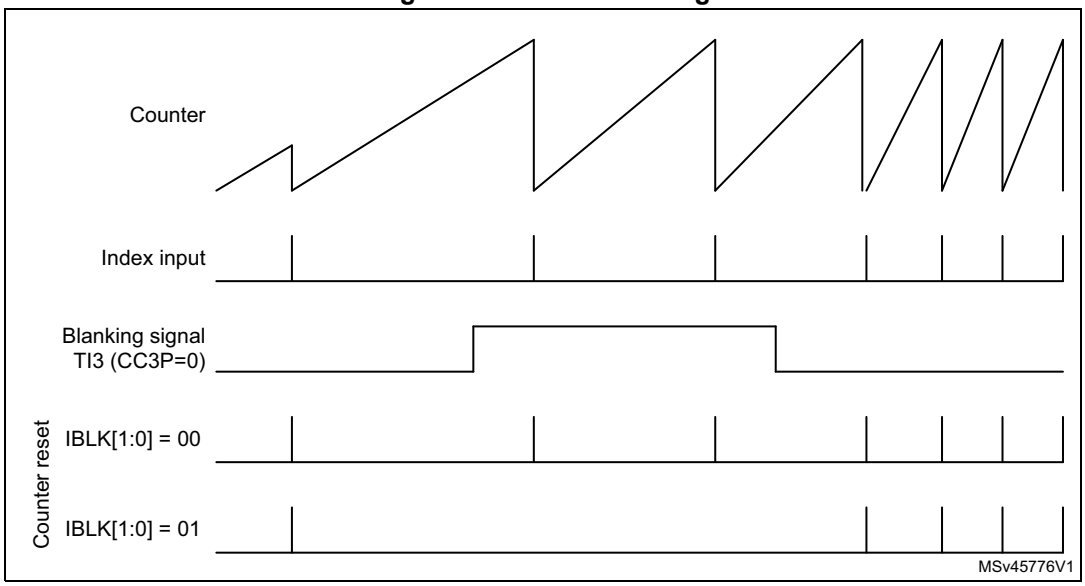
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 306* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIM_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 306. Index blanking



Index management in nonquadrature mode

Figure 307 and *Figure 308* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[4:0] bitfield is equal to 01010, 01011, 01100, 01101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 307. Index behavior in clock + direction mode, IPOS[0] = 1

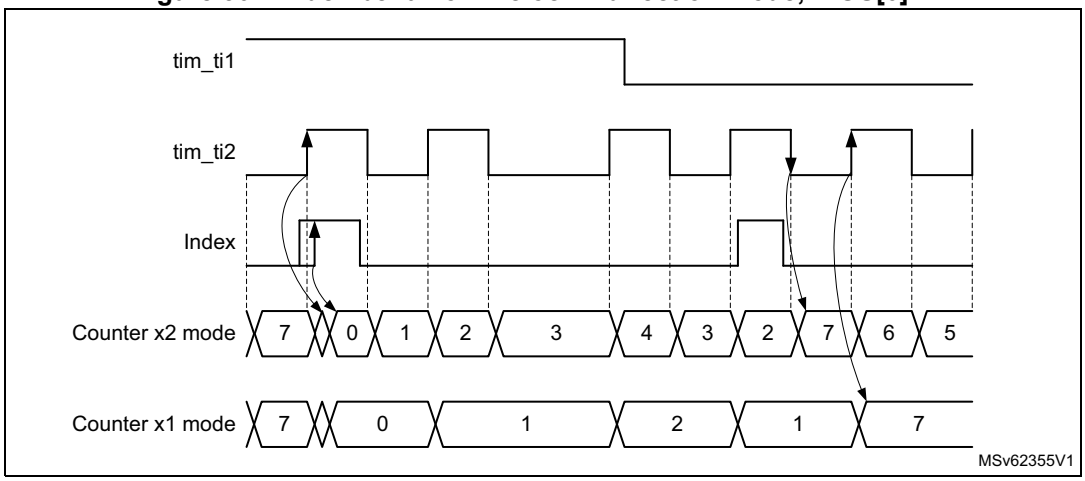
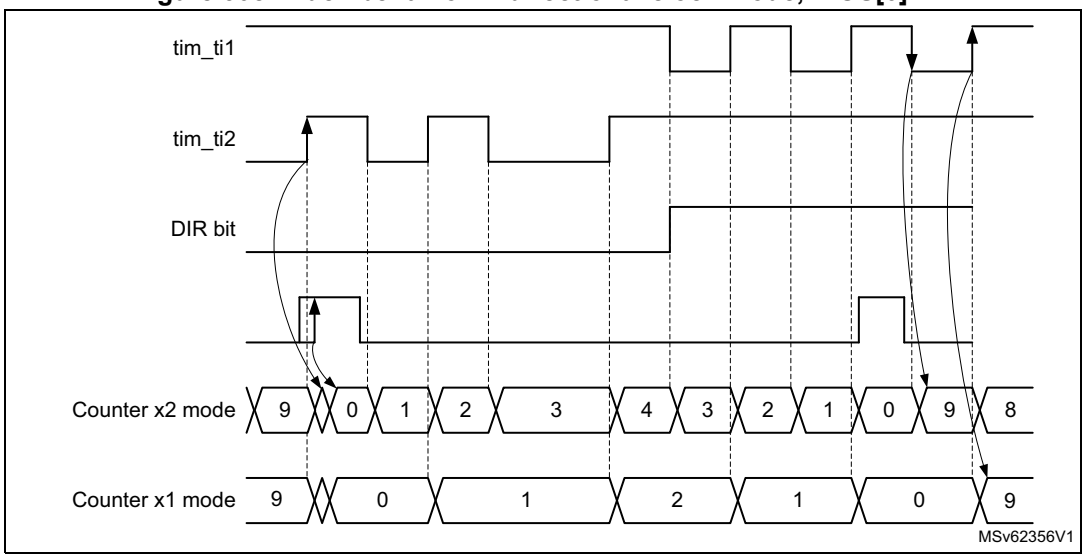


Figure 308. Index behavior in directional clock mode, IPOS[0] = 1

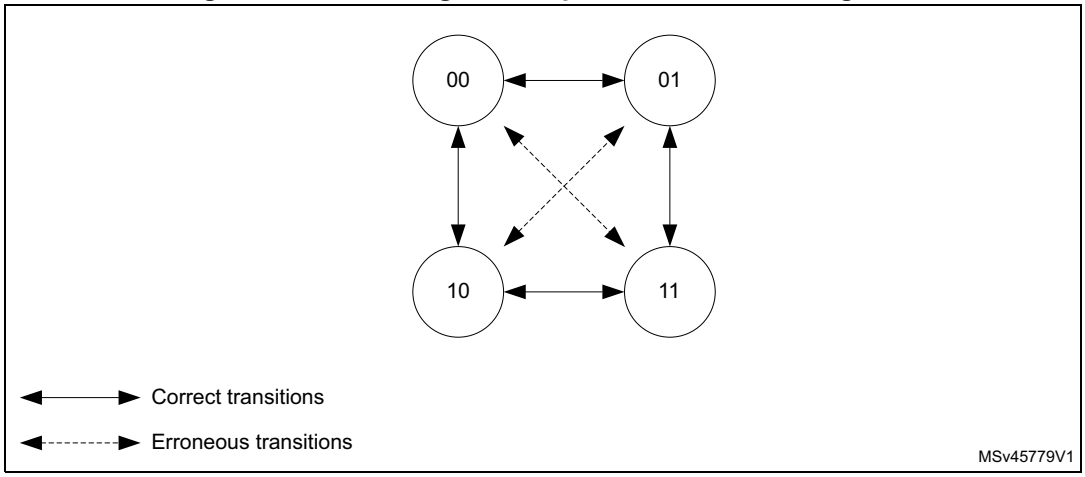


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 309.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status

register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 309. State diagram for quadrature encoded signals



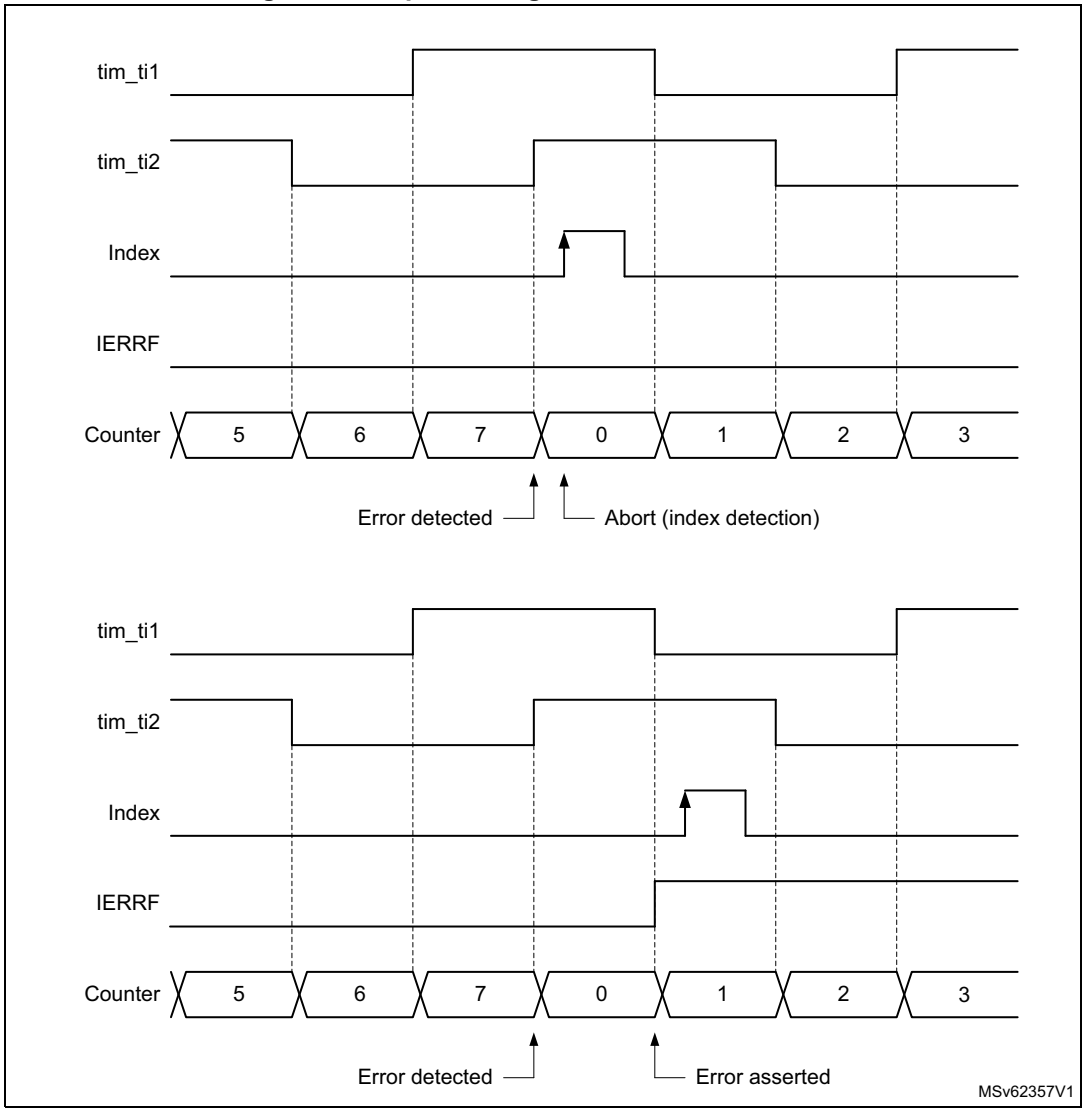
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIM_ARR to 0 or decremented from 0 to TIM_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIM_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIM_ARR = 3999 + 1 = 4000.

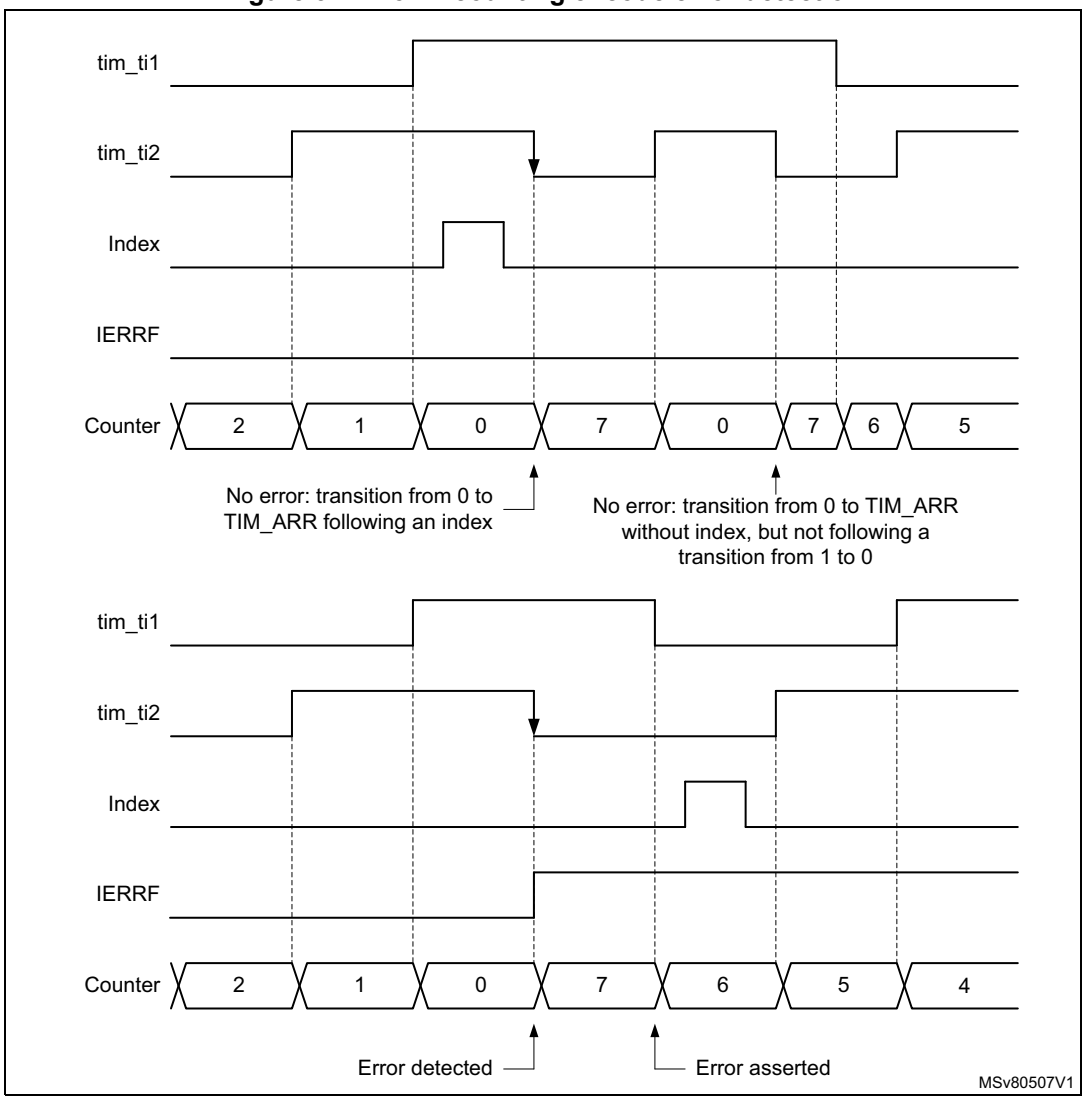
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 310*.

Figure 310. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 311*, to avoid any false error detection in case the encoder dithers between TIM_ARR and 0 immediately after the index detection.

Figure 311. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIM_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIM_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIM_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIM_DIER register.

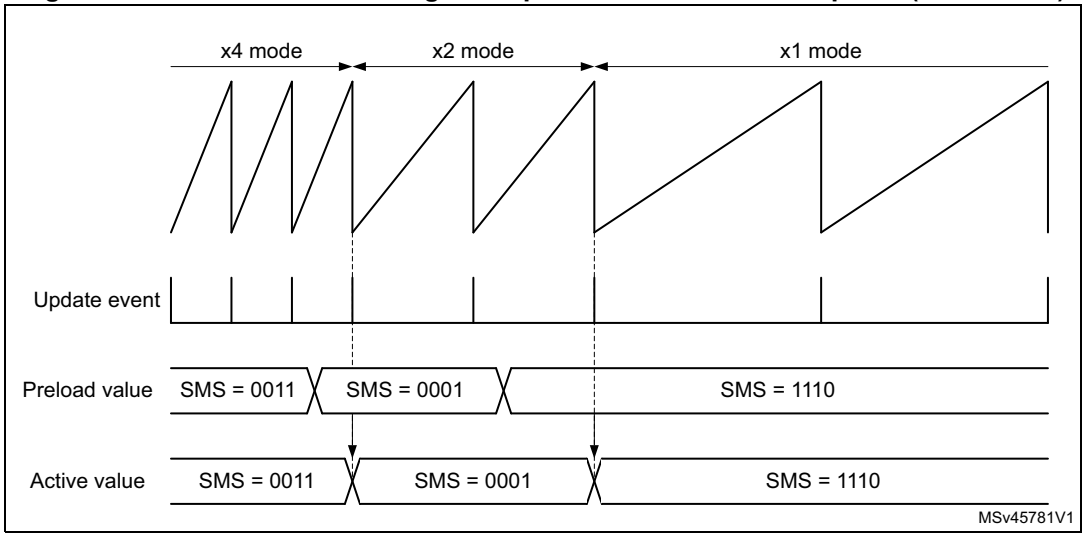
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 312*.

For this purpose, the SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value can be selected with the SMSPS bit in the TIM_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 312. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register.It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

31.3.29 Encoder with built-in debouncer

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 313* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 314* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between 2 positions.

These 2 modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 313. Encoder with built-in debouncer, x2 mode

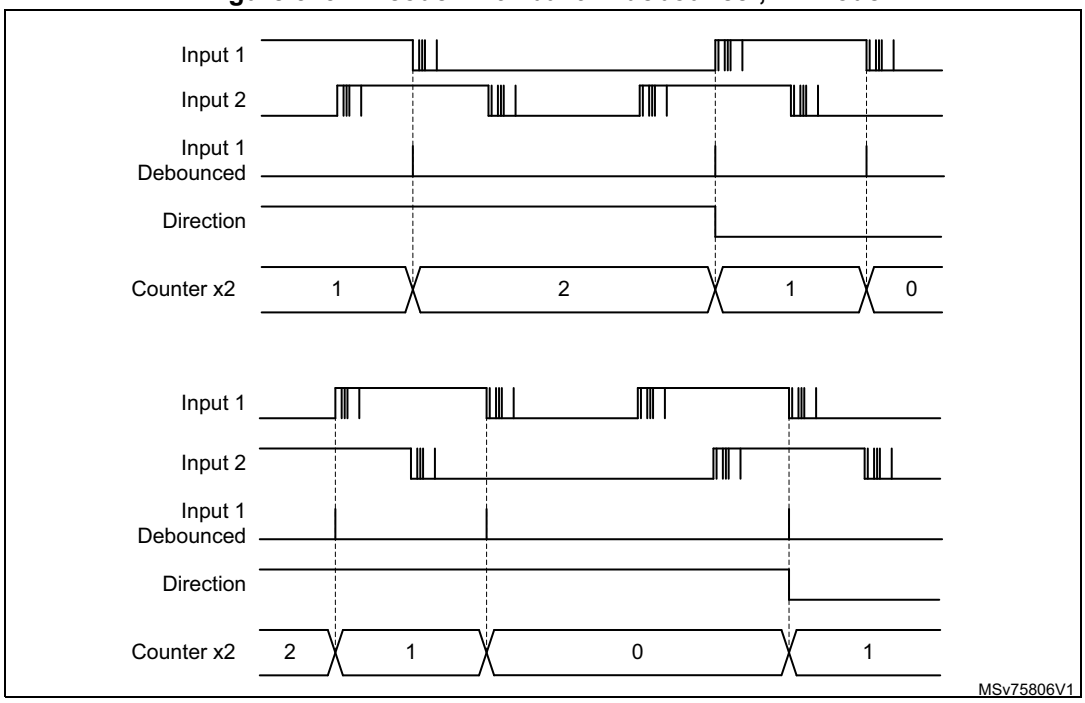
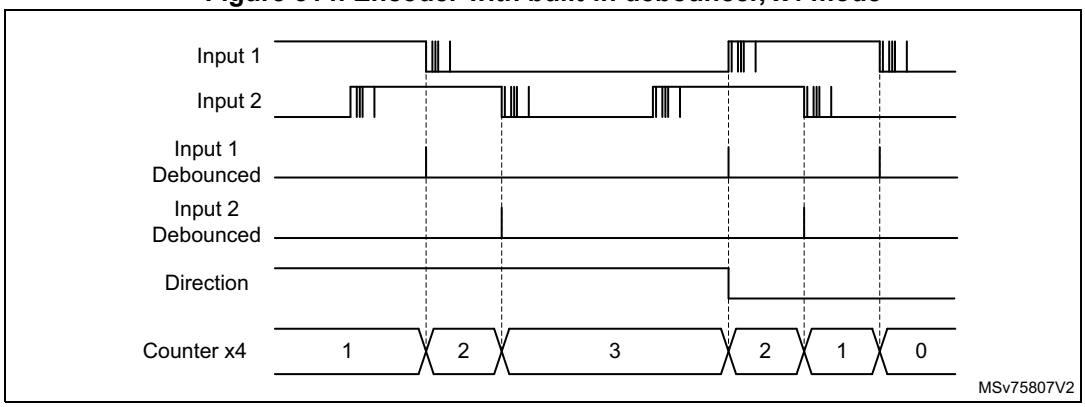


Figure 314. Encoder with built-in debouncer, x4 mode



31.3.30 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIM_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIM_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

31.3.31 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

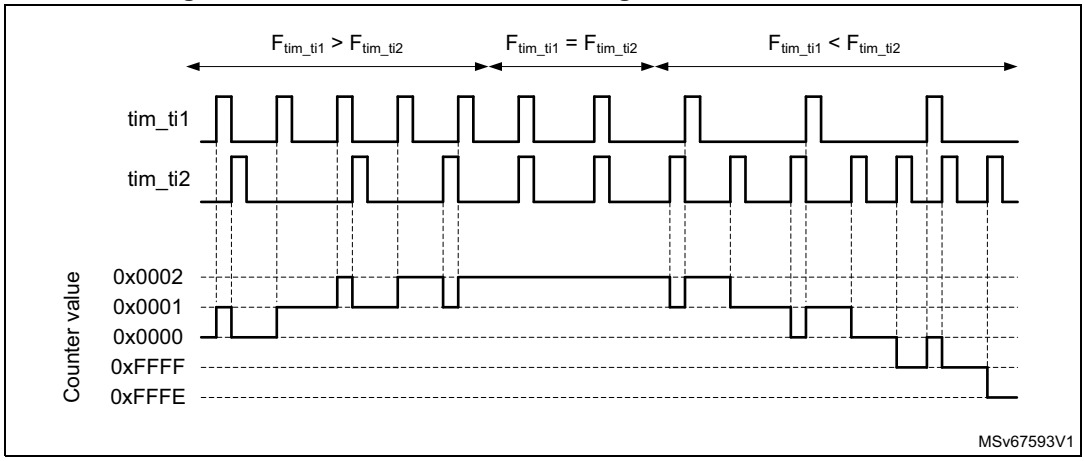
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 315*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value changes as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value dithers between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occuring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 315* below).

Figure 315. Clock drift measure using directional clock mode



31.3.32 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

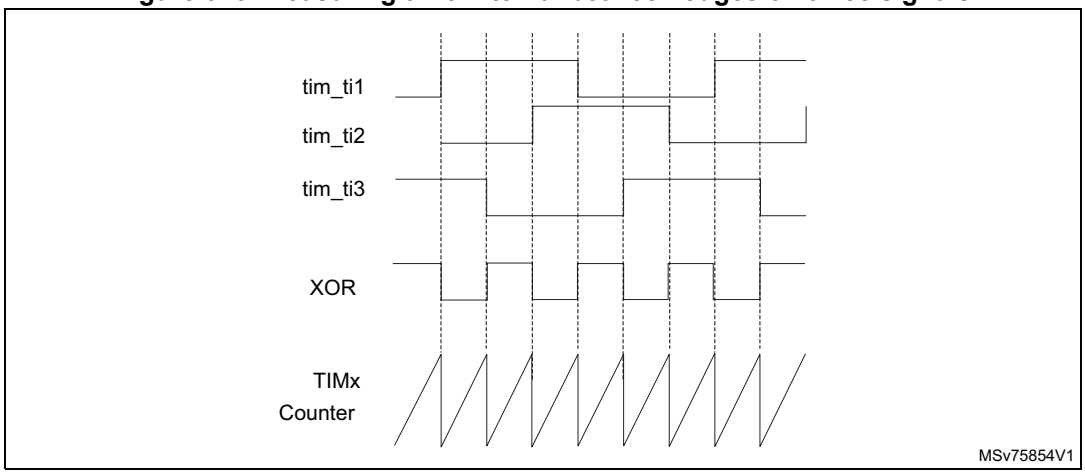
There is no latency between the UIF and UIFCPY flags assertion.

31.3.33 Timer input XOR function

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 316*.

Figure 316. Measuring time interval between edges on three signals



The XORPS control bit selects the position of the XOR gate, as shown on the *Figure 317* and *Figure 318* below, when the TI1S bit is set.

Figure 317. XOR gate connection with TI1S = 1 and XORPS = 0

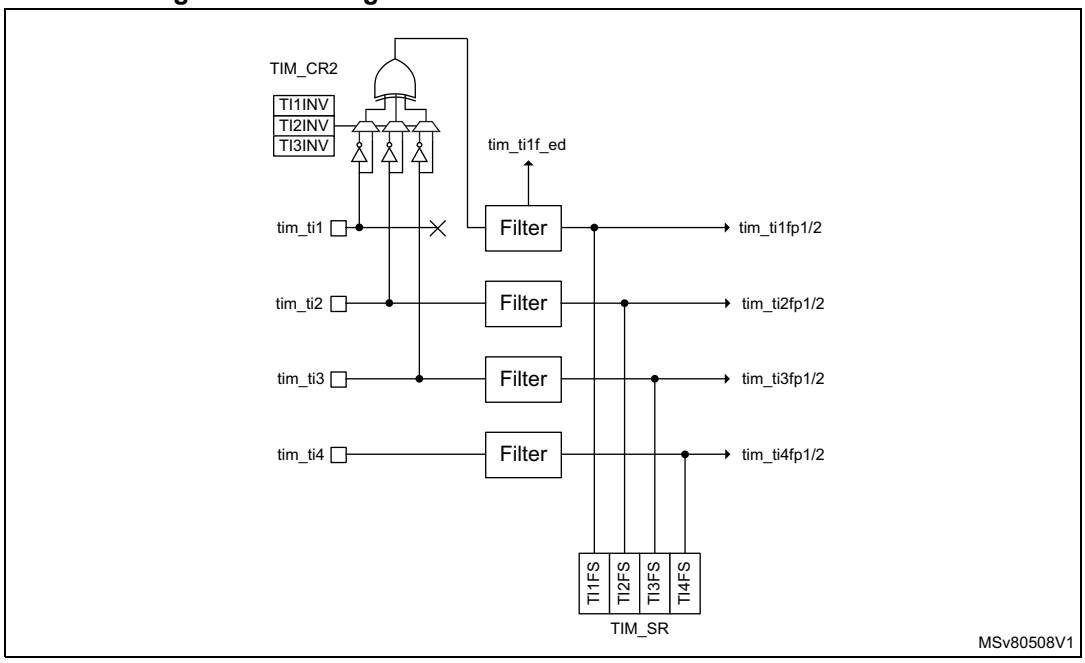
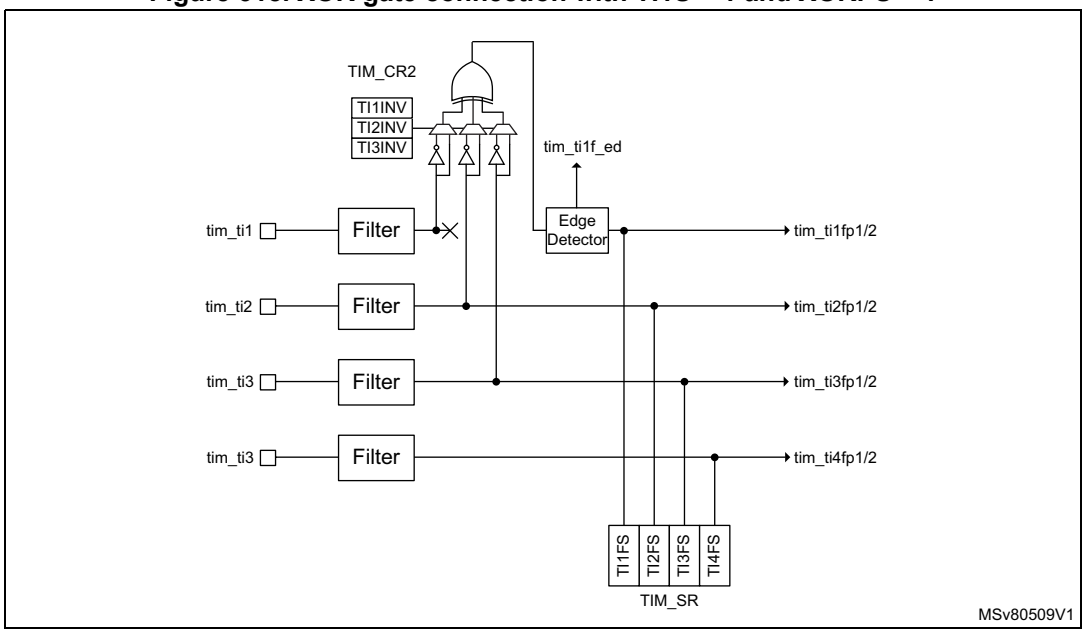


Figure 318. XOR gate connection with TI1S = 1 and XORPS = 1



Three control bits T1INV, T2INV and T3INV in the TIM_CR2 register can invert the tim_in1, tim_in2 and tim_in3 signals prior entering the XOR gate.

The status of the tim_in1, tim_in2, tim_in3 and tim_in4 signals can be software polled using the TI1FS, TI2FS, TI3FS and TI4FS bits in the TIM_SR register (tim_ti4 is not represented on figures above and do not enter into the XOR gate).

31.3.34 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as “interfacing timer” in *Figure 319*. The “interfacing timer” captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIM_CR2 register, with XORPS bit = 0).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 245*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

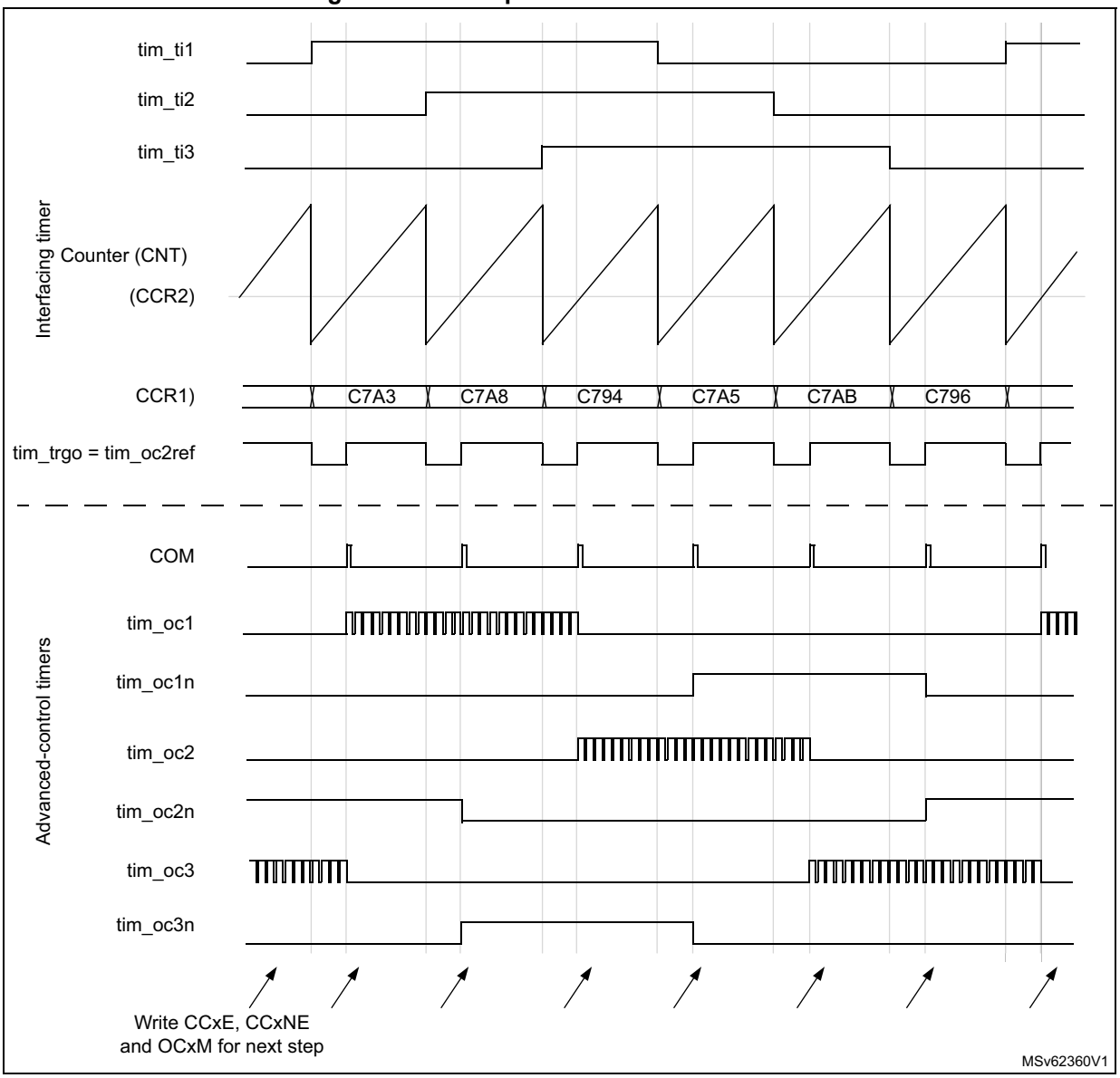
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIM_CR2 register to 1.
- Program the time base: write the TIM_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIM_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIM_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIM_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIM_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIM_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 319 describes this example.

Figure 319. Example of Hall sensor interface



31.3.35 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 32.4.27: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

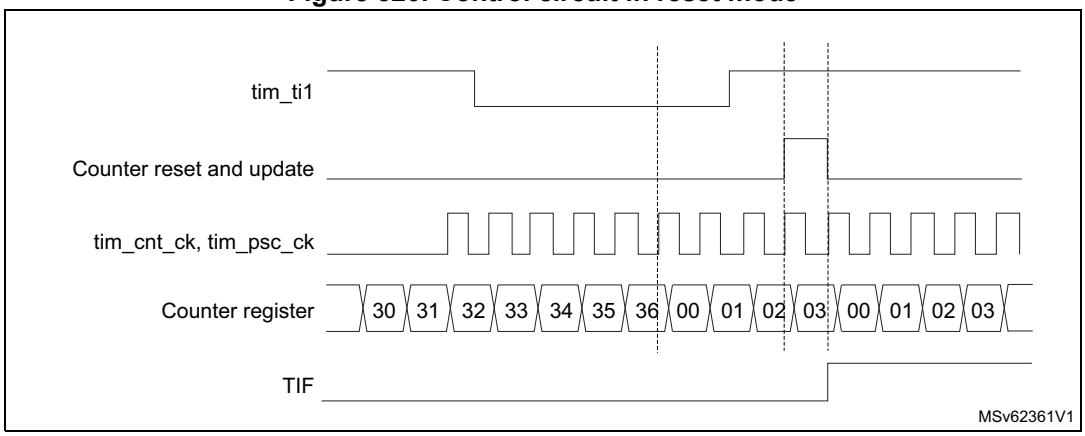
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
- Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 320. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

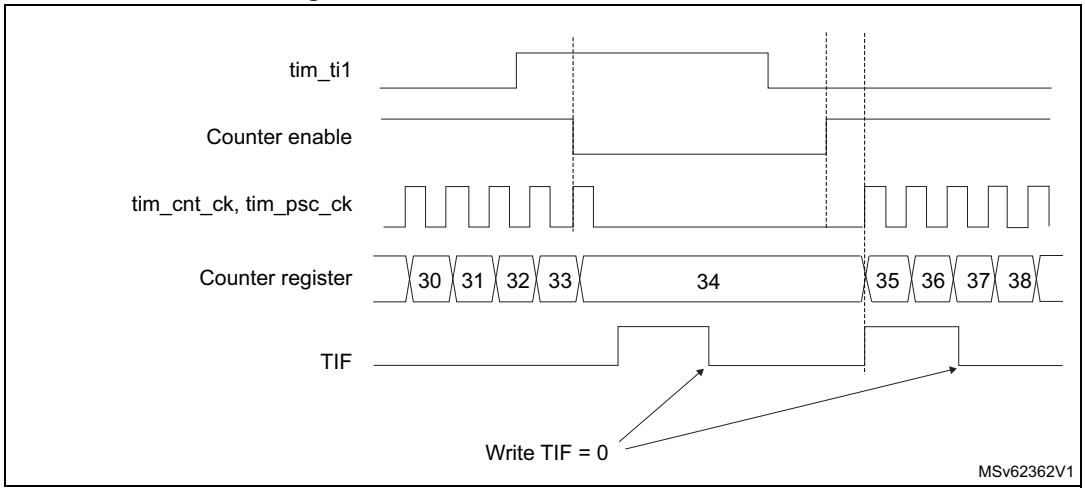
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
- Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 321. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

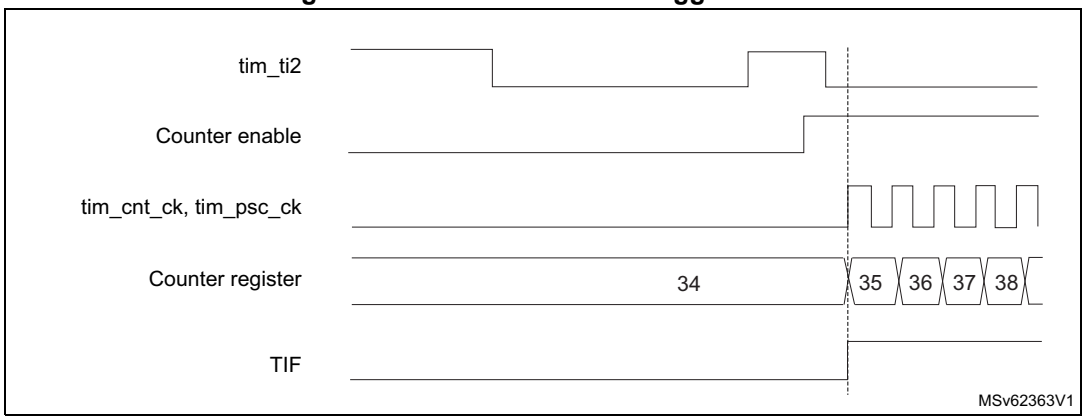
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 322. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIM_SMCR register.

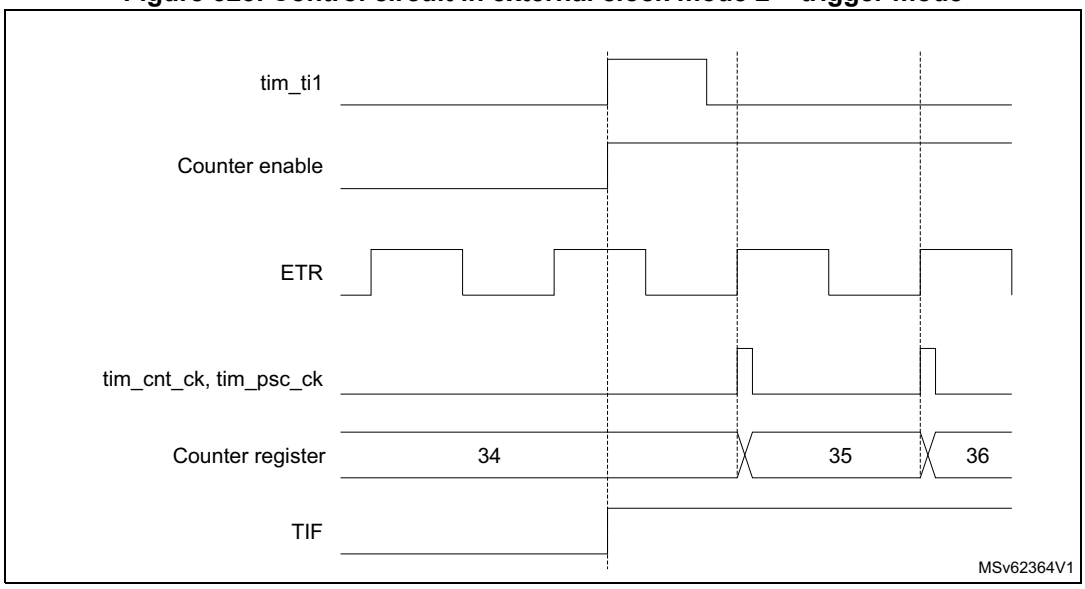
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIM_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIM_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 323. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

31.3.36 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

Refer to MMS2[4:0] bitfield description in TIM_CR2 for the complete list of triggering options.

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 24 possible events, which can be selected using the MMS2[4:0] bits in the TIM_CR2 register.

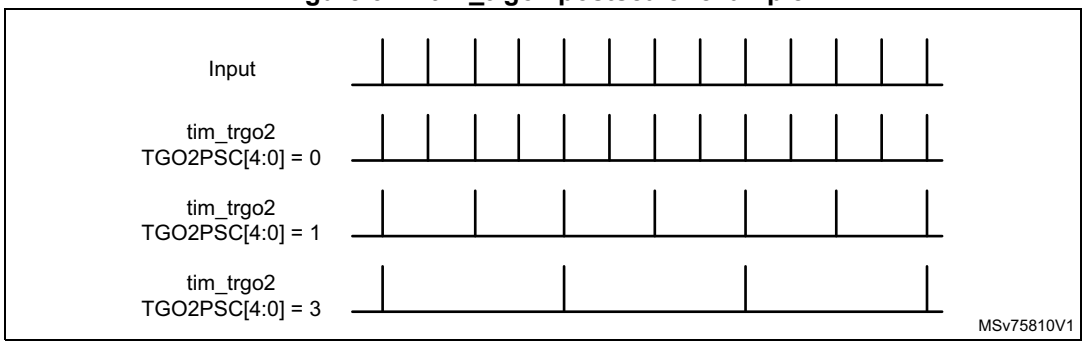
An example of an application for 3-phase motor drives is given in *Figure 263*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

It is possible to reduce the ADC triggering rate with the tim_tro2 post scaler. This is done by periodically discarding tim_trgo2 events, as per the value programmed in TGO2PSC[4:0] bitfield in TIM_CR1, as shown on *Figure 324* below.

Figure 324. tim_trgo2 postscaler example



The post scaler is able to operate with:

– pulses as shown on *Figure 324* above, when selecting for instance MMS2[4:0] = 10001 to have a trigger on tim_oc7refc rising rising or falling edges

– the tim_ocxrefc signal itself (ie with user-defined duty cycle), when selecting for instance MMS2[4:0] = 10000 to have a trigger on tim_oc7refc

31.3.37 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period:

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period:

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

31.3.38 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (in words).

The DBA[4:0] bits in the TIM_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register:

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 31.6.34: TIM DMA control register (TIM_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (see note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 2, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

31.3.39 TIM1/TIM8 DMA requests

The TIM1/TIM8 can generate a DMA request, as shown in the table below.

Table 301. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE

Table 301. DMA request (continued)

DMA request signal	DMA request	Enable control bit
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

31.3.40 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIM_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

31.4 TIM1/TIM8 low-power modes

Table 302. Effect of low-power modes on TIM1/TIM8

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

31.5 TIM1/TIM8 interrupts

The TIM1/TIM8 can generate multiple interrupts, as shown in *Table 303*.

Table 303. Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD		Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC		Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
		Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
		Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
		Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI_COM_DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_IERR	TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
		Break2	B2IF		write 0 in B2IF	Yes	No
		System Break	SBIF		write 0 in SBIF	Yes	No
	TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
	TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

31.6 TIM1/TIM8 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

31.6.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	TGO2PSC[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **TGO2PSC[4:0]**: Trigger output 2 postcaler

The postscaler discards periodically events sent on the tim_trgo2 output, to reduce the ADC triggering rate.
The number of events on tim_trgo2 is divided by (TGO2PSC[4:0] + 1).
When TGO2PSC[4:0] = 0x0, the tim_trgo2 event rate is not affected.
With TGO2PSC[4:0] = 0x1, one event out of 2 is sent on tim_trgo2, with TGO2PSC[4:0] = 0x2, one event out of 3 is sent on tim_trgo2, up to 1 event out of 31 when TGO2PSC[4:0] = 0x1F,...

Note: The postscaler is only applicable when tim_trgo2 transfers a pulse (reset, update, compare pulse). When the tim_trgo2 outputs a level-based information (compare signal or enable bit for gated mode), the postscaler must be set set to 0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Autoreload preload enable
0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.
Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

31.6.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	MMS2[4:0]					OIS7	OIS6	Res.	OIS5
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **TI3INV**: tim_ti3 signal inversion on XOR gate input

0: tim_ti3 non-inverted

1: tim_ti3 inverted

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input

0: tim_ti2 non-inverted

1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input

0: tim_ti1 non-inverted

1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 31.3.37* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection

This bit defines the position of the XOR gate with respect to input filters.

0: XOR gate placed before TI1 filter

1: XOR gate placed after TI1, TI2 and TI3 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:20 **MMS2[4:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

00000:**Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

00001:**Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIM_SMCR register).

00010:**Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

00011:**Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

00100:**Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

00101:**Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

00110:**Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

00111:**Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

01000:**Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

01001:**Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

01010:**Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

01011:**Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

01100:**Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

01101:**Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

01110:**Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

01111:**Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

10000: **Compare** - tim_oc7refc signal is used as trigger output (tim_trgo2)

10001: **Compare Pulse** - tim_oc7refc rising or falling edges generate pulses on tim_trgo2

10010: **Compare Pulse** tim_oc4refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10011: **Compare Pulse** - tim_oc4refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

10100: **Compare Pulse** -tim_oc5refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10101: **Compare Pulse** - tim_oc5refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

10110: **Compare Pulse** - tim_oc6refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10111: **Compare Pulse** - tim_oc6refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 **OIS7**: Output idle state 7 (tim_oc7 output)

Refer to OIS1 bit

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 (after a dead-time) when MOE = 0
1: tim_oc1 = 1 (after a dead-time) when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
This bit enables the insertion of a XOR gate to combine tim_ti1_in[15:0], tim_ti2_in[15:0], and tim_ti3_in[15:0] inputs into tim_ti1.
The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE, CCxP, CCxNP and OCxM bits are not preloaded

1: CCxE, CCxNE, CCxP, CCxNP and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

31.6.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
		rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **SETPS[3:0]**: Synchronous external trigger prescaler

The linear synchronous prescaler divides the frequency of the tim_etrf signal, by a factor (SETPS[3:0] + 1).

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[4:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

Bits 19:18 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 31.3.2: TIM1/TIM8 pins and internal signals* for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 17:16, 2:0 **SMS[4:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001:Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010:Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011:Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111:External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010:Encoder mode: Clock plus direction, x2 mode.

01011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

01100:Encoder mode: Directional Clock, x2 mode.

01101:Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000:Quadrature encoder with built-in debouncer: x2 mode.

10001:Quadrature encoder with built-in debouncer: x4 mode.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

31.6.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

31.6.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	ODS	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	CC7IF	CC6IF	CC5IF
r	r	r	r			r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B2GF	BGF	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 31 **TI4FS**: tim_ti4f status

This flag indicates the value of the tim_ti4f signal (after the digital filtering stage), for polling purpose.
0: tim_ti4f signal is low
1: tim_ti4f signal is high

Bit 30 **TI3FS**: tim_ti3f status

This flag indicates the value of the tim_ti3f signal (after the digital filtering stage), for polling purpose.
0: tim_ti3f signal is low
1: tim_ti3f signal is high

Bit 29 **TI2FS**: tim_ti2f status

This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **ODS**: Output disable status

This bit indicates the global output state when a break or break2 event occured, to discriminate the source.
0: Idle state: break was triggered (or MOE was written to 0), and OIS states are forced on tim_ocx and tim_ocxn outputs
1: Disabled state: break2 was triggered, tim_ocx and tim_ocxn outputs are inactive

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag

This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag

This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag

This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CC7IF**: Compare 7 interrupt flag
Refer to CC1IF description
Note: Channel 7 can only be configured as output.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bit 15 **B2GF**: Break 2 generation flag
This flag is set by hardware as soon as a break 2 event is forced by software, using the B2G bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break 2 event occurred.
1: A break 2 event was forced with the B2G bit

Bit 14 **BGF**: Break generation flag
This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break event occurred.
1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 31.6.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

31.6.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIM_ARR) if DIR = 1 (downcounting).

31.6.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter
Refer to IC1F[3:0] bit description

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler
Refer to IC1PSC[1:0] bit description

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

31.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

0: tim_oc2ref is not affected by the tim_ocref_clr_int signal

1: tim_oc2ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 31.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode

Refer to OC1M[4:0] bit description

Bit 11 **OC2PE**: Output compare 2 preload enable

0: Preload register on TIM_CCR2 disabled. TIM_CCR2 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR2 enabled. Read/Write operations access the preload register. TIM_CCR2 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC2FE**: Output compare 2 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC2 behaves normally depending on counter and CCR2 values even when the trigger is ON. The minimum delay to activate CC2 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC2 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC2 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 31.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000:Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs.This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001:Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010:Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011:Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100:Force inactive level - tim_oc1ref is forced low.

00101:Force active level - tim_oc1ref is forced high.

00110:PWM mode 1 - In upcounting, channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIM_CNT>TIM_CCR1 else active (tim_oc1ref = 1).

00111:PWM mode 2 - In upcounting, channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active. In downcounting, channel 1 is active as long as TIM_CNT>TIM_CCR1 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 296* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 295* for details.

10000-10001:Combined PWM mode 3 and 4. Refer to *Table 296* for details.

10010-11001: Asymmetric PWM mode 3 to 10. Refer to *Table 295* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

31.6.9 TIM capture/compare mode register 2 (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter
Refer to IC3F[3:0] bit description

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC3PSC[1:0] bit description

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter
This bitfield defines the frequency used to sample tim_ti3 input and the length of the digital filter applied to tim_ti3. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
This bitfield defines the ratio of the prescaler acting on CC3 input (tim_ic3). The prescaler is reset as soon as CC3E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

31.6.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

0: tim_oc4ref is not affected by the tim_ocref_clr_int signal

1: tim_oc4ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 31.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC4M[4:0]**: Output compare 4 mode

Refer to OC3M[4:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

0: Preload register on TIM_CCR4 disabled. TIM_CCR4 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR4 enabled. Read/Write operations access the preload register. TIM_CCR4 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC4FE**: Output compare 4 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC4 behaves normally depending on counter and CCR4 values even when the trigger is ON. The minimum delay to activate CC4 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC4 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC4 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

0: tim_oc3ref is not affected by the tim_ocref_clr_int signal

1: tim_oc3ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 31.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC3M[4:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

00000:Frozen - The comparison between the output compare register TIM_CCR3 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00010:Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00011:Toggle - tim_oc3ref toggles when TIM_CNT = TIM_CCR3.

00100:Force inactive level - tim_oc3ref is forced low.

00101:Force active level - tim_oc3ref is forced high.

00110:PWM mode 1 - In upcounting, channel 3 is active as long as TIM_CNT<TIM_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIM_CNT>TIM_CCR3 else active (tim_oc3ref = 1).

00111:PWM mode 2 - In upcounting, channel 3 is inactive as long as TIM_CNT<TIM_CCR3 else active. In downcounting, channel 3 is active as long as TIM_CNT>TIM_CCR3 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010:Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIM_ECR.

01011:Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 296* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 295* for details.

10000-10001:Combined PWM mode 3 and 4. Refer to *Table 296* for details.

10010-11001: Asymmetric PWM mode 3 to 10. Refer to *Table 295* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

0: Preload register on TIM_CCR3 disabled. TIM_CCR3 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR3 enabled. Read/Write operations access the preload register. TIM_CCR3 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC3FE**: Output compare 3 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC3 behaves normally depending on counter and CCR3 values even when the trigger is ON. The minimum delay to activate CC3 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC3 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC3 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

31.6.11 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC7P	CC7E	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
						rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CC7P**: Capture/compare 7 output polarity
Refer to CC1P description

Bit 24 **CC7E**: Capture/compare 7 output enable
Refer to CC1E description

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: In input mode, this bit is preloaded as per CCPC bit in TIM_CR2.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: In input mode, this bit is preloaded as per CCPC bit in TIM_CR2.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 304* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 304. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	0	0	0	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0, tim_ocxn = 0	
		1	0	0	Off-State (output enabled with inactive state) tim_ocx=CCxP, tim_ocxn=CCxNP	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref x or CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered). Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). *Note: tim_brk2 can only be used if OSSI = OSSR = 1.*	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

31.6.12 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in the TIM_CR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

31.6.13 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

31.6.14 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 31.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

31.6.15 TIM repetition counter register (TIM_RCR)

Address offset: 0x030

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

31.6.16 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

31.6.17 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

31.6.18 TIM capture/compare register 3 (TIM_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIM_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

31.6.19 TIM capture/compare register 4 (TIM_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc4 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.
If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIM_CCR4 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

31.6.20 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BK2E bit is set.
Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection, see *Figure 269: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).
Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BK2E bit is set.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM_CCER register).
See OC/OCN enable description for more details (*Section 31.6.11: TIM capture/compare enable register (TIM_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 269: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 31.6.11: TIM capture/compare enable register (TIM_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 31.6.11: TIM capture/compare enable register (TIM_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.
DTG[7:5] = 110 => DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.
DTG[7:5] = 111 => DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

31.6.21 TIM capture/compare register 5 (TIM_CCR5)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	GC5C4	GC5C4O	GC5C3O	GC5C2O	GC5C1O	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3

Distortion on channel 3 output:

0: No effect of tim_oc5ref on tim_oc3refc

1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2

Distortion on channel 2 output:

0: No effect of tim_oc5ref on tim_oc2refc

1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1

Distortion on channel 1 output:

0: No effect of oc5ref on oc1refc

1: oc1refc is the logical AND of oc1ref and oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 28 **GC5C4**: Group channel 5 and channel 4

Distortion on channel 1 output:

0: No effect of oc5ref on oc4refc

1: oc4refc is the logical AND of oc4ref and oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 27 **GC5C4O**: Group channel 5 and channel 4 (ORed)

Distortion on channel 4 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc4refc

1: oc4refc is the logical OR of oc4ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 26 **GC5C3O**: Group channel 5 and channel 3 (ORed)

Distortion on channel 3 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc3refc

1: oc4refc is the logical OR of oc3ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 25 **GC5C2O**: Group channel 5 and channel 2 (ORed)

Distortion on channel 2 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc2refc

1: oc4refc is the logical OR of oc2ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 24 **GC5C1O**: Group channel 5 and channel 1 (ORed)

Distortion on channel 1 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc1refc

1: oc4refc is the logical OR of oc1ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: compare 5 value

CCR5 is the value to be loaded in the actual compare 5 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC5PE). Else the preload value is copied in the active compare 5 register when an update event occurs.

The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc5 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

31.6.22 TIM capture/compare register 6 (TIM_CCR6)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Compare 6 value

CCR6 is the value to be loaded in the actual compare 6 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC6PE). Else the preload value is copied in the active compare 6 register when an update event occurs.

The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc6 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

31.6.23 TIM capture/compare mode register 3 (TIM_CCMR3)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC6M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC5M[4:3]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6CE	OC6M[2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

0: tim_oc6ref is not affected by the tim_ocref_clr_int signal

1: tim_oc6ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 31.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC6M[4:0]**: Output compare 6 mode

Refer to OC5M[4:0] bit description

Bit 11 **OC6PE**: Output compare 6 preload enable

0: Preload register on TIM_CCR6 disabled. TIM_CCR6 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR6 enabled. Read/Write operations access the preload register. TIM_CCR6 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC6FE**: Output compare 6 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC6 behaves normally depending on counter and CCR6 values even when the trigger is ON. The minimum delay to activate CC6 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC6 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC6 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

0: tim_oc5ref is not affected by the tim_ocref_clr_int signal

1: tim_oc5ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 31.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC5M[4:0]**: Output compare 5 mode

These bits define the behavior of the output reference signal tim_oc5ref5. tim_oc5ref is active high whereas tim_oc5 active level depends on CC5P bit.

00000:Frozen - The comparison between the output compare register TIM_CCR5 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 5 to active level on match. tim_oc5ref signal is forced high when the counter TIM_CNT matches the capture/compare register 5 (TIM_CCR5).

00010:Set channel 5 to inactive level on match. tim_oc5ref signal is forced low when the counter TIM_CNT matches the capture/compare register 5 (TIM_CCR5).

00011:Toggle - tim_oc5ref toggles when TIM_CNT = TIM_CCR5.

00100:Force inactive level - tim_oc5ref is forced low.

00101:Force active level - tim_oc5ref is forced high.

00110:PWM mode 1 - In upcounting, channel 5 is active as long as TIM_CNT<TIM_CCR5 else inactive. In downcounting, channel 5 is inactive (tim_oc5ref = 0) as long as TIM_CNT>TIM_CCR5 else active (tim_oc5ref = 1).

00111:PWM mode 2 - In upcounting, channel 5 is inactive as long as TIM_CNT<TIM_CCR5 else active. In downcounting, channel 5 is active as long as TIM_CNT>TIM_CCR5 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010-01011:Reserved.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 296* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 295* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC5PE**: Output compare 5 preload enable

0: Preload register on TIM_CCR5 disabled. TIM_CCR5 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR5 enabled. Read/Write operations access the preload register. TIM_CCR5 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC5FE**: Output compare 5 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC5 behaves normally depending on counter and CCR5 values even when the trigger is ON. The minimum delay to activate CC5 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC5 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC5 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 Reserved, must be kept at reset value.

31.6.24 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx => DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

31.6.25 TIM timer encoder control register (TIM_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[4:0] = 00001, 00010, 00011, 01110, 01111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[4:0] = 01010, 01011, 01100, 01101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input

00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

31.6.26 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIM_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIM_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIM_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIM_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

31.6.27 TIM alternate function option register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCM P20E	BKCM P19E	BKCM P18E	BKCM P17E	BKCM P16E	BKCM P15E	BKCM P14E	BKCM P13E	BKCM P12E	BKCM P11E	BKCM P10E	BKCM P9E	Res.	ETRSEL[4:2]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BKCM P4P	BKCM P3P	BKCM P2P	BKCM P1P	BKINP	BKCM P8E	BKCM P7E	BKCM P6E	BKCM P5E	BKCM P4E	BKCM P3E	BKCM P2E	BKCM P1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable

This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp20 input disabled

1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable
This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp13 input disabled
1: tim_brk_cmp13 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable
This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp12 input disabled
1: tim_brk_cmp12 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable
This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp11 input disabled
1: tim_brk_cmp11 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable
This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp10 input disabled
1: tim_brk_cmp10 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable
This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp9 input disabled
1: tim_brk_cmp9 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **ETRSEL[4:0]**: etr_in source selection
These bits select the etr_in input source.
00000:tim_etr0: TIM_ETR input
00001:tim_etr1
...
11111:tim_etr31
Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation.
Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIM_BKIN input polarity

This bit selects the TIM_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIM_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIM_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIM_BKIN input enable

This bit enables the TIM_BKIN alternate function input for the timer's tim_brk input. TIM_BKIN input is 'ORed' with the other tim_brk sources.

0: TIM_BKIN input disabled

1: TIM_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Refer to Section 31.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

31.6.28 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BK2CMP20E	BK2CMP19E	BK2CMP18E	BK2CMP17E	BK2CMP16E	BK2CMP15E	BK2CMP14E	BK2CMP13E	BK2CMP12E	BK2CMP11E	BK2CMP10E	BK2CMP9E	OCRSEL[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BK2CMP20E**: tim_brk2_cmp20 enable

This bit enables the tim_brk2_cmp20 for the timer's tim_brk2 input. tim_brk2_cmp20 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp20 input disabled

1: tim_brk2_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BK2CMP19E**: tim_brk2_cmp19 enable

This bit enables the tim_brk2_cmp19 for the timer's tim_brk2 input. tim_brk2_cmp19 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp19 input disabled

1: tim_brk2_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BK2CMP18E**: tim_brk2_cmp18 enable

This bit enables the tim_brk2_cmp18 for the timer's tim_brk2 input. tim_brk2_cmp18 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp18 input disabled

1: tim_brk2_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BK2CMP17E**: tim_brk2_cmp17 enable

This bit enables the tim_brk2_cmp17 for the timer's tim_brk2 input. tim_brk2_cmp17 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp17 input disabled

1: tim_brk2_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BK2CMP16E**: tim_brk2_cmp16 enable

This bit enables the tim_brk2_cmp16 for the timer's tim_brk2 input. tim_brk2_cmp16 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp16 input disabled

1: tim_brk2_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BK2CMP15E**: tim_brk2_cmp15 enable

This bit enables the tim_brk2_cmp15 for the timer's tim_brk2 input. tim_brk2_cmp15 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp15 input disabled

1: tim_brk2_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BK2CMP14E**: tim_brk2_cmp14 enable

This bit enables the tim_brk2_cmp14 for the timer's tim_brk2 input. tim_brk2_cmp14 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp14 input disabled

1: tim_brk2_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BK2CMP13E**: tim_brk2_cmp13 enable

This bit enables the tim_brk2_cmp13 for the timer's tim_brk2 input. tim_brk2_cmp13 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp13 input disabled

1: tim_brk2_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BK2CMP12E**: tim_brk2_cmp12 enable

This bit enables the tim_brk2_cmp12 for the timer's tim_brk2 input. tim_brk2_cmp12 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp12 input disabled

1: tim_brk2_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BK2CMP11E**: tim_brk2_cmp11 enable

This bit enables the tim_brk2_cmp11 for the timer's tim_brk2 input. tim_brk2_cmp11 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp11 input disabled

1: tim_brk2_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BK2CMP10E**: tim_brk2_cmp10 enable

This bit enables the tim_brk2_cmp10 for the timer's tim_brk2 input. tim_brk2_cmp10 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp10 input disabled

1: tim_brk2_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BK2CMP9E**: tim_brk2_cmp9 enable

This bit enables the tim_brk2_cmp9 for the timer's tim_brk2 input. tim_brk2_cmp9 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp9 input disabled

1: tim_brk2_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

0000:tim_ocref_clr0

0001:tim_ocref_clr1

...

1111: tim_ocref_clr15

Refer to *Section 31.3.2: TIM1/TIM8 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BK2INP**: TIM_BKIN2 input polarity

This bit selects the TIM_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIM_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIM_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BK2INE**: TIM_BKIN2 input enable

This bit enables the TIM_BKIN2 alternate function input for the timer's tim_brk2 input. TIM_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIM_BKIN2 input disabled

1: TIM_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Refer to Section 31.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

31.6.29 TIM capture/compare register 7 (TIM_CCR7)

Address offset: 0x070

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR7[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR7[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR7[19:0]**: Capture/compare 7 value

CCR7 is the value to be loaded in the actual compare 7 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR4 register (bit OC7PE). Else the preload value is copied in the active compare 7 register when an update event occurs.

The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc7 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR7[15:0]. The CCR7[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR7[19:4]. The CCR7[3:0] bitfield contains the dithered part.

31.6.30 TIM capture/compare mode register 4 (TIM_CCMR4)

Address offset: 0x078

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channel 7 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7M[4:3]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7CE	OC7M[2:0]			OC7PE	OC7FE	Res.	Res.
								rw	rw	rw	rw	rw	rw		

Bits 31:18 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC7CE:** Output compare 7 clear enable

0: tim_oc7ref is not affected by the tim_ocref_clr_int signal

1: tim_oc7ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 31.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC7M[4:0]**: Output compare 7 mode

These bits define the behavior of the output reference signal tim_ocref7. tim_oc7ref is active high whereas tim_oc7 active level depends on CC7P bit.

00000:Frozen - The comparison between the output compare register TIM_CCR7 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 7 to active level on match. tim_oc7ref signal is forced high when the counter TIM_CNT matches the capture/compare register 7 (TIM_CCR7).

00010:Set channel 7 to inactive level on match. tim_oc7ref signal is forced low when the counter TIM_CNT matches the capture/compare register 7 (TIM_CCR7).

00011:Toggle - tim_oc7ref toggles when TIM_CNT = TIM_CCR7.

00100:Force inactive level - tim_oc7ref is forced low.

00101:Force active level - tim_oc7ref is forced high.

00110:PWM mode 1 - In upcounting, channel 7 is active as long as TIM_CNT<TIM_CCR7 else inactive. In downcounting, channel 7 is inactive (tim_oc7ref = 0) as long as TIM_CNT>TIM_CCR7 else active (tim_oc5ref = 1).

00111:PWM mode 2 - In upcounting, channel 7 is inactive as long as TIM_CNT<TIM_CCR7 else active. In downcounting, channel 7 is active as long as TIM_CNT>TIM_CCR7 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC7PE**: Output compare 7 preload enable

0: Preload register on TIM_CCR7 disabled. TIM_CCR7 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR7 enabled. Read/Write operations access the preload register. TIM_CCR7 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC7FE**: Output compare 7 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC7 behaves normally depending on counter and CCR7 values even when the trigger is ON. The minimum delay to activate CC7 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC7 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC7 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 Reserved, must be kept at reset value.

31.6.31 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BK4NM[1:0]		BK4M[1:0]		BK3NM[1:0]		BK3M[1:0]		BK2NM[1:0]		BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **BK4NM[1:0]**: Break mode for tim_oc4n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 13:12 **BK4M[1:0]**: Break mode for tim_oc4 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 11:10 **BK3NM[1:0]**: Break mode for tim_oc3n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **BK3M[1:0]**: Break mode for tim_oc3 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 7:6 **BK2NM[1:0]**: Break mode for tim_oc2n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 5:4 **BK2M[1:0]**: Break mode for tim_oc2 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

31.6.32 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for t_{DTG} selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK2[6:0] if DBK2[7] = 0
- DBK2[5:0] if DBK2[7:6] = 10
- DBK2[4:0] if DBK2[7:5] = 110 or 111

DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for t_{DTG} selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK1[6:0] if DBK1[7] = 0
- DBK1[5:0] if DBK1[7:6] = 10
- DBK1[4:0] if DBK1[7:5] = 110 or 111

DBK1[7:5]=0xx => Delay1=DBK1[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK1[7:5]=10x => Delay1=(64+DBK1[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK1[7:5]=110 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK1[7:5]=111 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

31.6.33 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	OOS4N	OOS4	OOS3N	OOS3	OOS2N	OOS2	OOS1N	OOS1
			rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control
This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.
0: tim_ocx and tim_ocxn outputs are in the default idle state
1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits
Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **OOS4N**: Output override state for output tim_oc4n
This bit defines the state for output tim_oc4n when OOC = 1
0: tim_oc4n = 0
1: tim_oc4n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **OOS4**: Output override state for output tim_oc4
This bit defines the state for output tim_oc4 when OOC = 1
0: tim_oc4 = 0
1: tim_oc4 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **OOS3N**: Output override state for output tim_oc3n
This bit defines the state for output tim_oc3n when OOC = 1
0: tim_oc3n = 0
1: tim_oc3n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **OOS3**: Output override state for output tim_oc3
This bit defines the state for output tim_oc3 when OOC = 1
0: tim_oc3 = 0
1: tim_oc3 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **OOS2N**: Output override state for output tim_oc2n
This bit defines the state for output tim_oc2n when OOC = 1
0: tim_oc2n = 0
1: tim_oc2n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **OOS2**: Output override state for output tim_oc2
This bit defines the state for output tim_oc2 when OOC = 1
0: tim_oc2 = 0
1: tim_oc2 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **OOS1N**: Output override state for output tim_oc1n
This bit defines the state for output tim_oc1n when OOC = 1
0: tim_oc1n = 0
1: tim_oc1n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1
This bit defines the state for output tim_oc1 when OOC = 1
0: tim_oc1 = 0
1: tim_oc1 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

31.6.34 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).
0000: Disabled
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers.

00000: 1 transfer

00001: 2 transfers

00010: 3 transfers

...

11111: 32 transfers

Example: Let us consider the following transfer: DBL = 6 & DBA = TIM2_CR1.

–If DBL = 6 and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIM_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIM_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIM_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

...

31.6.35 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

31.6.36 TIM1/TIM8 register map

TIM1/TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

Table 305. TIM1/TIM8 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIM_CR1	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	TGO2PSC[4:0]					Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0				0	0	0	0	0				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIM_CR2	TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	MMS2[4:0]				OIS7	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	TIM_SMCR	Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS[4:3]		ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value			0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TIM_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	TIM_SR	TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	ODS	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	CC7IF	CC6IF	CC5IF	B2GF	BGF	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	TIM_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x018	TIM_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 305. TIM1/TIM8 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	TIM_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]		OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIM_CCER	Res.	Res.	Res.	Res.	Res.	Res.	CC7P	CC7E	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value							0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TIM_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIM_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	TIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIM_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIM_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIM_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIM_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	TIM_CCR5	GC5C3	GC5C2	GC5C1	GC5C4	GC5C4O	GC5C3O	GC5C2O	GC5C1O	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 305. TIM1/TIM8 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04C	TIM_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIM_CCMR3	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC5M[4:3]		OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value							0	0							0	0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	TIM_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	TIM_ECR	Res.	Res.	Res.	Res.	Res.	PWPR SC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIM_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIM_AF1	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	ETRSEL [4:0]					BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	TIM_AF2	BK2CMP20E	BK2CMP19E	BK2CMP18E	BK2CMP17E	BK2CMP16E	BK2CMP15E	BK2CMP14E	BK2CMP13E	BK2CMP12E	BK2CMP11E	BK2CMP10E	BK2CMP9E	OCRSEL[3:0]				Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x06C	Reserved	Res.																															
0x070	TIM_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR719:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	Reserved	Res.																															
0x078	TIM_CCMR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7CE	OC7M [2:0]			OC7PE	OC7FE	Res.	Res.
	Reset value															0	0									0	0	0	0	0	0		
0x07C.. 0x8C	Reserved	Res.																															
0x090	TIM_MPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK4NM[1:0]		BK4M[1:0]		BK3NM[1:0]		BK3M[1:0]		BK2NM[1:0]		BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	TIM_MPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0

Table 305. TIM1/TIM8 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x098.. 0x09C	Reserved	Res.																															
0x0A0	TIM_OOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	OOS4N	OOS4	OOS3N	OOS3	OOS2N	OOS2	OOS1N	OOS1
	Reset value																				0					0	0	0	0	0	0	0	0
0x0A4.. 0x3D8	Reserved	Res.																															
0x3DC	TIM_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIM_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

32 General-purpose timers(TIM2/TIM3/TIM4/TIM5)

32.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 32.4.27: Timer synchronization*.

32.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.
- ADC synchronization for jitter-free sampling points.

32.3 TIM2/TIM3/TIM4/TIM5 implementation

Table 306. Timer instance implementation on STM32C5

Timer instance	STM32C53x/C54x	STM32C55x/C56x	STM32C59x/C5Ax
TIM2	X	X	X
TIM3	-	-	X
TIM4	-	-	X
TIM5	X	X	X

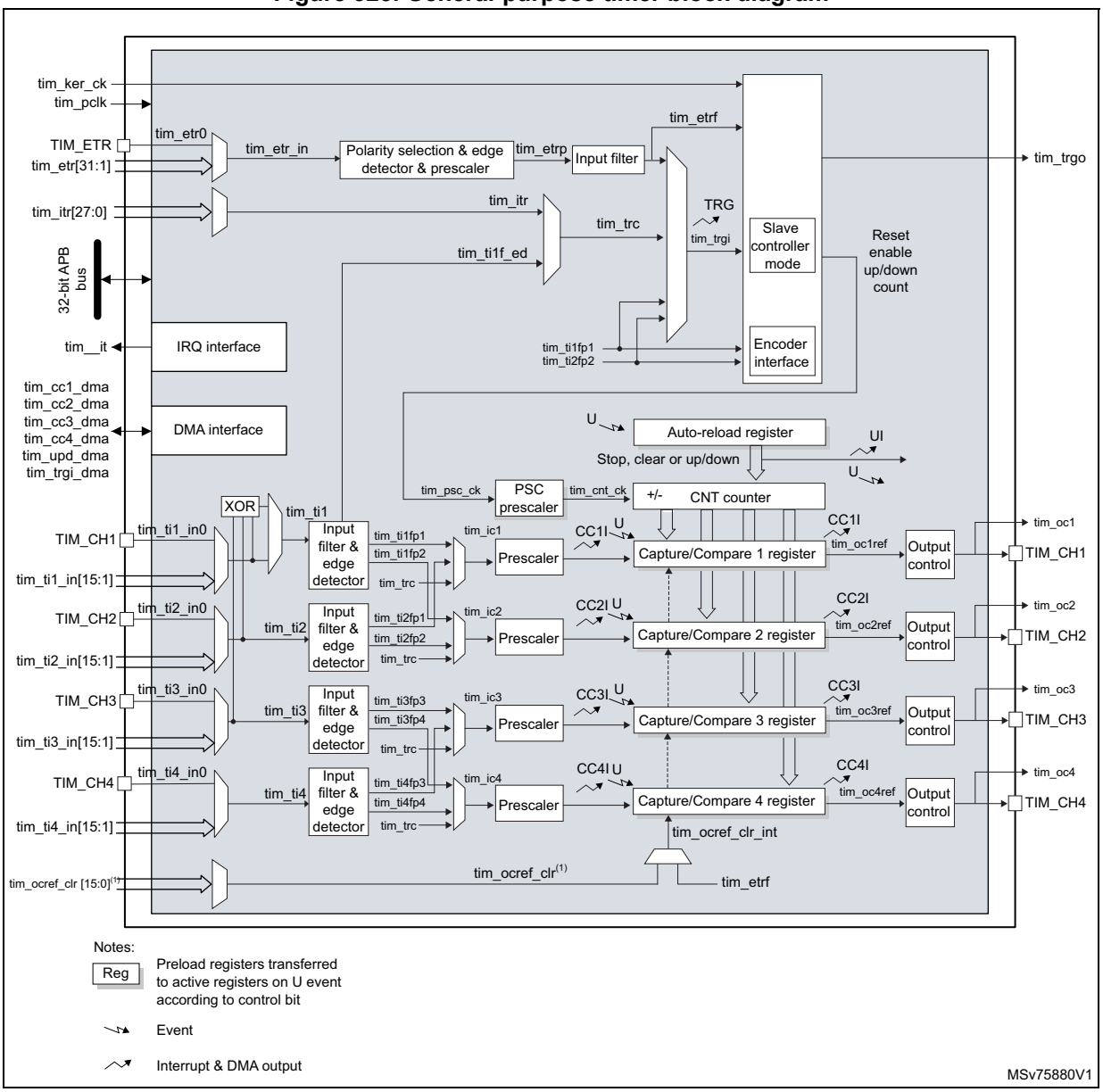
Table 307. STM32C5 general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	32-bit	32-bit	32-bit
OCREF clear selection	Yes	Yes	Yes	Yes
Sources	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr

32.4 TIM2/TIM3/TIM4/TIM5 functional description

32.4.1 Block diagram

Figure 325. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 32.3: TIM2/TIM3/TIM4/TIM5 implementation*.

32.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 308 and *Table 309* in this section summarize the TIM inputs and outputs.

Table 308. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 309. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[31:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 309. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.

Tables below list the sources connected to the tim_ti[4:1] input multiplexers.

Table 310. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	comp1_out			
tim_ti1_in2	comp2_out(1)/ eth_ptp_pps_o(2)	eth_ptp_pps_o	Reserved	
tim_ti1_in3	LSI	Reserved		
tim_ti1_in4	LSE	Reserved		
tim_ti1_in5	rtc_wut_trg	Reserved		
tim_ti1_in6	TIM5_CH1	Reserved		
tim_ti1_in7	fdcan1_rxeof_evt	fdcan2_rxeof_evt	Reserved	
tim_ti1_in[15:8]	Reserved			

1. comp2_out is only available on STM32C53x/542 devices.
2. eth_ptp_pps_o is only available on STM32C59x/5A3 devices.

Table 311. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in1	Reserved			
tim_ti2_in2	Reserved			
tim_ti2_in3	hse_1M_ck	Reserved		
tim_ti2_in4	MCO1	Reserved		
tim_ti2_in5	MCO2	Reserved		

Table 311. Interconnect to the tim_ti2 input multiplexer (continued)

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in6	Reserved			
tim_ti2_in7	fdcan1_txeof_evt	fdcan2_txeof_evt	Reserved	
tim_ti2_in[15:8]	Reserved			

Table 312. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in1	Reserved			
tim_ti3_in[6:2]	Reserved			
tim_ti3_in7	fdcan2_rxeof_evt	Reserved		
tim_ti3_in[15:8]	Reserved			

Table 313. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in1	comp1_out	Reserved		
tim_ti4_in2	comp2_out	Reserved		
tim_ti4_in[6:3]	Reserved			
tim_ti4_in7	fdcan2_txeof_evt	Reserved		
tim_ti4_in[15:8]	Reserved			

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 314. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo			
tim_itr1	tim1_trgo2			
tim_itr2	Reserved	tim2_trgo		
tim_itr3	tim3_trgo	Reserved	tim3_trgo	
tim_itr4	tim4_trgo		Reserved	tim4_trgo
tim_itr5	tim5_trgo			Reserved

Table 314. TIMx internal trigger connection (continued)

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr6	tim8_trgo			
tim_itr7	tim8_trgo2			
tim_itr8	tim12_trgo			
tim_itr9	tim15_trgo			
tim_itr10	tim16_oc1			
tim_itr11	tim17_oc1			
tim_itr[27:12]	Reserved			

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 315. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	comp1_out			
tim_etr2	comp2_out	Reserved		
tim_etr3	adc1_awd1	adc3_awd1		
tim_etr4	adc1_awd2	adc3_awd2		
tim_etr5	adc1_awd3	adc3_awd3		
tim_etr6	LSE	Reserved		
tim_etr7	MCO1	Reserved		
tim_etr8	Reserved	TIM2_ETR		
tim_etr9	TIM3_ETR	Reserved	TIM3_ETR	
tim_etr10	TIM4_ETR		Reserved	TIM4_ETR
tim_etr11	TIM5_ETR			Reserved
tim_etr[31:12]	Reserved			

The table below lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 316. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr0	comp1_out			
tim_ocref_clr1	comp2_out	Reserved		
tim_ocref_clr[15:2]	Reserved			

32.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 326 and *Figure 327* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 326. Counter timing diagram with prescaler division change from 1 to 2

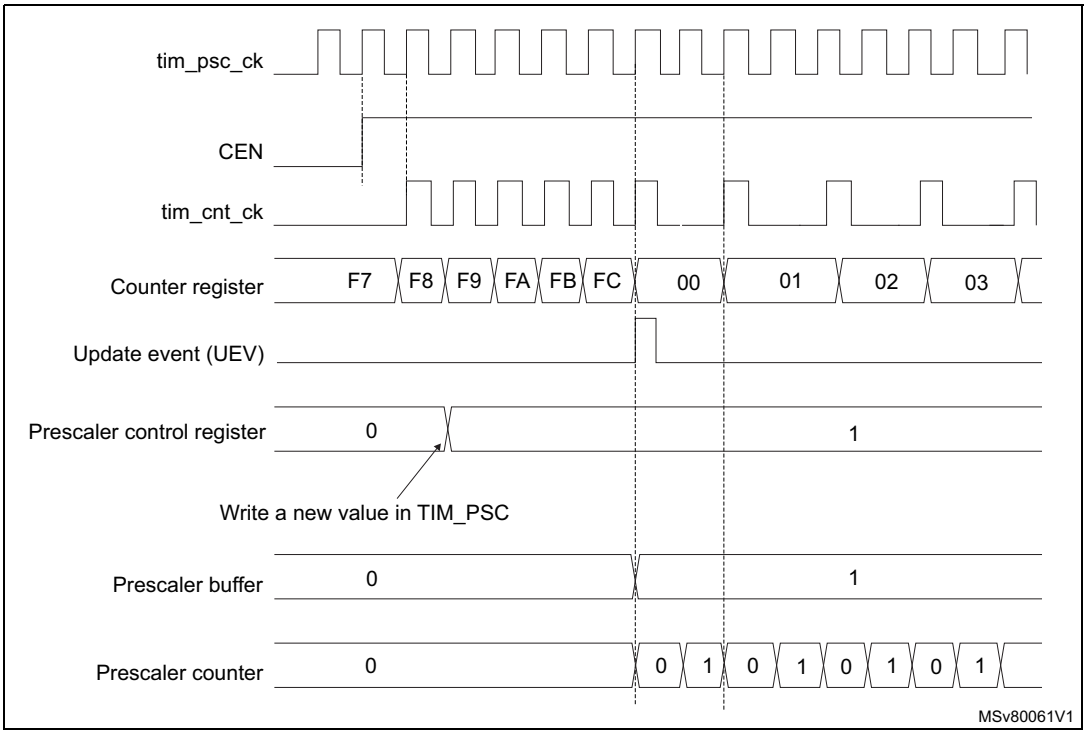
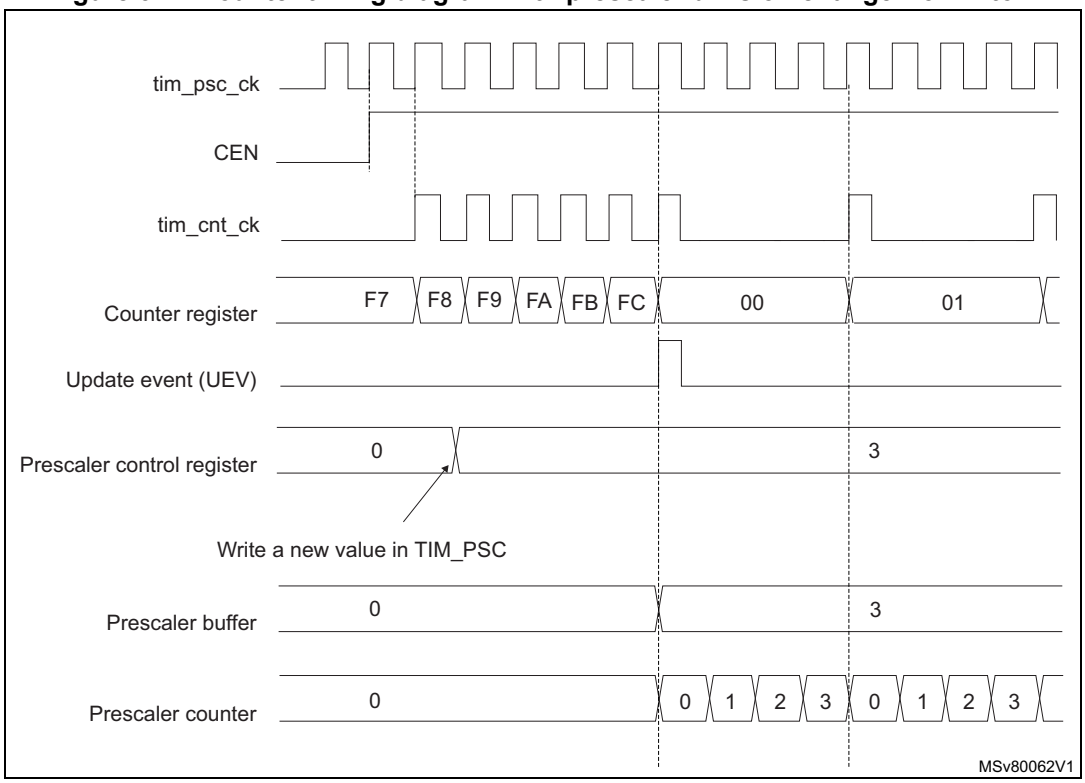


Figure 327. Counter timing diagram with prescaler division change from 1 to 4



32.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload shadow register is updated with the preload value (TIM_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 328. Counter timing diagram, internal clock divided by 1

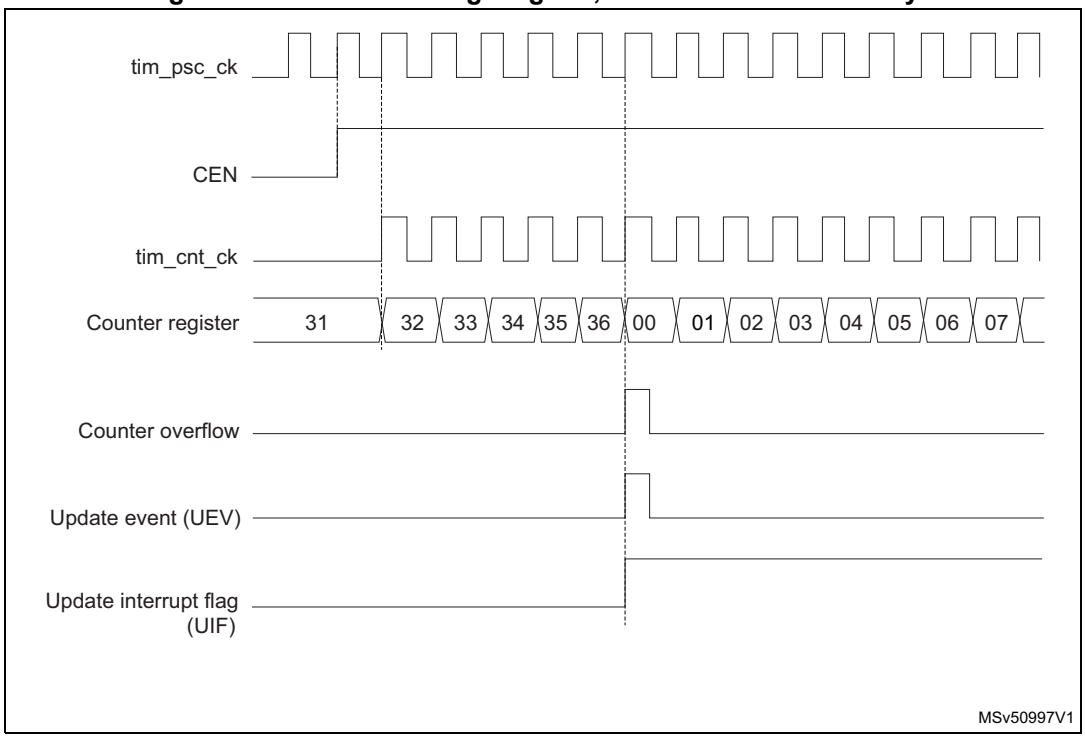


Figure 329. Counter timing diagram, internal clock divided by 2

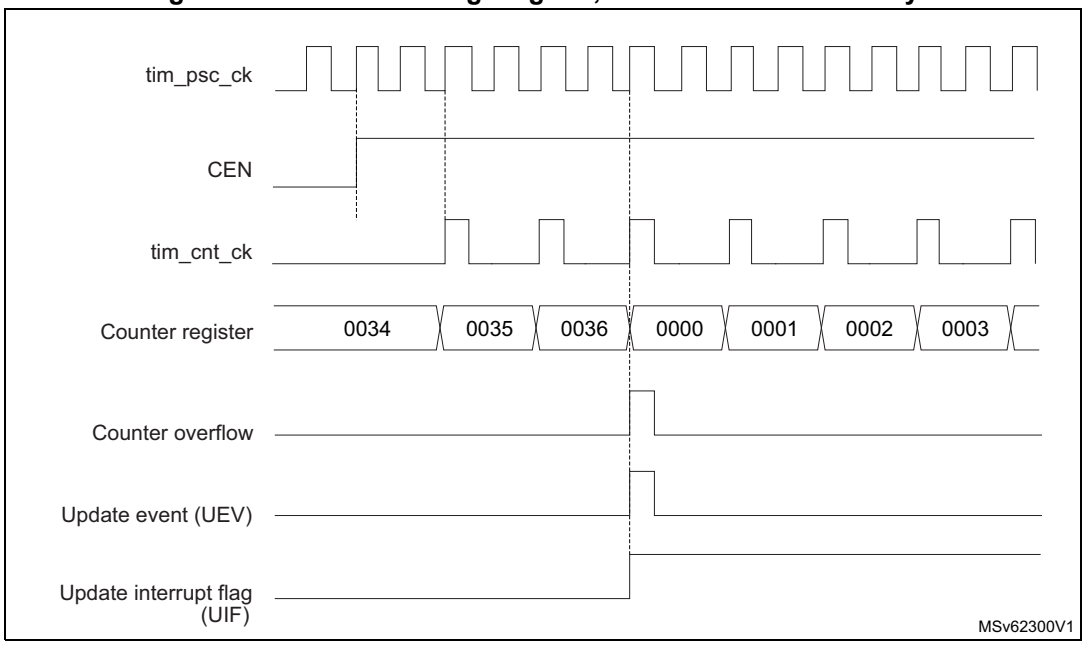


Figure 330. Counter timing diagram, internal clock divided by 4

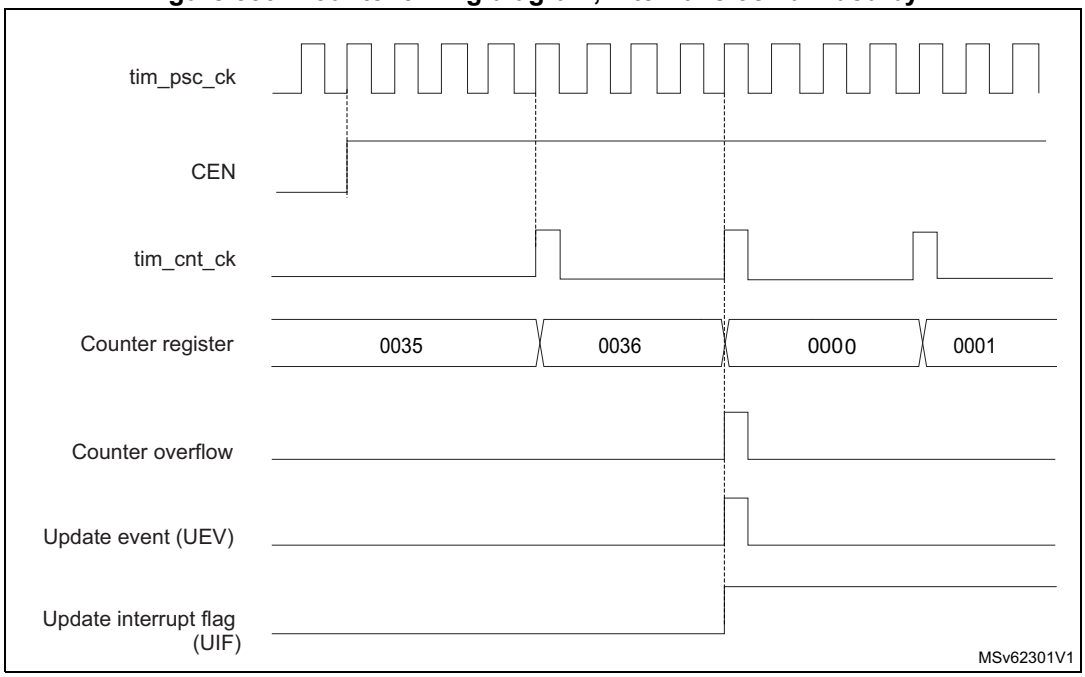


Figure 331. Counter timing diagram, internal clock divided by N

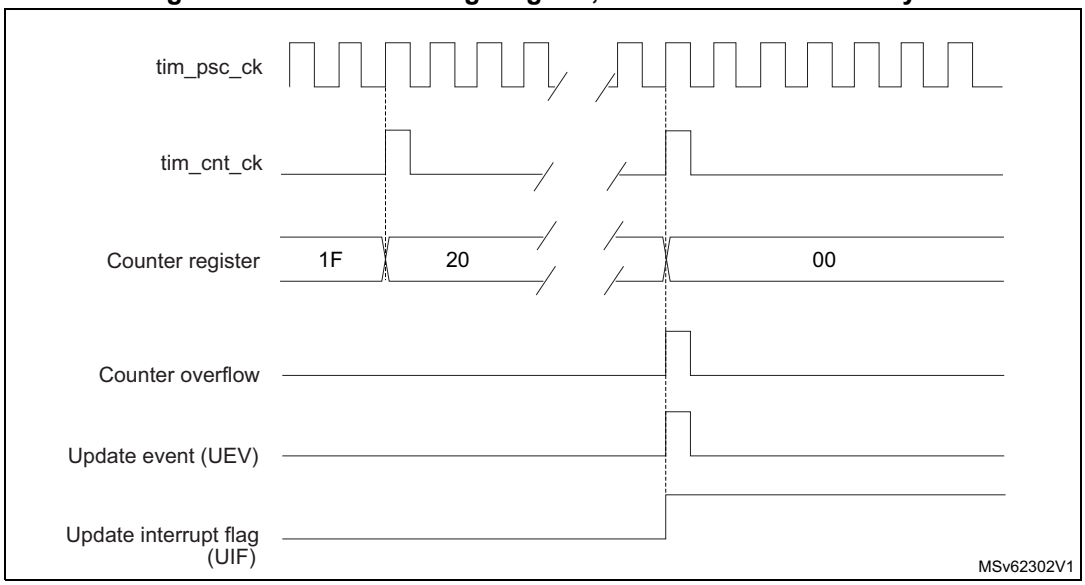


Figure 332. Counter timing diagram, Update event when ARPE = 0 (TIM_ARR not preloaded)

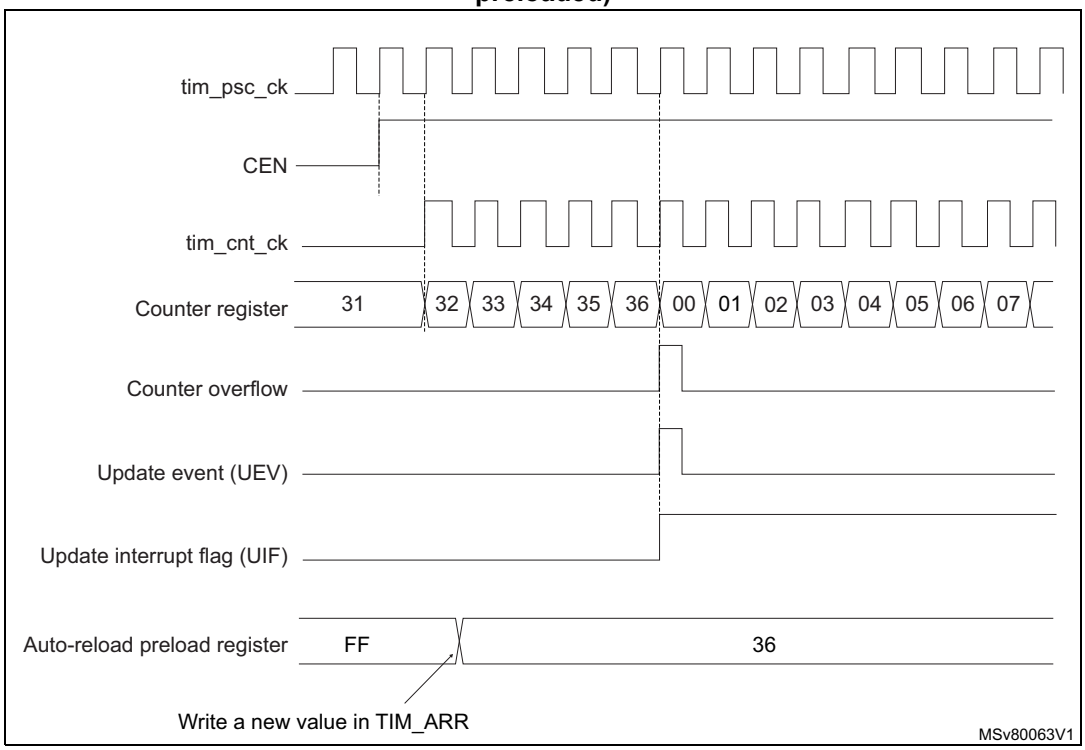
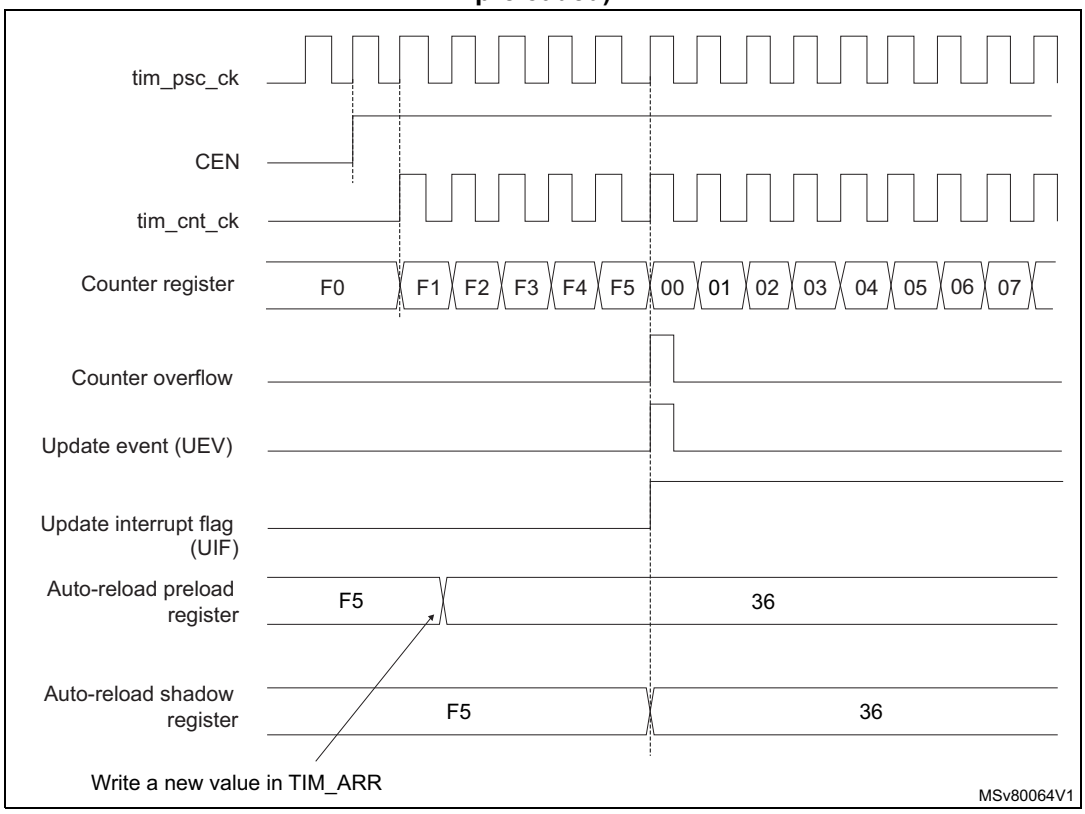


Figure 333. Counter timing diagram, Update event when ARPE = 1 (TIM_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 334. Counter timing diagram, internal clock divided by 1

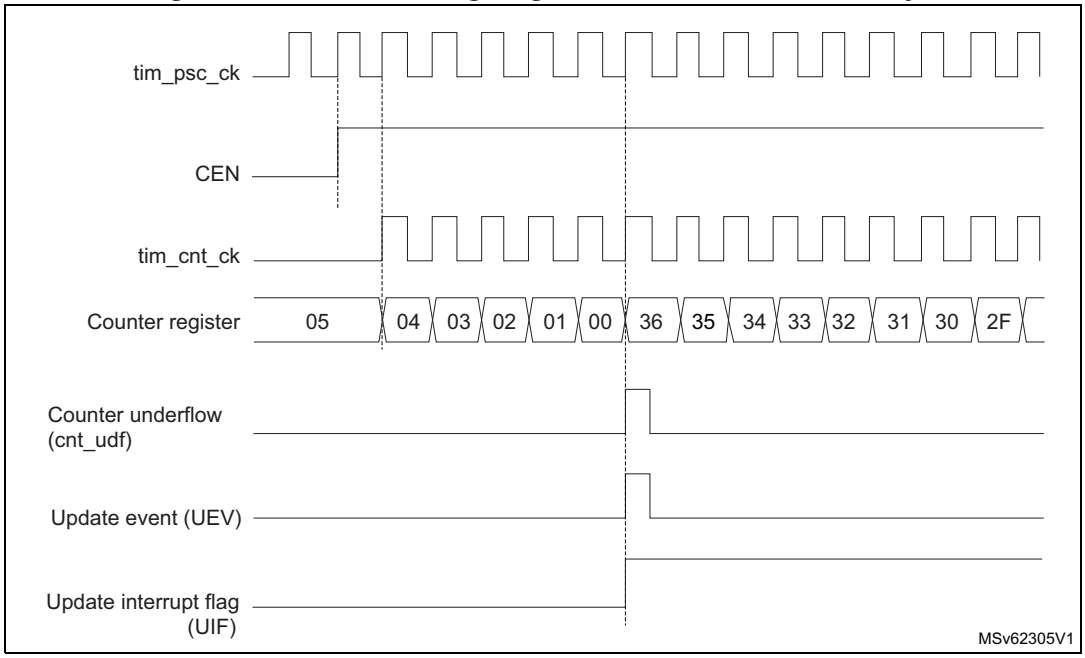


Figure 335. Counter timing diagram, internal clock divided by 2

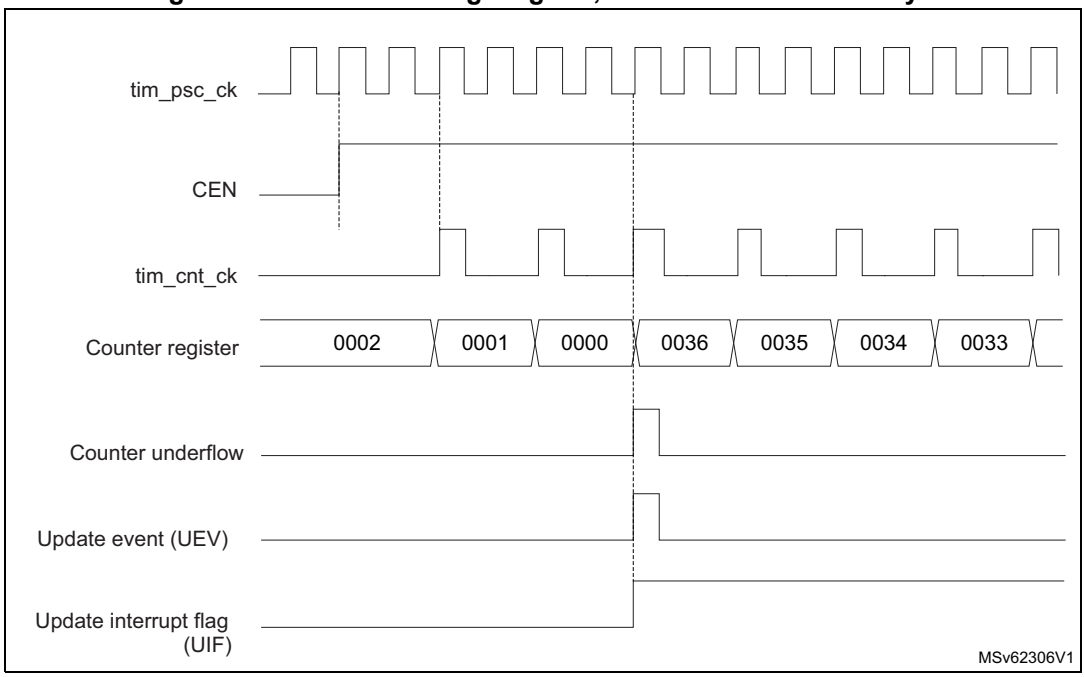


Figure 336. Counter timing diagram, internal clock divided by 4

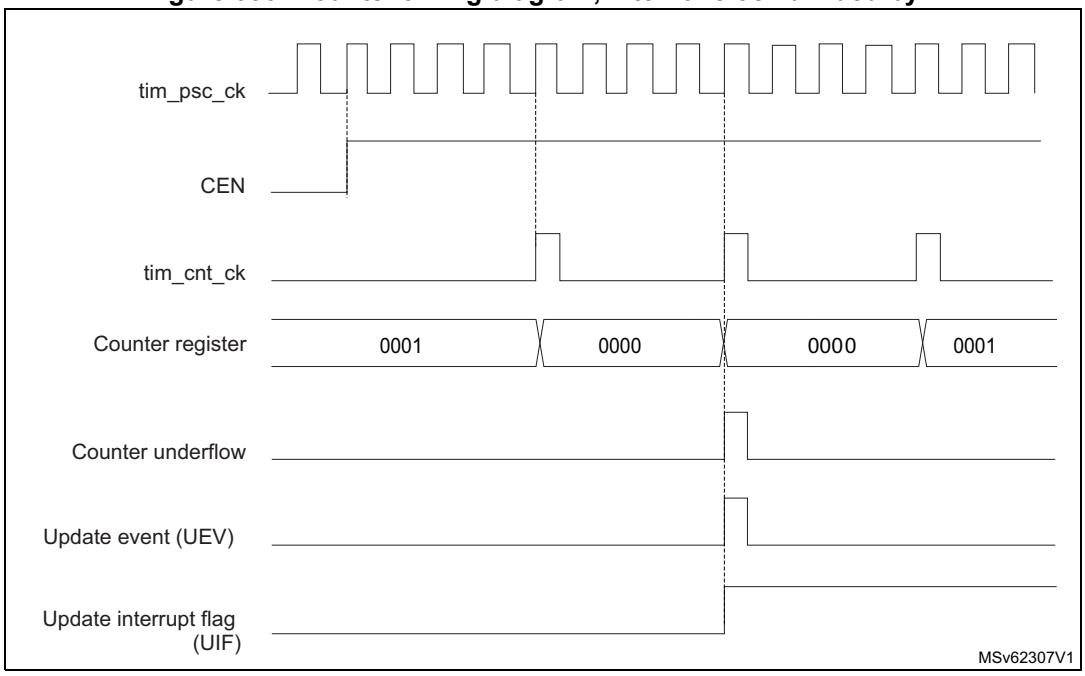


Figure 337. Counter timing diagram, internal clock divided by N

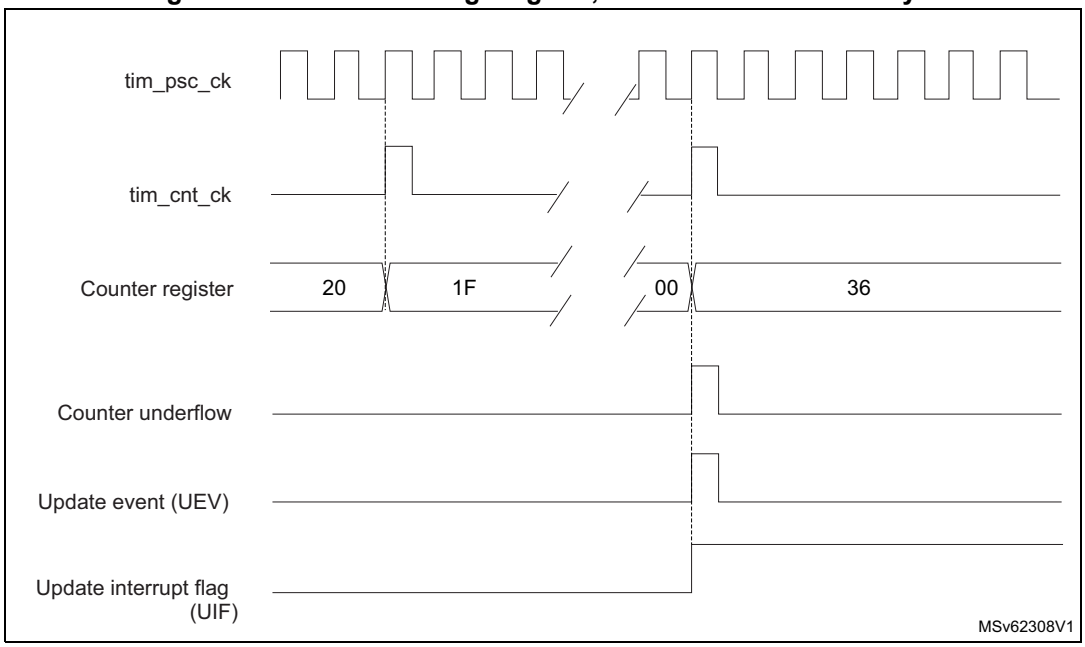
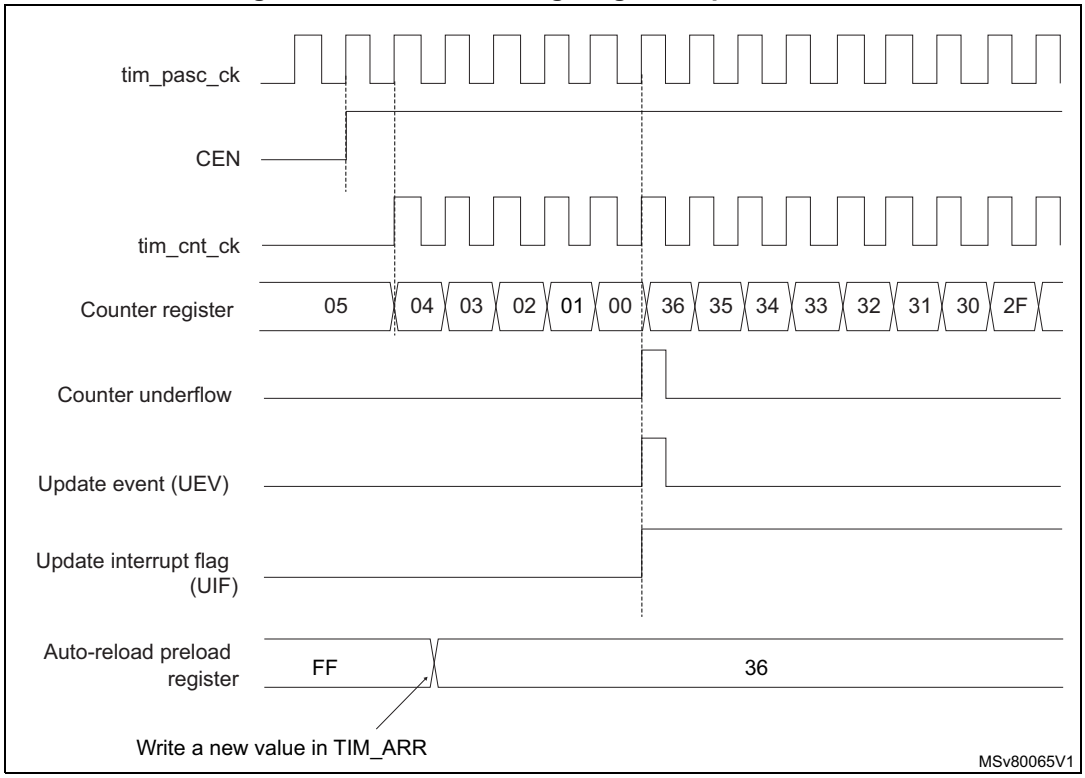


Figure 338. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (center aligned mode 1, CMS = 01), the counter counts up (center aligned mode 2, CMS = 10) the counter counts up and down (center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIM_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

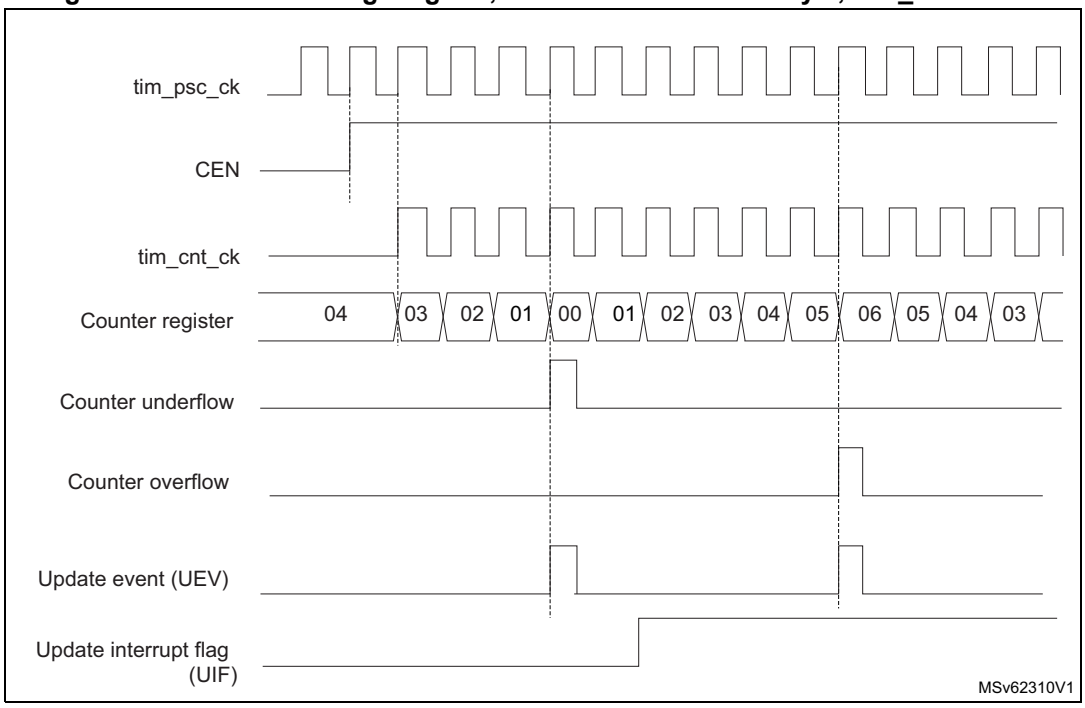
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 339. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 32.5.1: TIM control register 1 (TIM_CR1)*).

Figure 340. Counter timing diagram, internal clock divided by 2

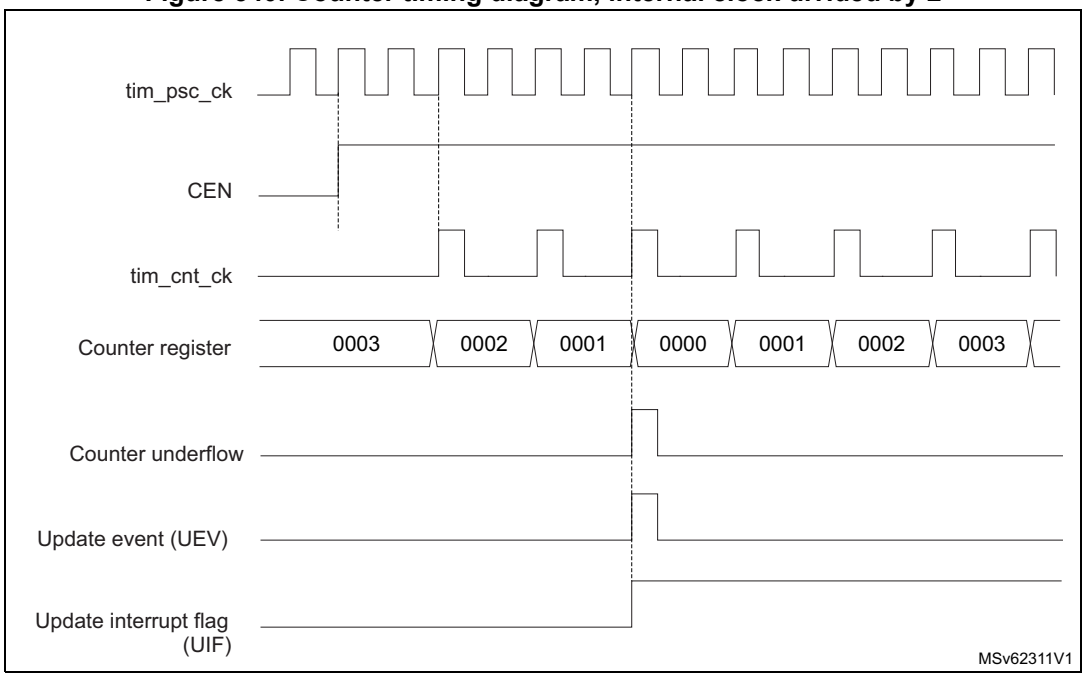
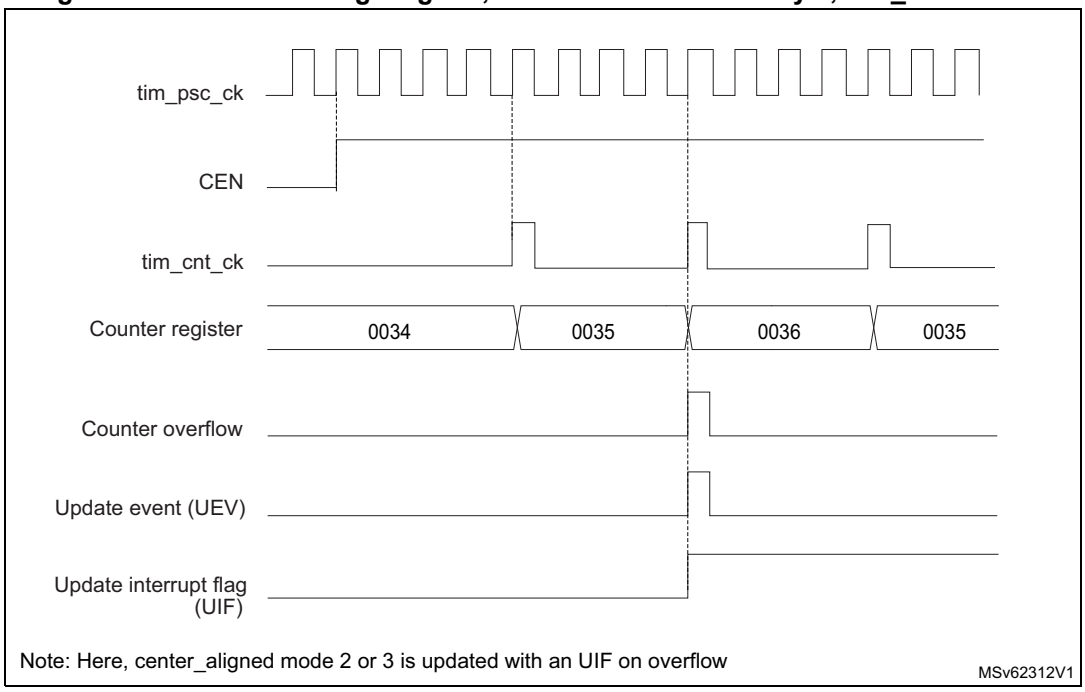


Figure 341. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 342. Counter timing diagram, internal clock divided by N

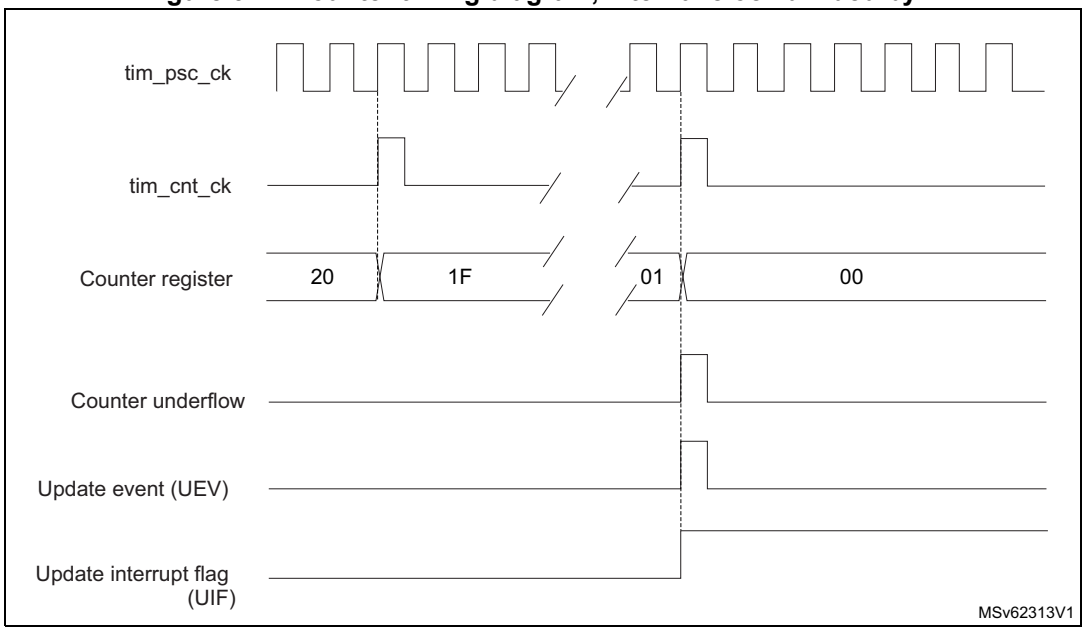


Figure 343. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

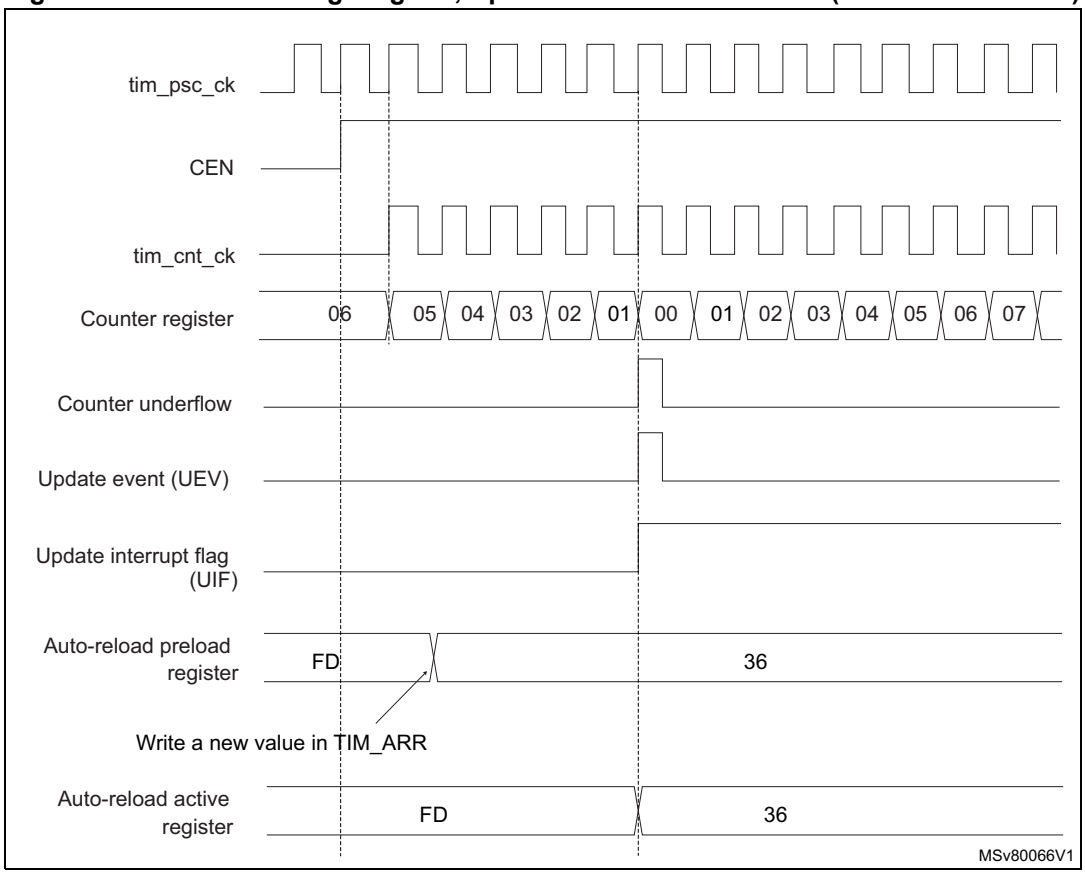
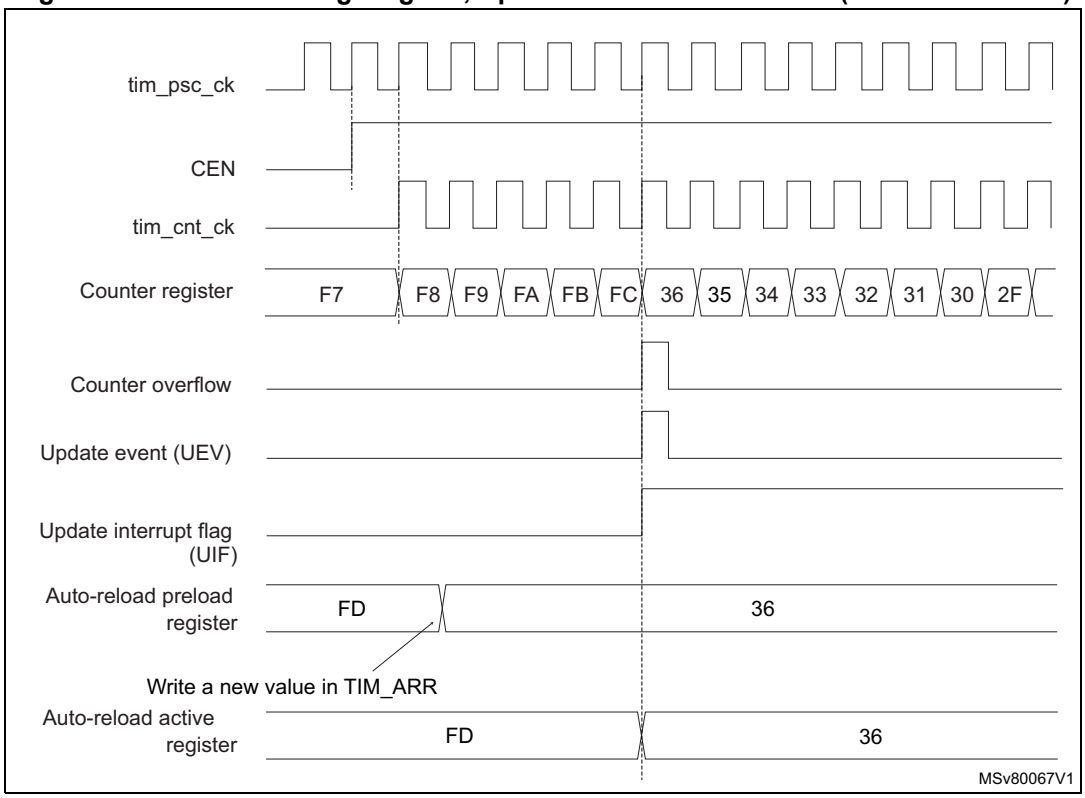


Figure 344. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



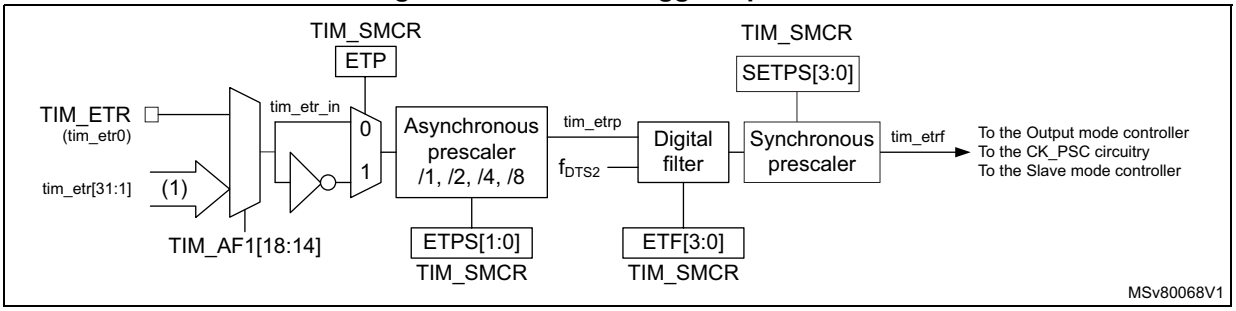
32.4.5 External trigger input

The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 32.4.6*)
- trigger for the slave mode (see *Section 32.4.27*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 32.4.16*)

Figure 345 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIM_SMCR register. The trigger frequency can be decreased with the prescalers programmed by the ETPS[1:0] and SETPS[3:0] bitfields and digitally filtered as per the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the slave mode controller.

Figure 345. External trigger input block



The external trigger input block features 2 prescalers with different functionalities:

- An asynchronous prescaler (division by 1, 2, 4 or 8), able to process tim_etr_in signals with frequencies above tim_ker_ck
- A linear synchronous prescaler (division by 1,2,3,..15), only able to operate with tim_etrp signal frequencies below tim_ker_ck/3

Note: When tim_etrf is used as PWM reset input for cycle-by-cycle current regulation (see Section 32.4.16), SETPS[3:0] must be set to 0 (no prescaling).

Note: The SETPS[3:0] bitfield is always preloaded: a new value will become active only once an update event has occured.

The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[4:0] bitfield in the TIM_AF1 register.

Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for the list of sources connected to the etr_in input in the product.

32.4.6 Clock selection

The counter clock can be provided by the following clock sources:

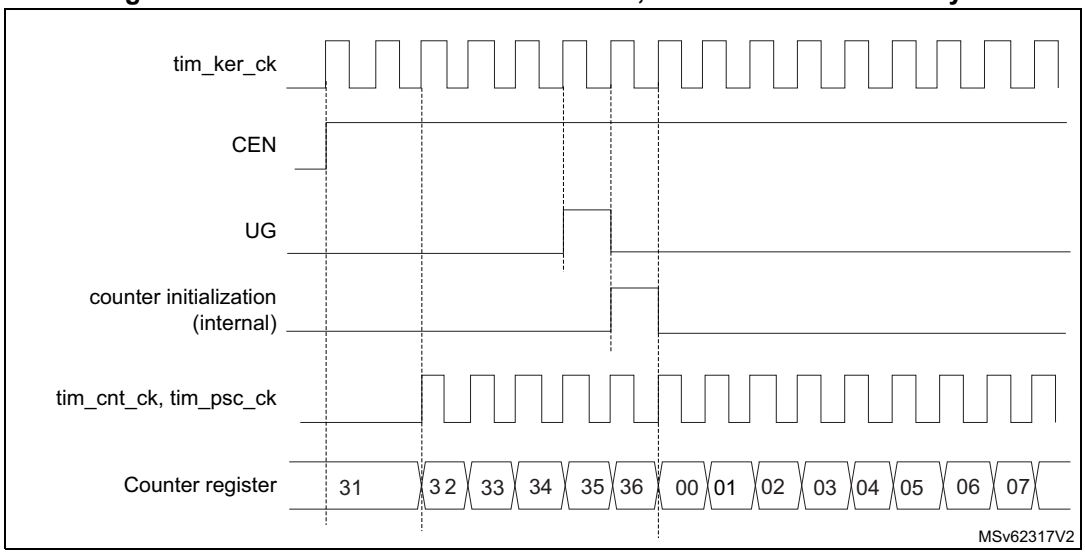
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIM_SMCR register), then the CEN, DIR (in the TIM_CR1 register), and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 346 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

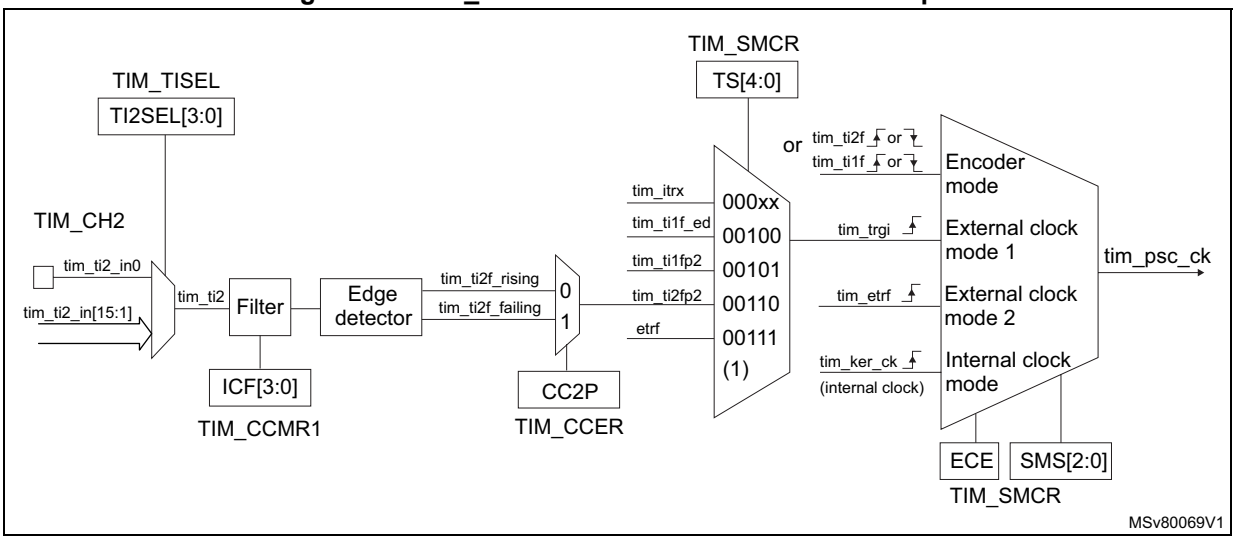
Figure 346. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 347. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).

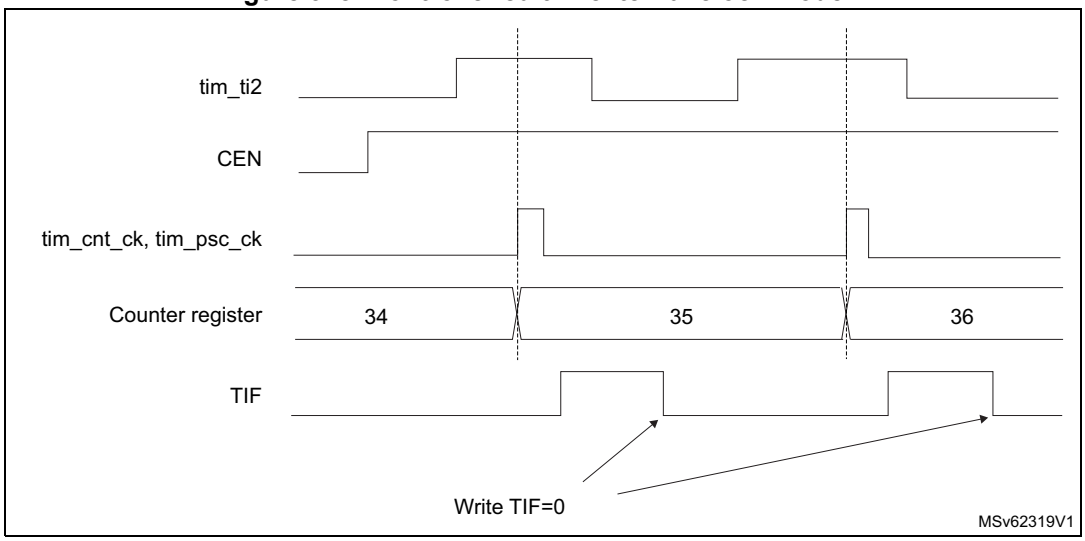
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 348. Control circuit in external clock mode 1



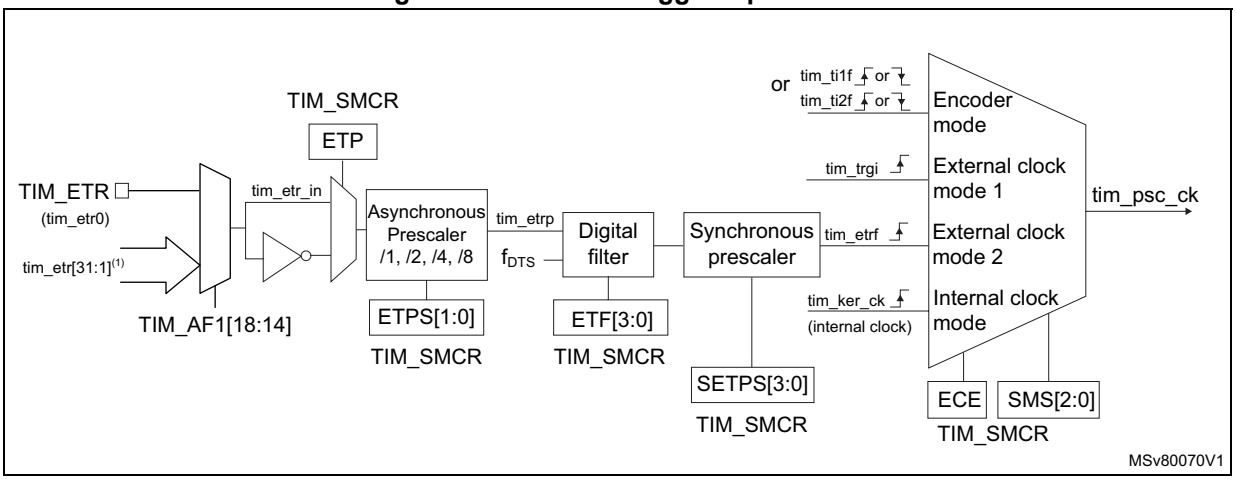
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIM_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 349 gives an overview of the external trigger input block.

Figure 349. External trigger input block



1. Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals*

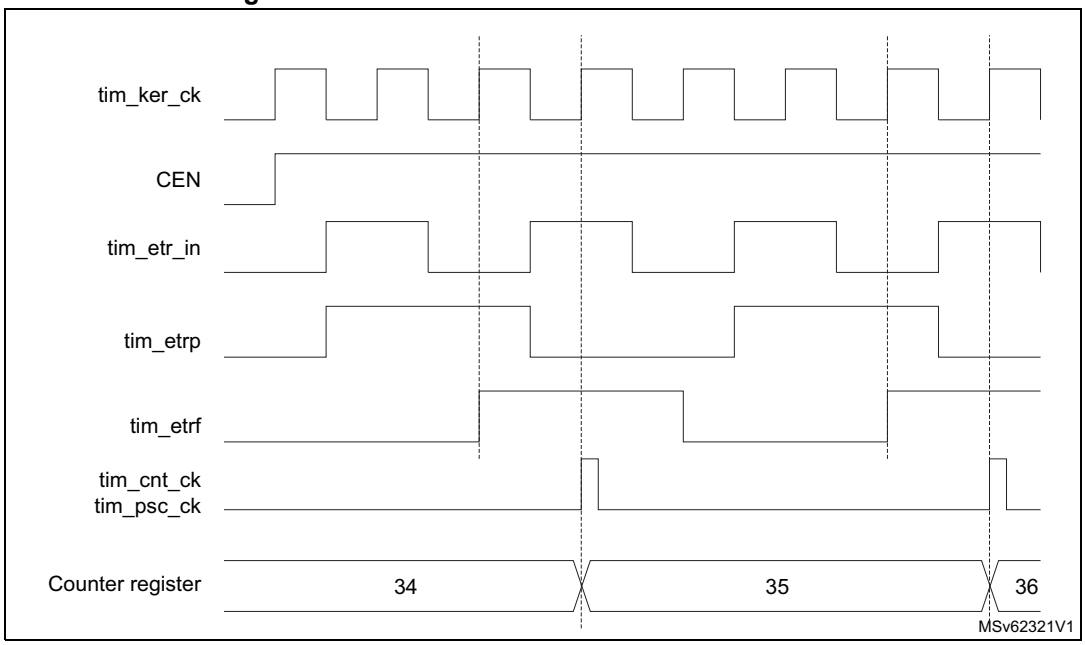
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[4:0] bits in the TIM_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIM_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 and SETPS[3:0] = 0 in the TIM_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIM_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIM_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 350. Control circuit in external clock mode 2



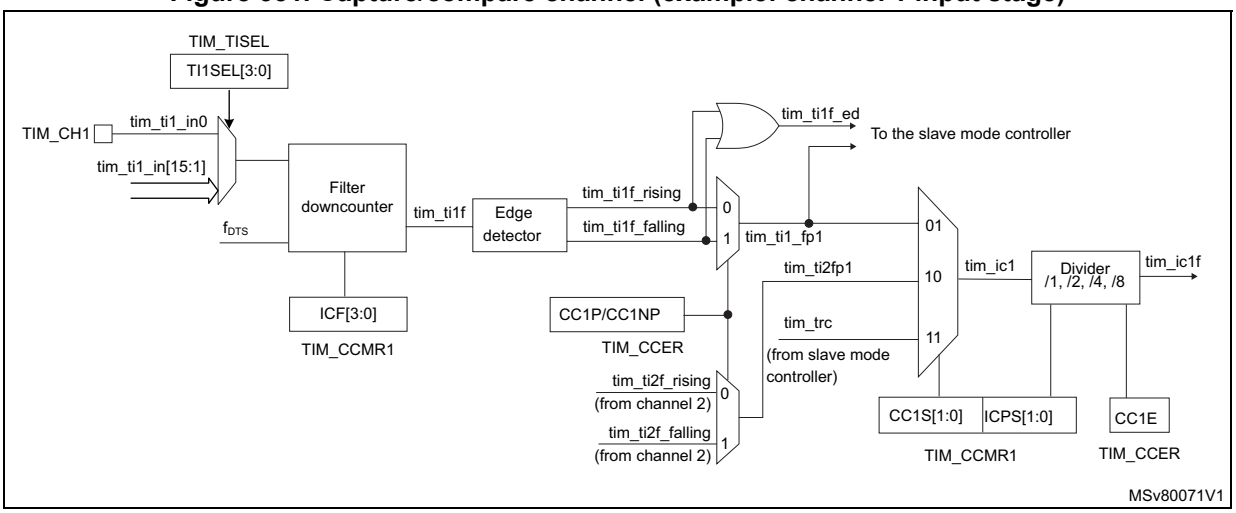
32.4.7 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 351. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 352. Capture/compare channel 1 main circuit

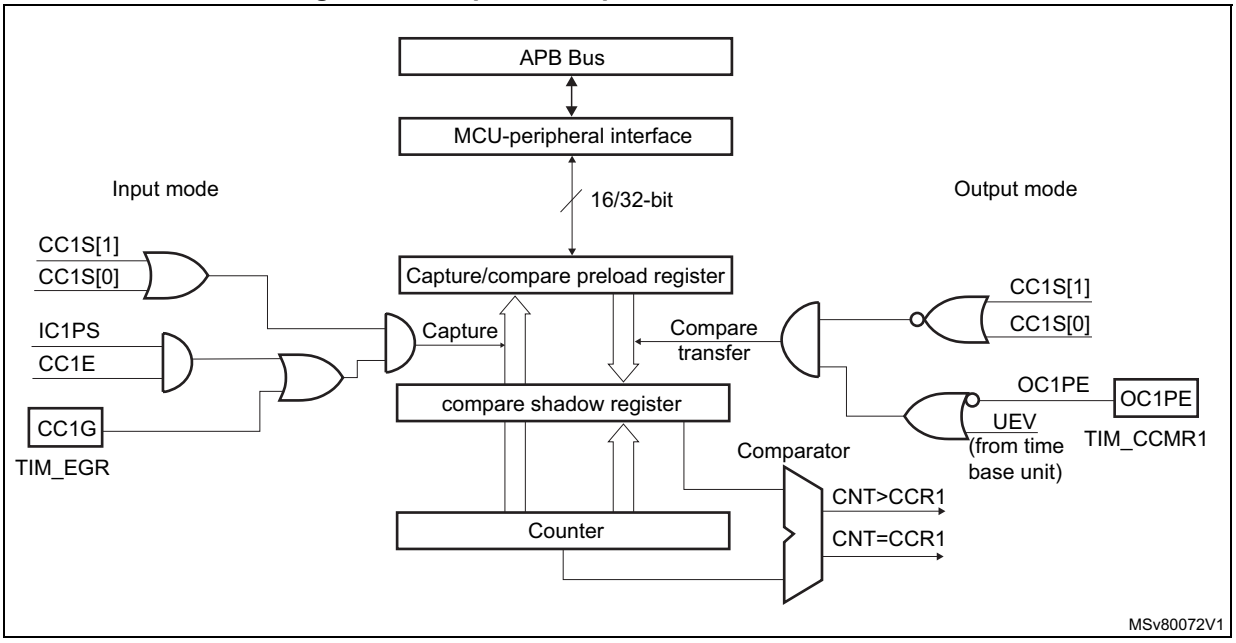
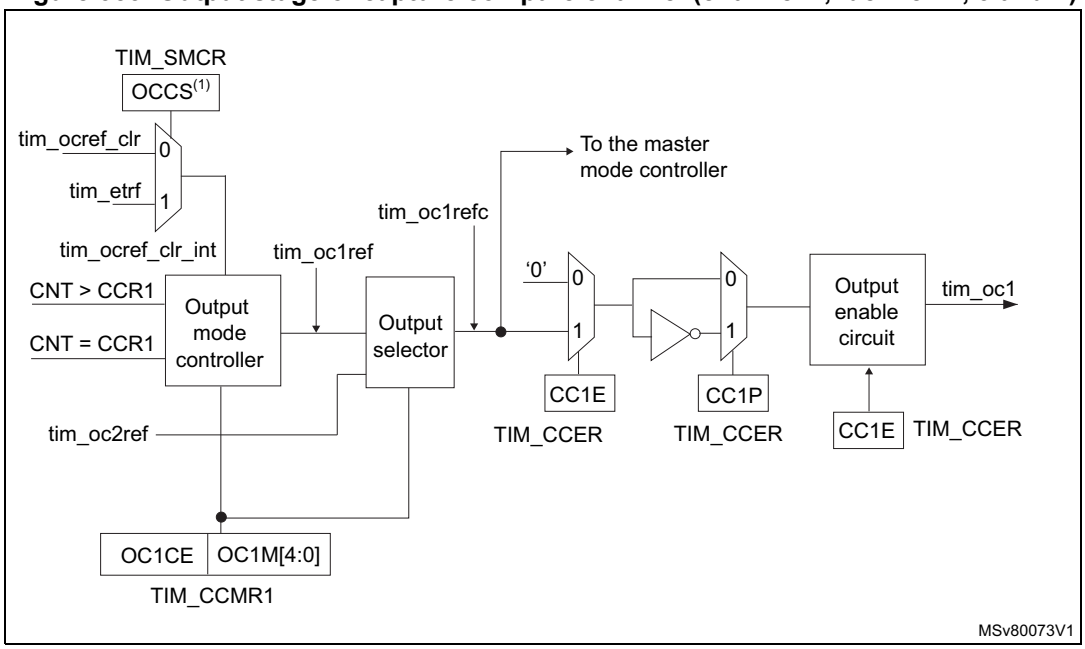


Figure 353. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

32.4.8 Input capture mode

In input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIM_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

32.4.9 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

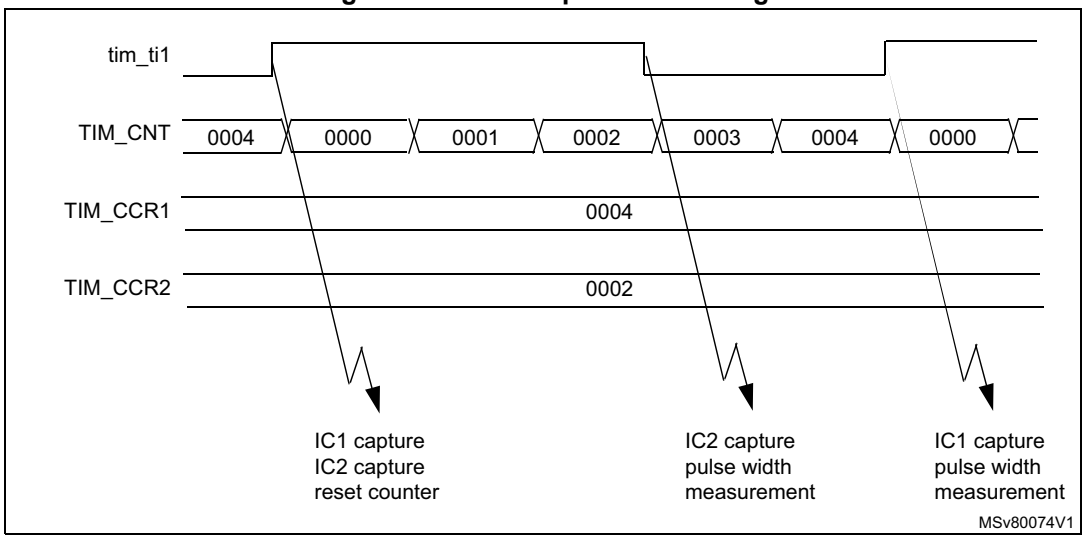
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 354. PWM input mode timing



1. The PWM input mode can be used only with the TIM_CH1/TIM_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

32.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section.

32.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 00000), be

set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

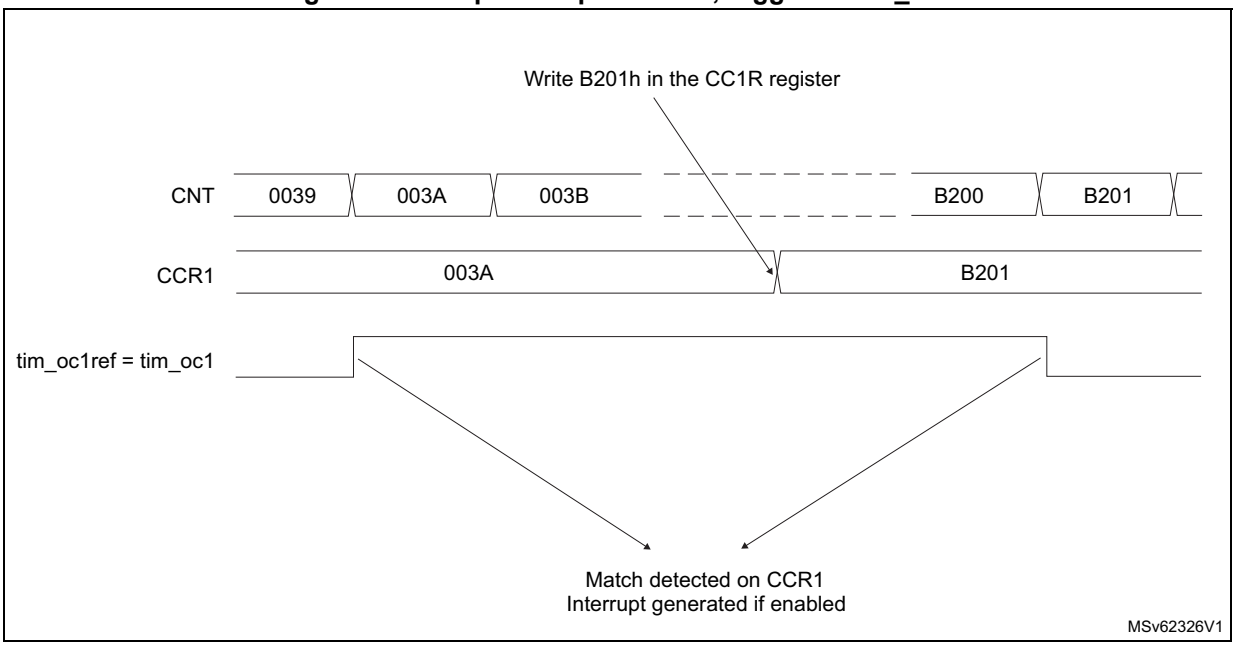
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in one-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 355*.

Figure 355. Output compare mode, toggle on tim_oc1



32.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIM_CCER register. Refer to the TIM_CCERx register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIM_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 00000) to one of the PWM modes (OCxM = 00110 or 00111). This forces the PWM by software while the timer is running.

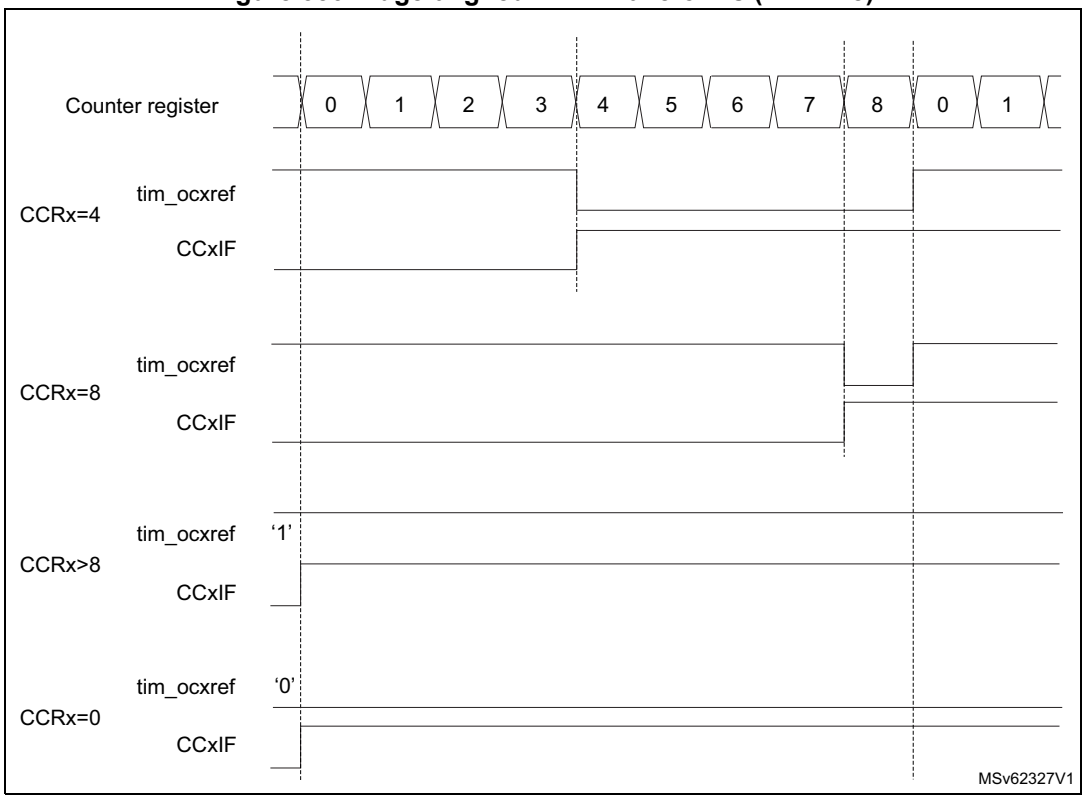
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT <TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 356* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 356. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIM_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT>TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

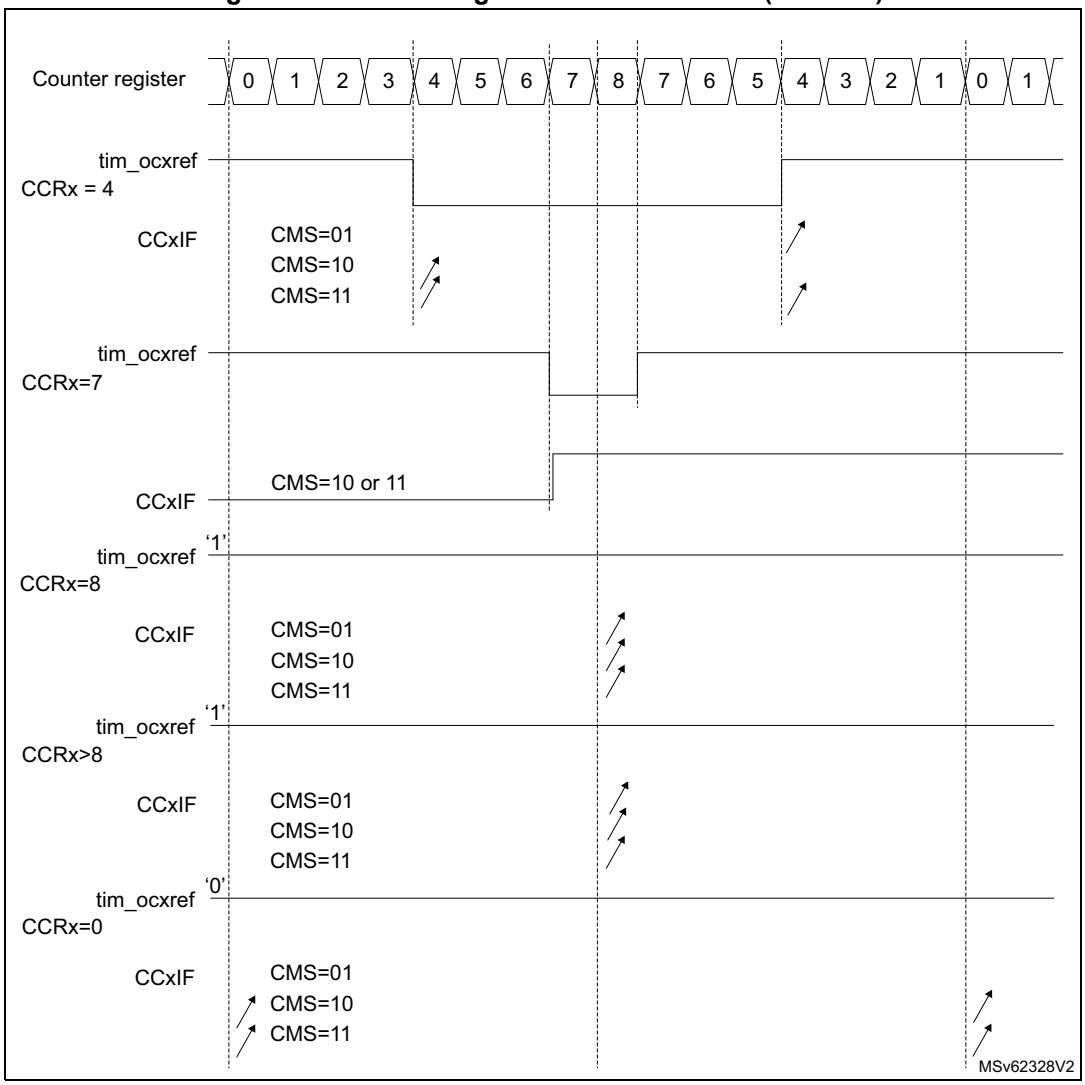
Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 357 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 357. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

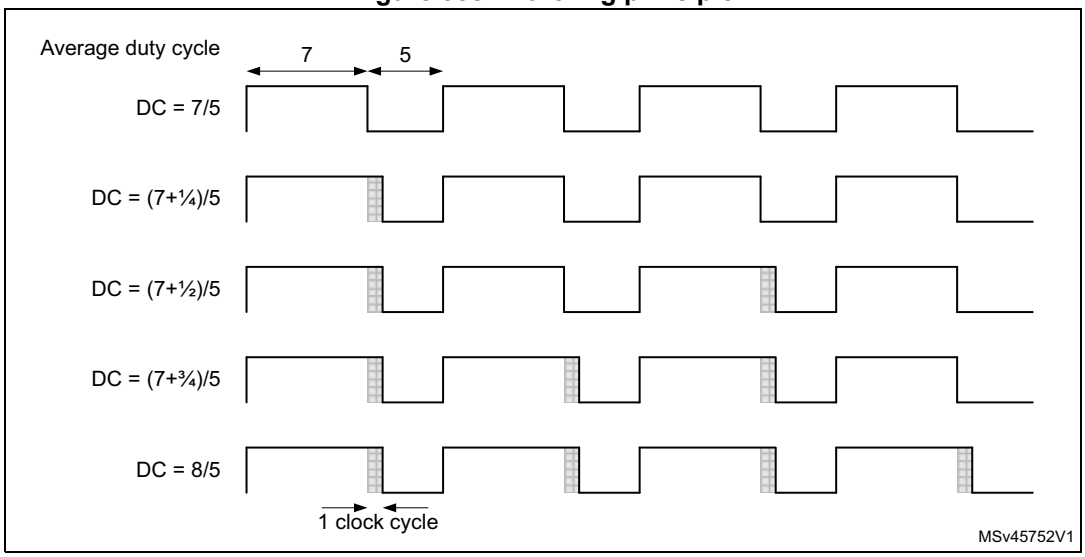
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT>TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 358* presents the dithering principle applied to four consecutive PWM cycles.

Figure 358. Dithering principle



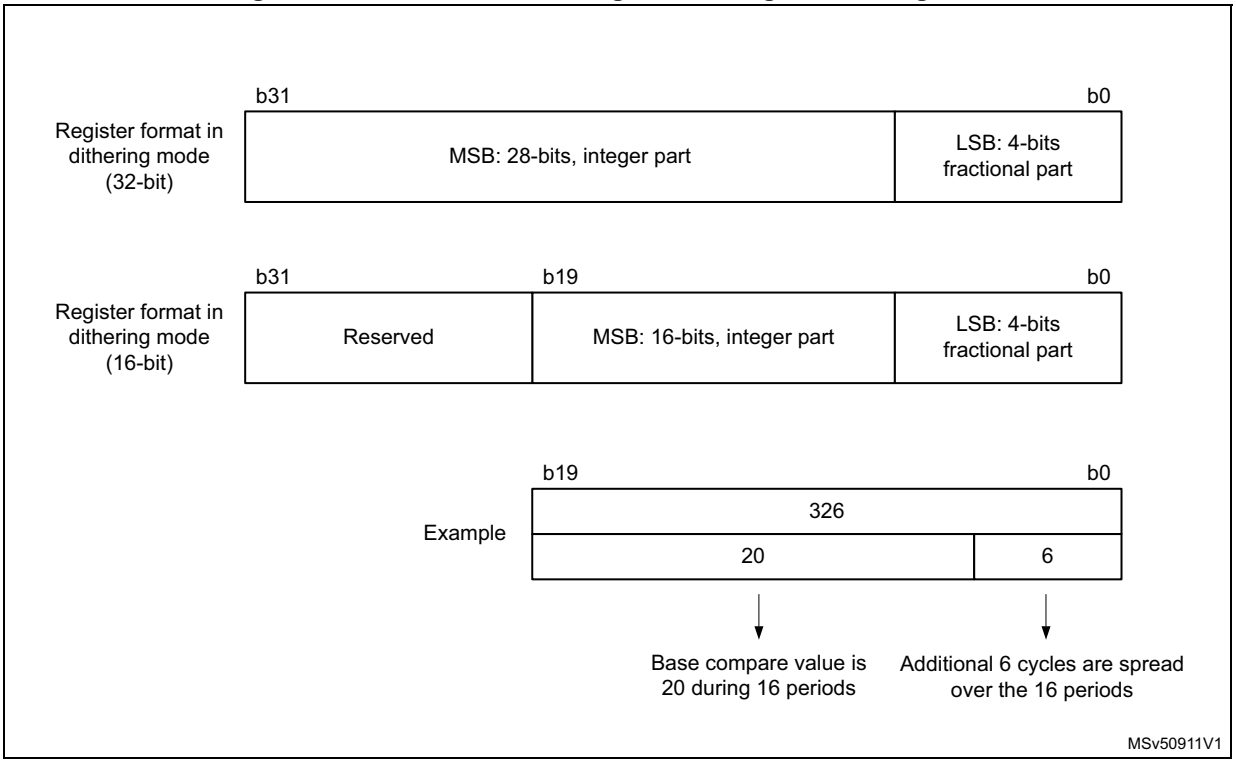
When the dithering mode is enabled, the register coding is changed as following (see *Figure 359* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 359. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIM_ARR and TIM_CCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 360* and *Figure 361*, the dithering mode is used to increase the PWM resolution.

Figure 360. PWM resolution vs frequency (16-bit mode)

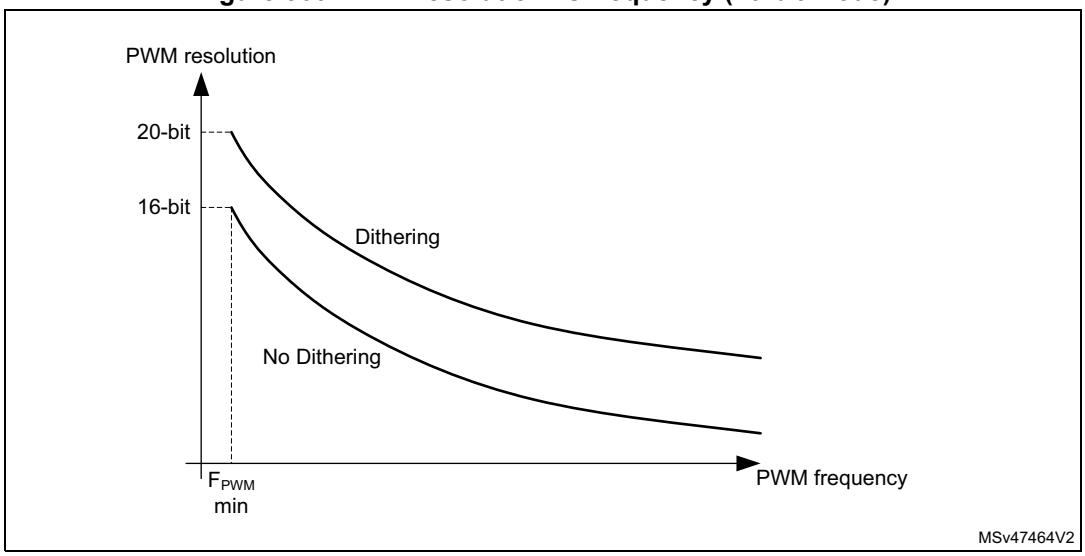
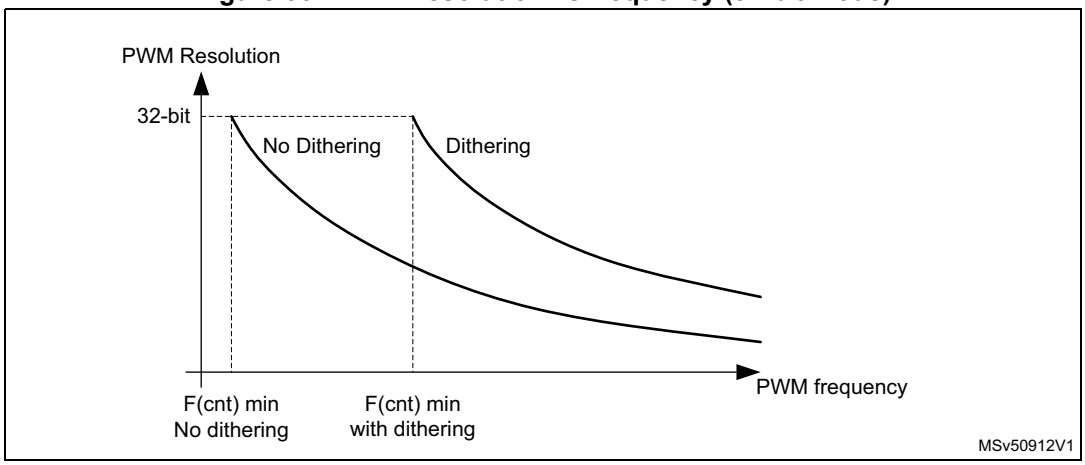
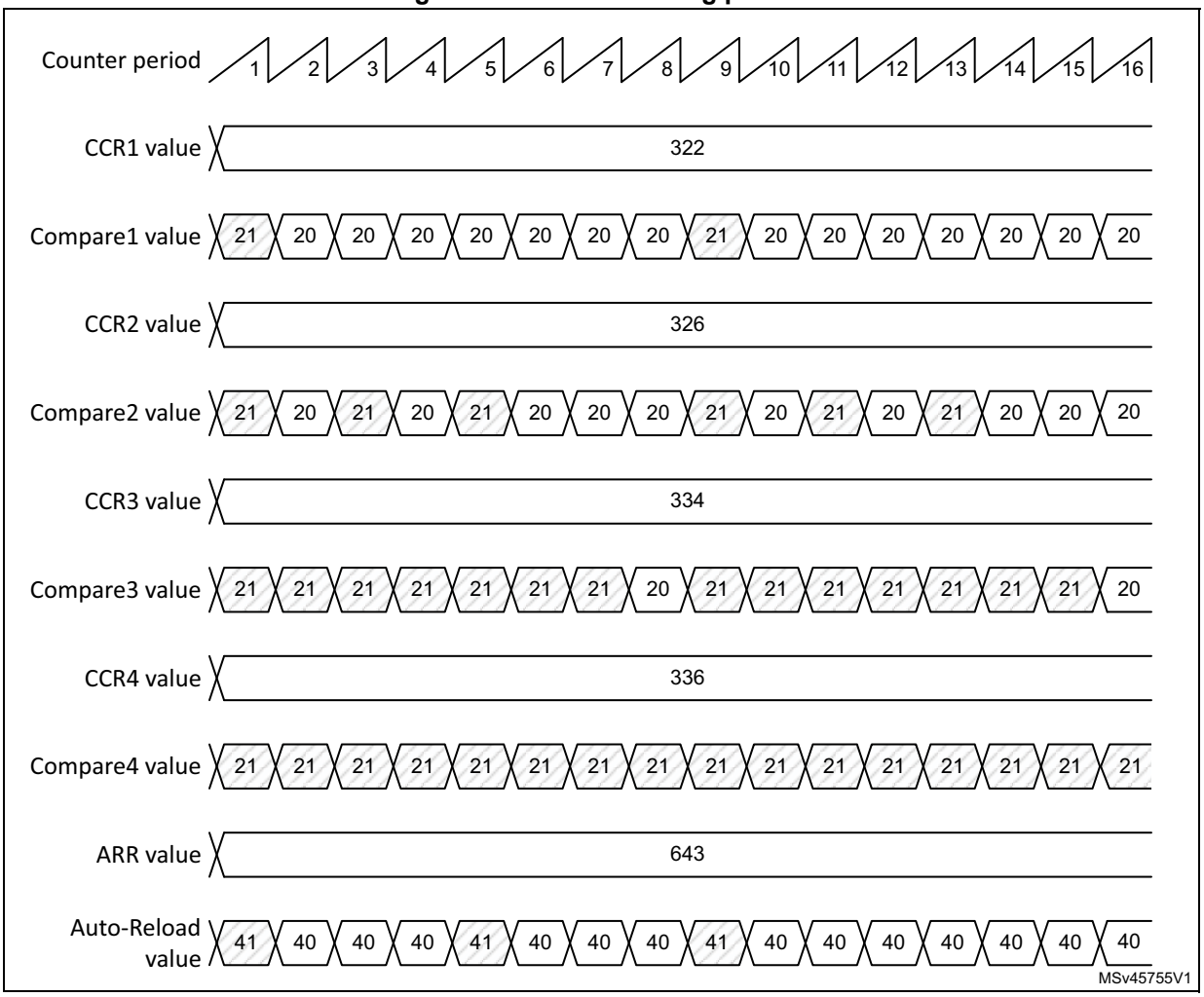


Figure 361. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 362*.

Figure 362. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 317*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 317. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 363*.

Figure 363. Dithering effect on duty cycle in center-aligned PWM mode

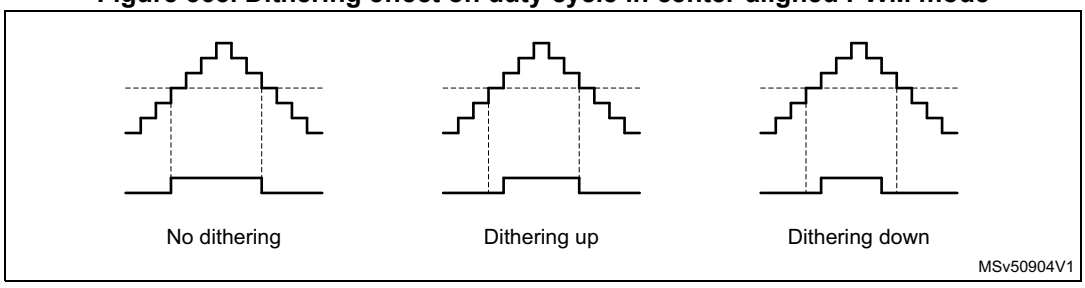


Table 318 shows how the dithering pattern is added in center-aligned PWM mode.

Table 318. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

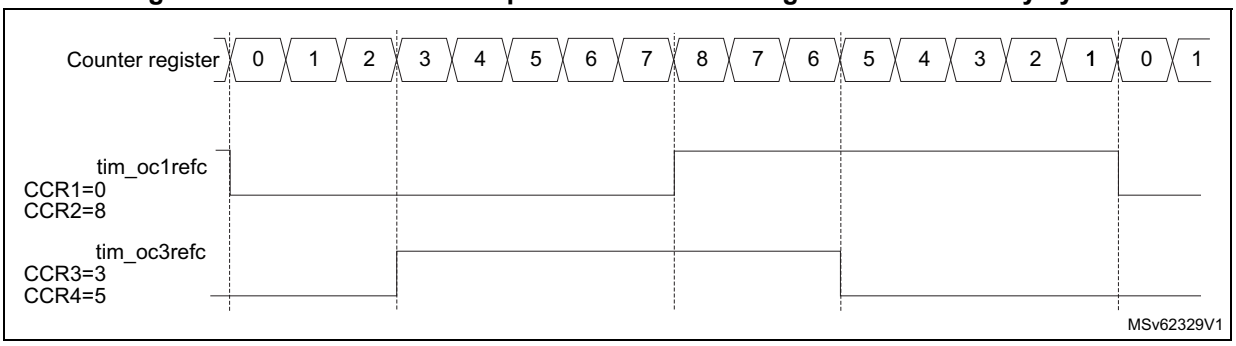
32.4.13 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle, for instance:

- tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2.
- tim_oc3refc is controlled by TIM_CCR3 and TIM_CCR4.

Figure 364 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 364. Generation of two phase-shifted PWM signals with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 319*. The selection is done with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (e.g. Asymmetric PWM mode 1 and 2), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 and 2, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 01110 for the asymmetric PWM mode 1 on oc1refc.

For the asymmetric PWM modes 7 to 10, it is mandatory to have opposite PWM mode for both channels (one being in PWM mode 1 and the other in PWM mode 2). Refer to *Table 319* for configuration.

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Table 319. Asymmetric PWM modes(1)

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	PWM Mode(2)
Asymmetric PWM mode 1	01110	oc1ref (up-counting) oc2ref (down-counting)		oc3ref (up-counting) oc4ref (down-counting)		1
Asymmetric PWM mode 2	01111					2
Asymmetric PWM mode 7	10110	-	-	oc3refc = 1 (up-counting) oc3ref OR oc4ref (down-counting)	-	1(3)
Asymmetric PWM mode 8	10111	-	-	oc3refc = 0 (up-counting) oc3ref AND oc4ref (down-counting)	-	2(4)
Asymmetric PWM mode 9	11000	-	-	oc3ref OR oc4ref (up-counting) oc3refc = 1 (down-counting)	-	1(3)
Asymmetric PWM mode 10	11001	-	-	oc3ref AND oc4ref (up-counting) oc3refc = 0 (down-counting)	-	2(4)

1. Asymmetric PWM mode 7 to 10 are only available on OC3M.

2. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value. PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the two channels used for the asymmetric mode.
3. oc4ref must be programmed in PWM mode 1.
4. oc4ref must be programmed in PWM mode 2.

32.4.14 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs,

- tim_oc1refc (or tim_oc2refc) is controlled by TIM_CCR1 and TIM_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIM_CCR3 and TIM_CCR4

Table 320 summarizes different combinations of PWM combined mode:

Table 320. Combined PWM mode overview

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	PWM mode(1)
Combined PWM mode 1	01100	oc1ref OR oc2ref		oc3ref OR oc4ref		1
Combined PWM mode 2	01101	oc1ref AND oc2ref		oc3ref AND oc4ref		2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 01100 (combined PWM mode 1) or 01101 (combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

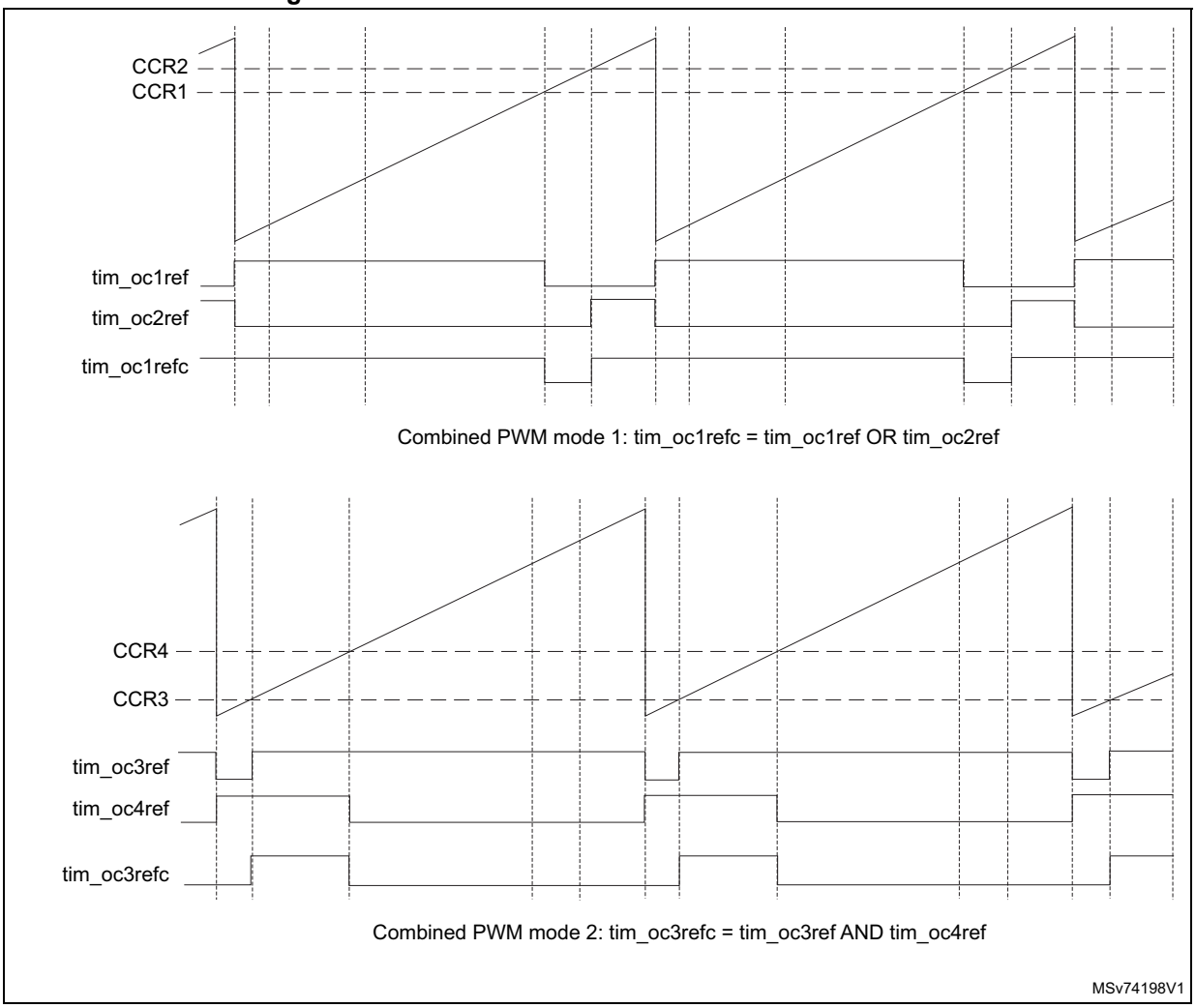
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in PWM mode 1 and the other in PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

Figure 365 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 1.
- Channel 2 is configured in PWM mode 2.
- Channel 3 is configured in combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 365. Combined PWM mode on channels 1 and 3

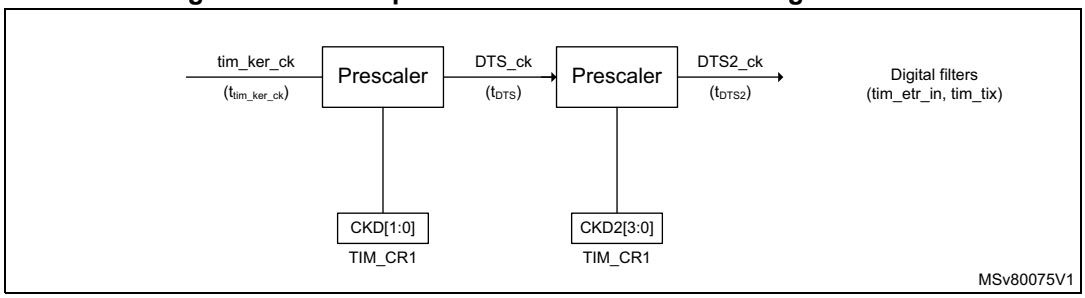


32.4.15 Clock dividers for dead-time generators and digital filters

The the digital filters (for tim_etr_in, tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the DTS_ck clock period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 366. Clock prescalers for dead-time and digital filters



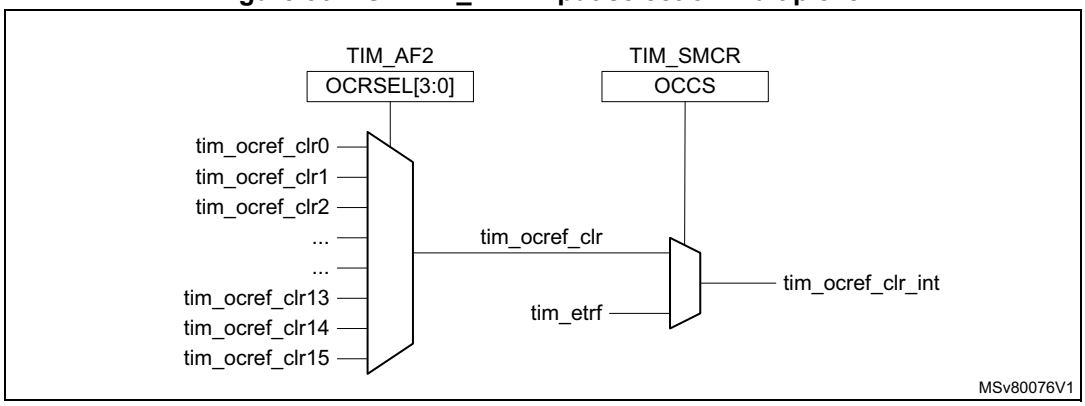
32.4.16 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in output compare and PWM modes. It must not be used in forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 32.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIM_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[15:0] inputs, using the OCRSEL[3:0] bitfield in the TIM_AF2 register, as shown in *Figure 367*.

Figure 367. OCREF_CLR input selection multiplexer



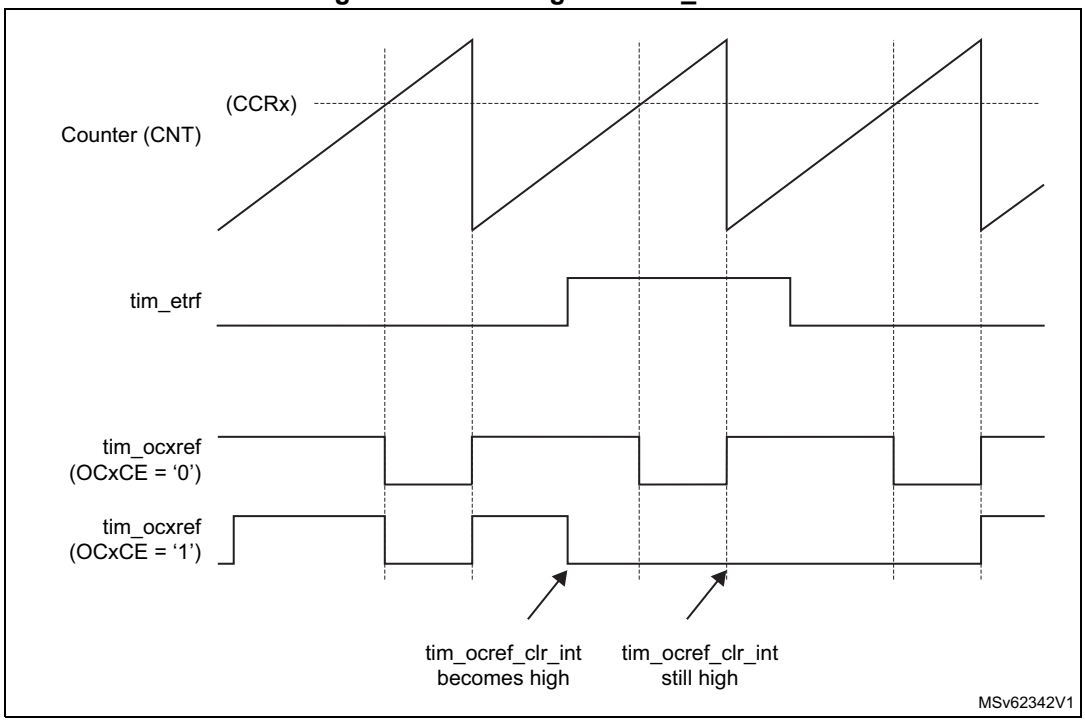
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIM_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 368 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 368. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to enable the OCxCE bit for the channel. The OCxCE bit of the complementary channel (for exemple: OC2CE = X for oc2ref in *Figure 369*) is ignored. *Figure 369* to *Figure 371* describe the output waveforms for specific use cases when the ocref_clr_int signal is asserted.

Figure 369. Asymmetric PWM mode 1

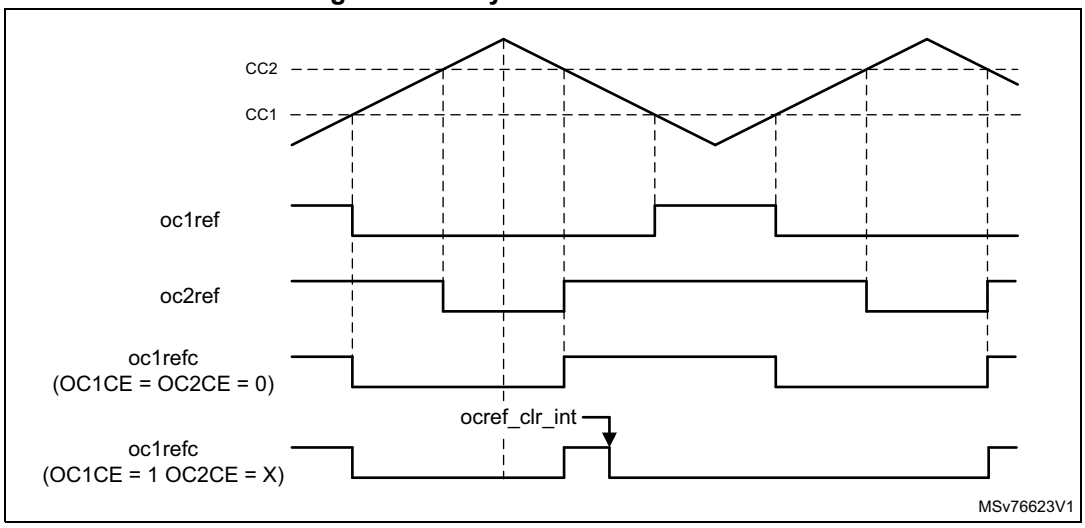


Figure 370. Asymmetric PWM mode 2

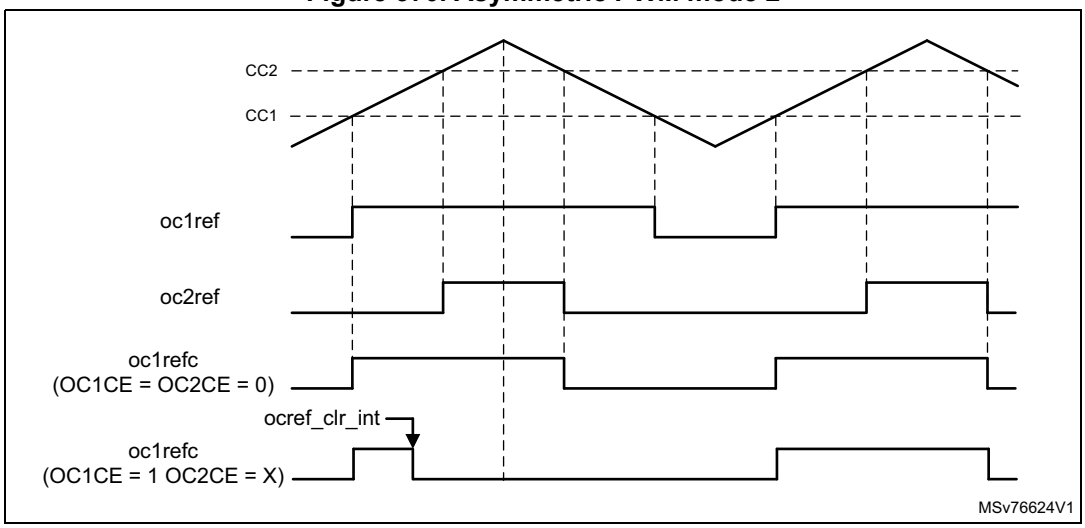
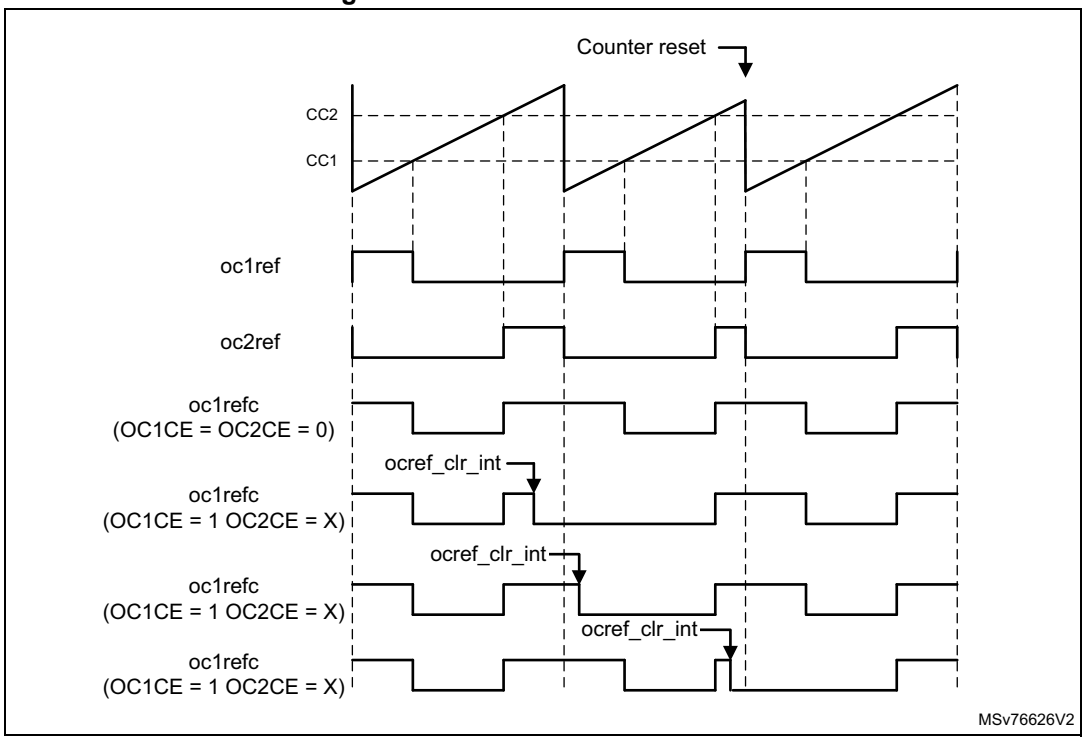


Figure 371. Combined PWM mode 1



Conditions of use

This feature supports cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used in the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One-pulse mode with PWM active on match (OPM = 1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E)
- Combined PWM mode 1 or 3 in center-aligned mode (CMS[1:0] = 1, 2, 3 and OCxM[4:0] = 0x0C)

32.4.17 One-pulse mode

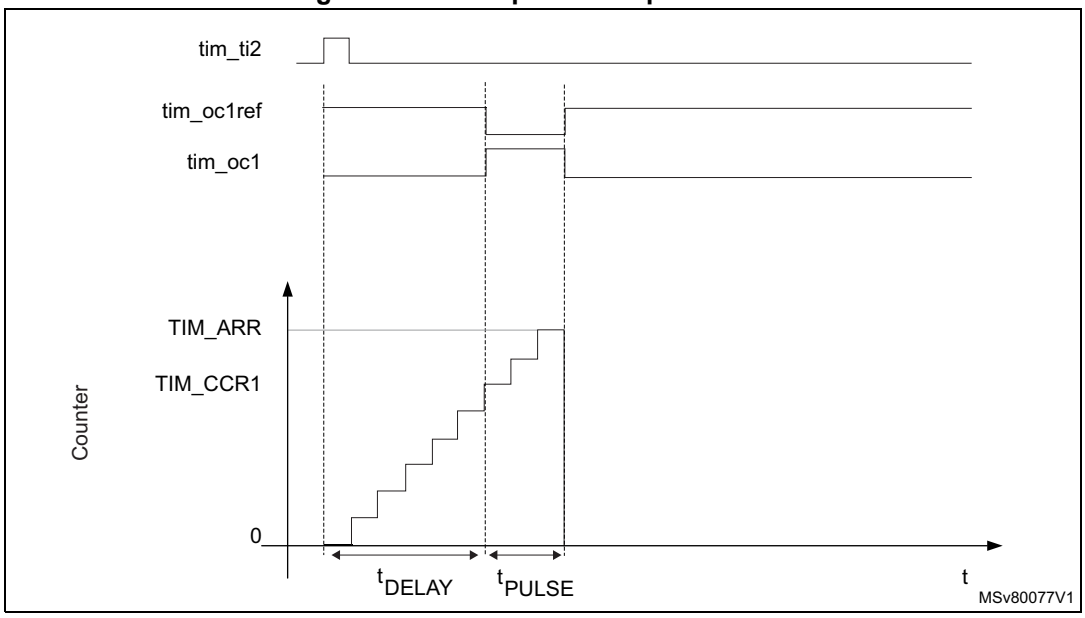
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 372. Example of one-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 00111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIM_CR1 register must be low.

Since only one pulse (single mode) is needed, a one must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the repetitive mode is selected.

Particular case: tim_ocx fast enable:

In one-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

32.4.18 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 32.4.17*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in slave mode, with the bits SMS[4:0] = 01000 (combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for retriggerable OPM mode 1 or 2.

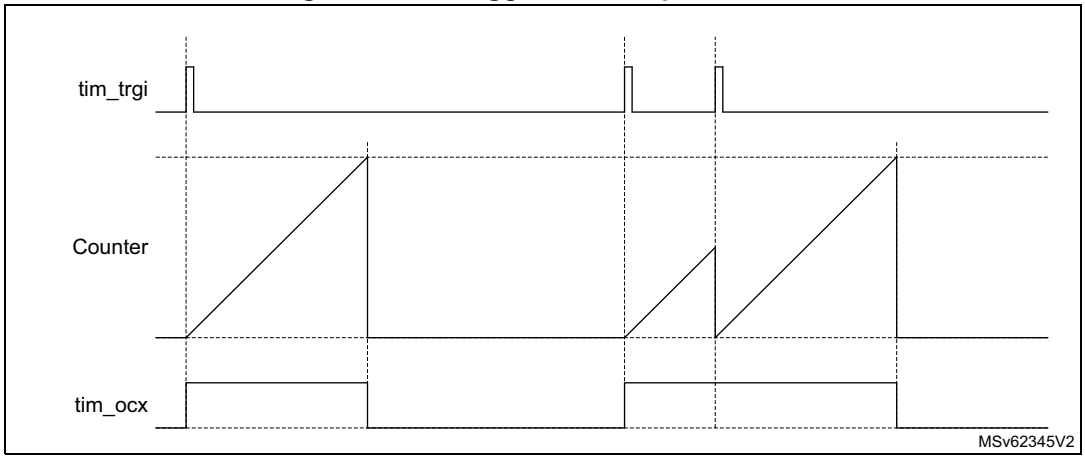
If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[4:0] and SMS[4:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 373. Retriggerable one-pulse mode



32.4.19 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 374*.

Figure 374. Pulse generator circuitry

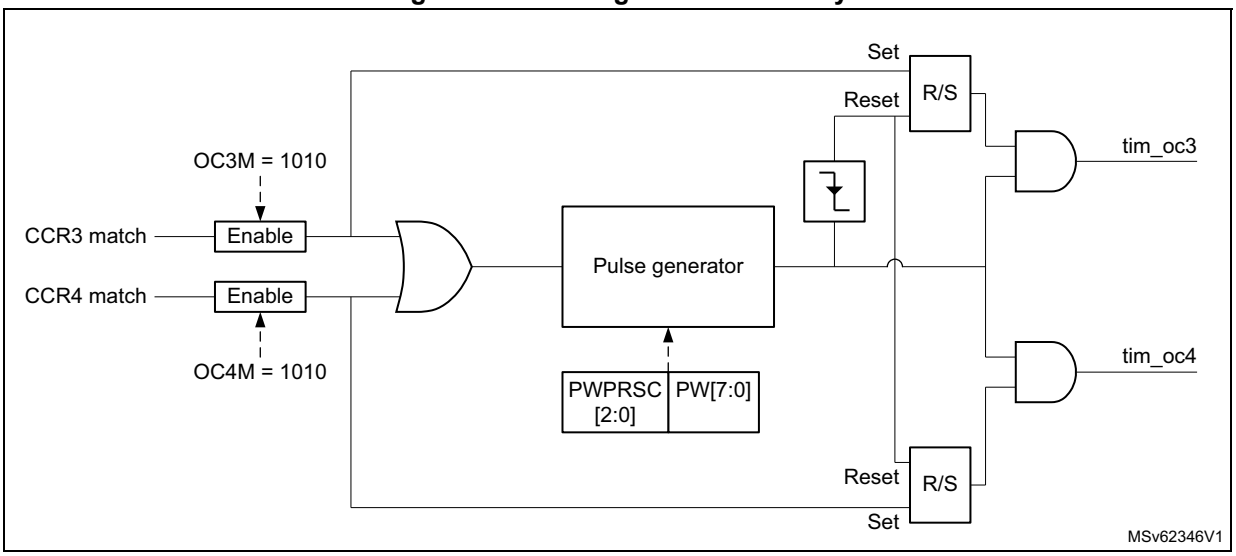
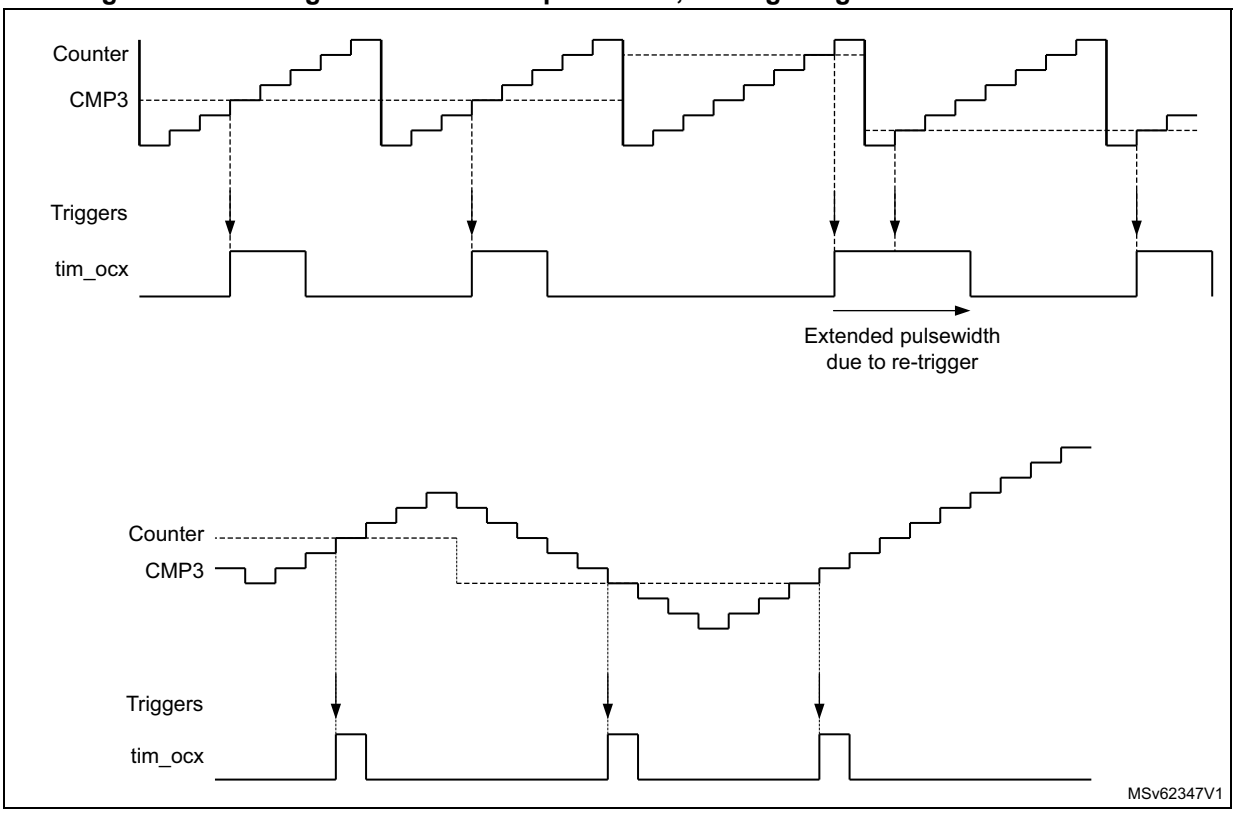


Figure 375 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 375. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIM_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

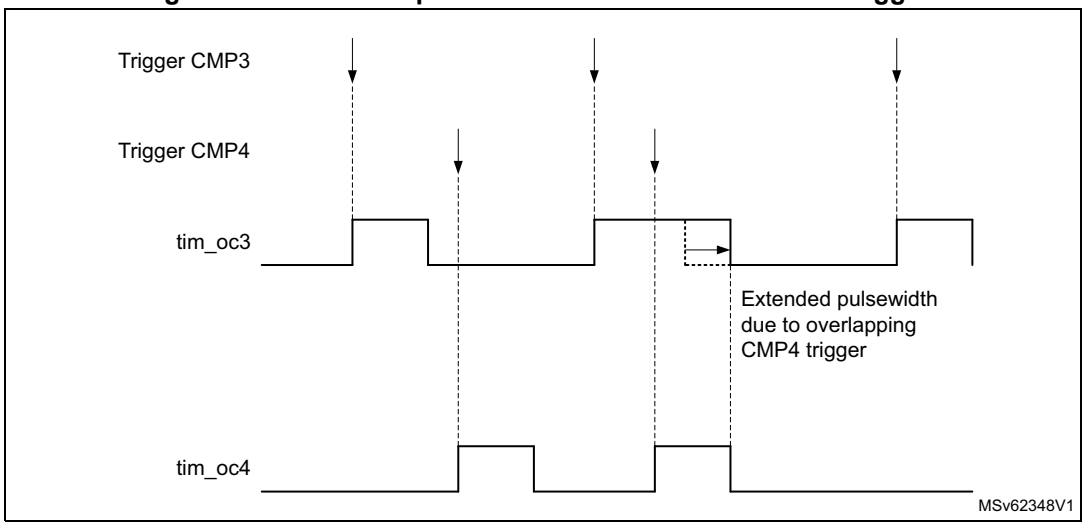
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

Gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 376).

Figure 376. Extended pulse width in case of concurrent triggers



32.4.20 Encoder interface mode

Quadrature encoder

To select encoder interface mode write SMS = 0001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 321*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and external clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 321. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 377 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIM_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, counter is enabled).

Figure 377. Example of counter operation in encoder interface mode

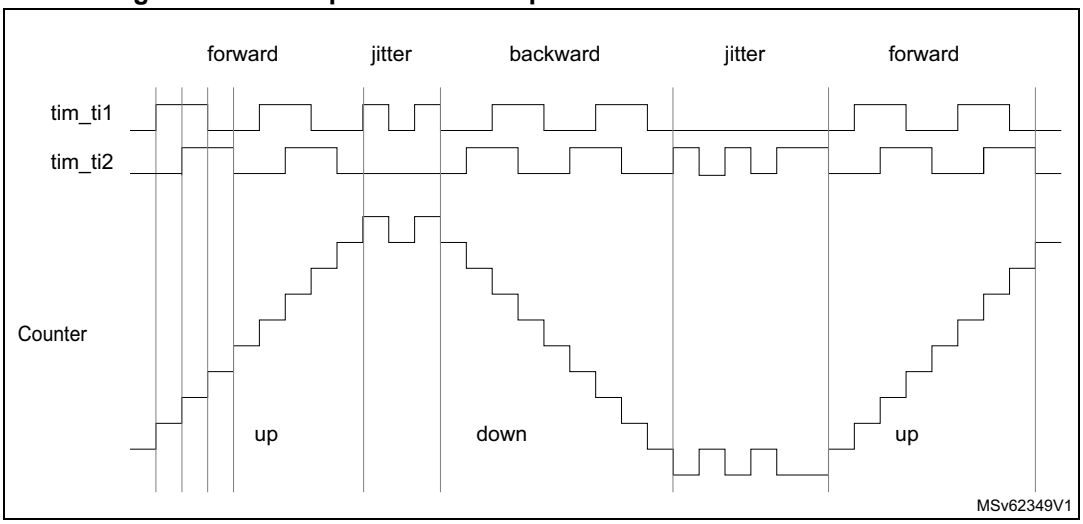


Figure 378 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 378. Example of encoder interface mode with tim_ti1fp1 polarity inverted

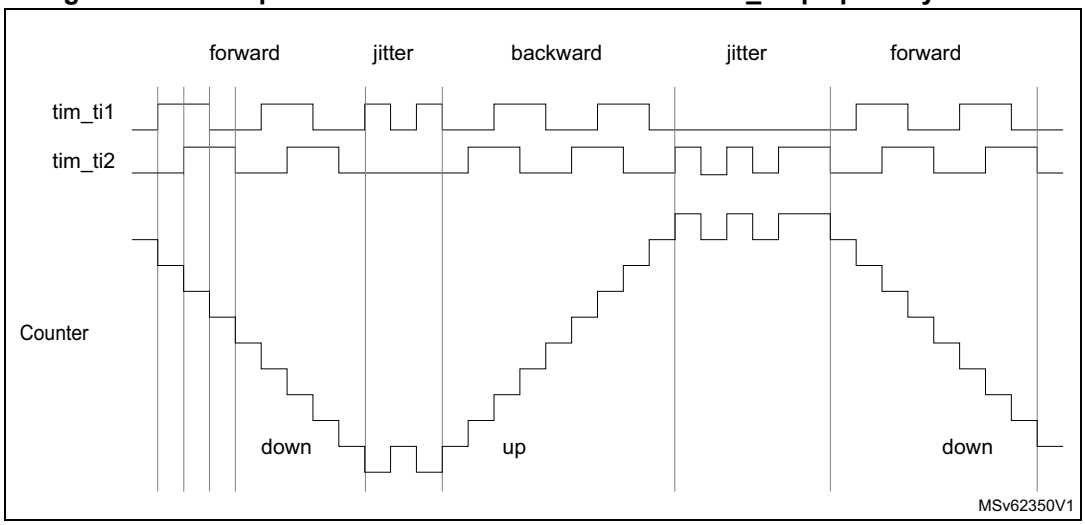
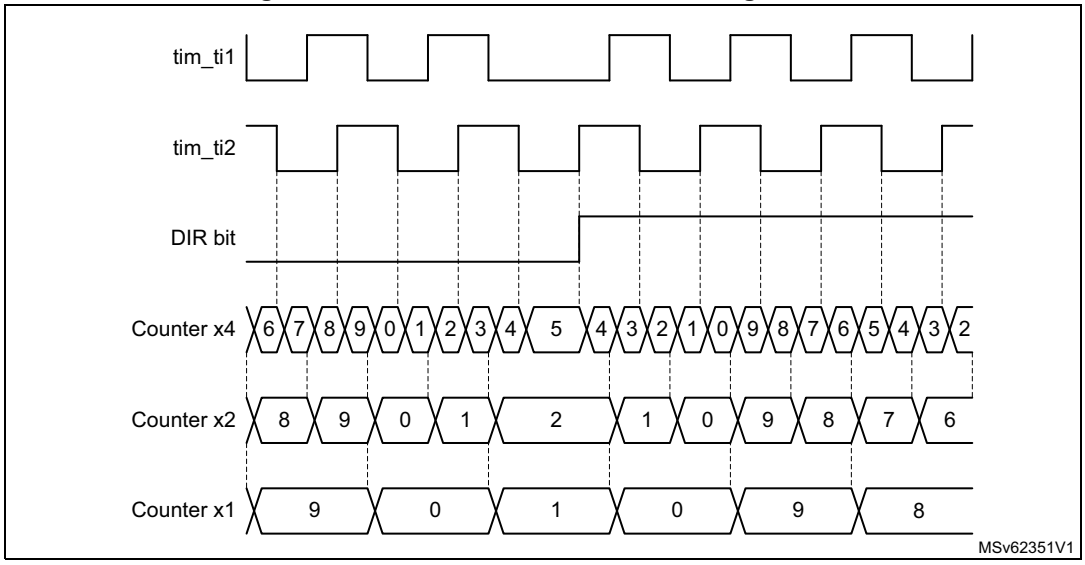


Figure 379 shows the timer counter value during a speed reversal, for various counting modes.

Figure 379. Quadrature encoder counting modes



The timer, when configured in encoder interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

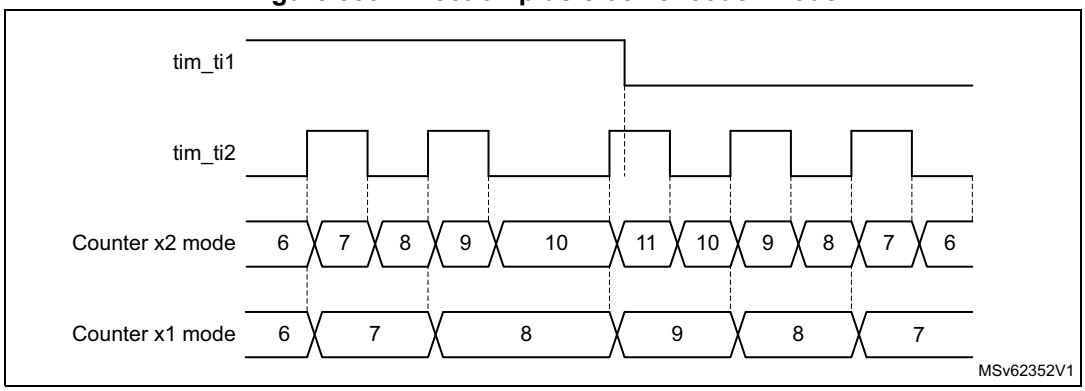
In the "clock plus direction" mode shown on *Figure 380*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 380. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 381*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 381*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 382*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 381*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 382*).

Figure 381. Directional clock encoder mode (CC1P = CC2P = 0)

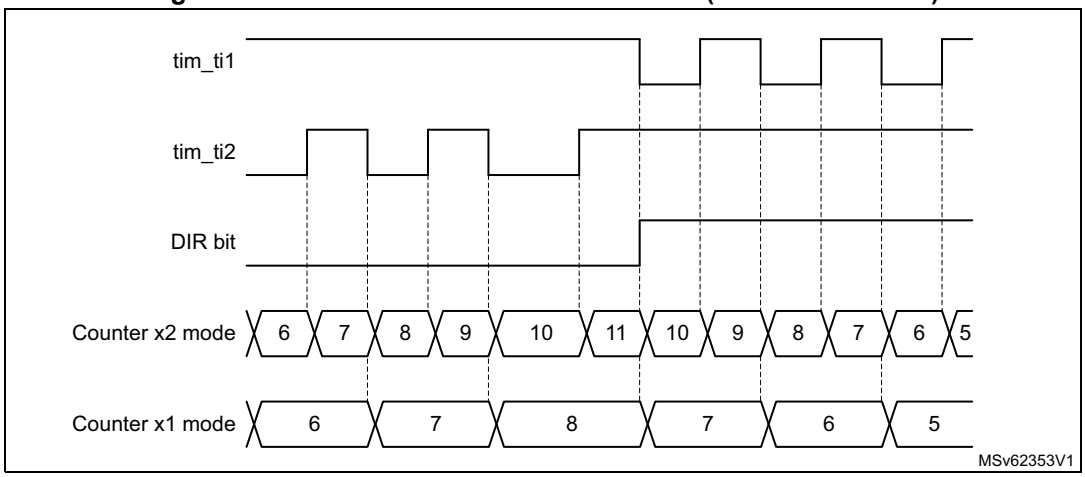


Figure 382. Directional clock encoder mode (CC1P = CC2P = 1)

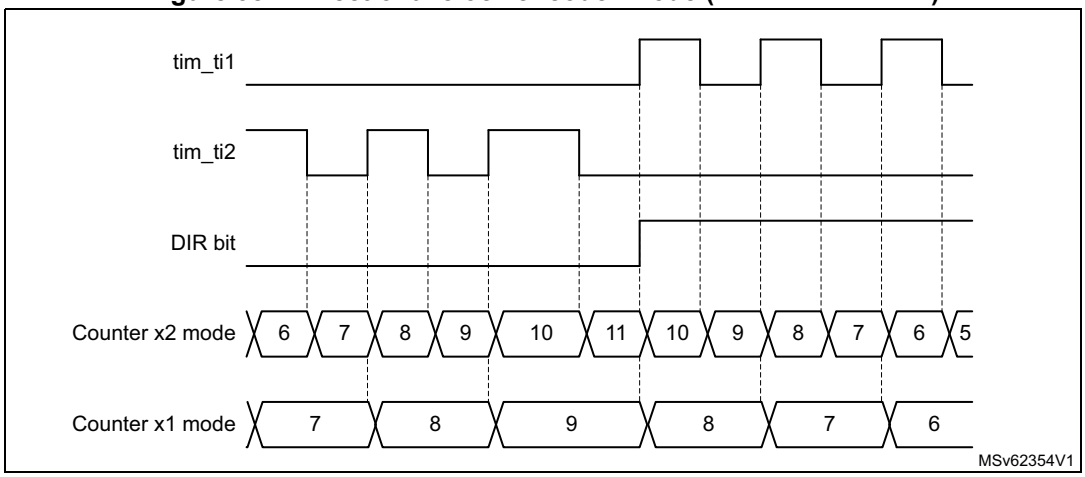


Table 322 details how the directional clock mode operates, for any input transition.

Table 322. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

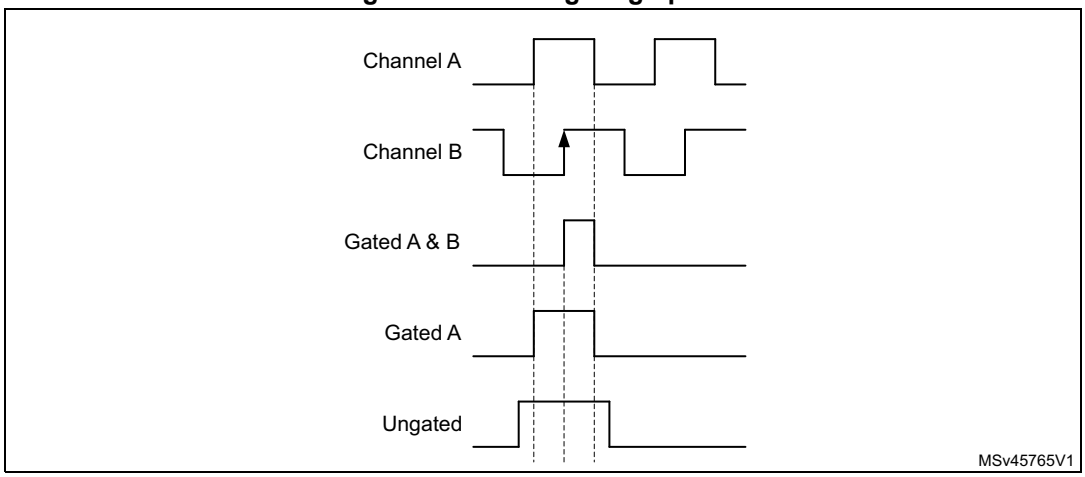
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIM_ECR register. The IE bit must be set only in encoder mode, when the SMS[4:0] bitfield has the following values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 383*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

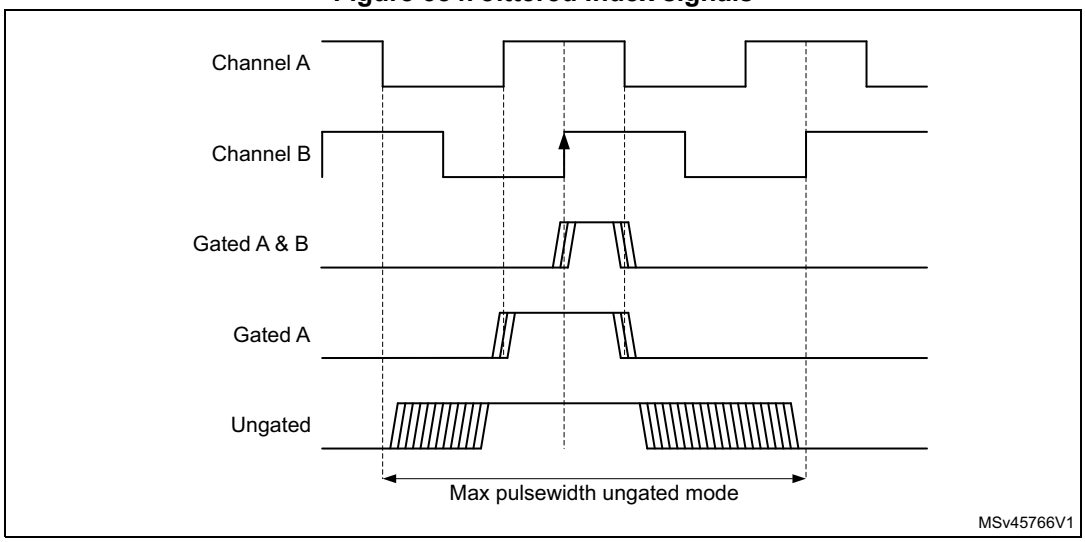
Figure 383. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 384*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 384. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIM_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIM_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 385* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 385. Index generation for IPOS[1:0] = 11

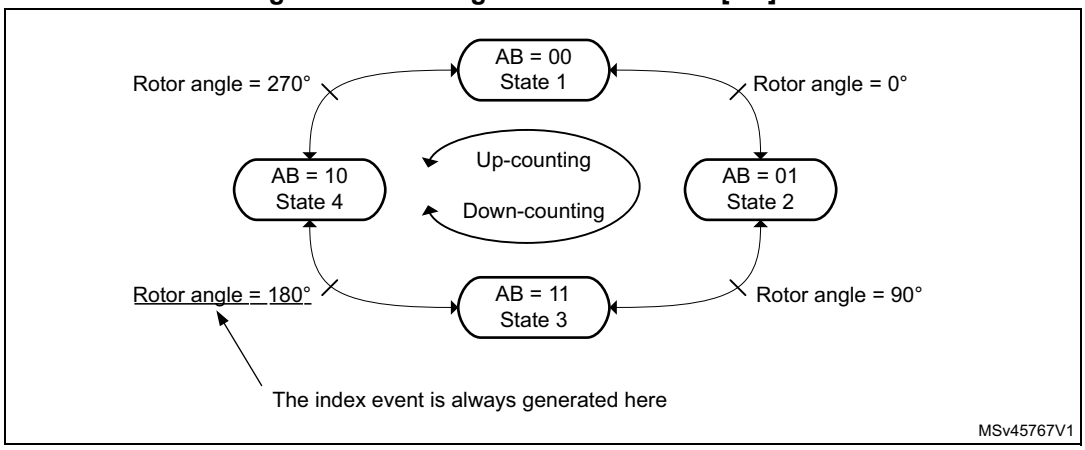


Figure 386 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIM_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 386. Counter reading with index gated on channel A (IPOS[1:0] = 11)

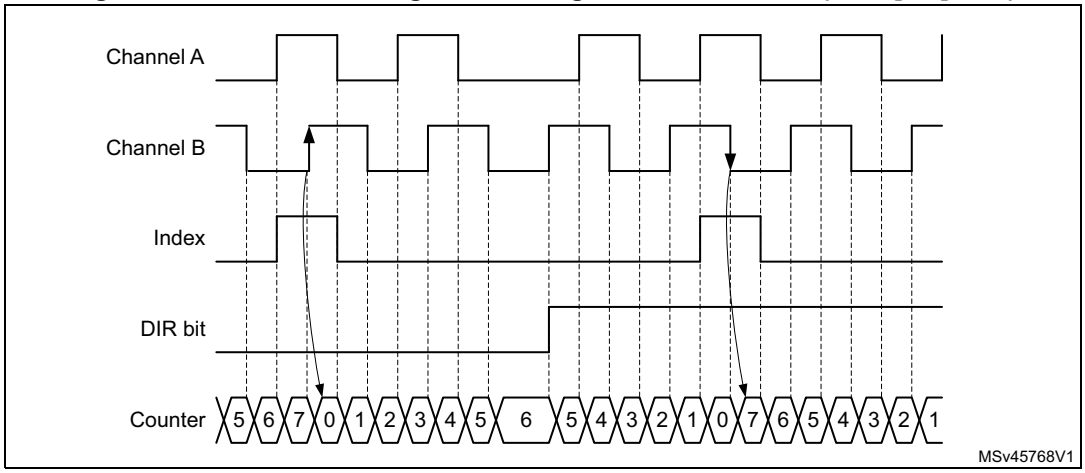


Figure 387 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 387. Counter reading with index ungated (IPOS[1:0] = 00)

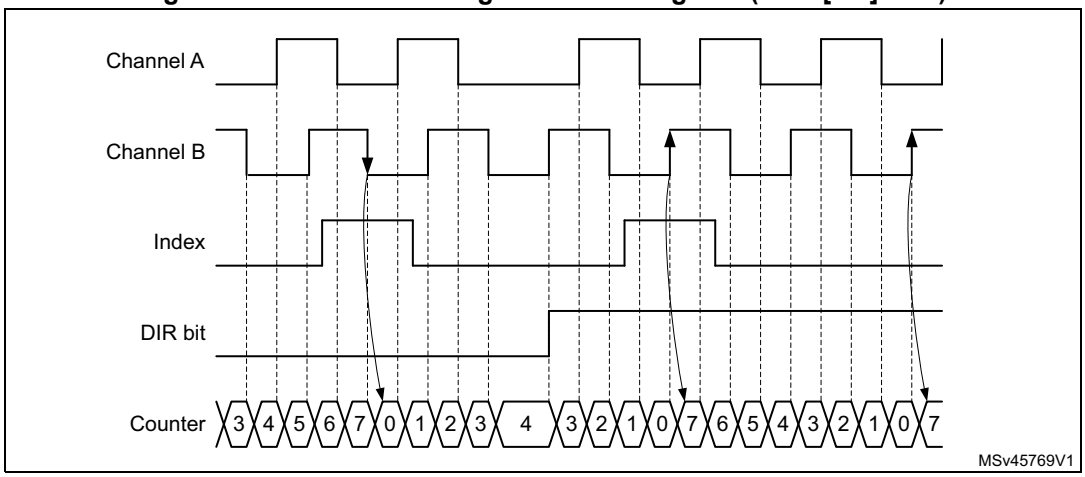


Figure 388 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 388. Counter reading with index gated on channel A and B

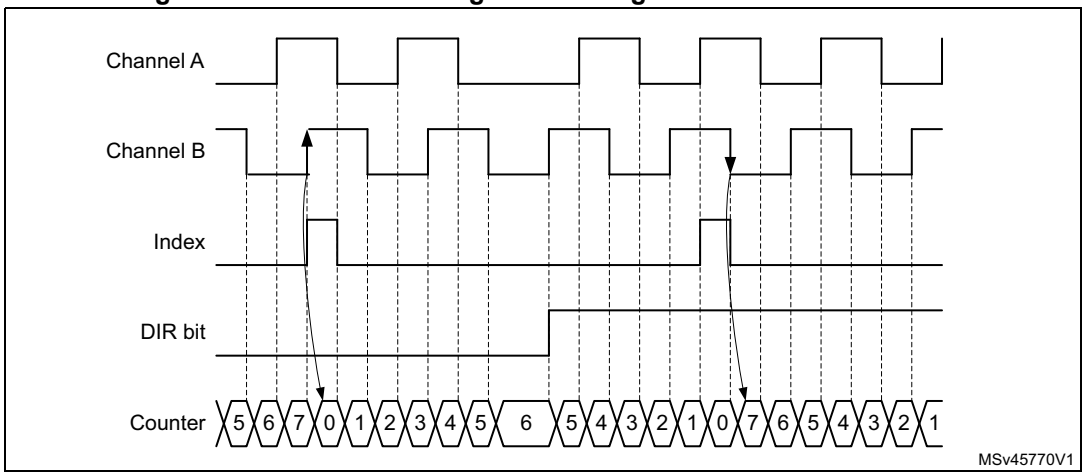


Figure 389 and *Figure 390* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 389. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

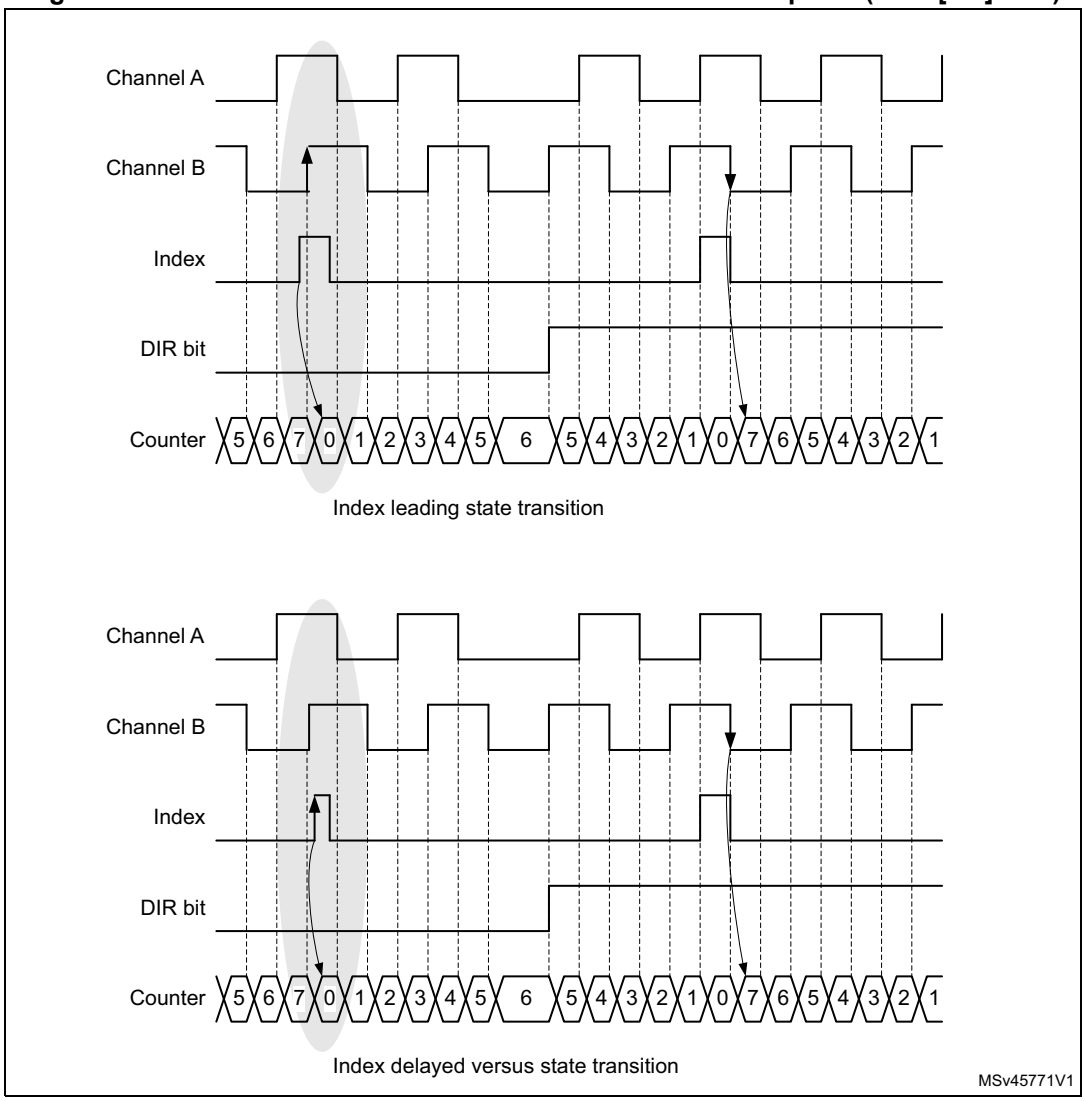


Figure 390. Counter reset Narrow index pulse (closer view, ARR = 0x07)

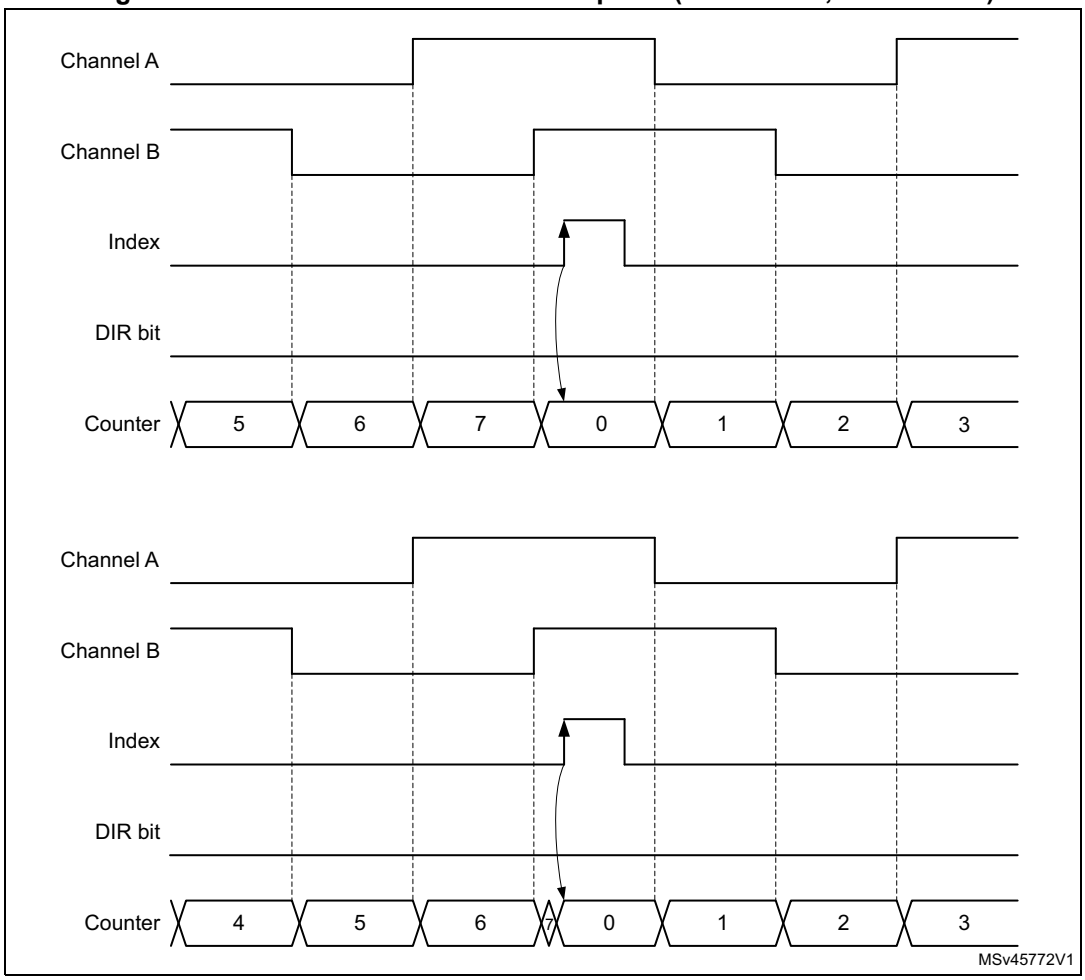
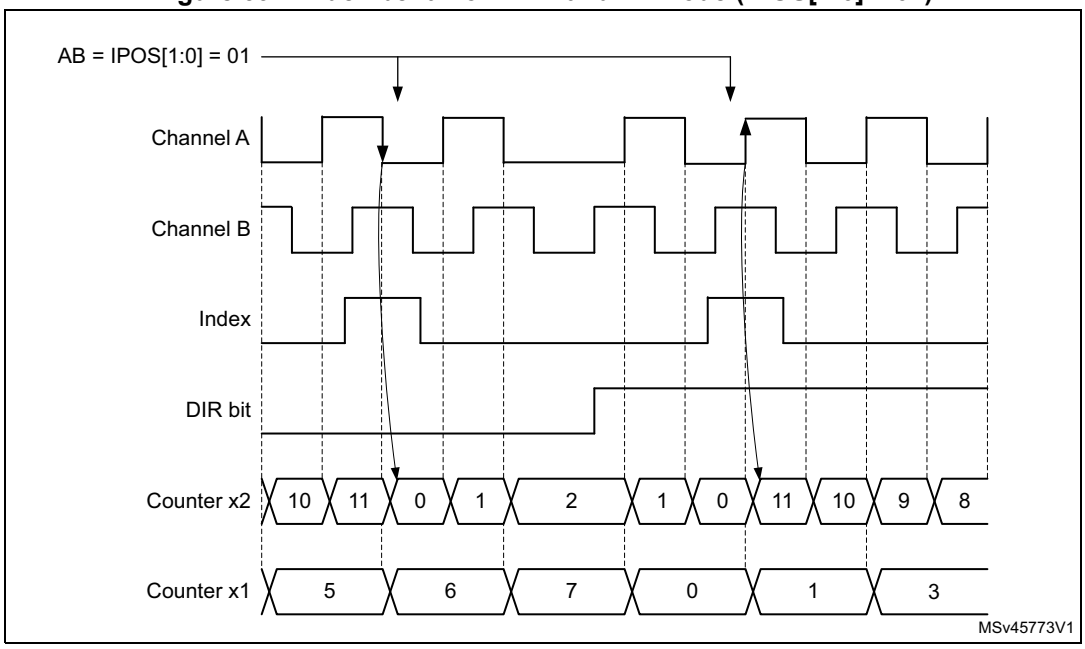


Figure 391 shows how the index is managed in x1 and x2 modes.

Figure 391. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

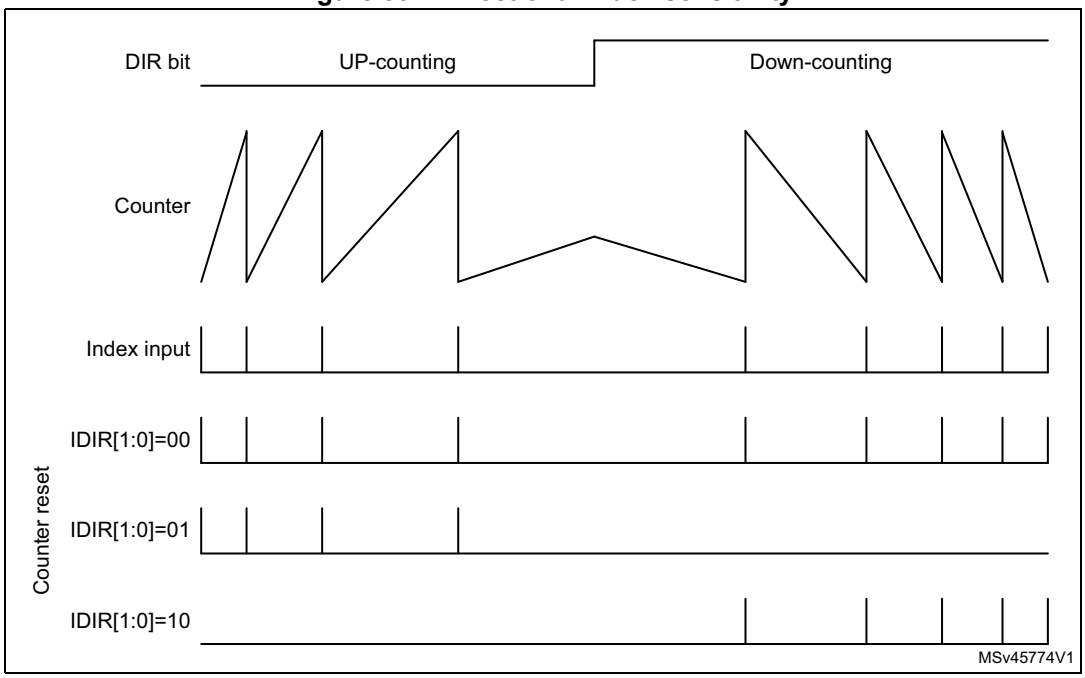


Directional index sensitivity

The IDIR[1:0] bitfield in the TIM_ECR register allows the index to be active only in a selected counting direction.

Figure 392 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

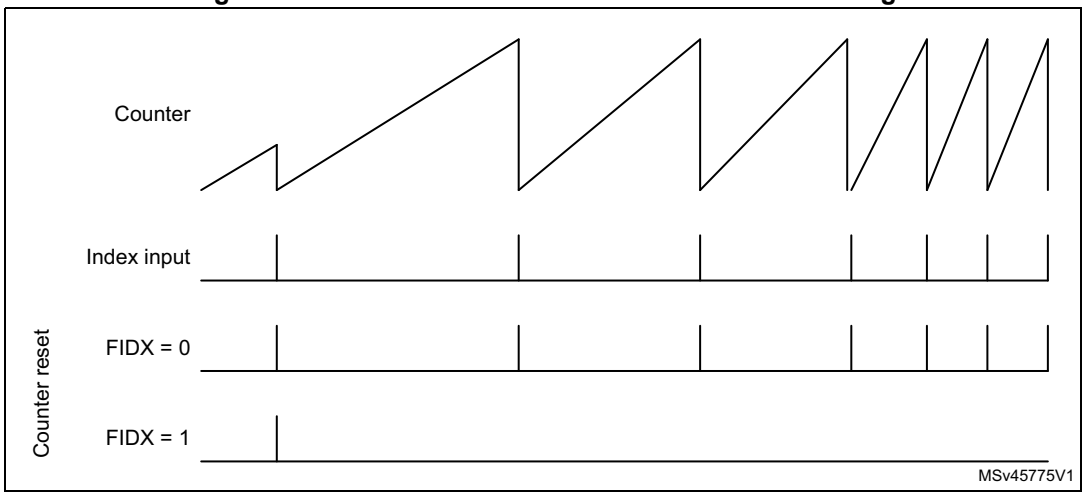
Figure 392. Directional index sensitivity



Special first index event management

The FIDX bit in the TIM_ECR register allows the index to be taken only once, as shown on *Figure 393*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 393. Counter reset as function of FIDX bit setting



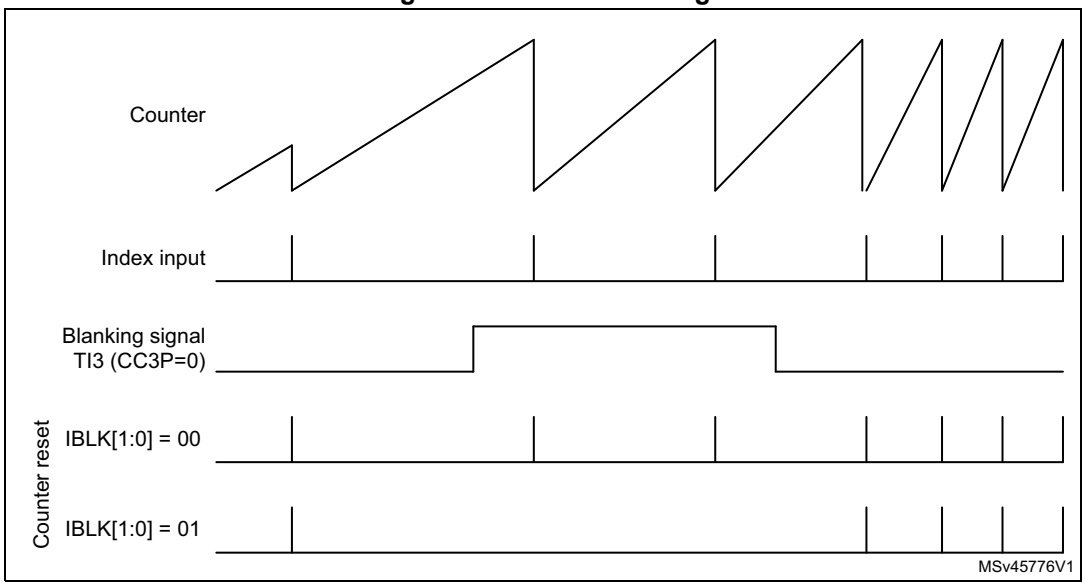
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 394*.

This mode is enabled using the IBLK[1:0] bitfield in the TIM_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 394. Index blanking



Index management in nonquadrature mode

Figure 395 and *Figure 396* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[4:0] bitfield is equal to 01010, 01011, 01100, 01101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 395. Index behavior in clock + direction mode, IPOS[0] = 1

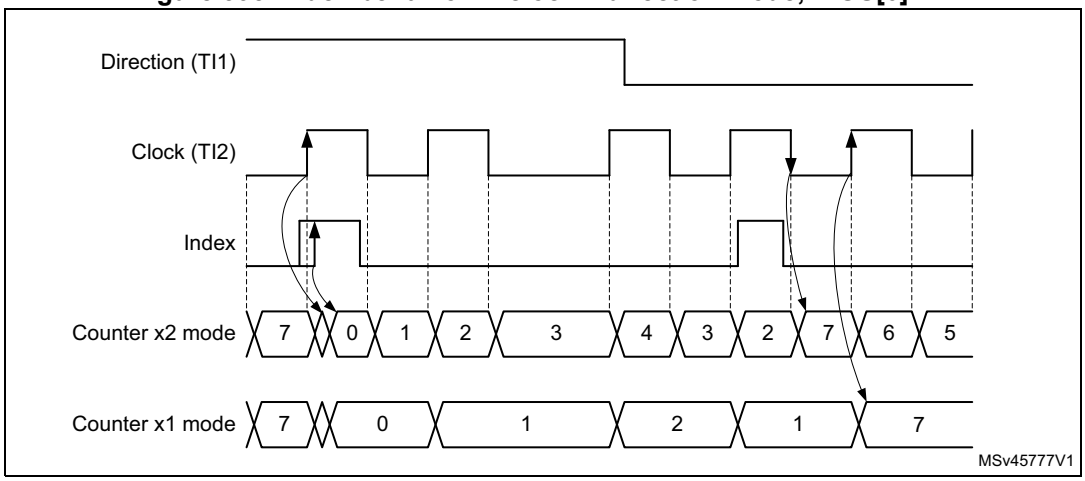
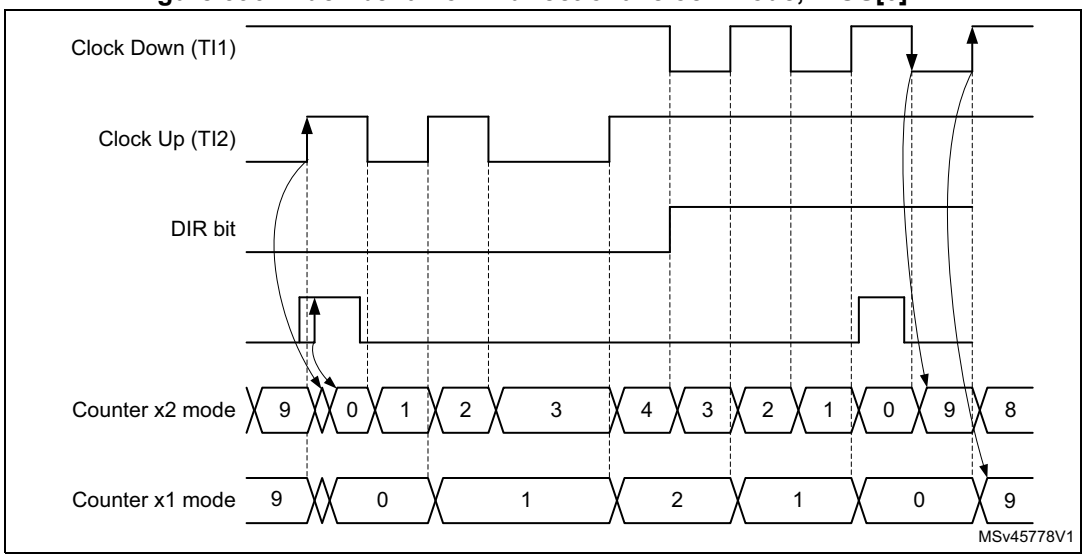


Figure 396. Index behavior in directional clock mode, IPOS[0] = 1

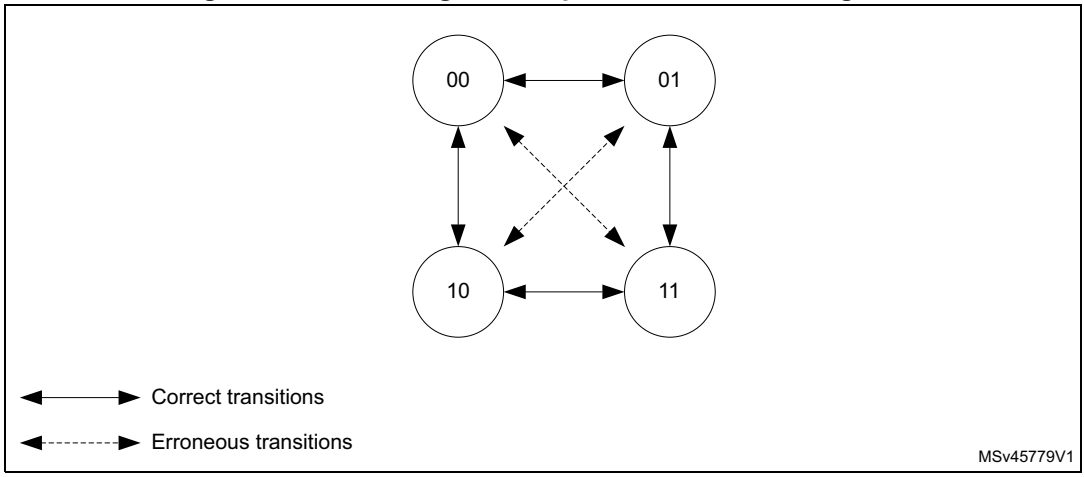


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 397*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status

register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 397. State diagram for quadrature encoded signals



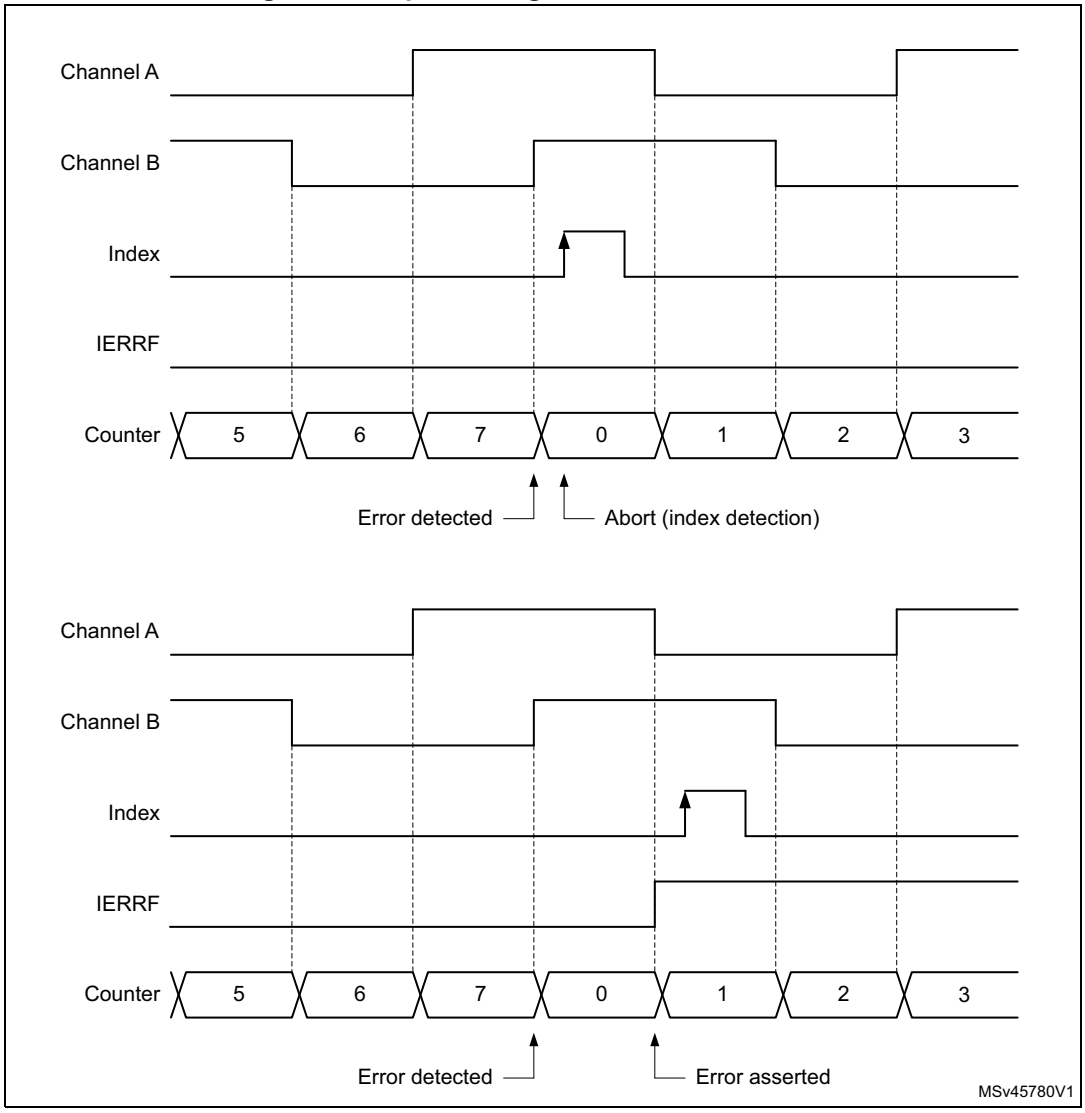
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIM_ARR to 0 or decremented from 0 to TIM_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIM_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIM_ARR = 3999 + 1 = 4000.

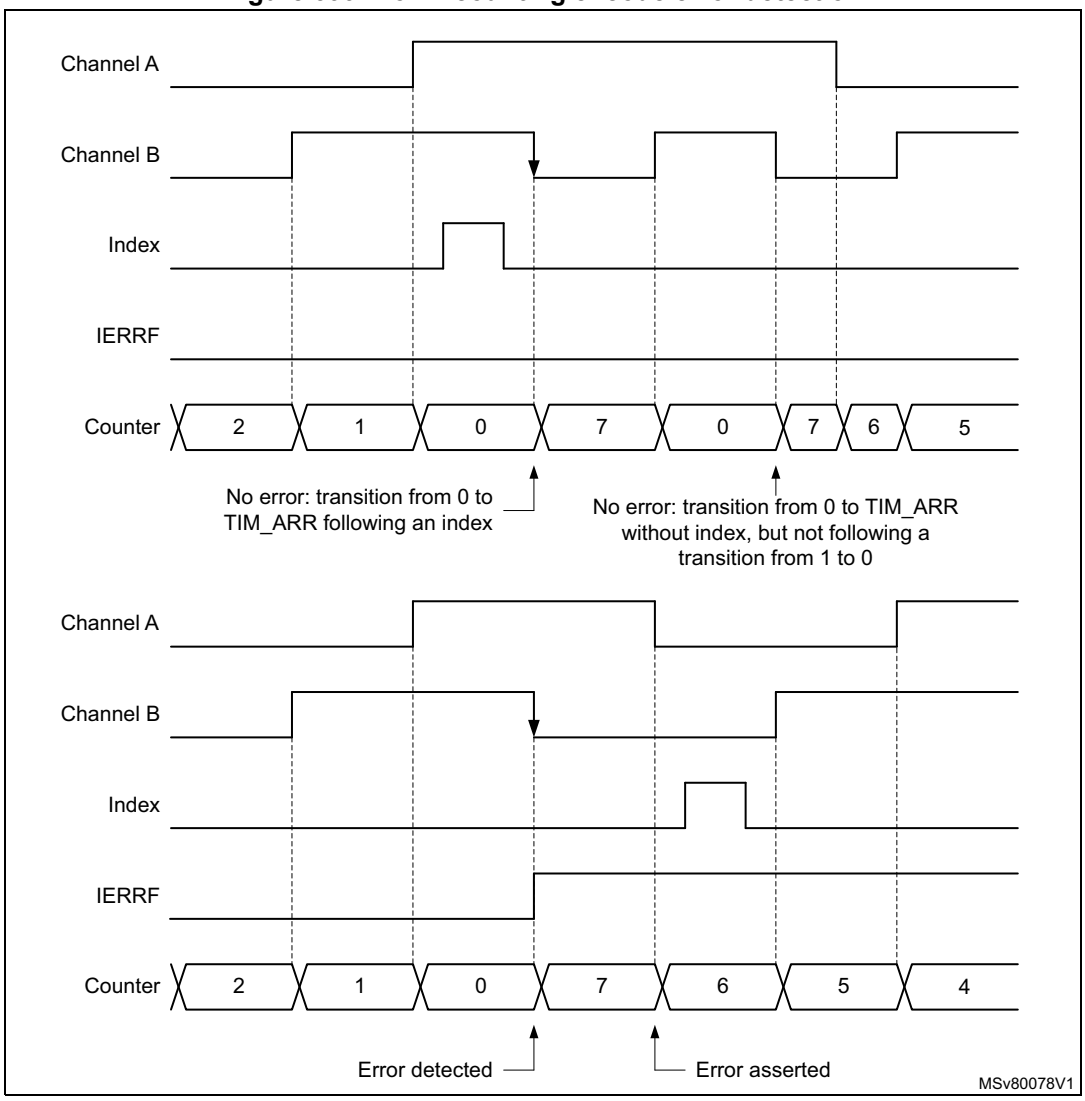
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 398*.

Figure 398. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 399*, to avoid any false error detection in case the encoder dithers between TIM_ARR and 0 immediately after the index detection.

Figure 399. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIM_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIM_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIM_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIM_DIER register.

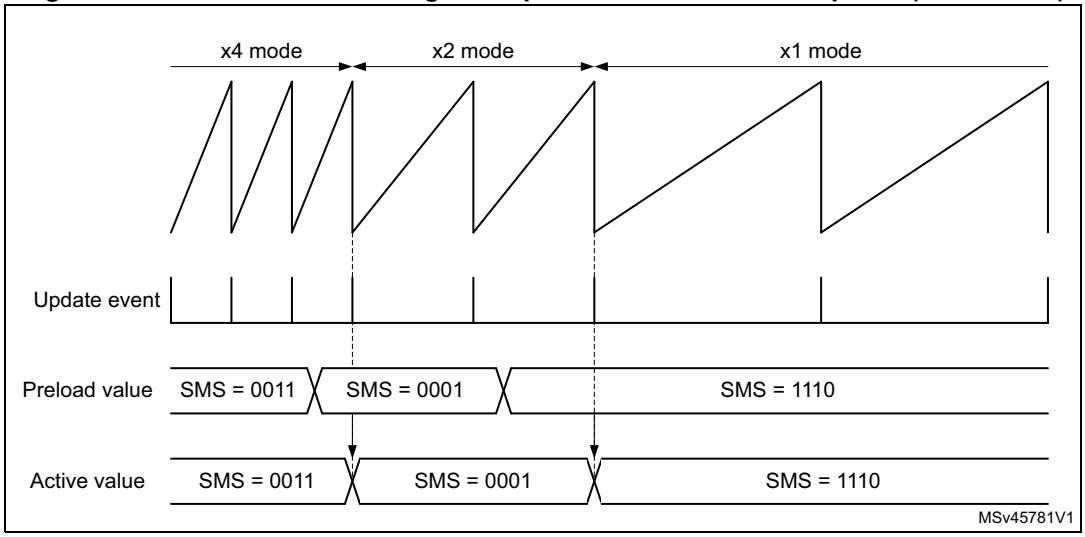
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 400*.

For this purpose, the SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value can be selected with the SMSPS bit in the TIM_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 400. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register. It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

32.4.21 Encoder with built-in debouncer

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 401* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 402* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between two positions.

These two modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 401. Encoder with built-in debouncer, x2 mode

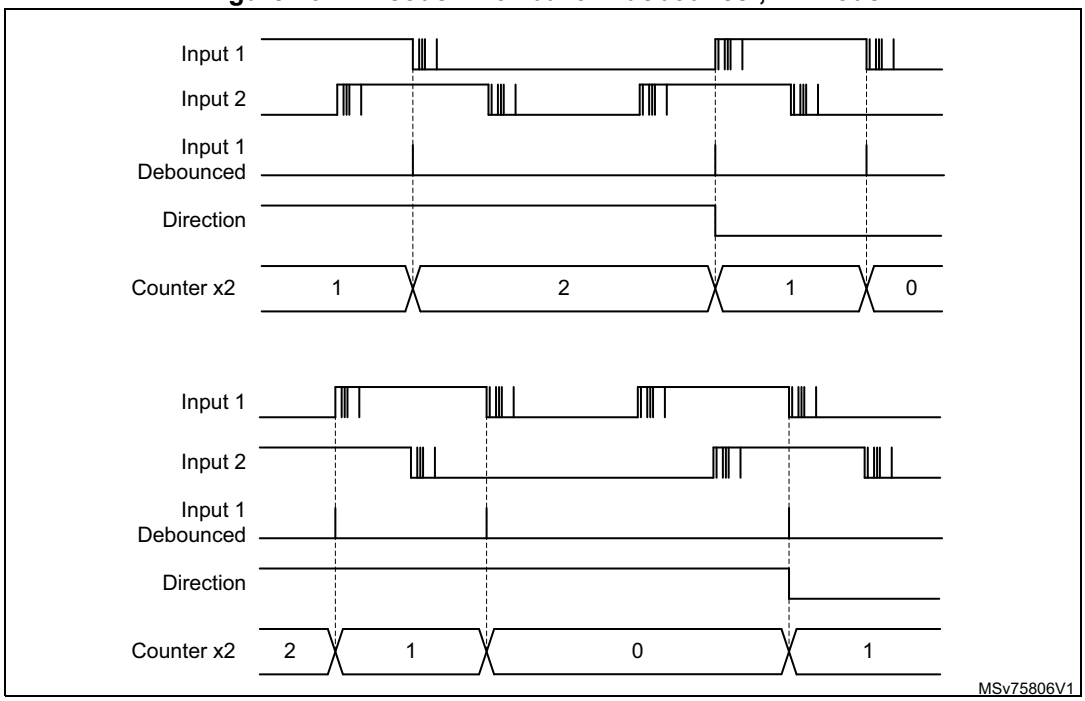
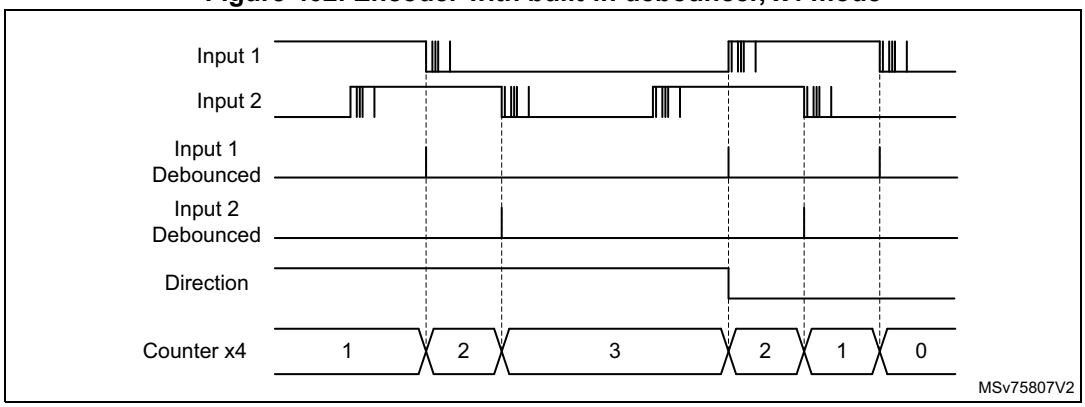


Figure 402. Encoder with built-in debouncer, x4 mode



32.4.22 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIM_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIM_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

32.4.23 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

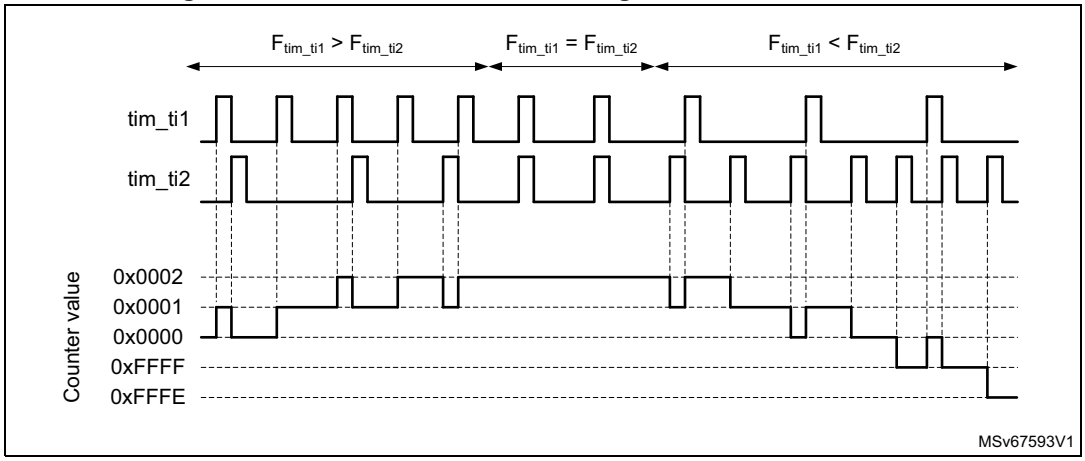
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 403*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value will change as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases.
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases.
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value will dither between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occurring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 403*).

Figure 403. Clock drift measure using directional clock mode



32.4.24 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

32.4.25 Timer input XOR function

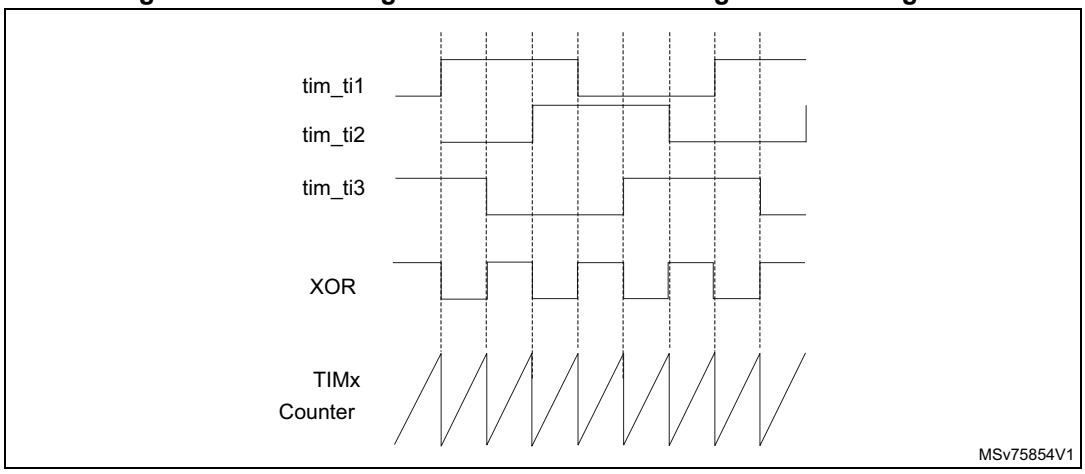
The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 31.3.34: Interfacing with Hall sensors*.

It is convenient to measure the interval between edges on two input signals, as per *Figure 404*.

Figure 404. Measuring time interval between edges on three signals



The XORPS control bit selects the position of the XOR gate, as shown on the *Figure 405* and *Figure 406* below, when the TI1S bit is set.

Figure 405. XOR gate connection with TI1S = 1 and XORPS = 0

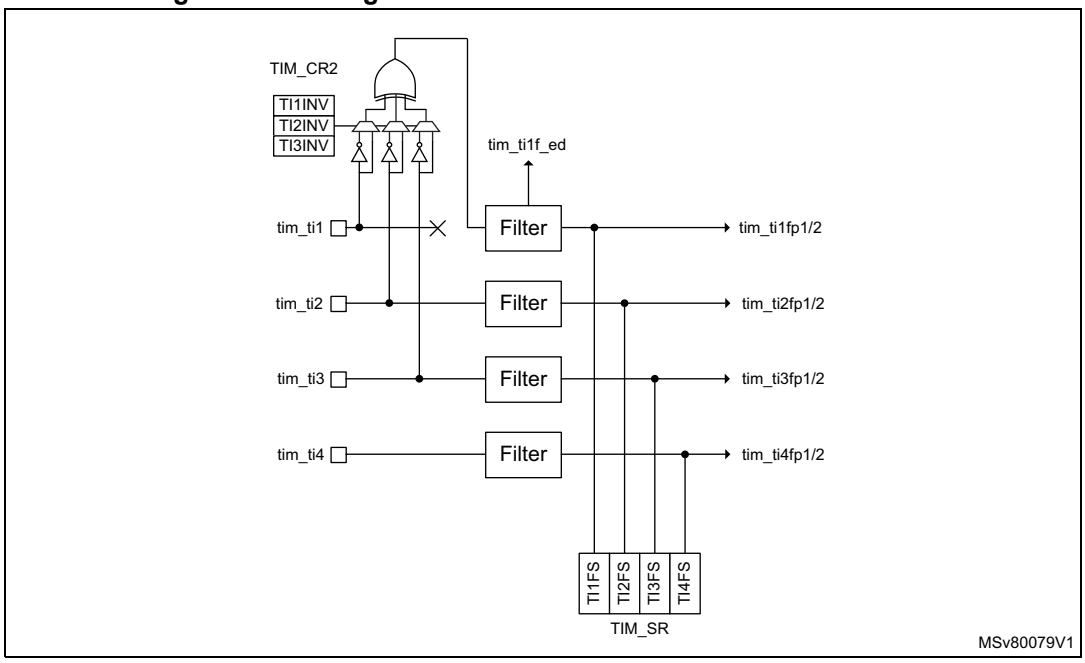
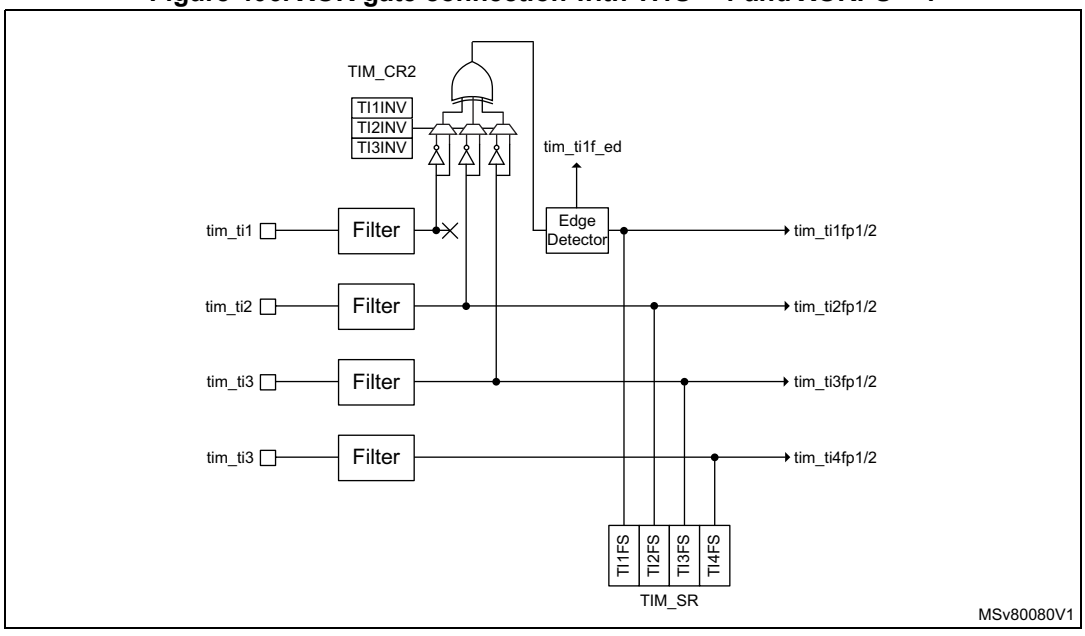


Figure 406. XOR gate connection with TI1S = 1 and XORPS = 1



Three control bits T1INV, T2INV and T3INV in the TIM_CR2 register can invert the tim_in1, tim_in2 and tim_in3 signals prior entering the XOR gate.

The status of the tim_in1, tim_in2, tim_in3 and tim_in4 signals can be software polled using the TI1FS, TI2FS, TI3FS and TI4FS bits in the TIM_SR register (tim_ti4 is not represented on figures above and do not enter into the XOR gate).

32.4.26 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: reset mode, gated mode, trigger mode, reset + trigger and gated + reset modes.

Slave mode: reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

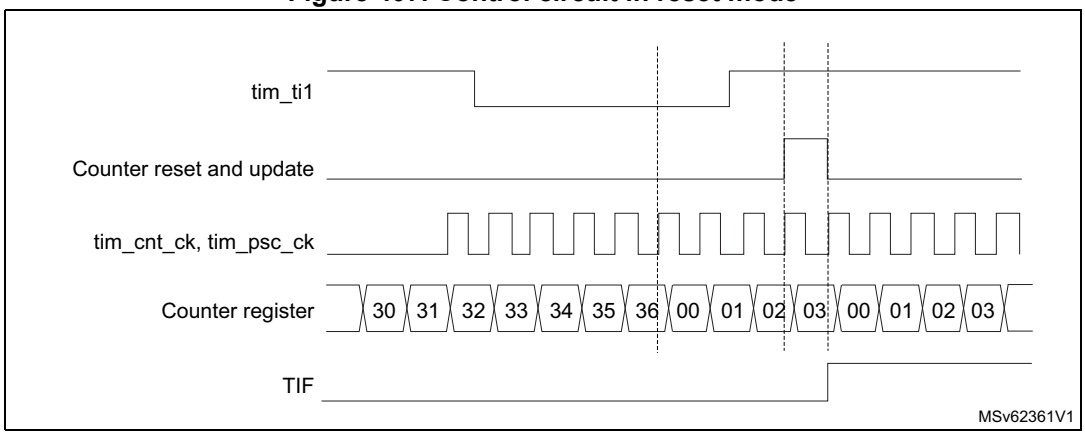
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 407. Control circuit in reset mode



Slave mode: gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The

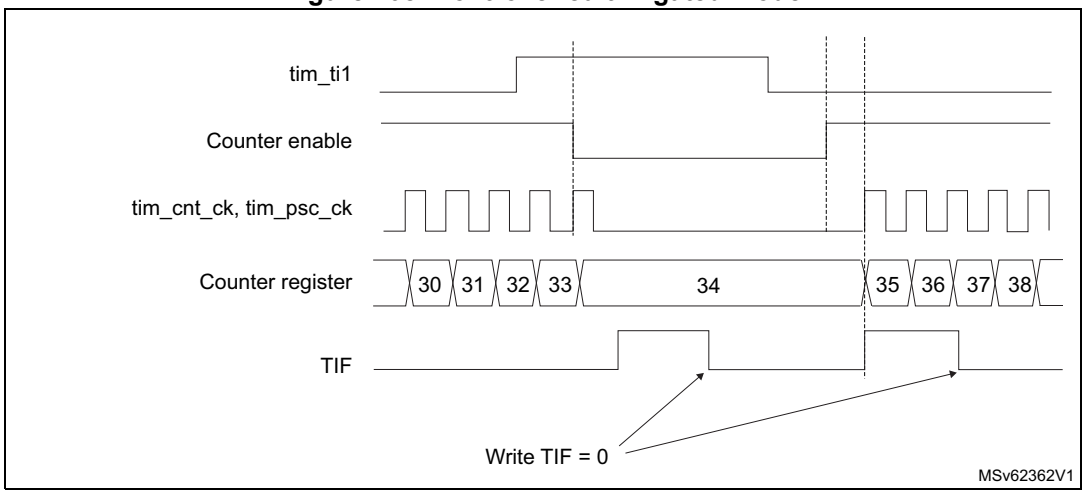
capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 408. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: trigger mode

The counter can start in response to an event on a selected input.

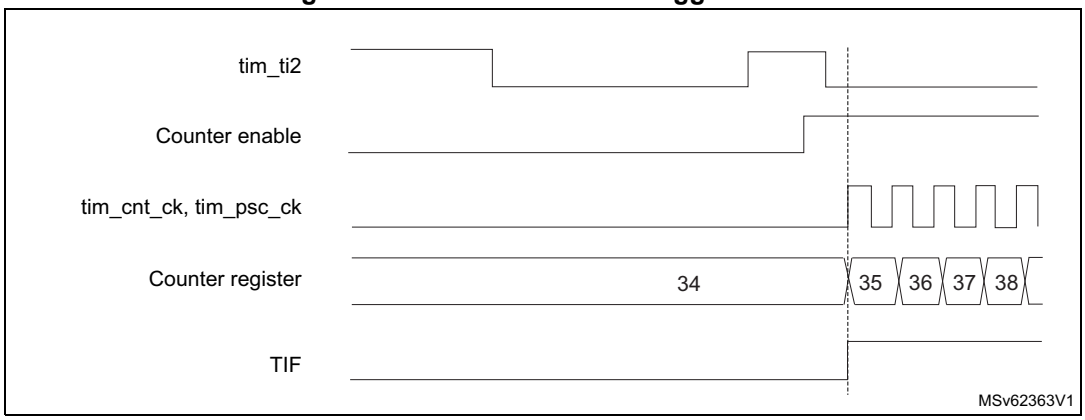
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 409. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIM_SMCR register.

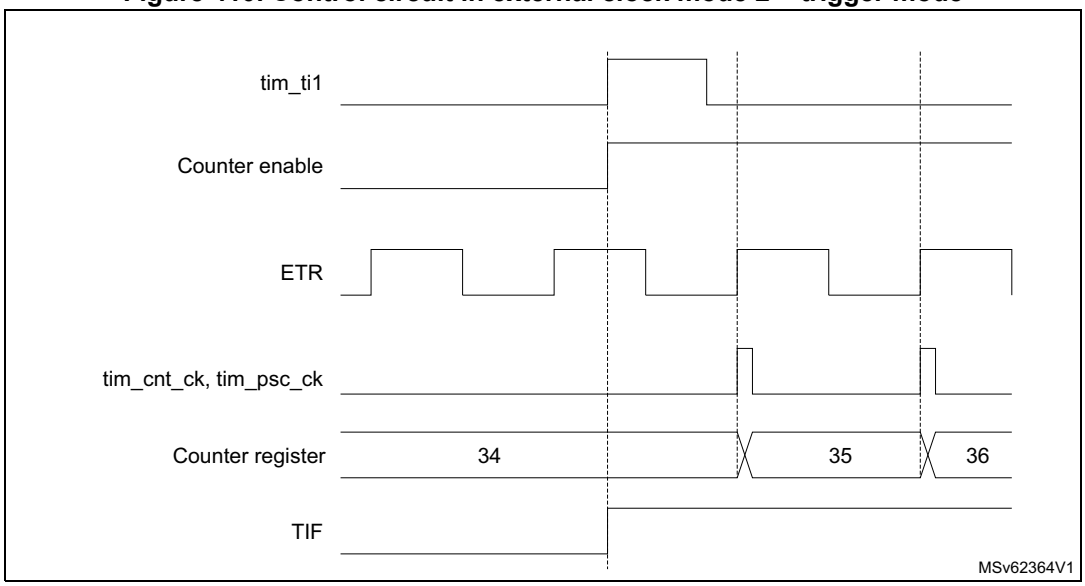
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIM_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIM_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 410. Control circuit in external clock mode 2 + trigger mode



32.4.27 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

Figure 411 and *Figure 412* show examples of master/slave timer connections.

Figure 411. Master/Slave timer example

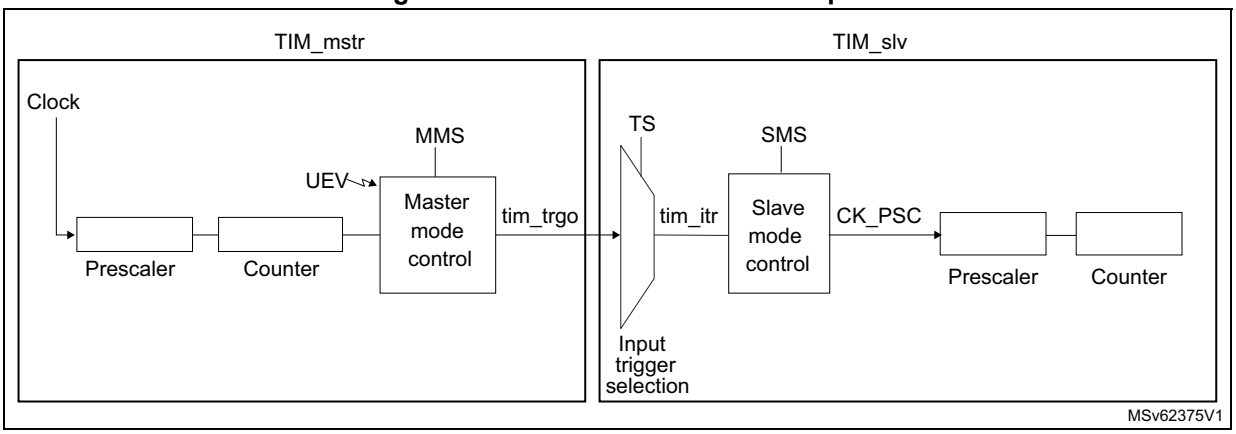
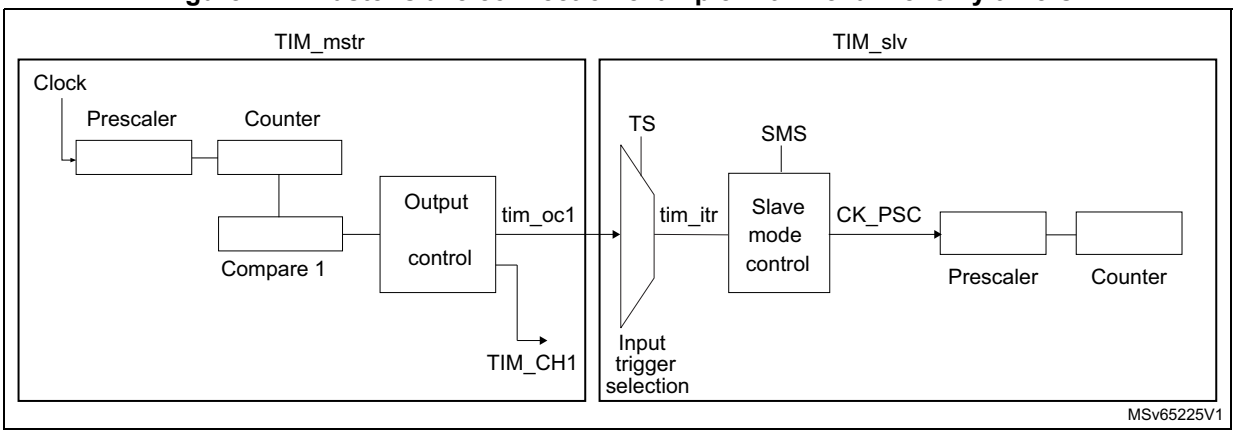


Figure 412. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 412) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals). The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 411*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge

of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).

4. Finally both timers must be enabled by setting their respective CEN bits (TIM_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

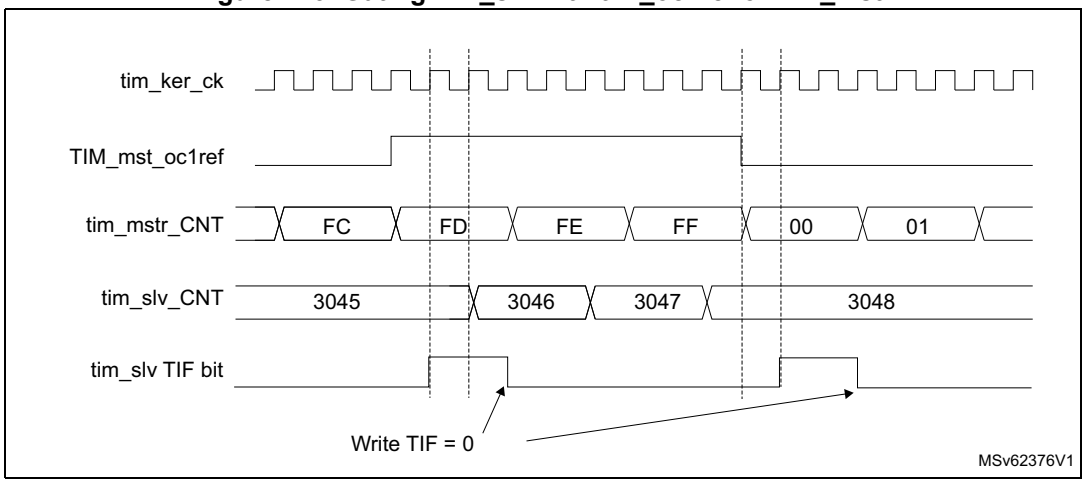
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 411* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 413. Gating TIM_slv with tim_oc1ref of TIM_mstr

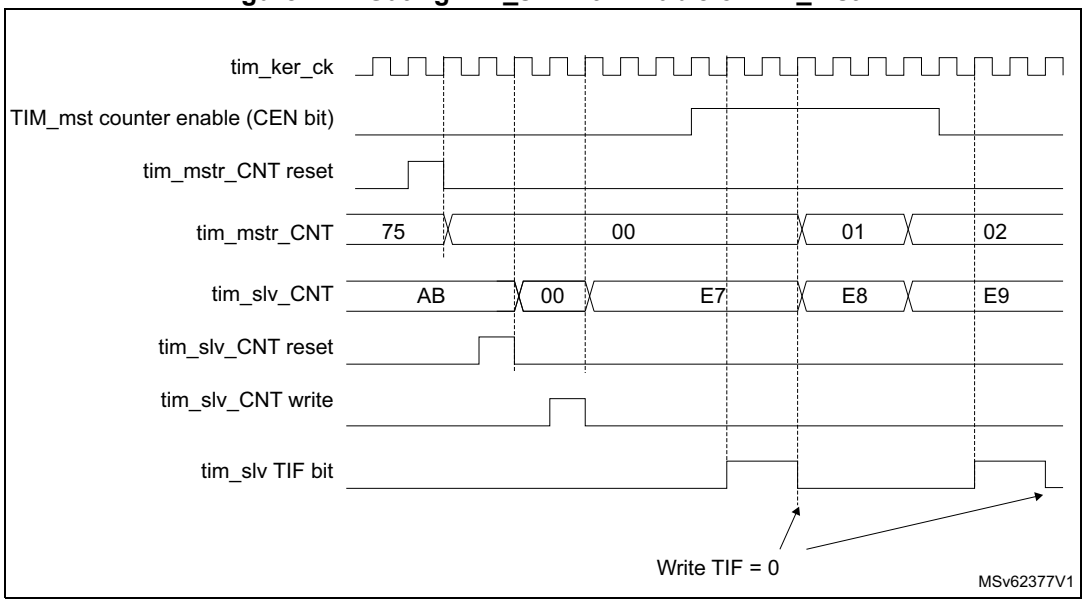


In the example in *Figure 413*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIM_EGR registers.

In the next example (refer to *Figure 414*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 414. Gating TIM_slv with Enable of TIM_mstr

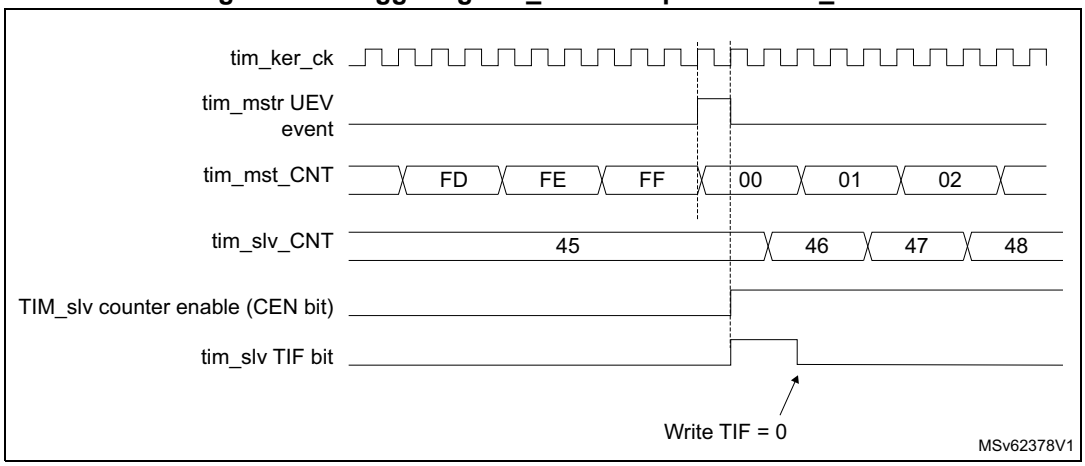


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 411* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

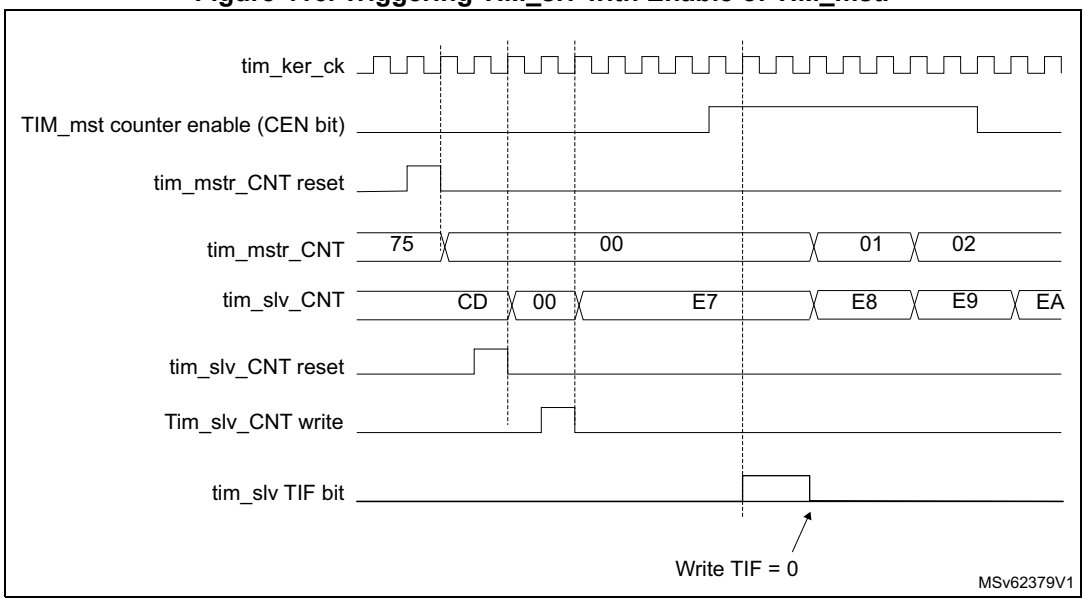
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 415. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 416* shows the behavior with the same configuration as in *Figure 415* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 416. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

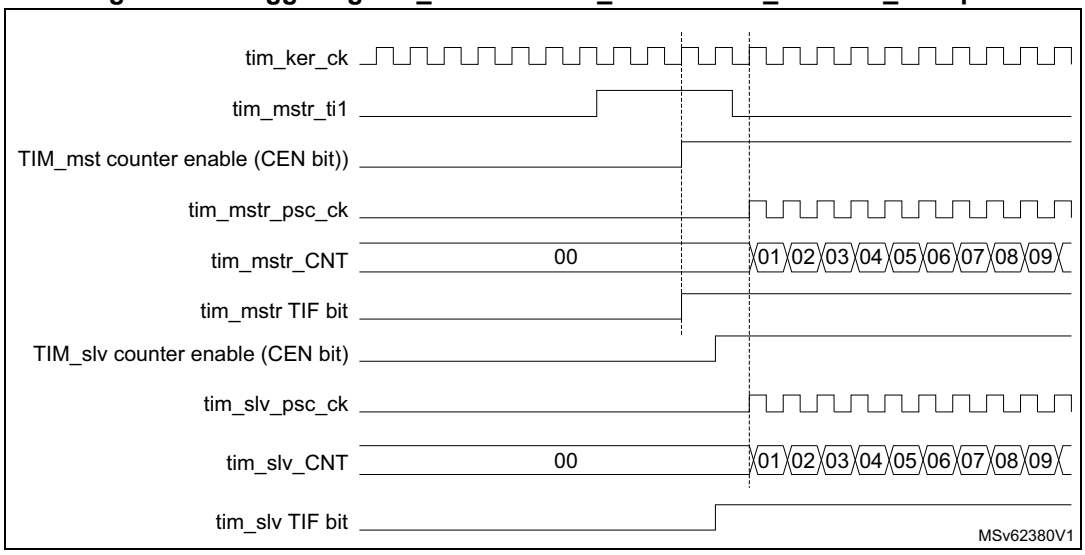
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 411* for connections. To ensure the counters are aligned, TIM_mstr must be configured in master/slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in master/slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIM_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 417. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

32.4.28 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

32.4.29 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period.

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period.

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

32.4.30 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIM_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register:

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 32.5.23: TIM DMA control register (TIM_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

32.4.31 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 323*.

Table 323. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

32.4.32 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

32.4.33 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 324. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

32.4.34 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 325*.

Table 325. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

32.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

32.5.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITH EN	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Autoreload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in center-aligned mode or encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

32.5.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI3INV	TI2INV	TI1INV	AD SYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw		rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bit 31 **TI3INV**: tim_ti3 signal inversion on XOR gate input
0: tim_ti3 non-inverted
1: tim_ti3 inverted

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input
0: tim_ti2 non-inverted
1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input
0: tim_ti1 non-inverted
1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization
0: The timer operates independently from the ADC
1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 32.4.29* for requirements.
The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection
This bit defines the position of the XOR gate with respect to input filters.
0: XOR gate placed before TI1 filter
1: XOR gate placed after TI1, TI2 and TI3 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection
This bit enables the insertion of a XOR gate to combine tim_ti1_in[15:0], tim_ti2_in[15:0], and tim_ti3_in[15:0] inputs into tim_ti1.
The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.
0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 31.3.34: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

32.5.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
		rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **SETPS[3:0]**: Synchronous external trigger prescaler

The linear synchronous prescaler divides the frequency of the tim_etrf signal, by a factor (SETPS[3:0] + 1).

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[4:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[4:0] bitfield is preloaded

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables external clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etrp frequency divided by 2
10: tim_etrp frequency divided by 4
11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

0: tim_ocref_clr_int is connected to the tim_ocref_clr input

1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 32.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 17:16, 2:0 **SMS[4:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010: Encoder mode: Clock plus direction, x2 mode.

01011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

01100: Encoder mode: Directional clock, x2 mode.

01101: Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000: Quadrature encoder with built-in debouncer: x2 mode.

10001: Quadrature encoder with built-in debouncer: x4 mode.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

32.5.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

32.5.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
r	r	r	r				rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 31 **TI4FS**: tim_ti4f status
This flag indicates the value of the tim_ti4f signal (after the digital filtering stage), for polling purpose.
0: tim_ti4f signal is low
1: tim_ti4f signal is high

Bit 30 **TI3FS**: tim_ti3f status
This flag indicates the value of the tim_ti3f signal (after the digital filtering stage), for polling purpose.
0: tim_ti3f signal is low
1: tim_ti3f signal is high

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIM_CR1 register.
When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

32.5.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIM_ARR) if DIR = 1 (down-counting).

32.5.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000:No filter, sampling is done at f_{DTS}

0001:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100:$f_{SAMPLING} = f_{DTS}/2$, N = 6

0101:$f_{SAMPLING} = f_{DTS}/2$, N = 8

0110:$f_{SAMPLING} = f_{DTS}/4$, N = 6

0111:$f_{SAMPLING} = f_{DTS}/4$, N = 8

1000:$f_{SAMPLING} = f_{DTS}/8$, N = 6

1001:$f_{SAMPLING} = f_{DTS}/8$, N = 8

1010:$f_{SAMPLING} = f_{DTS}/16$, N = 5

1011:$f_{SAMPLING} = f_{DTS}/16$, N = 6

1100:$f_{SAMPLING} = f_{DTS}/16$, N = 8

1101:$f_{SAMPLING} = f_{DTS}/32$, N = 5

1110:$f_{SAMPLING} = f_{DTS}/32$, N = 6

1111:$f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

32.5.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M [4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M [4:3]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

0: tim_oc2ref is not affected by the tim_ocref_clr_int signal

1: tim_oc2ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 32.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode

Refer to OC1M description on bits 17:16, 6:4.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Note: This mode is not available for all PWM modes. Refer to Section 32.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - In up-counting, channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIM_CNT>TIM_CCR1 else active (tim_oc1ref = 1).

00111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active. In down-counting, channel 1 is active as long as TIM_CNT>TIM_CCR1 else inactive.

01000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Reserved.

01011: Reserved.

01100-01101: Combined PWM mode 1 and 2 - Refer to *Table 320: Combined PWM mode overview* for details.

01110-01111: Asymmetric PWM mode 1 and 2 - Refer to *Table 319: Asymmetric PWM modes* for details.

Others: Reserved.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

32.5.9 TIM capture/compare mode register 2 (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

32.5.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
0: tim_oc4ref is not affected by the tim_ocref_clr_int signal
1: tim_oc4ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)
Note: This mode is not available for all PWM modes. Refer to Section 32.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC4M[4:0]**: Output compare 4 mode
Refer to OC3M[4:0].

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

0: tim_oc3ref is not affected by the tim_ocref_clr_int signal
1: tim_oc3ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 32.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC3M[4:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR3 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00011: Toggle - tim_oc3ref toggles when TIM_CNT = TIM_CCR3.

00100: Force inactive level - tim_oc3ref is forced low.

00101: Force active level - tim_oc3ref is forced high.

00110: PWM mode 1 - In up-counting, channel 3 is active as long as TIM_CNT<TIM_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIM_CNT>TIM_CCR3 else active (tim_oc3ref = 1).

00111: PWM mode 2 - In up-counting, channel 3 is inactive as long as TIM_CNT<TIM_CCR3 else active. In down-counting, channel 3 is active as long as TIM_CNT>TIM_CCR3 else inactive.

01000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Pulse on compare - A pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIM_ECR.

01011: Direction output - The tim_oc3ref signal is overridden by a copy of the DIR bit.

01100-01101: Combined PWM mode 1 and 2 - Refer to *Table 320: Combined PWM mode overview* for details.

01110-01111: Asymmetric PWM mode 1 and 2 - Refer to *Table 319: Asymmetric PWM modes* for details.

10110-11001: Asymmetric PWM mode 7 to 10 - Available on OC3M only. Refer to *Table 319: Asymmetric PWM modes* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

32.5.11 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: Capture/compare 4 output Polarity.
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/compare 4 output Polarity.
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable.
refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 output Polarity.
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/compare 3 output Polarity.
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable.
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 output Polarity.
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/compare 2 output Polarity.
refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable.
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/compare 1 output Polarity.

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1:non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:this configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/compare 1 output enable.

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 326. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

32.5.12 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY_CNT[31]	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIM_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIM_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

32.5.13 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

32.5.14 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 32.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

32.5.15 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

32.5.16 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

32.5.17 TIM capture/compare register 3 (TIM_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIM_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

32.5.18 TIM capture/compare register 4 (TIM_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIM_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

32.5.19 TIM timer encoder control register (TIM_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[4:0] = 00001, 00010, 00011, 01110, 01111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[4:0] = 01010, 01011, 01100, 01101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

32.5.20 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

32.5.21 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[4:2]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:14 **ETRSEL[4:0]**: etr_in source selection
These bits select the etr_in input source.
00000: tim_etr0: TIM_ETR input
00001: tim_etr1
...
11111: tim_etr31
Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

32.5.22 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
0000: tim_ocref_clr0
0001: tim_ocref_clr1
...
0111: tim_ocref_clr15
Refer to *Section 32.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

32.5.23 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Example: Let us consider the following transfer: DBL = 6 bytes & DBA = TIM2_CR1.

–If DBL = 6 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation: (TIM_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIM_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIM_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:
00000: TIM_CR1,
00001: TIM_CR2,
00010: TIM_SMCR,
...

32.5.24 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

32.5.25 TIM2/TIM3/TIM4/TIM5 register map

TIM2/TIM3/TIM4/TIM5 registers are mapped as described in the table below.

Table 327. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0	0	0	0	0	0	0	0
0x004	**TIM_CR2**	TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value	0	0	0	0	0		0																		0	0	0	0	0			
0x008	**TIM_SMCR**	Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS [4:3]		ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value			0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0		0	0	0	0	0	0	0		0	0	0	0	0
0x010	**TIM_SR**	TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value	0	0	0	0				0	0	0	0	0								0	0	0	0			0		0	0	0	0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	**TIM_CCMR1** Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 327. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**TIM_CCMR2** Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR2 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]		O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	**TIM_CNT**	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIM_ARR**	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved																																
0x034	**TIM_CCR1**	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIM_CCR2**	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**TIM_CCR3**	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	**TIM_CCR4**	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	**TIM_ECR**	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	**TIM_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0	0	0														

Table 327. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x064	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

33 Basic timers (TIM6/TIM7)

33.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

33.2 TIM6/TIM7 main features

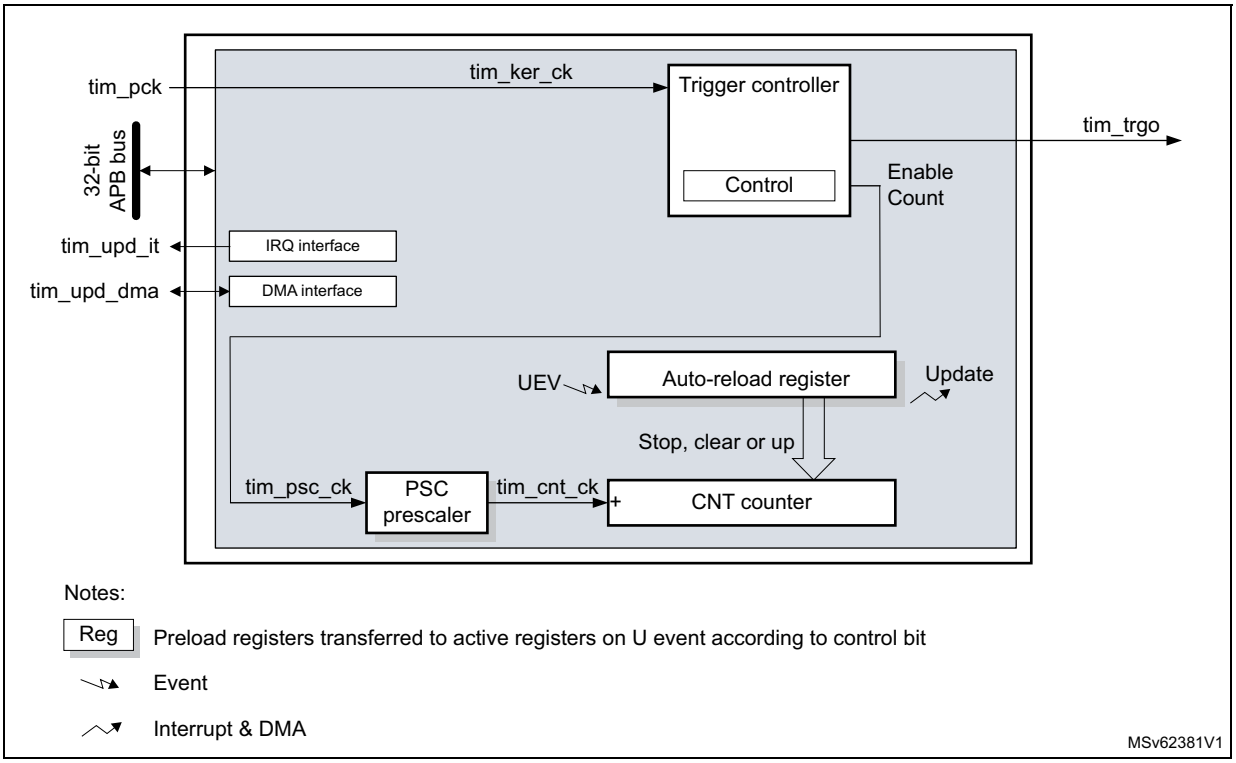
Basic timer (TIM6/TIM7) features include:

- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generation on the update event: counter overflow.
- ADC synchronization for jitter-free sampling points.

33.3 TIM6/TIM7 functional description

33.3.1 TIM6/TIM7 block diagram

Figure 418. Basic timer block diagram



33.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 328. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

33.3.3 TIM6/TIM7 clocks

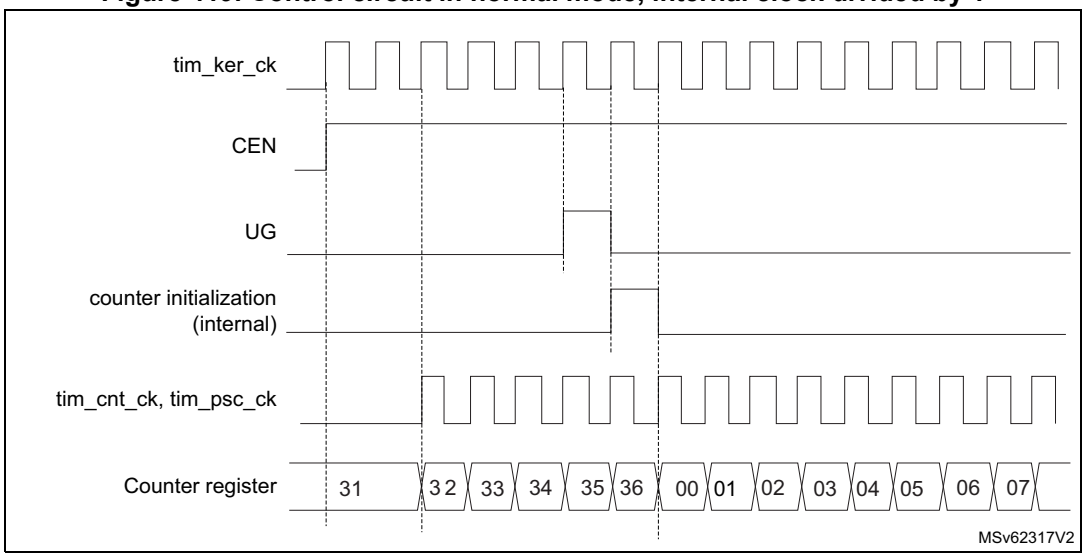
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 419 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 419. Control circuit in normal mode, internal clock divided by 1



33.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Auto-Reload register (TIM_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIM_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIM_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as the TIM_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 420 and *Figure 421* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 420. Counter timing diagram with prescaler division change from 1 to 2

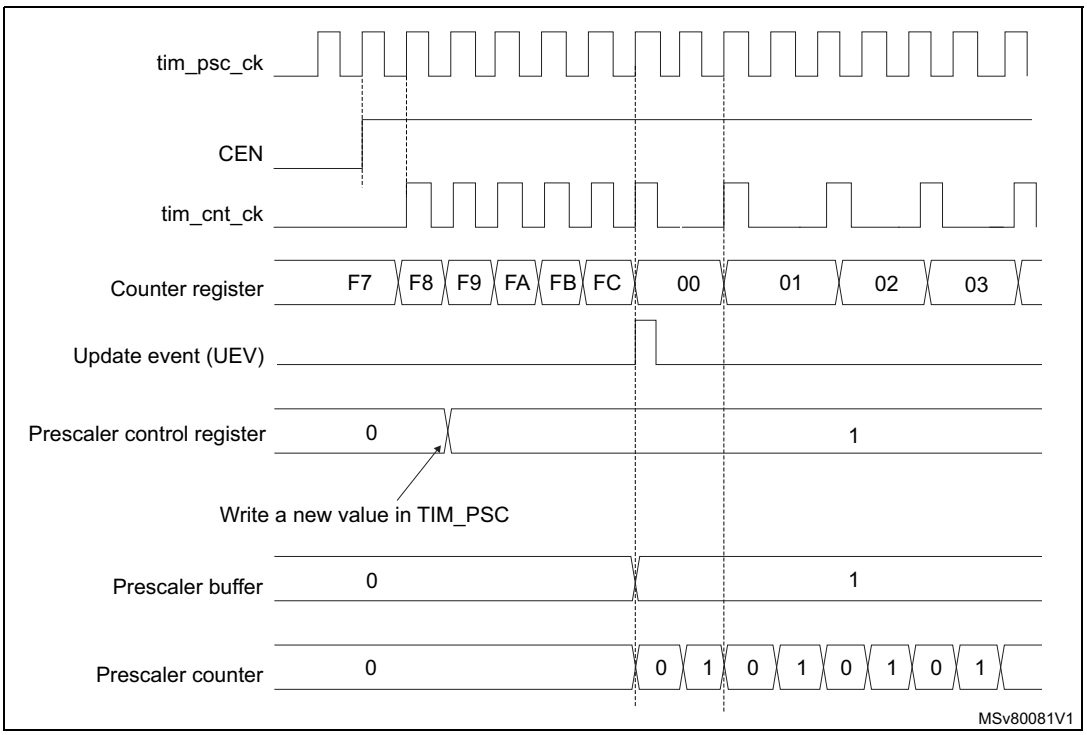
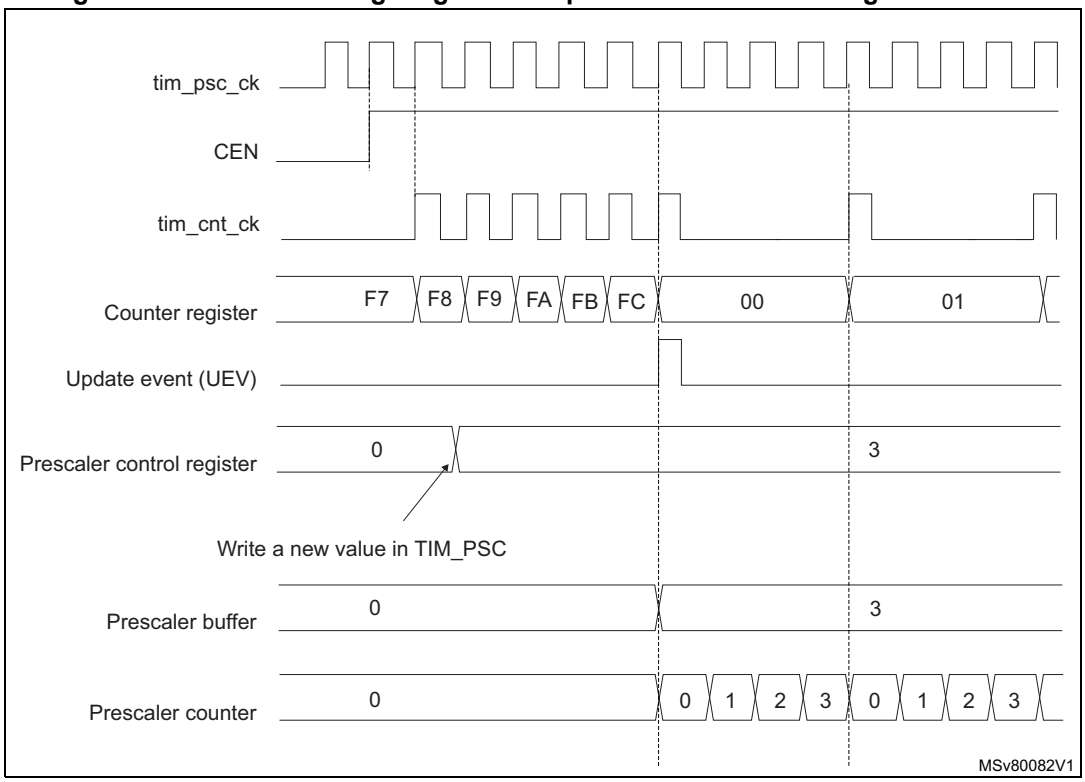


Figure 421. Counter timing diagram with prescaler division change from 1 to 4



33.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIM_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIM_PSC register).
- The autoreload shadow register is updated with the preload value (TIM_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 422. Counter timing diagram, internal clock divided by 1

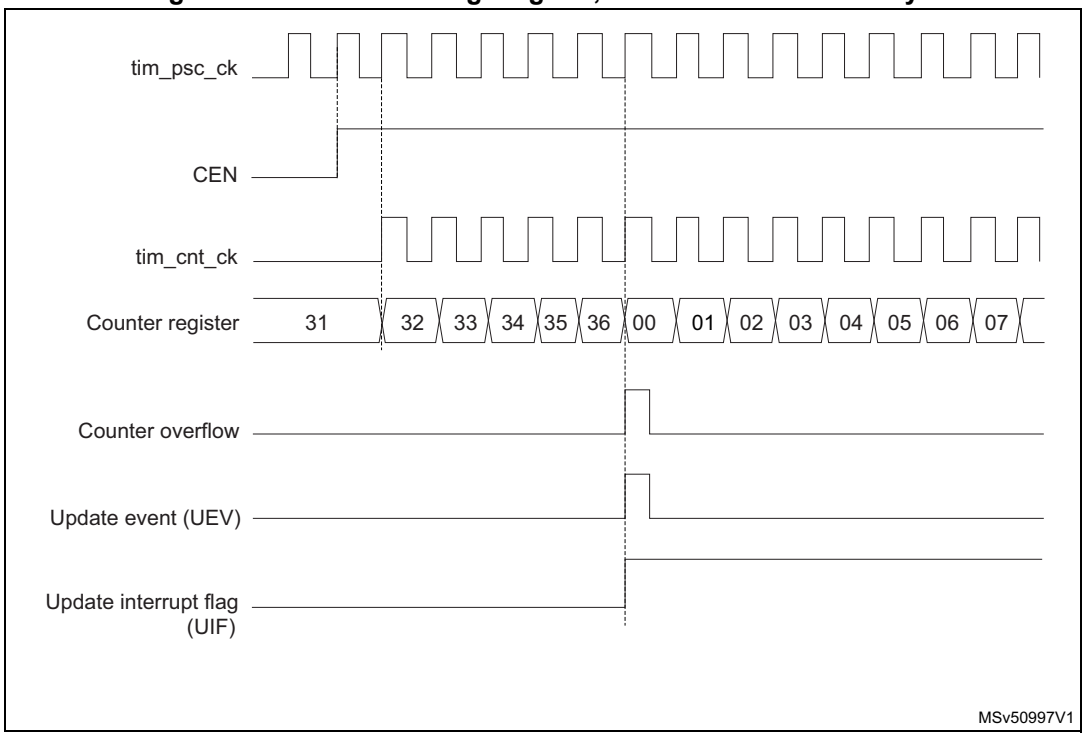


Figure 423. Counter timing diagram, internal clock divided by 2

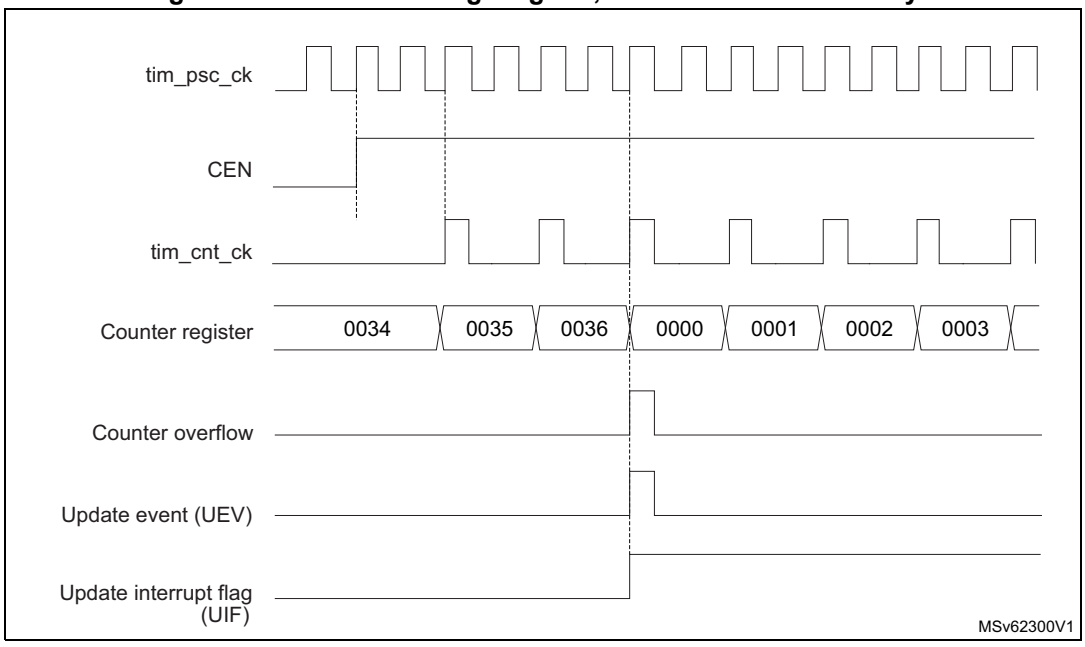


Figure 424. Counter timing diagram, internal clock divided by 4

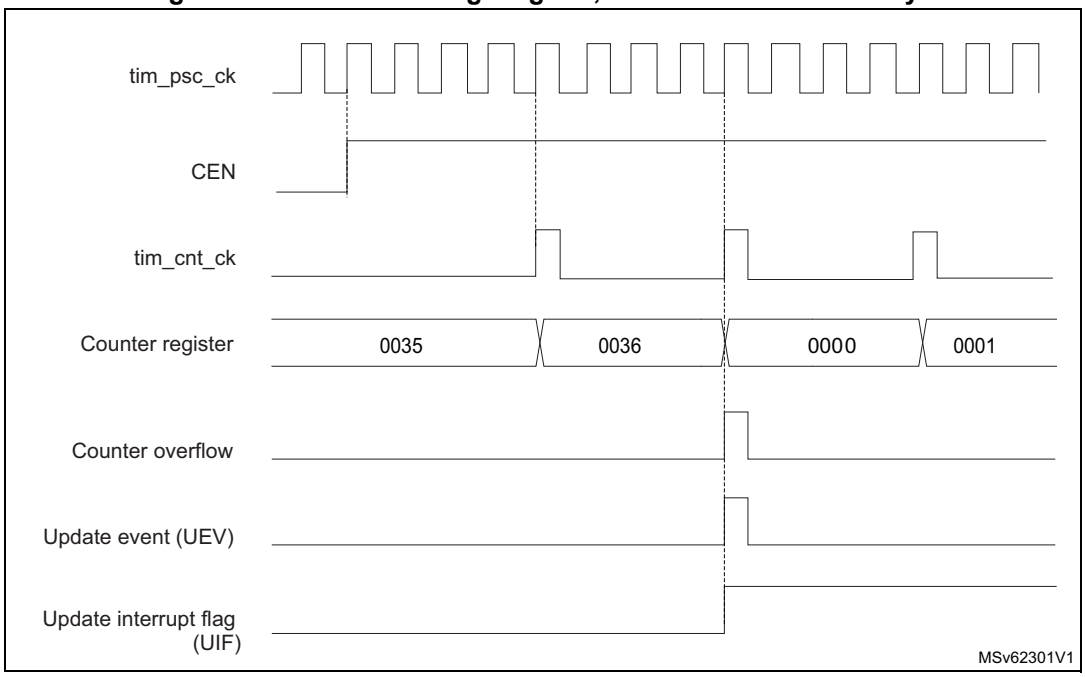


Figure 425. Counter timing diagram, internal clock divided by N

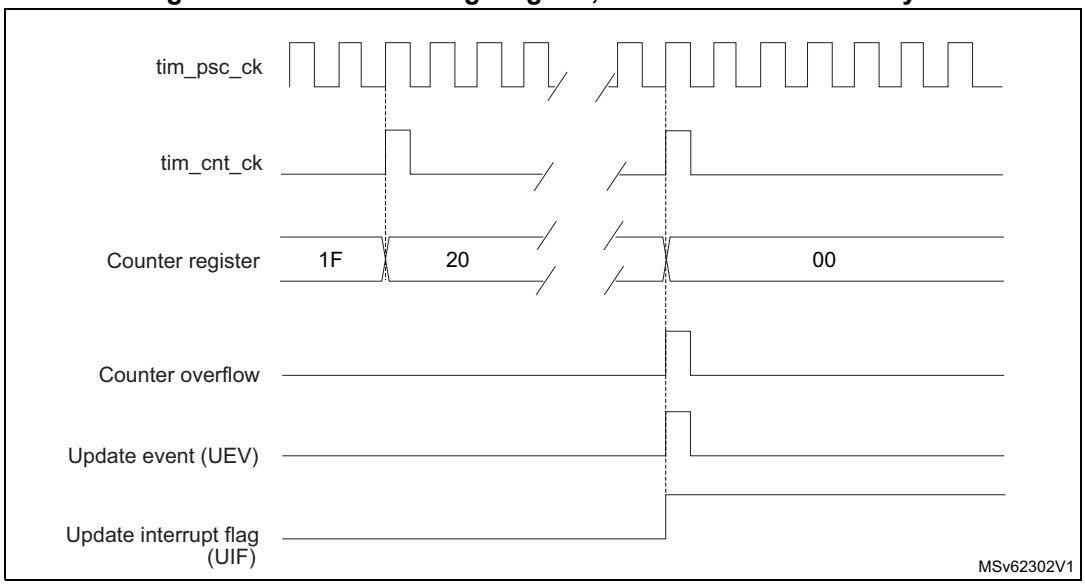


Figure 426. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

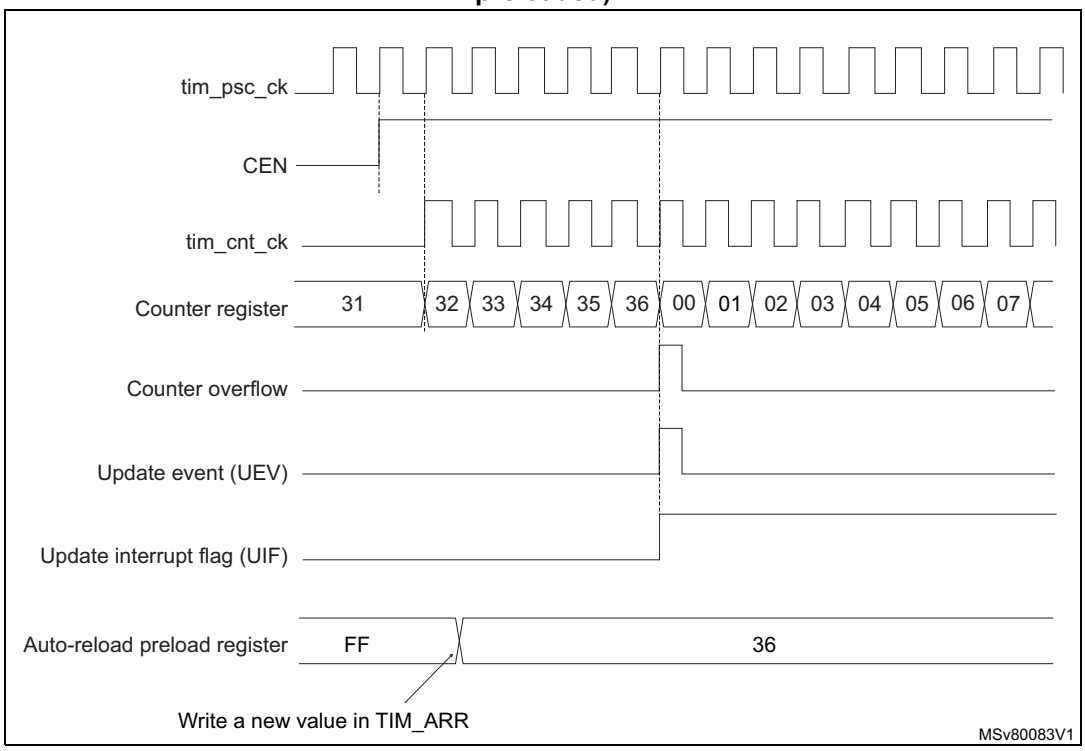
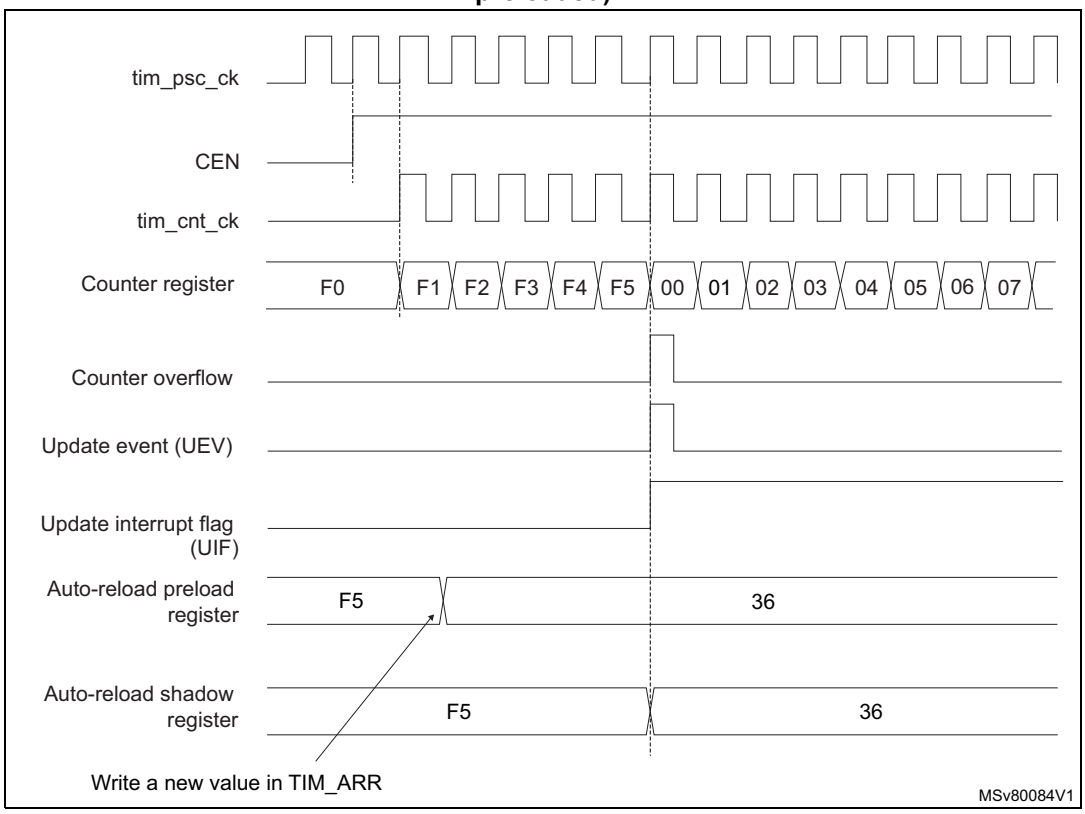


Figure 427. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



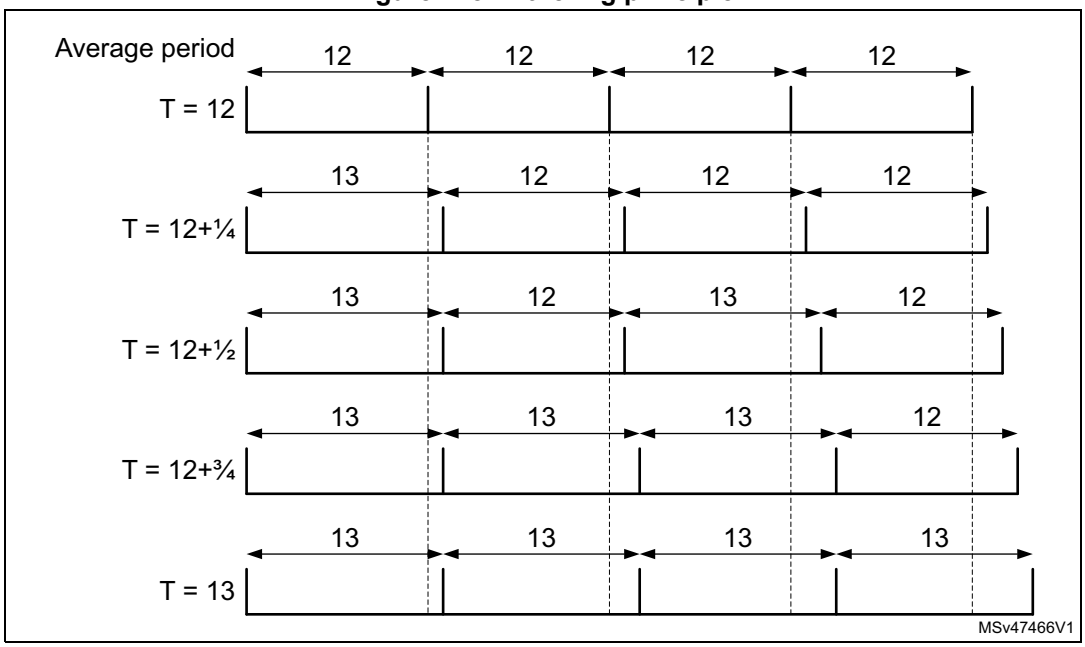
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This affects the way the TIM_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 428 presents the dithering principle applied to four consecutive counting periods.

Figure 428. Dithering principle

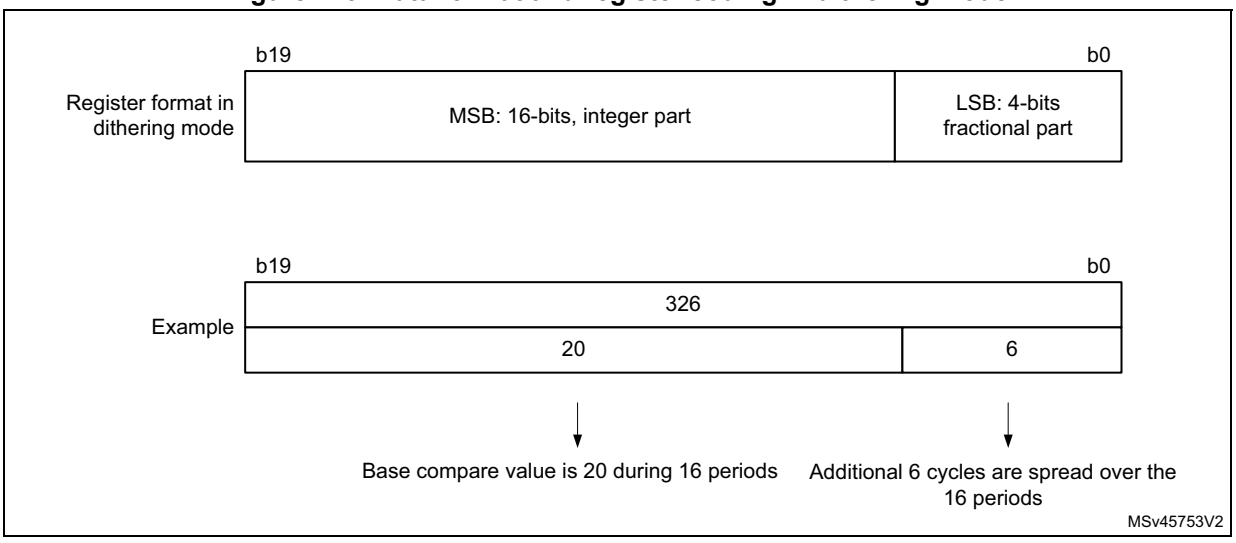


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 429* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 429. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

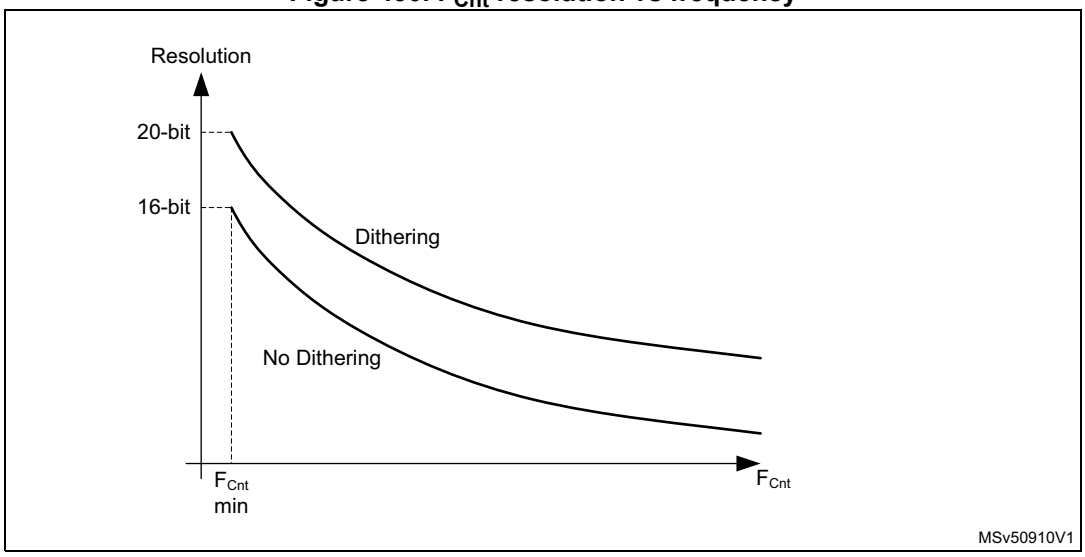
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

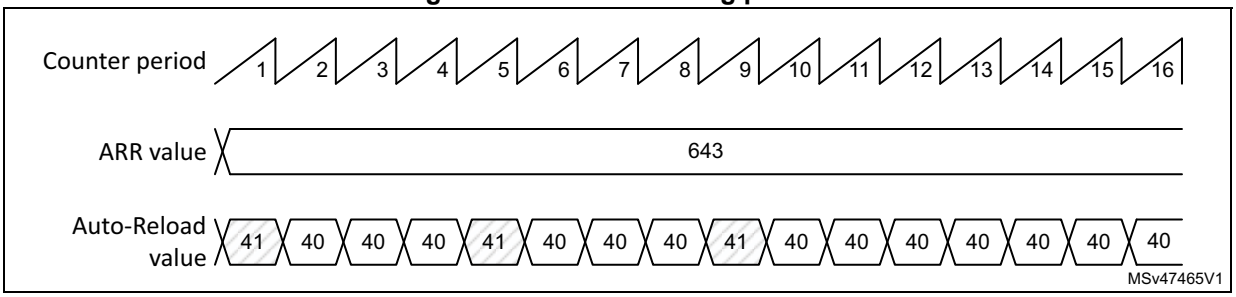
As shown on *Figure 430*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 430. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 431*.

Figure 431. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 329*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 329. TIM_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

33.3.6 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

33.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

33.3.8 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

33.3.9 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 330*.

Table 330. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE

33.3.10 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

33.3.11 TIM6/TIM7 low-power modes

Table 331. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

33.3.12 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 332*.

Table 332. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

33.4 TIM6/TIM7 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

33.4.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM_ARR register is not buffered.
1: TIM_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

33.4.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 33.3.8* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIM_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

33.4.3 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

33.4.4 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–On counter overflow if UDIS = 0 in the TIM_CR1 register.

–When CNT is reinitialized by software using the UG bit in the TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

33.4.5 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

33.4.6 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in TIM_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

33.4.7 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIM_EGR register.

33.4.8 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 33.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

33.4.9 TIM6/TIM7 register map

TIM6/TIM7 registers are mapped as 16-bit addressable registers as described in the table below:

Table 333. TIM6/TIM7 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	**TIM_CR2**	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value				0																						0	0	0				
0x08	Reserved	Res.																															
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIM_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value								0																								0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved	Res.																															
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

34 General-purpose timers (TIM12)

34.1 TIM12 introduction

The TIM12 general-purpose timers consist in a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM12 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 34.3.21: Timer synchronization*.

34.2 TIM12 main features

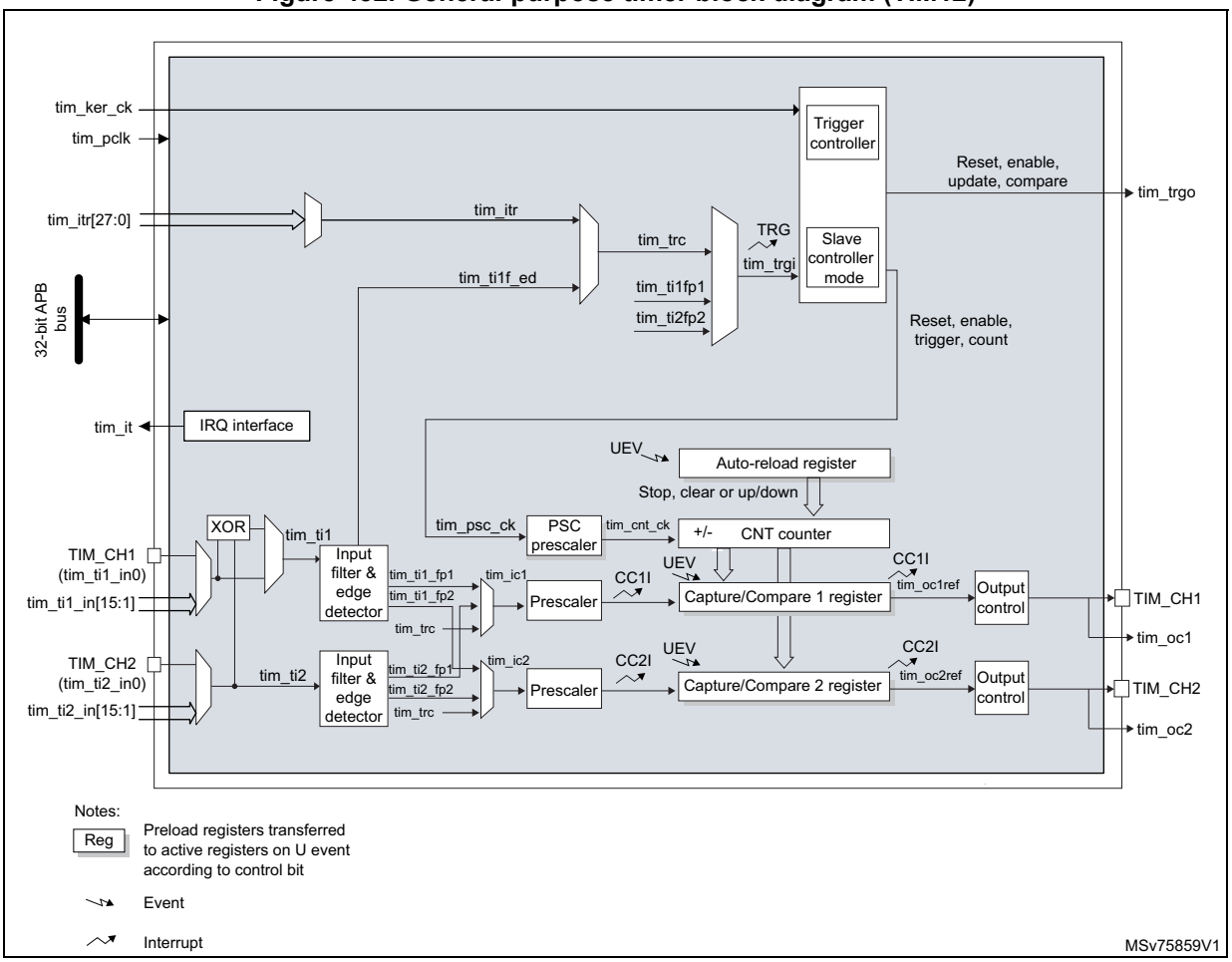
The features of the TIM12 general-purpose timer include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization, or count by internal trigger)
  - Input capture
  - Output compare
- ADC synchronization for jitter-free sampling points

34.3 TIM12 functional description

34.3.1 Block diagram

Figure 432. General-purpose timer block diagram (TIM12)



34.3.2 TIM12 pins and internal signals

Table 334 and *Table 335* in this section summarize the TIM inputs and outputs.

Table 334. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.

Table 335. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_oc1 tim_oc2	Output	Internal timer output. Can be used for triggering other timers or the ADC(s).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, and trigger requests

Table 336 and *Table 337* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 336. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources
	TIM12
tim_ti1_in0	TIM12_CH1
tim_ti1_in1	comp1_out
tim_ti1_in2	comp2_out
tim_ti1_in3	rcc_mco1
tim_ti1_in4	rcc_mco2
tim_ti1_in5	hse_1M_ck
tim_ti1_in6	Reserved
tim_ti1_in7	i3c1_ibi_ack
tim_ti1_in[15:18]	Reserved

Table 337. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM12
tim_ti2_in0	TIM12_CH2
tim_ti2_in[15:1]	Reserved

Table 338 lists the internal sources connected to the tim_itr input multiplexer.

Table 338. TIMx internal trigger connection

tim_itrx inputs	TIM12
tim_itr0	tim1_trgo
tim_itr1	tim1_trgo2
tim_itr2	tim2_trgo
tim_itr3	tim3_trgo
tim_itr4	tim4_trgo
tim_itr5	tim5_trgo
tim_itr6	tim8_trgo
tim_itr7	tim8_trgo2
tim_itr8	Reserved
tim_itr9	tim15_trgo
tim_itr10	tim16_oc1
tim_itr11	tim17_oc1
tim_itr[27:12]	Reserved

34.3.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR).

The autoreload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- *The update interrupt service routine is too long with respect to the interval between consecutive update events.*
- *The update interrupt service routine was served too late with respect to the subsequent update event.*

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 433 and *Figure 434* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 433. Counter timing diagram with prescaler division change from 1 to 2

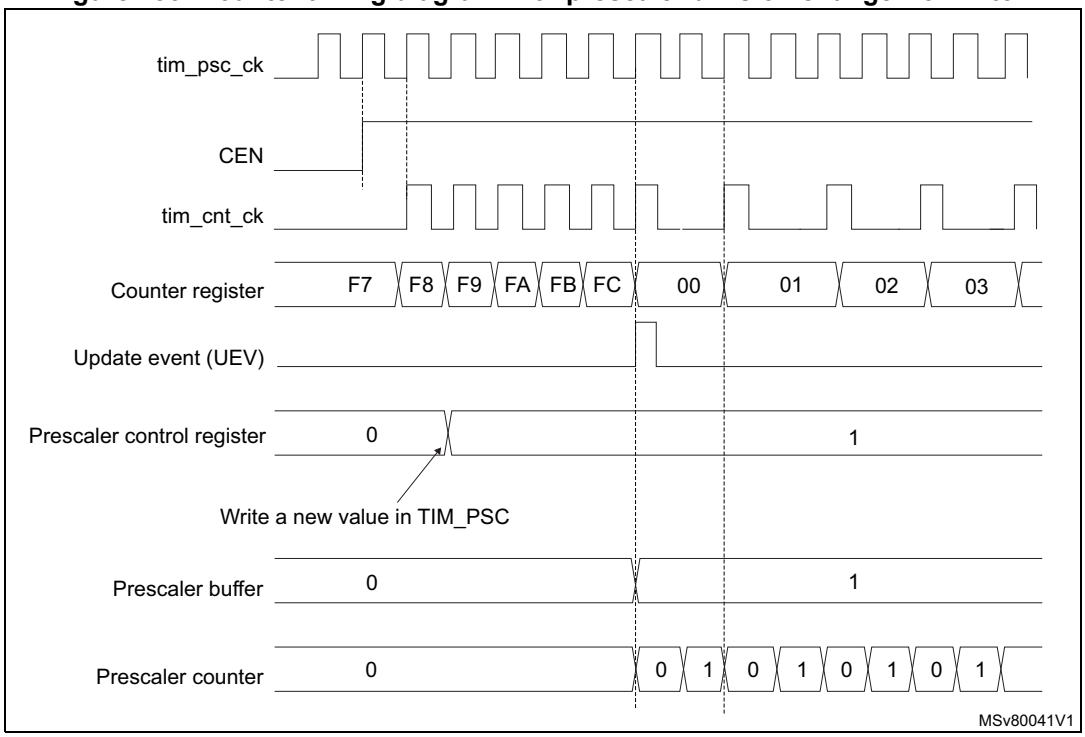
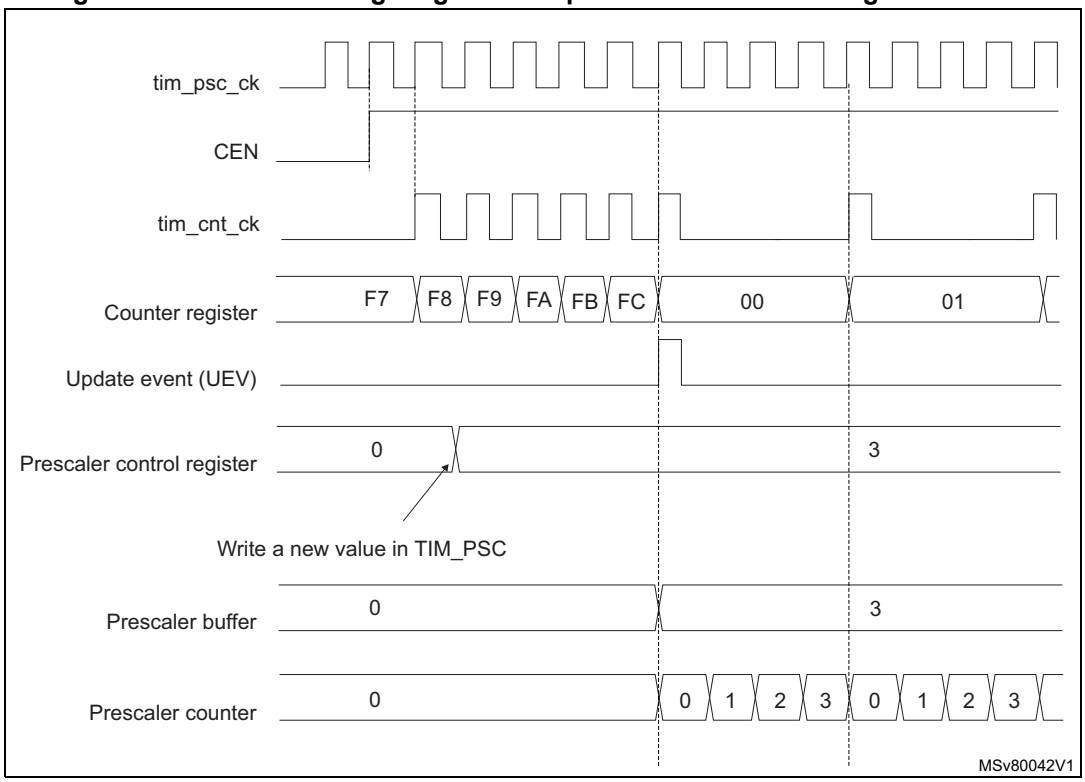


Figure 434. Counter timing diagram with prescaler division change from 1 to 4



34.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The autoreload shadow register is updated with the preload value (TIM_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 435. Counter timing diagram, internal clock divided by 1

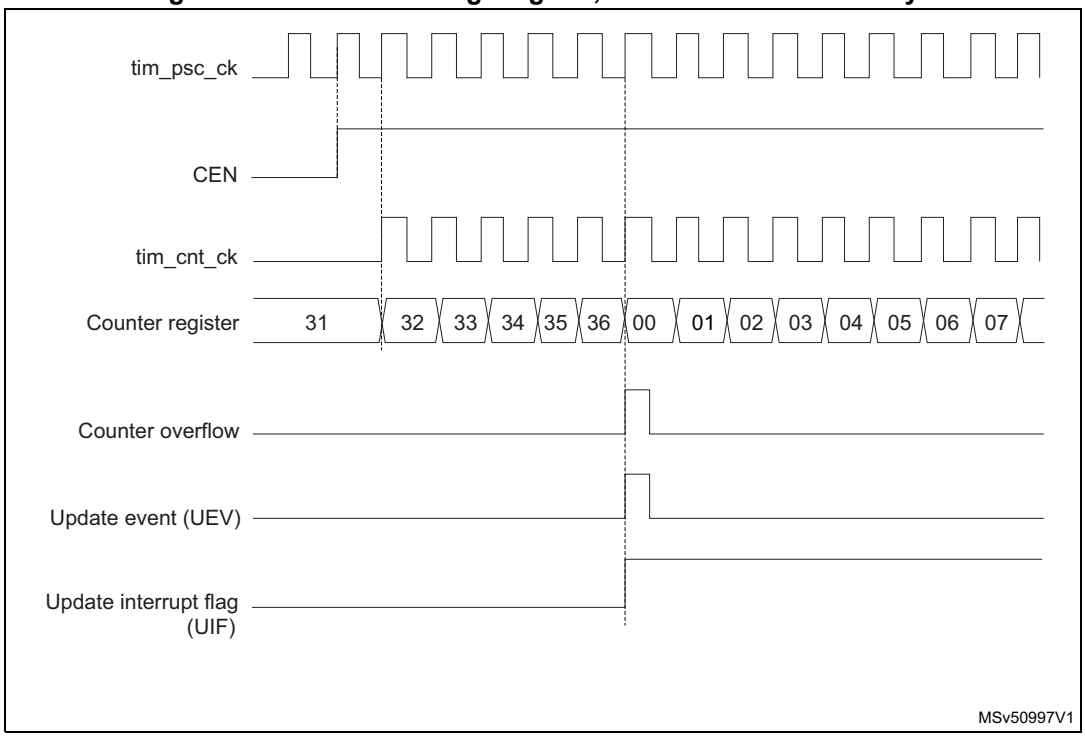


Figure 436. Counter timing diagram, internal clock divided by 2

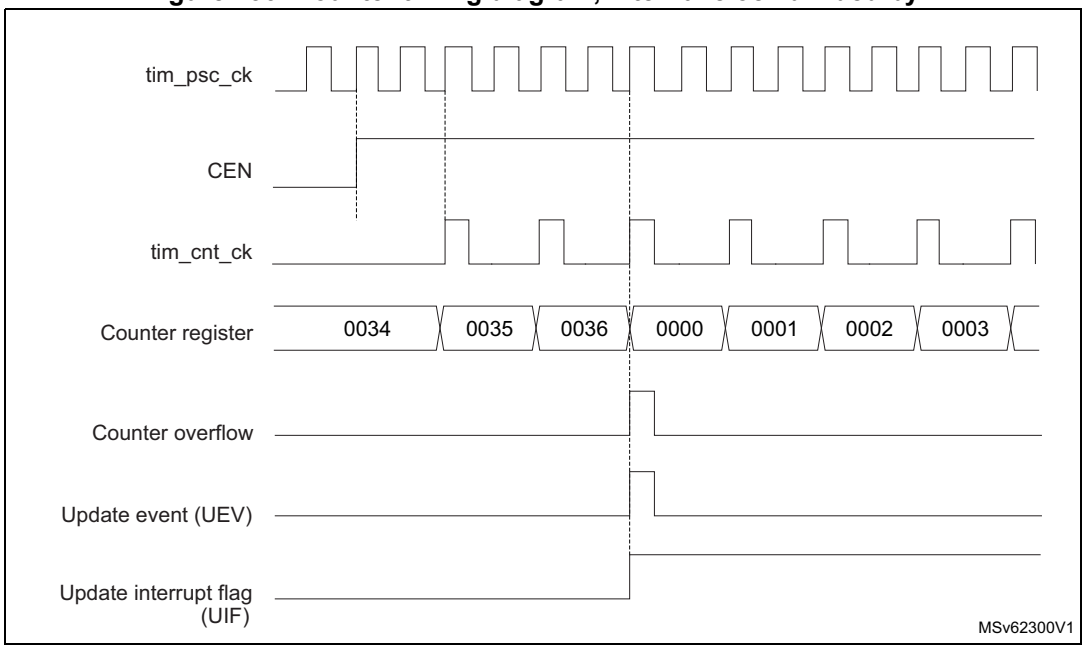


Figure 437. Counter timing diagram, internal clock divided by 4

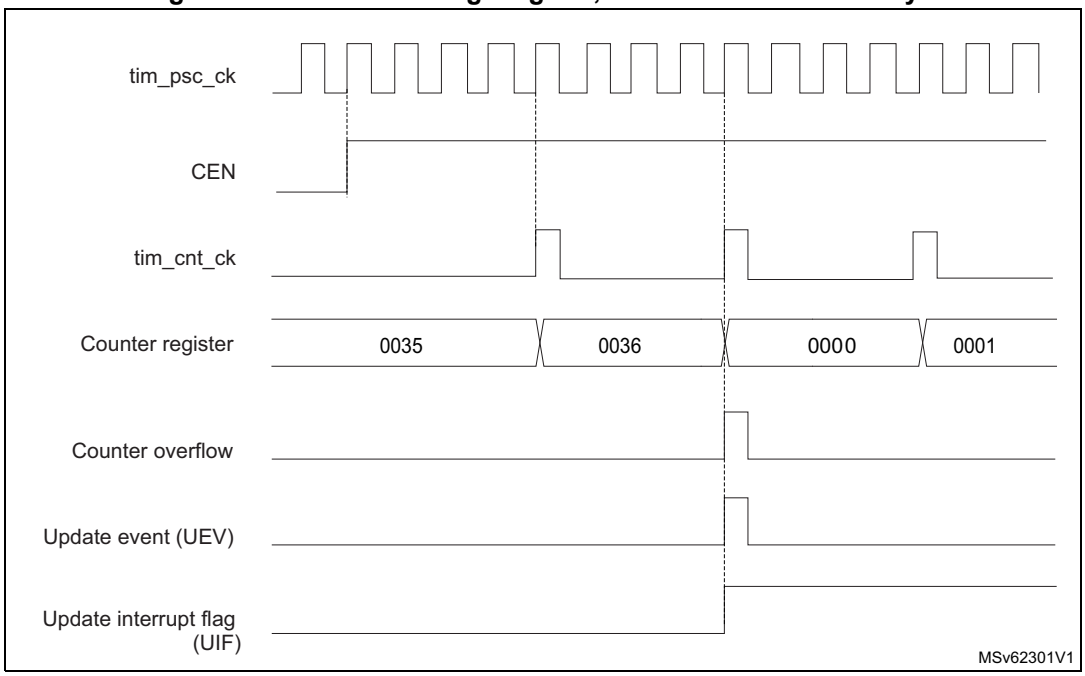


Figure 438. Counter timing diagram, internal clock divided by N

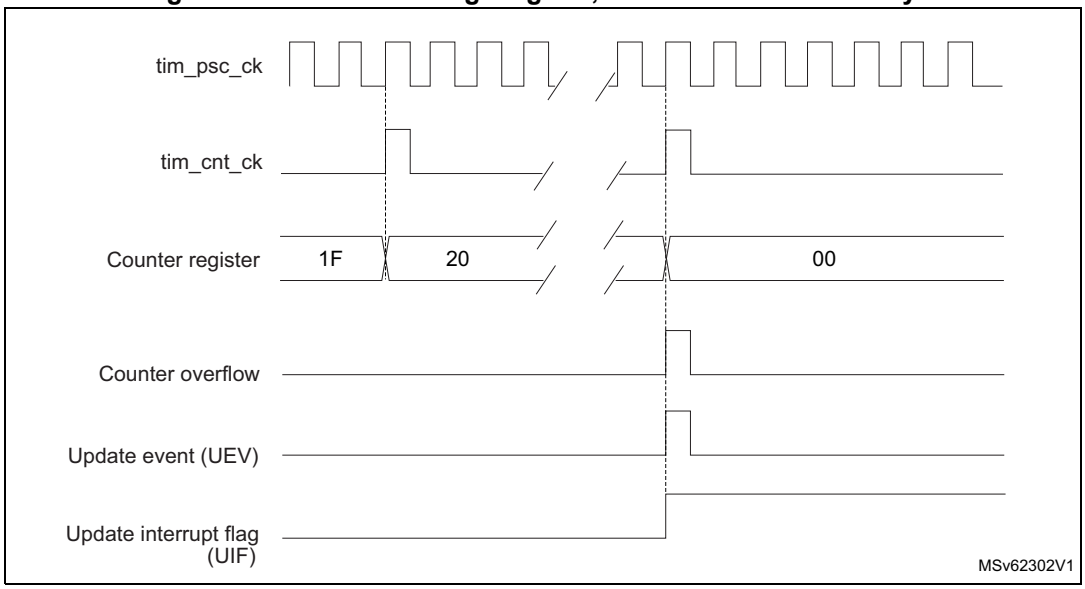


Figure 439. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

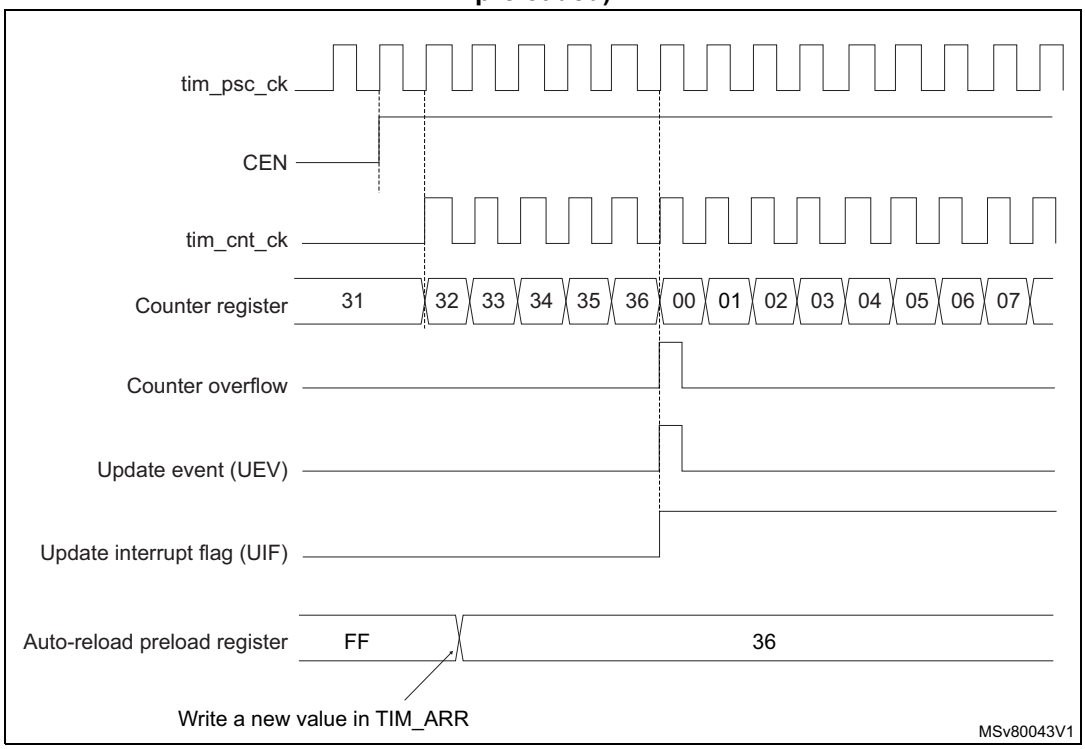
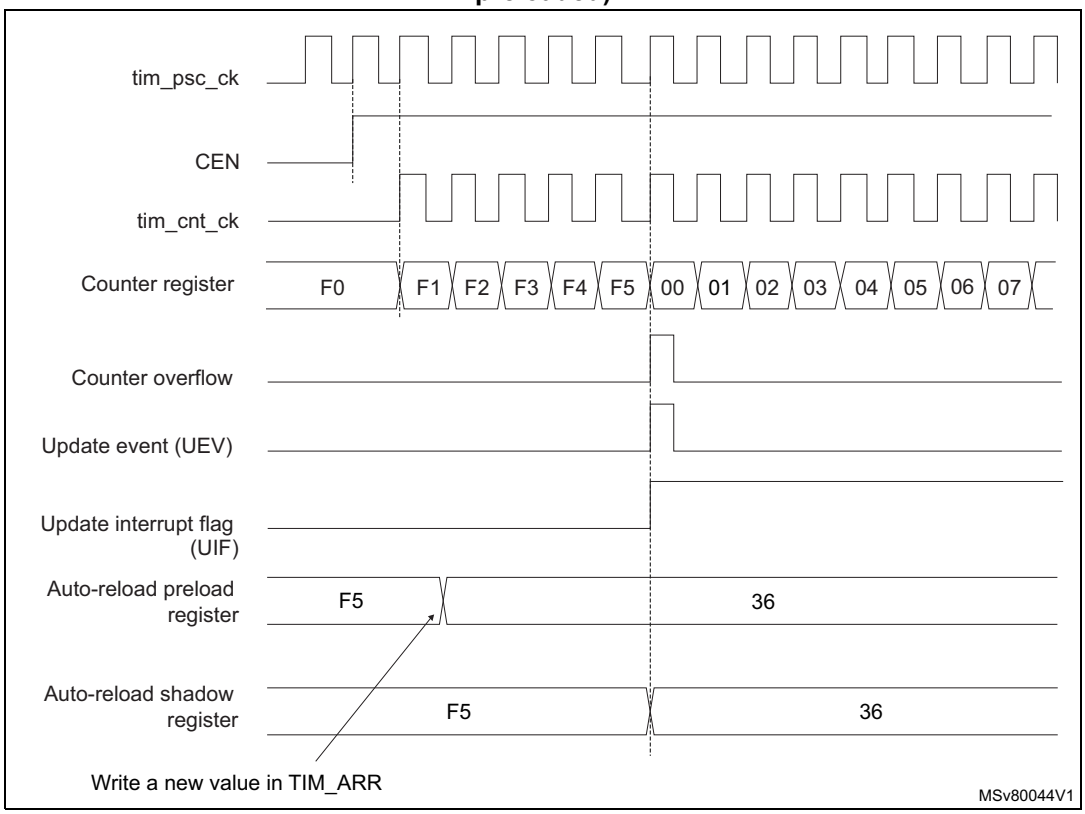


Figure 440. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



34.3.5 Clock selection

The counter clock can be provided by the following clock sources:

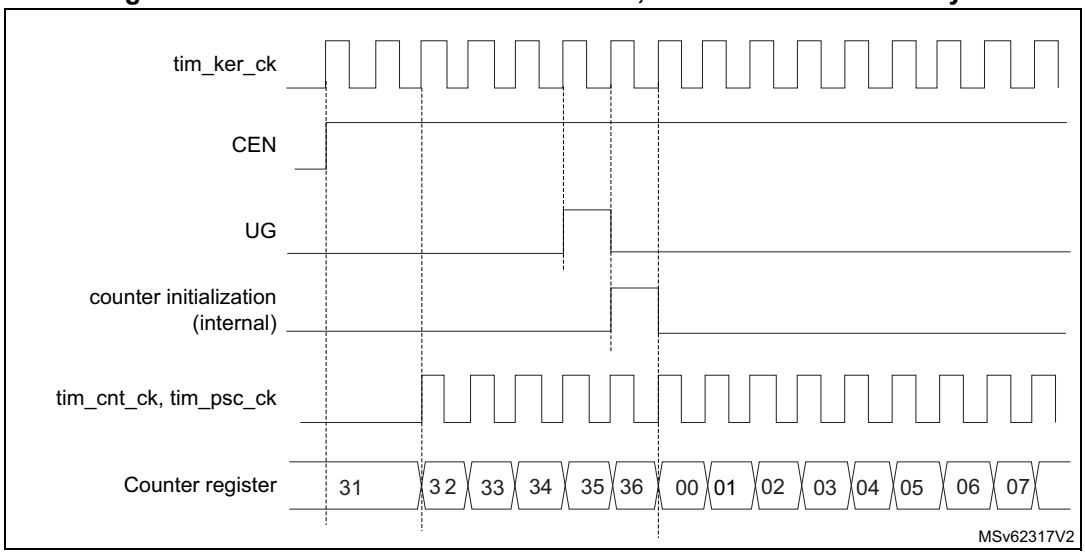
- Internal clock (tim_ker_ck)
- External clock mode1 (for TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (for TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

For TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS = 000). The CEN bit in the TIM_CR1 register and the UG bit in the TIM_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 441 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

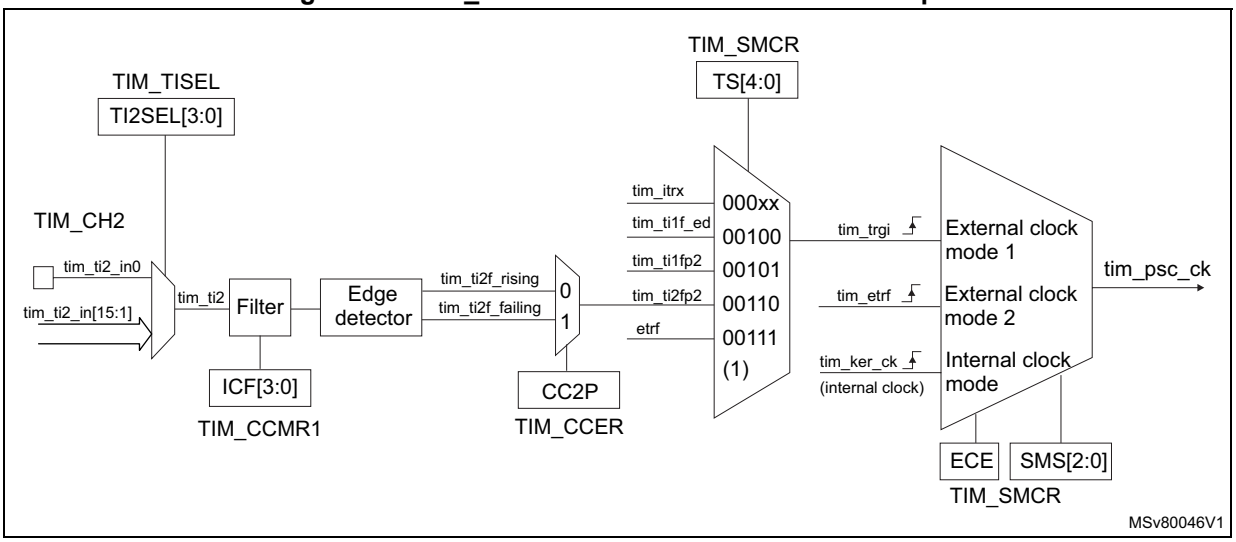
Figure 441. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 442. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

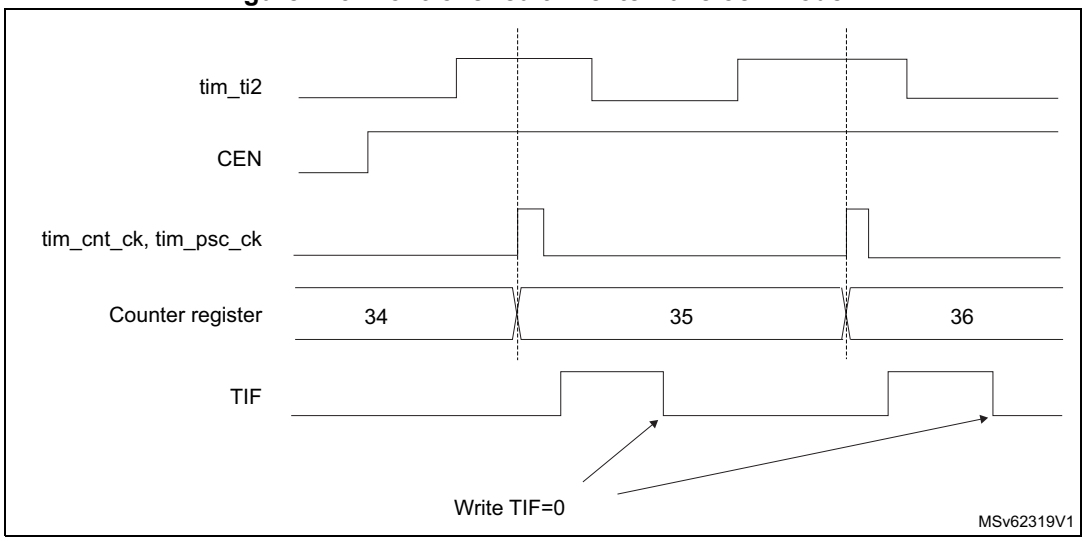
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select the rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 443. Control circuit in external clock mode 1



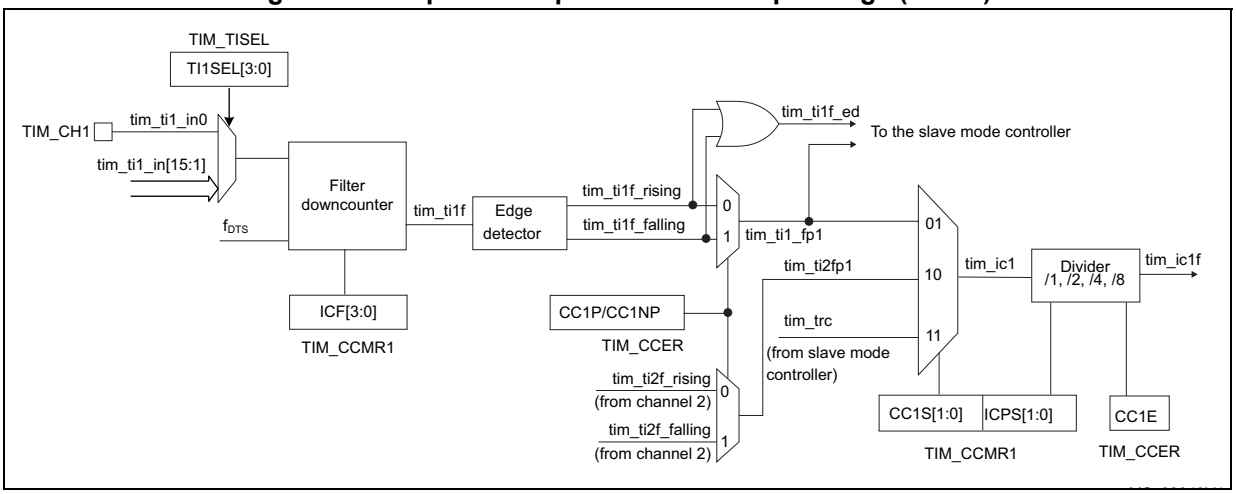
34.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler), and an output stage (with comparator and output control).

Figure 444, *Figure 445* and *Figure 446* give an overview of a capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 444. Capture/compare channel 1 input stage (TIM12)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 445. Capture/compare channel 1 main circuit

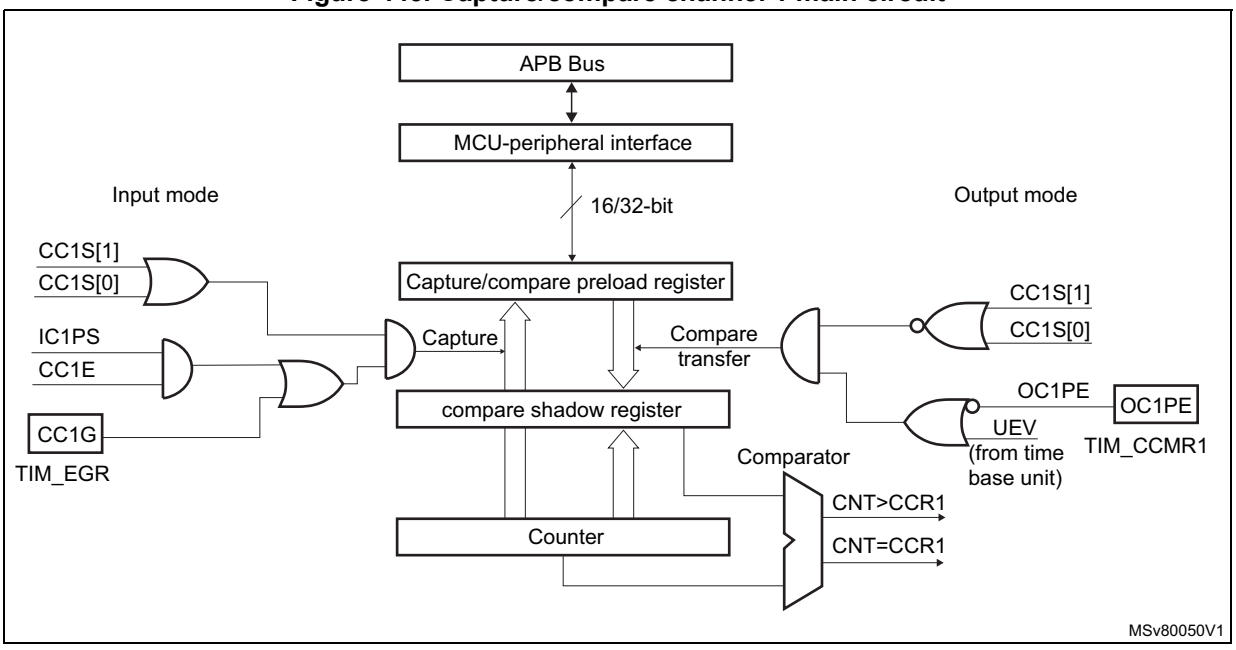
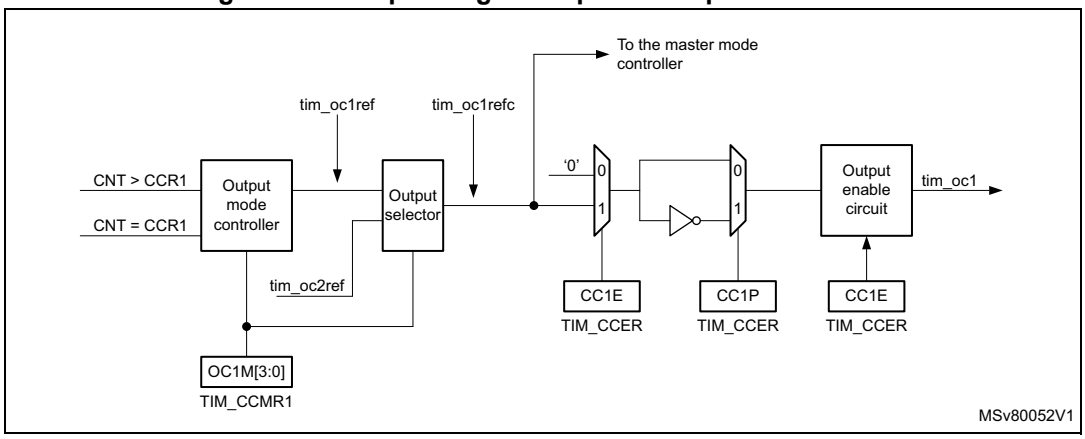


Figure 446. Output stage of capture/compare channel 1



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

34.3.7 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input mode, and the TIM_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIM_CCMRx register if the input is one of the tim_tix inputs). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight

consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.

4. Select the edge of the active transition on the tim_ti1 channel by programming CC1P and CC1NP bits to 00 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wishes the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

34.3.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges.

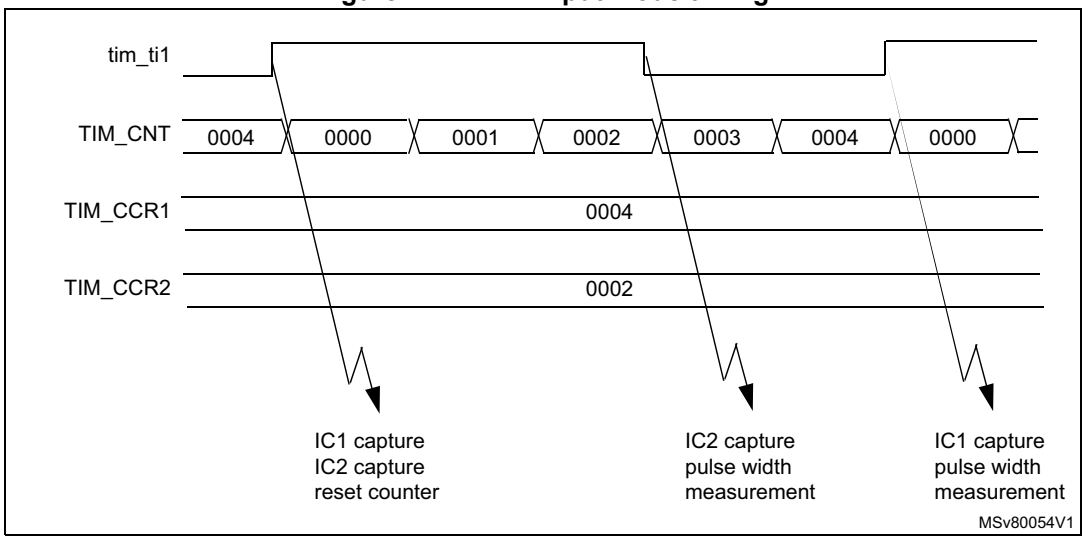
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIM_CCR1 register) and the duty cycle (in TIM_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): program the CC1P and CC1NP bits to 00 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): program the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 447. PWM input mode timing



34.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 0101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

34.3.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIM_DIER register).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

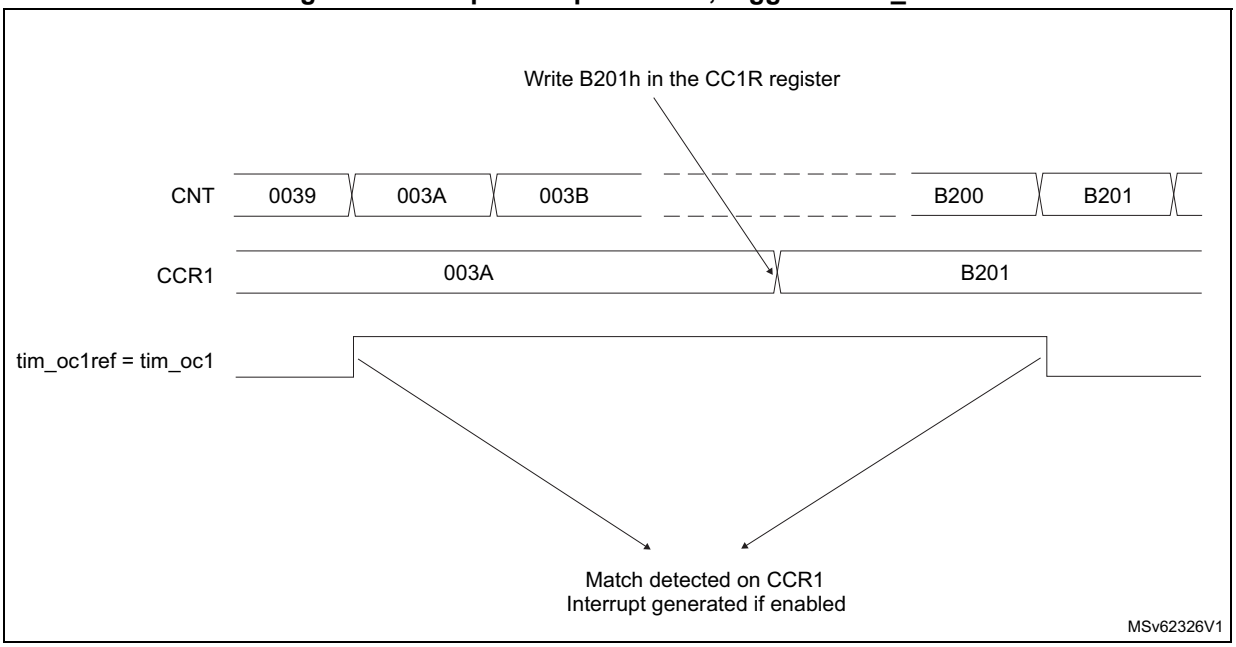
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 448*.

Figure 448. Output compare mode, toggle on tim_oc1.



34.3.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

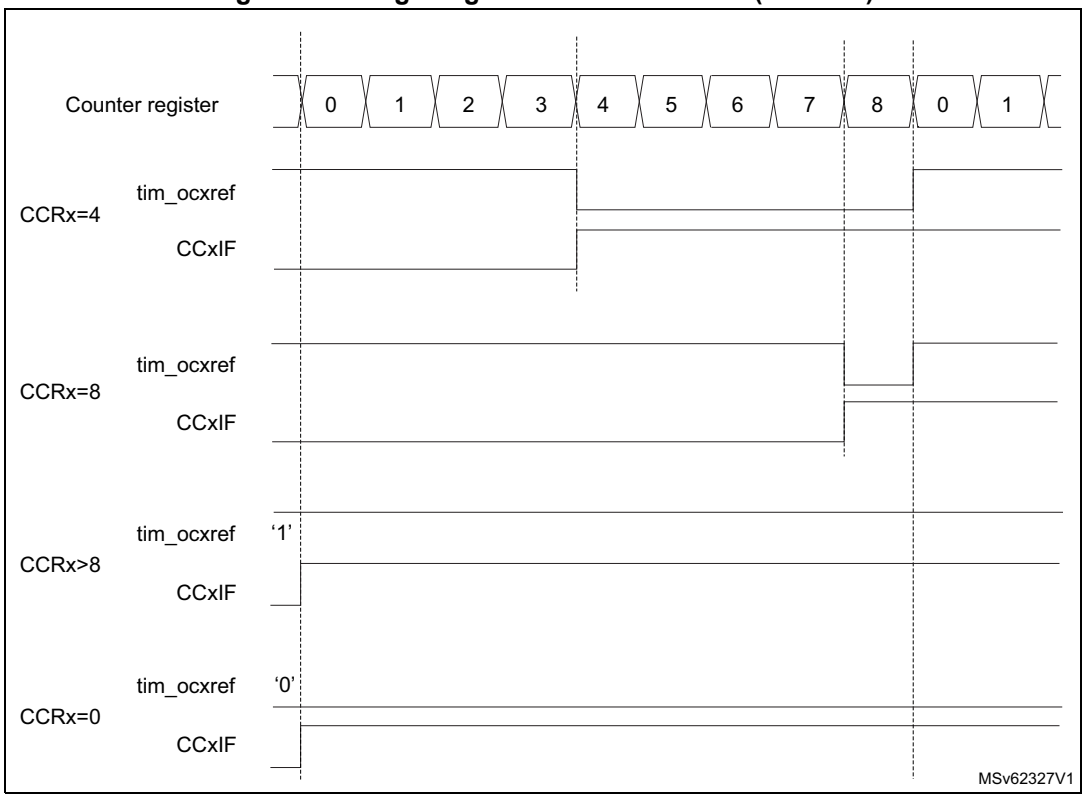
The tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. The tim_ocx output is enabled by the CCxE bit in the TIM_CCER register. Refer to the TIM_CCERx register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CNT ≤ TIM_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, the user considers PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 449* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 449. Edge-aligned PWM waveforms (ARR = 8)

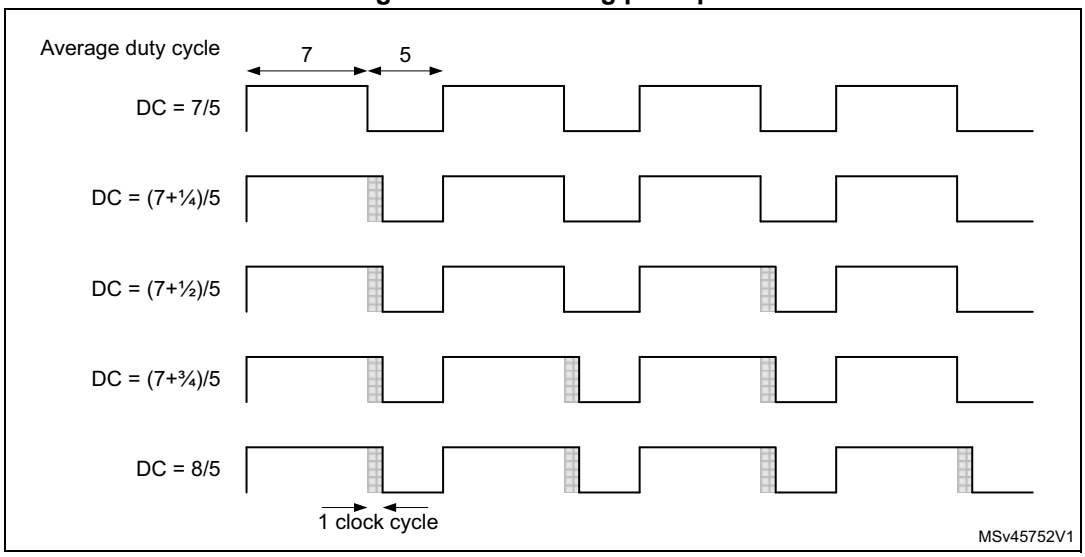


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 450* presents the dithering principle applied to four consecutive PWM cycles.

Figure 450. Dithering principle

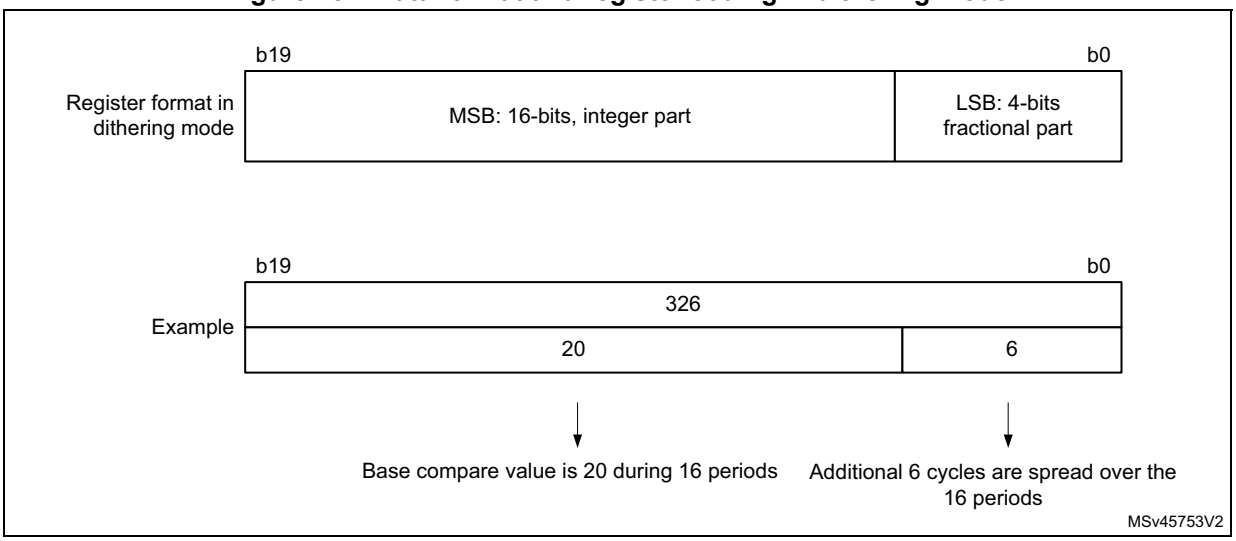


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 451* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 451. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

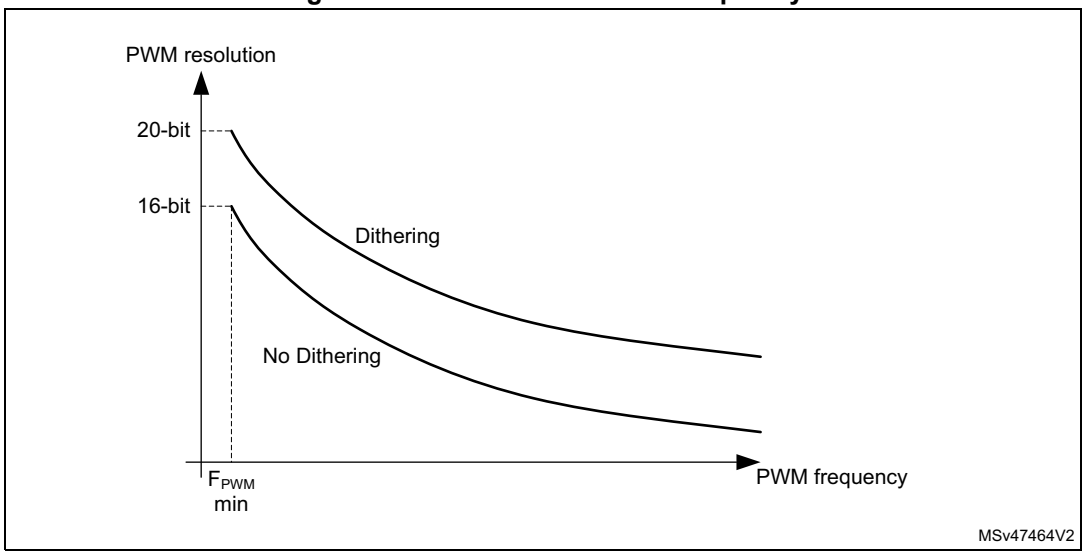
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

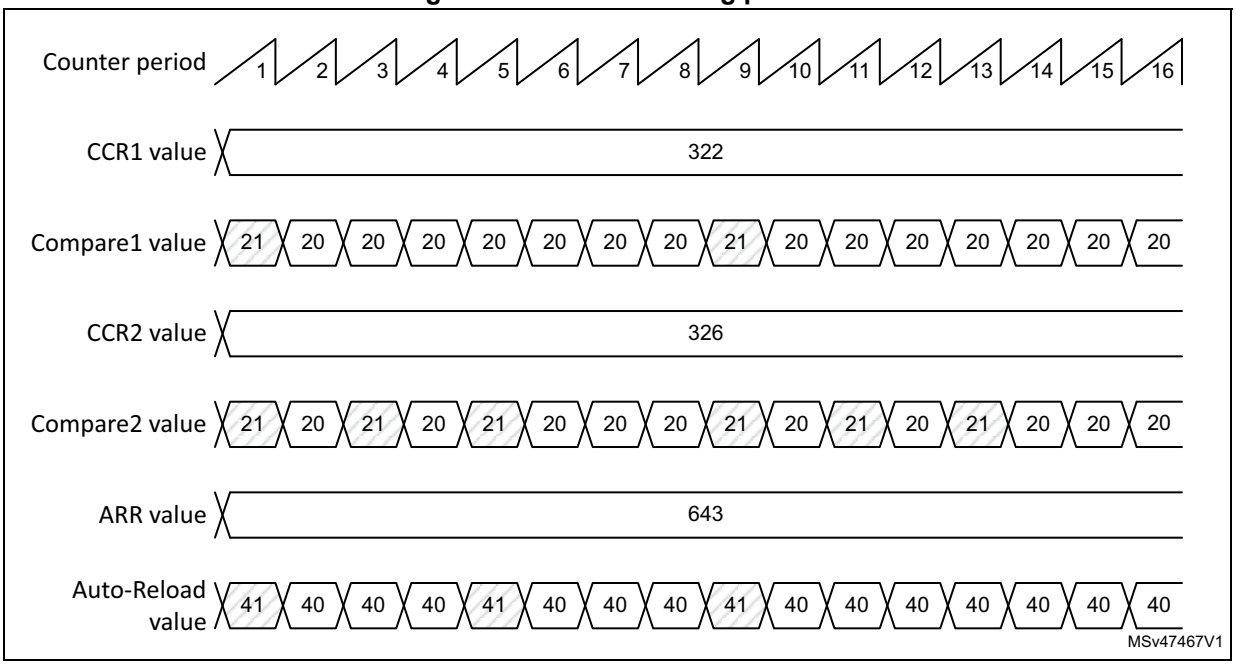
As shown on *Figure 452*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 452. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 453*.

Figure 453. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 339*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 339. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 339. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

34.3.12 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIM_CCR1 and TIM_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

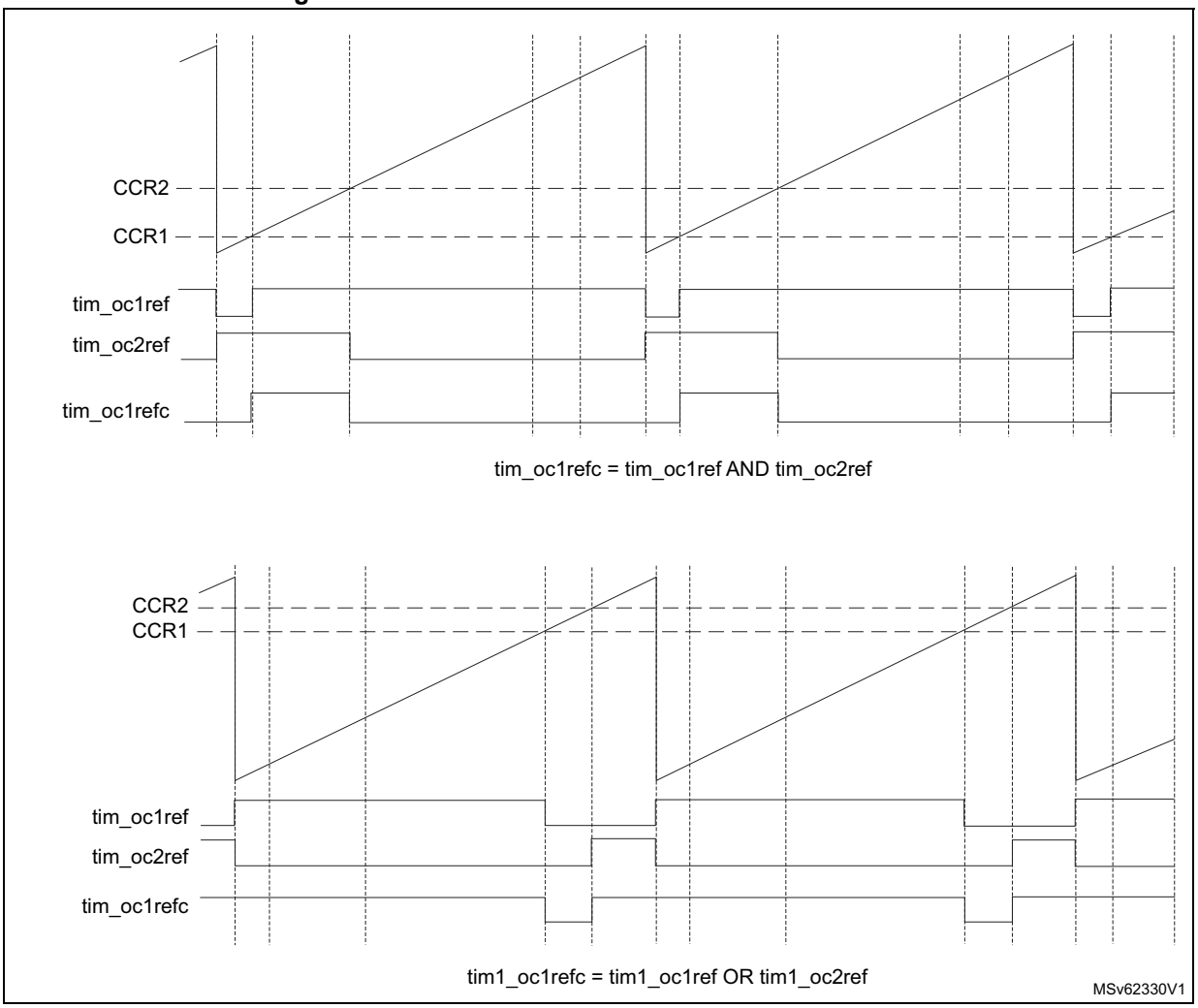
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 454 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 454. Combined PWM mode on channel 1 and 2

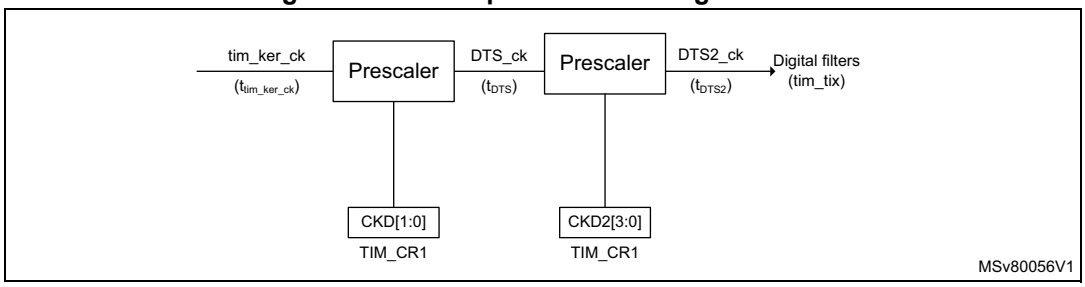


34.3.13 Clock dividers for digital filters

The digital filters (for tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the DTS_ck clock period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 455. Clock prescalers for digital filters



34.3.14 One-pulse mode

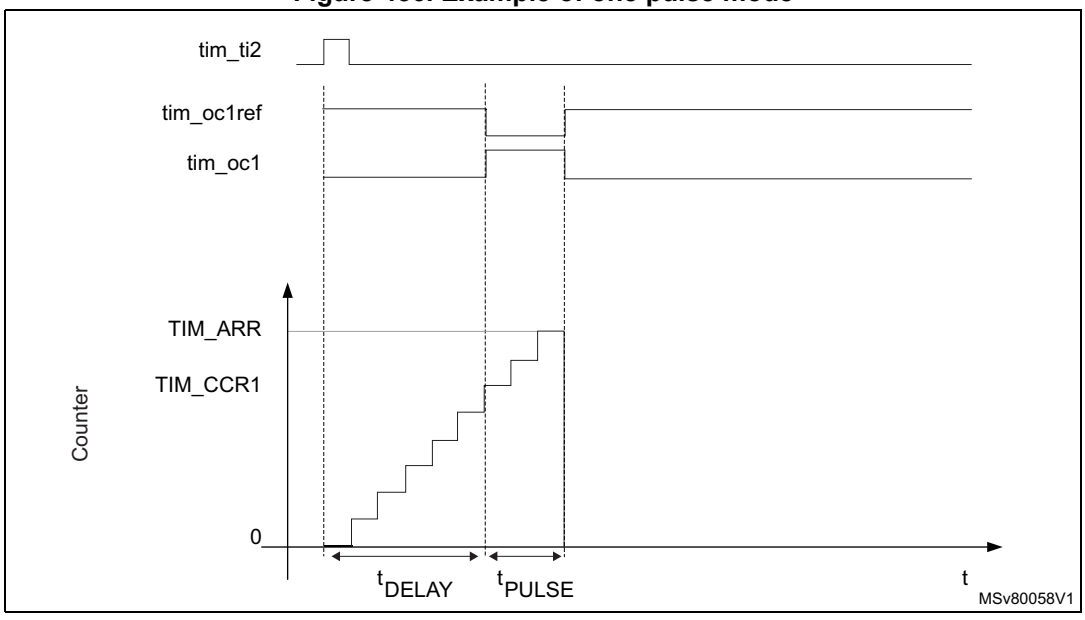
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 456. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Assuming the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 0111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

34.3.15 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one pulse mode described in *Section 34.3.14: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

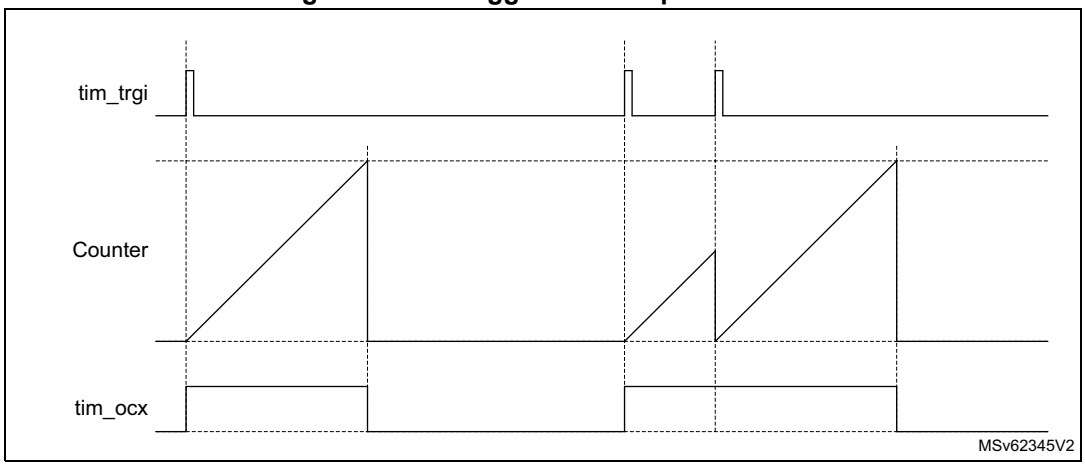
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 457. Retriggerable one pulse mode



34.3.16 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

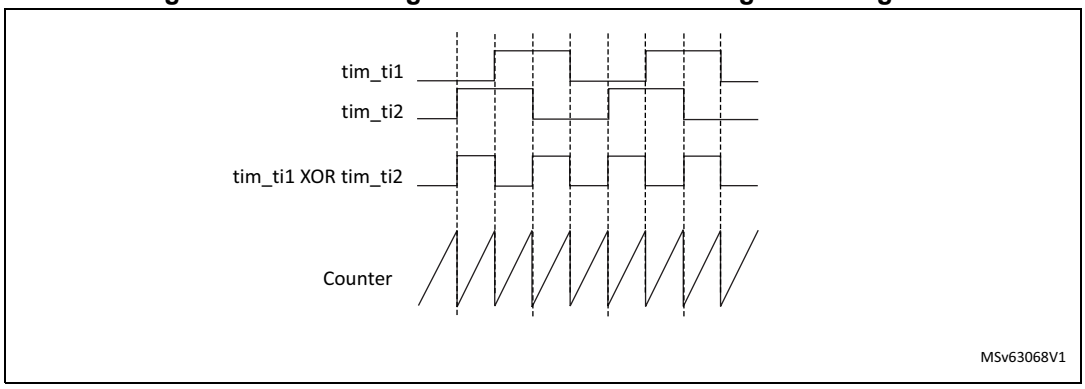
There is no latency between the assertions of the UIF and UIFCPY flags.

34.3.17 Timer input XOR function

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 458*.

Figure 458. Measuring time interval between edges on 2 signals



The XORPS control bit selects the position of the XOR gate, as shown in *Figure 459* and *Figure 460*, when the TI1S bit is set.

Figure 459. XOR gate connection with TI1S = 1 and XORPS = 0

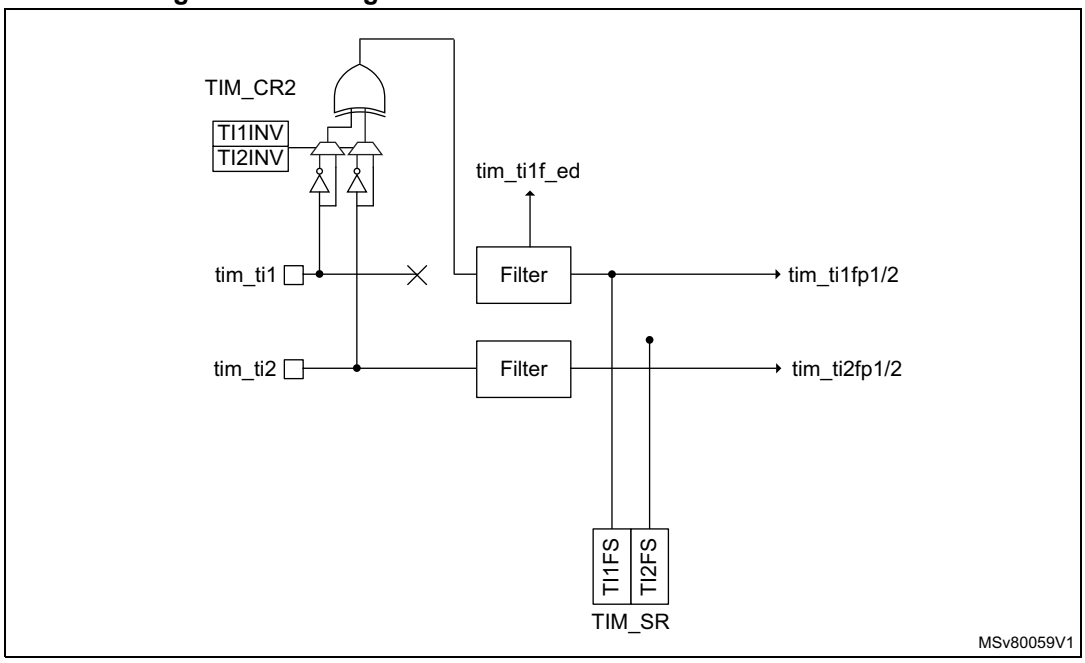
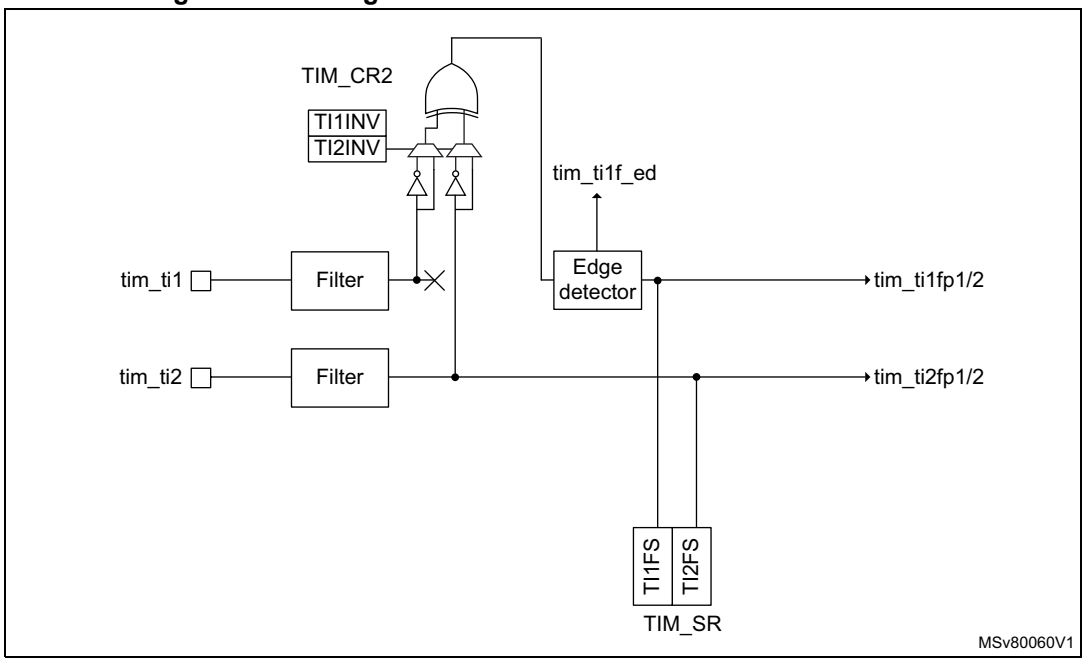


Figure 460. XOR gate connection with TI1S = 1 and XORPS = 1



Two control bits TI1INV and TI2INV in the TIM_CR2 register can invert the tim_in1 and tim_in2 signals prior entering the XOR gate.

The status of the tim_in1 and tim_in2 signals can be software polled using the TI1FS and TI2FS bits in the TIM_SR register.

34.3.18 TIM12 external trigger synchronization

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and Gated + reset mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

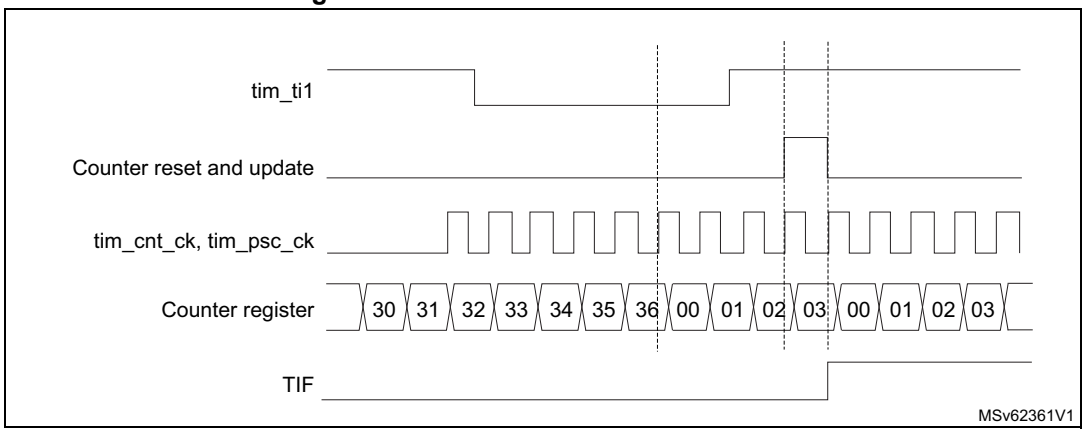
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Program CC1P and CC1NP to 00 in TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 461. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

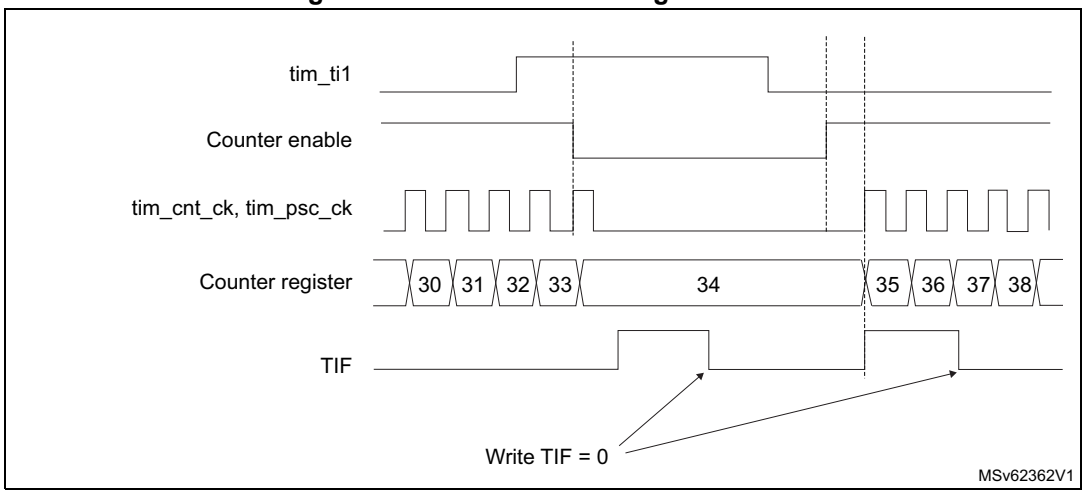
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 462. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

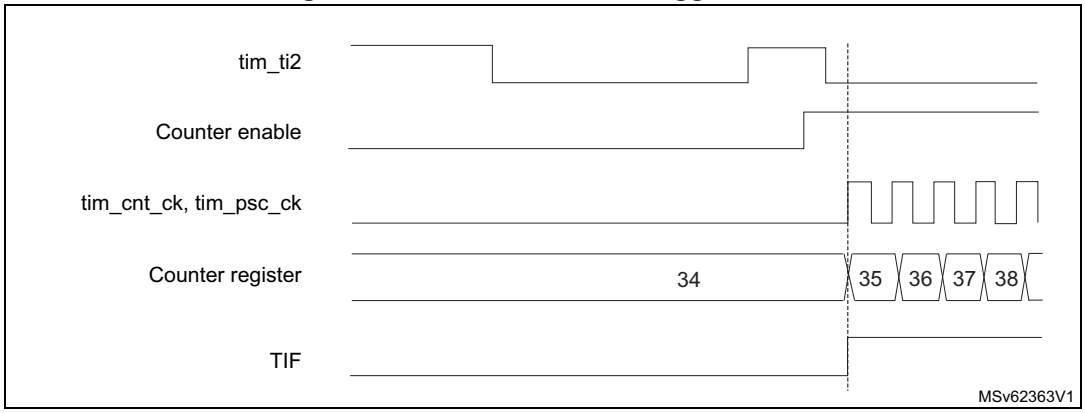
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Program CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 463. Control circuit in trigger mode



34.3.19 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

34.3.20 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

34.3.21 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 32.4.27: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

34.3.22 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

34.3.23 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CCRx} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

34.3.24 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the debug section.

34.4 TIM12 low-power modes

Table 340. Effect of low-power modes on TIM12

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

34.5 TIM12 interrupts

The TIM12 can generate multiple interrupts, as shown in *Table 341*.

Table 341. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
TIM	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Trigger	TIF	TIE	write 0 in TIF	Yes	No

34.6 TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

34.6.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) and the digital filters (tim_tix).

0000: No Division
0001: $t_{DTS2} = 4 \times t_{DTS}$
0010: $t_{DTS2} = 16 \times t_{DTS}$
0011: $t_{DTS2} = 64 \times t_{DTS}$
0100: $t_{DTS2} = 256 \times t_{DTS}$
0101: $t_{DTS2} = 1024 \times t_{DTS}$
0110: $t_{DTS2} = 4096 \times t_{DTS}$
0111: $t_{DTS2} = 16384 \times t_{DTS}$
1000: $t_{DTS2} = 65536 \times t_{DTS}$
1001: $t_{DTS2} = 262144 \times t_{DTS}$
Others: Reserved, do not program this value.

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered.
1: TIM_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled. These events can be:
– Counter overflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
CEN is cleared automatically in one-pulse mode, when an update event occurs.
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

34.6.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TI2INV	TI1INV	AD SYNC	XORPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input
0: tim_ti2 non-inverted
1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input
0: tim_ti1 non-inverted
1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 34.3.23* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection

This bit defines the position of the XOR gate with respect to input filters.

0: XOR gate placed before TI1 filter

1: XOR gate placed after TI1 and TI2 filters

Bits 26:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

This bit enables the insertion of an XOR gate to combine tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs into tim_ti1.

The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: Reset - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

010: Update - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: Compare - tim_oc1refc signal is used as trigger output (tim_trgo).

101: Compare - tim_oc2refc signal is used as trigger output (tim_trgo).

Bits 3:0 Reserved, must be kept at reset value.

34.6.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 34.3.2: TIM12 pins and internal signals* for more details on the meaning of tim_itrx for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see input control register and control register description.

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Other codes: reserved.

Note: The gated mode (including gated + reset mode) must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

34.6.4 TIM interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.

1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

0: CC2 interrupt disabled.

1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

34.6.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update interrupt overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No update interrupt overrun
1: Update interrupt overrun

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when he content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow and if UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 34.6.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

34.6.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIM_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

34.6.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

34.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas the active level of tim_oc1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The tim_oc1ref signal is forced high when the TIM_CNT counter matches the capture/compare register 1 (TIM_CCR1).

0010: Set channel 1 to inactive level on match. The tim_oc1ref signal is forced low when the TIM_CNT counter matches the capture/compare register 1 (TIM_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1

0100: Force inactive level - tim_oc1ref is forced low

0101: Force active level - tim_oc1ref is forced high

0110: PWM mode 1 - channel 1 is active as long as TIM_CNT<TIM_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else it is active

1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00:CC1 channel is configured as output
01:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11:CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

34.6.9 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active

1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 342. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The states of the external I/O pins connected to the standard tim_ocx *channels depend on the state of the* tim_ocx *channel and on the GPIO registers.*

34.6.10 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

34.6.11 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

34.6.12 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 34.3.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

34.6.13 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

34.6.14 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

34.6.15 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIM_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

0100: tim_ti2_in15

Refer to *Section 34.3.2: TIM12 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIM_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 34.3.2: TIM12 pins and internal signals* for interconnects list.

34.6.16 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 343. TIM12 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0				0	0	0	0
0x004	**TIM_CR2**	Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value		0	0	0	0																				0	0	0	0				
0x008	**TIM_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value								0																		0				0	0	0
0x010	**TIM_SR**	Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value			0	0				0														0	0			0				0	0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x018	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x024	**TIM_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 343. TIM12 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIM_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C to 0x058	Reserved	Res.																															
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

35 General purpose timers (TIM15/TIM16/TIM17)

35.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 35.5.33: Timer synchronization (TIM15 only)*.

35.2 TIM15 main features

TIM15 includes the following features:

- 16-bit up, down, up/down autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
- Incremental encoders and quadrature encoders and hall sensors support
- ADC synchronization for jitter-free sampling points

35.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit up, down, up/down autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

35.4 TIM15/TIM16/TIM17 implementation

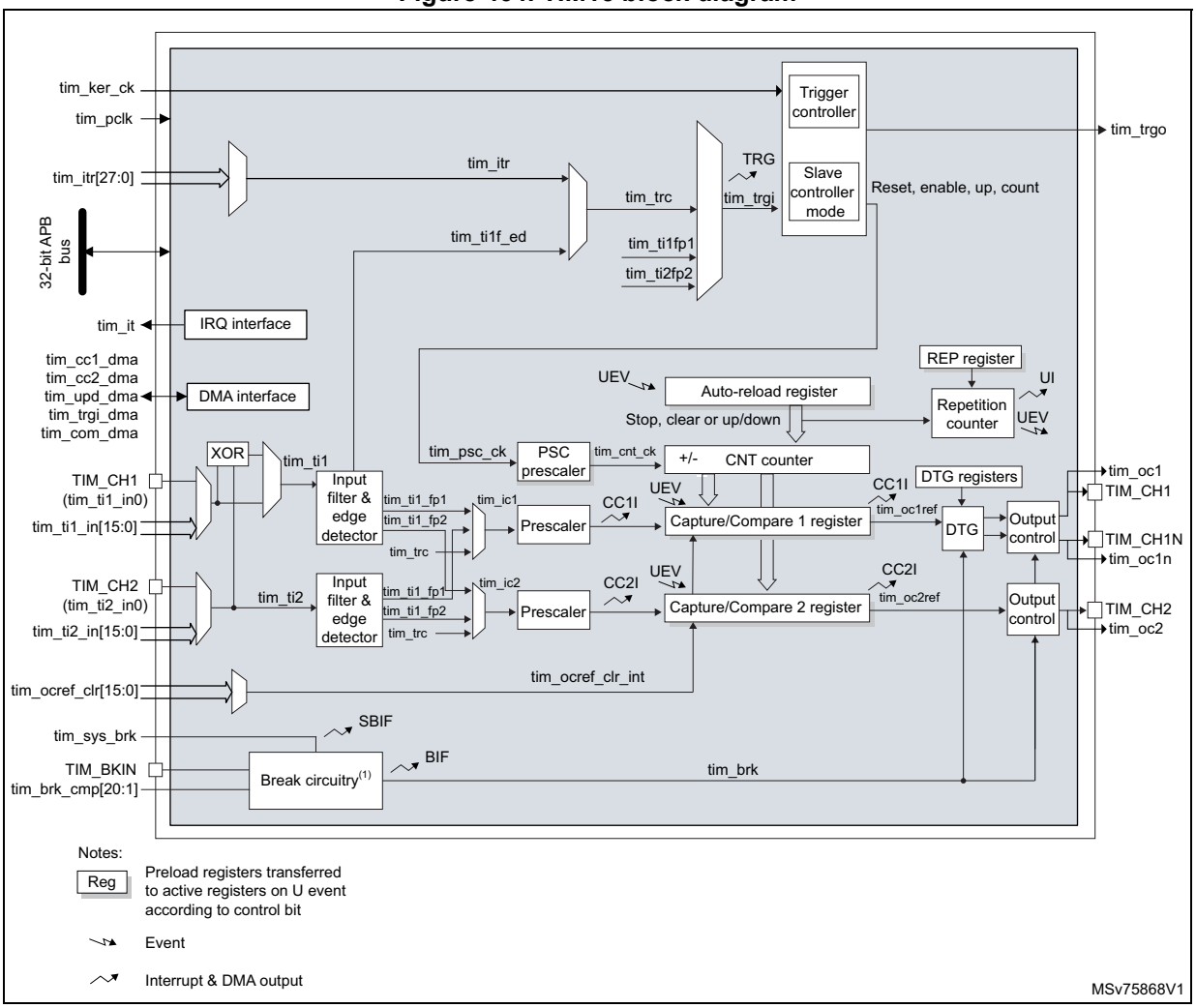
Table 344. Timer instance implementation on STM32C5

Timer instance	STM32C53x/C54x	STM32C55x/C56x	STM32C59x/C5Ax
TIM15	X	X	X
TIM16	-	X	X
TIM17	-	X	X

35.5 TIM15/TIM16/TIM17 functional description

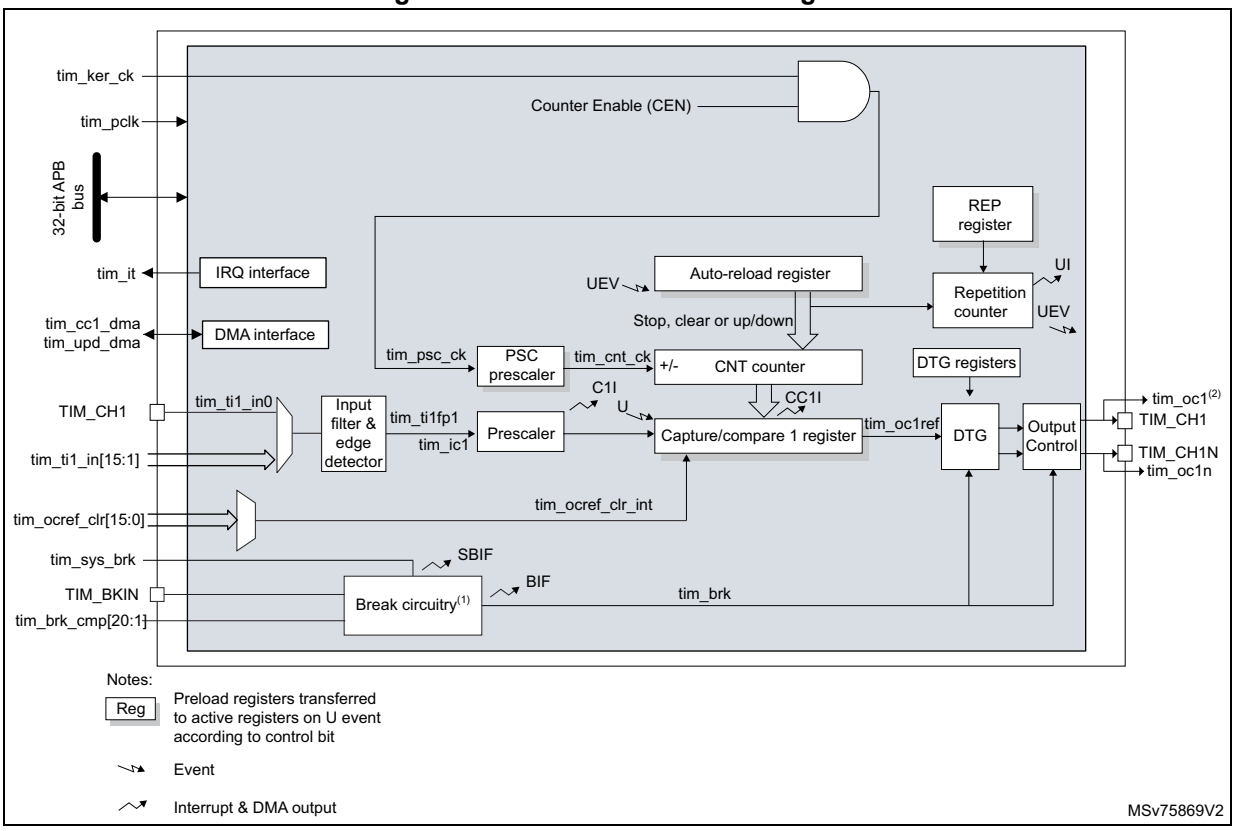
35.5.1 Block diagram

Figure 464. TIM15 block diagram



1. Refer to *Section 35.5.17: Using the break function* for details.

Figure 465. TIM16/TIM17 block diagram



1. Refer to *Section 35.5.17: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 35.5.34: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

35.5.2 TIM15/TIM16/TIM17 pins and internalsignals

Table 345 and *Table 346* in this section summarize the TIM inputs and outputs.

Table 345. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

Table 346. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocx (x = 1 to 2)	Output	Internal output compare signals of channels 1 to 2. These outputs can trigger or synchronize other internal peripherals.
tim_oc1n	Output	Internal timer complementary output derived from tim_oc1 internal output, with the option to insert dead time.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[20:1]	Input	Break input for internal signals
tim_sys_brk[5:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_cc2_dma	Output	Timer capture / compare 2 dma request
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 347 and *Table 348* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 347. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	comp1_out	comp1_out	comp1_out
tim_ti1_in2	comp2_out	Reserved	
tim_ti1_in3	Reserved	LSI	Reserved
tim_ti1_in4	Reserved	LSE	hse_1M_ck
tim_ti1_in5	LSE	rtc_wut_trg	MCO1
tim_ti1_in6	fdcan2_rxeof_evt	MCO1	MCO2
tim_ti1_in7	Reserved	MCO2	i3c1_ibiack
tim_ti1_in[15:8]	Reserved		

Table 348. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM15
tim_ti2_in0	TIM15_CH2
tim_ti2_in[6:1]	Reserved
tim_ti2_in7	fdcan2_txeof_evt
tim_ti2_in[15:8]	Reserved

Table 349 lists the internal sources connected to the tim_itr input multiplexer.

Table 349. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim1_trgo2
tim_itr2	tim2_trgo
tim_itr3	tim3_trgo
tim_itr4	tim4_trgo
tim_itr5	tim5_trgo
tim_itr6	tim8_trgo
tim_itr7	tim8_trgo2
tim_itr8	tim12_trgo
tim_itr9	Reserved

Table 349. TIMx internal trigger connection (continued)

tim_itrx inputs	TIM15
tim_itr10	tim16_oc1
tim_itr11	tim17_oc1
tim_itr[27:12]	Reserved

Table 350. Interconnect to the tim_brk_cmp multiplexer

tim_brk inputs	TIM15	TIM16	TIM17
TIM_BKIN	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp1	comp1_out		
tim_brk_cmp2	comp2_out	Reserved	
tim_brk_cmp[4:3]	Reserved		
tim_brk_cmp5	TIM1_BKIN		
tim_brk_cmp6	TIM8_BKIN		
tim_brk_cmp7	Reserved		
tim_brk_cmp8	Reserved	TIM15_BKIN	
tim_brk_cmp9	TIM16_BKIN1	Reserved	TIM16_BKIN
tim_brk_cmp10	TIM17_BKIN1		Reserved

Table 351 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 351. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment		
	TIM15	TIM16	TIM17
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	Reserved	
tim_ocref_clr[7:2]	Reserved		

Table 352. System break interconnect

tim_sys_brk inputs	TIM15/TIM16/TIM17	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	SRAM double ECC error	SEL
tim_sys_brk2	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk3	Non-Volatile memory double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)

35.5.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter can count up, down or both up and down.The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)
- Repetition counter register (TIM_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 466 and *Figure 467* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 466. Counter timing diagram with prescaler division change from 1 to 2

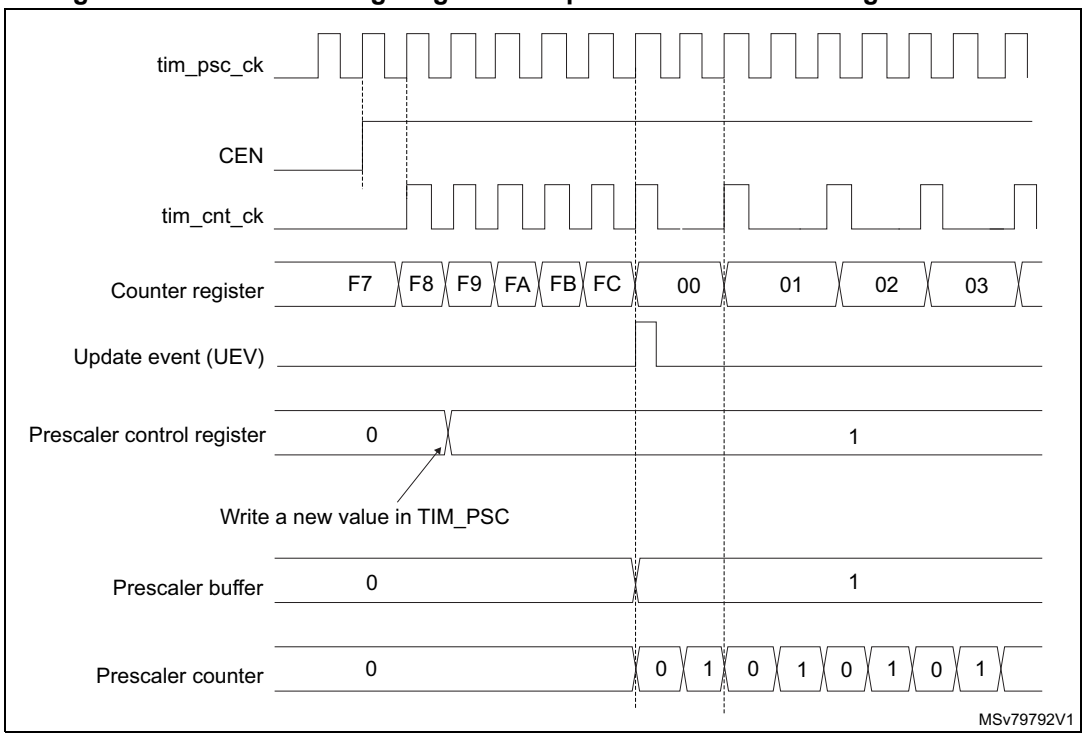
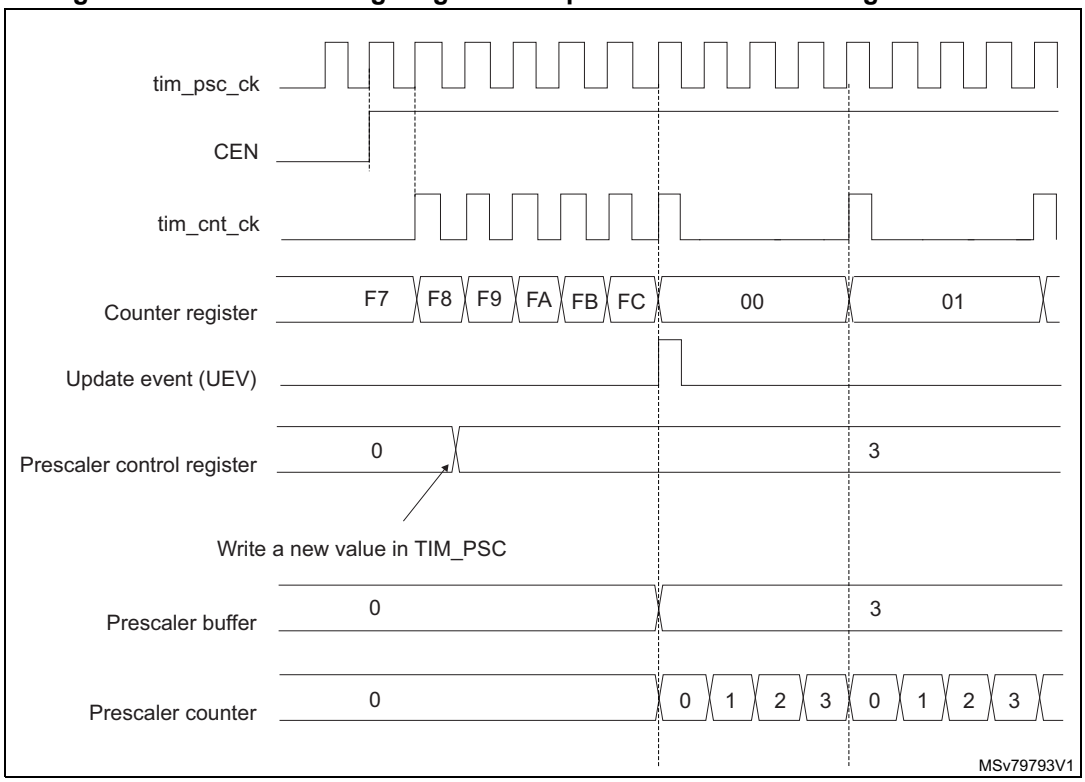


Figure 467. Counter timing diagram with prescaler division change from 1 to 4



35.5.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register,
- The autoreload shadow register is updated with the preload value (TIM_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 468. Counter timing diagram, internal clock divided by 1

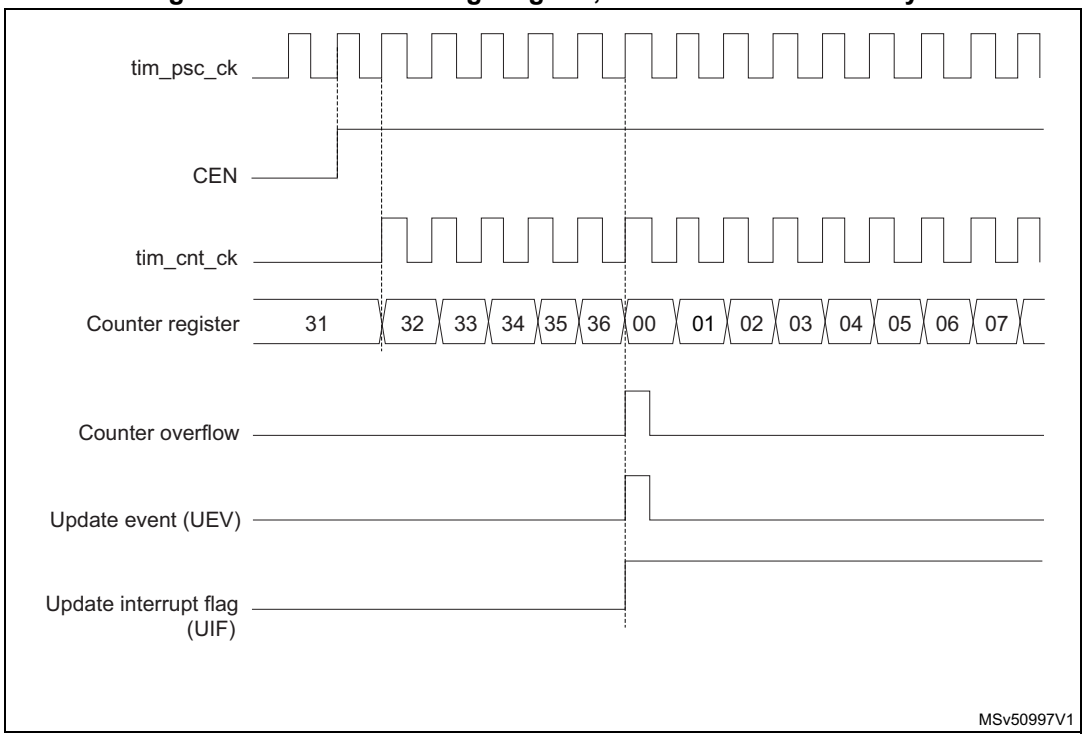


Figure 469. Counter timing diagram, internal clock divided by 2

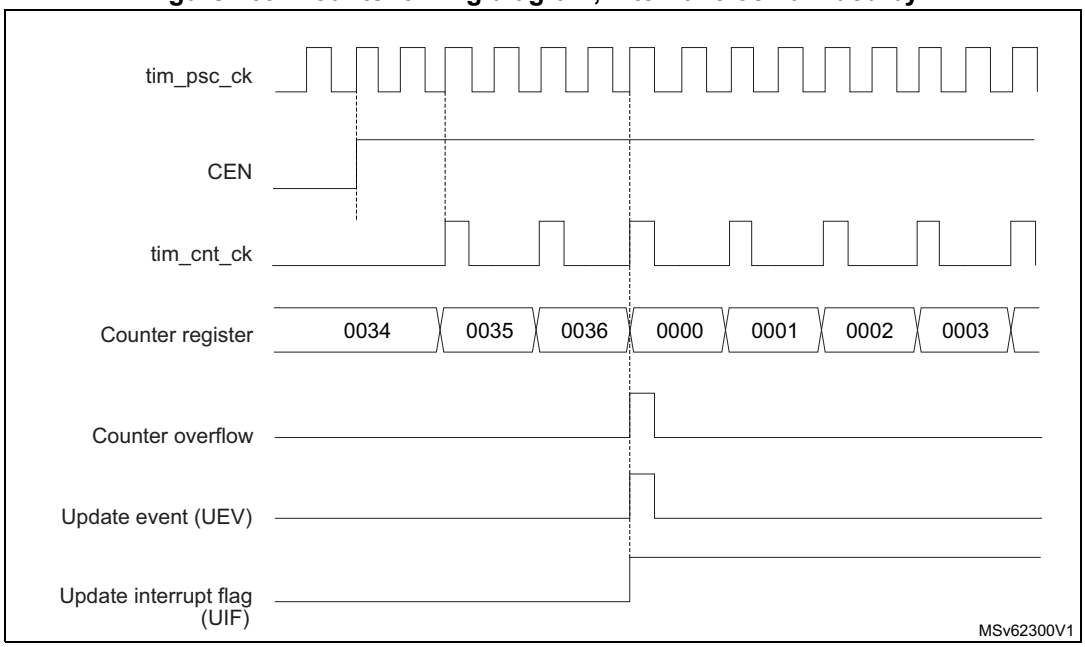


Figure 470. Counter timing diagram, internal clock divided by 4

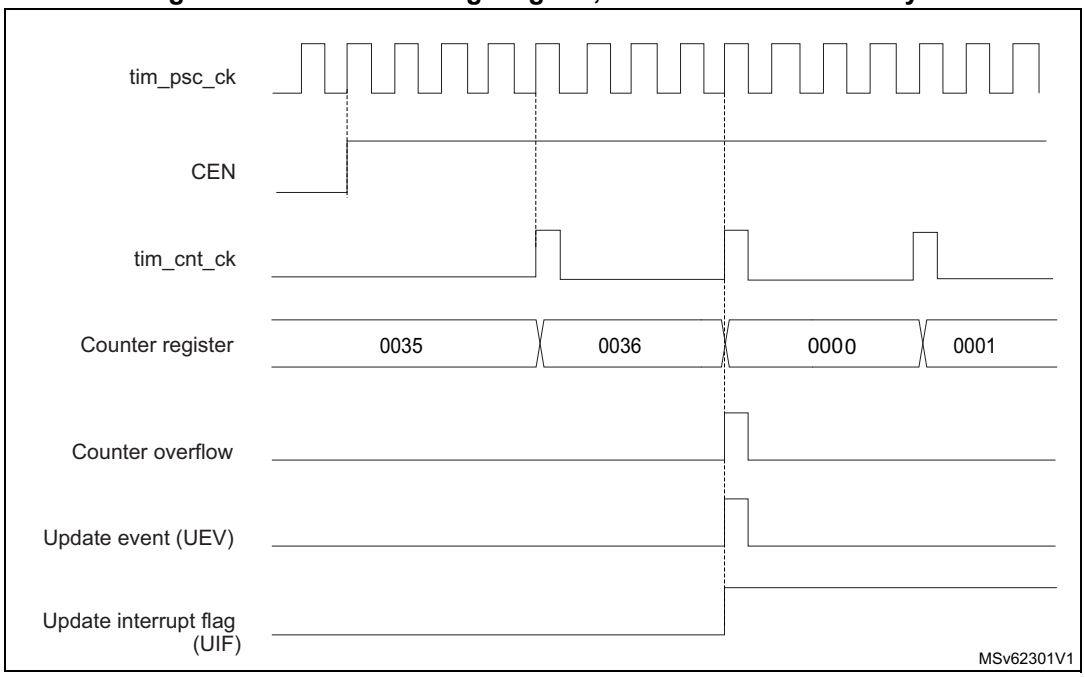


Figure 471. Counter timing diagram, internal clock divided by N

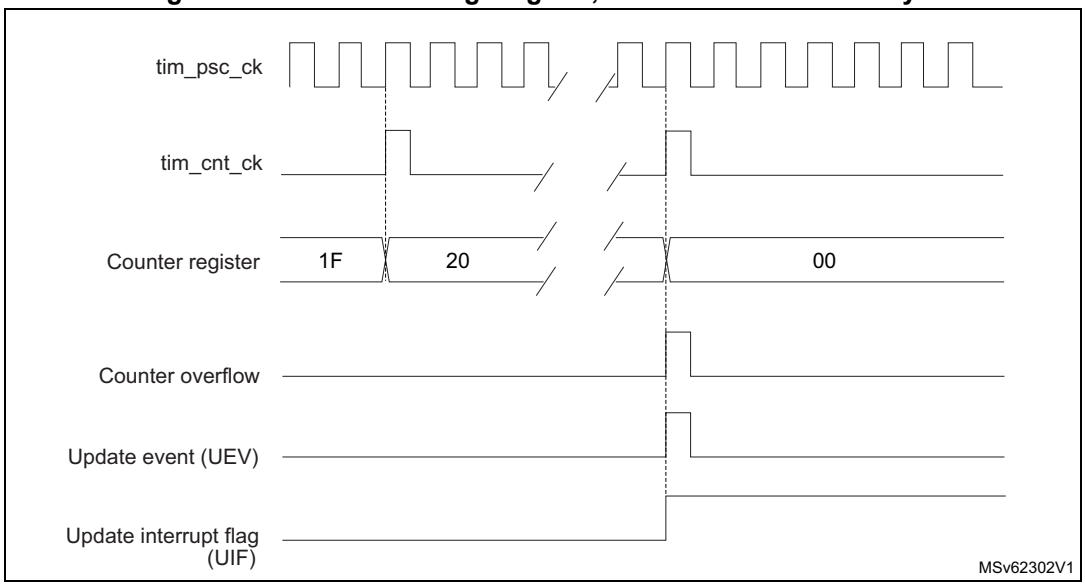


Figure 472. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

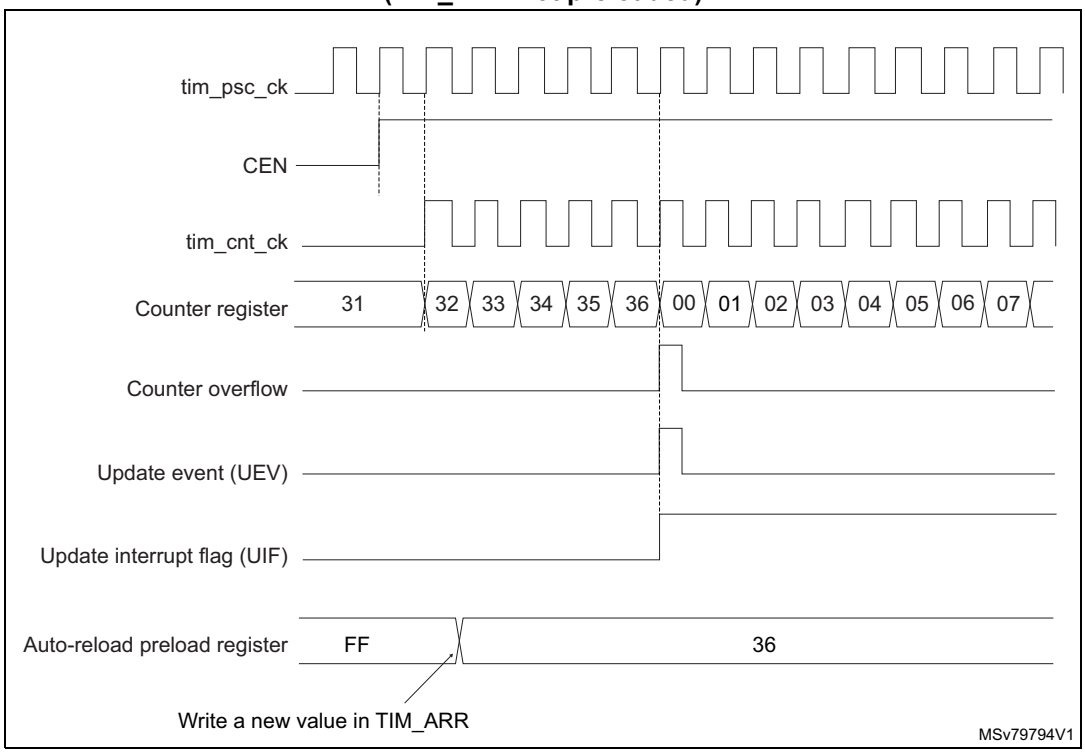
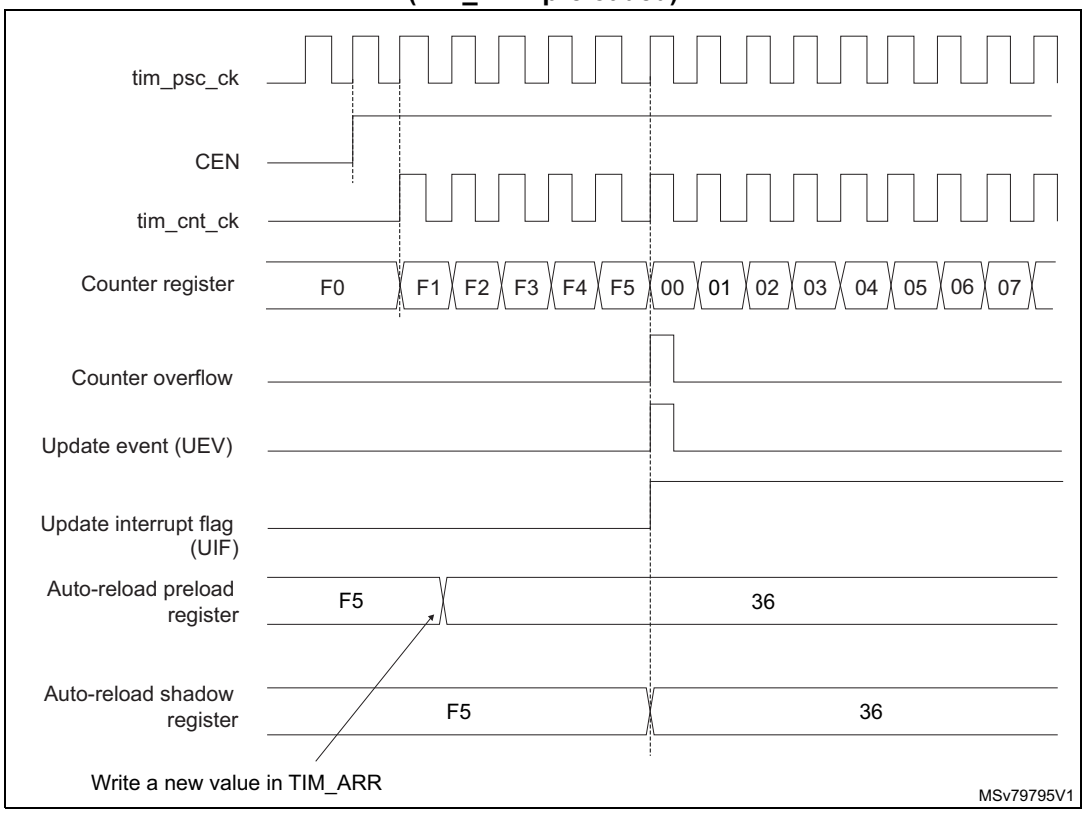


Figure 473. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 474. Counter timing diagram, internal clock divided by 1

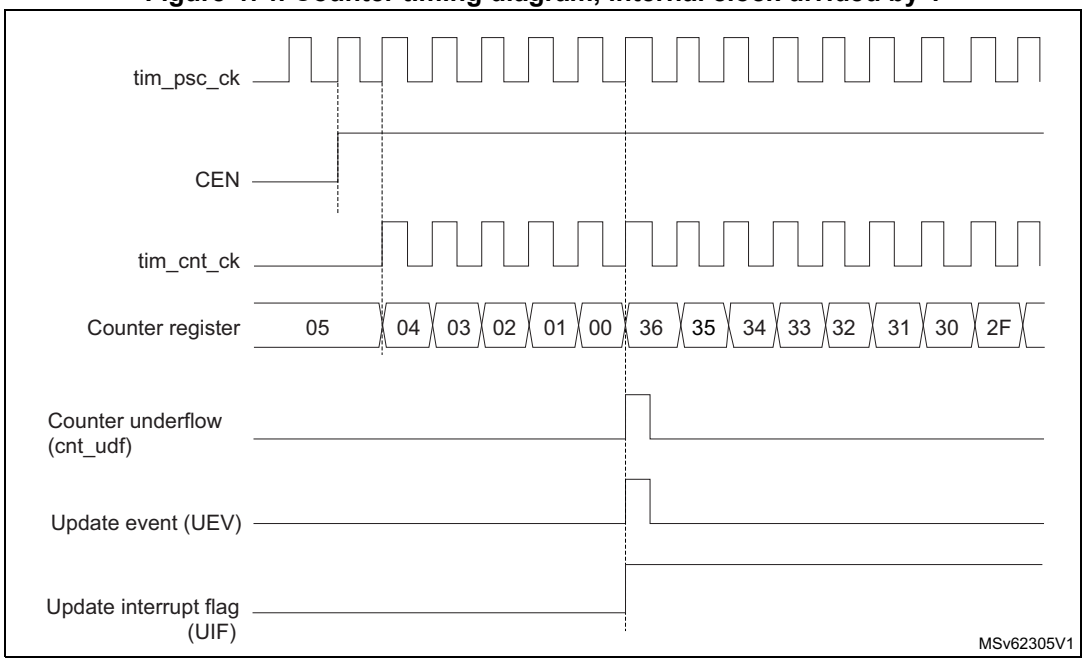


Figure 475. Counter timing diagram, internal clock divided by 2

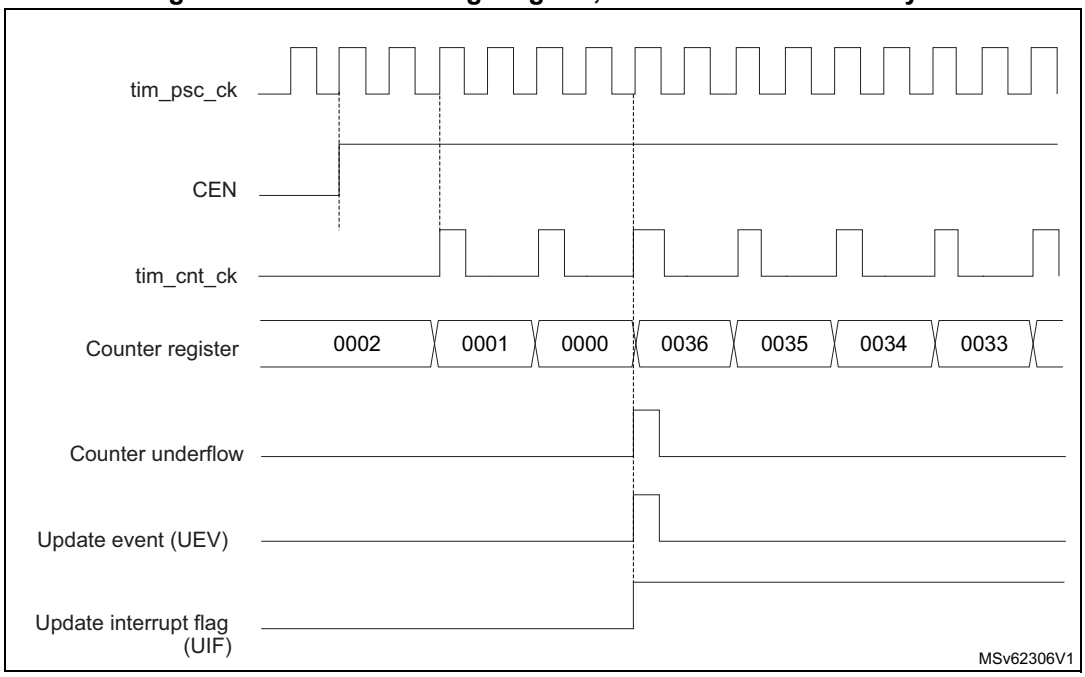


Figure 476. Counter timing diagram, internal clock divided by 4

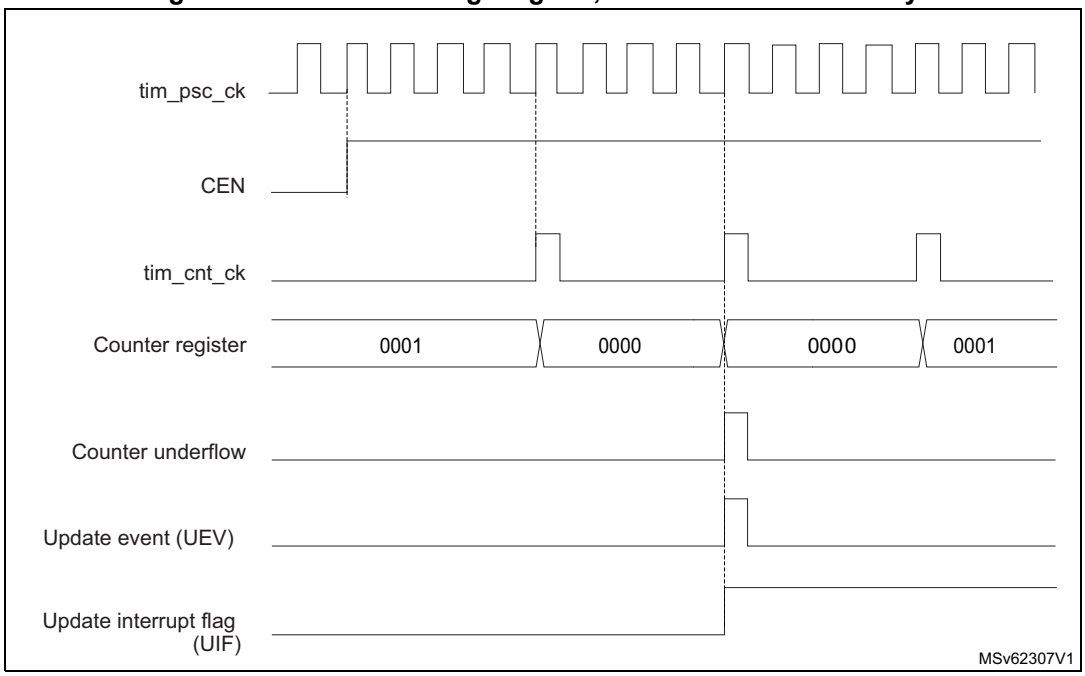


Figure 477. Counter timing diagram, internal clock divided by N

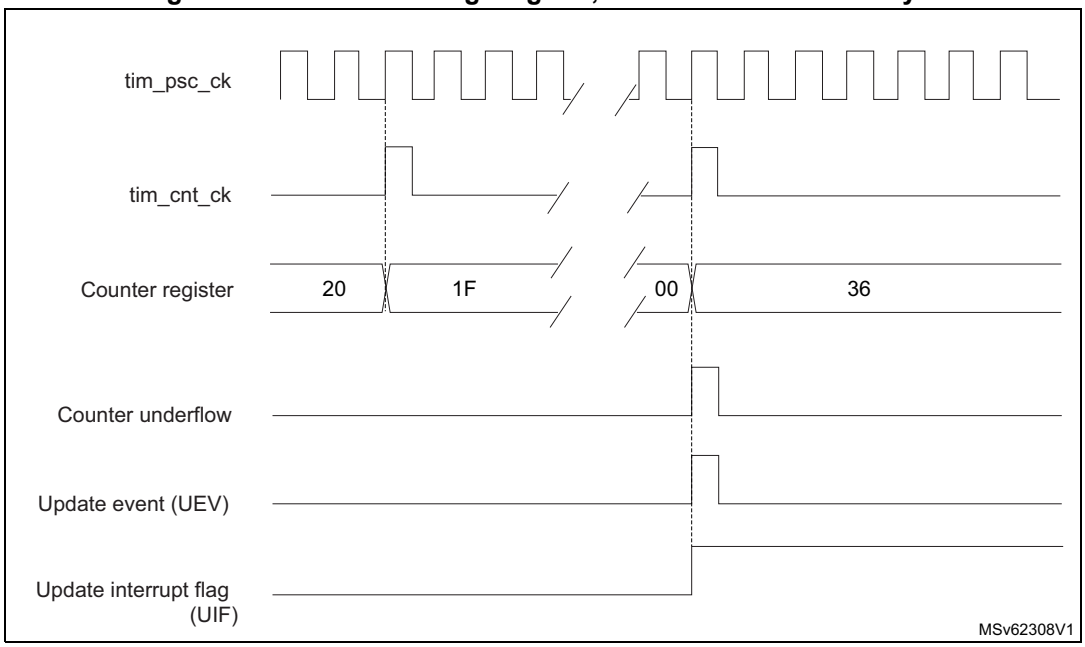
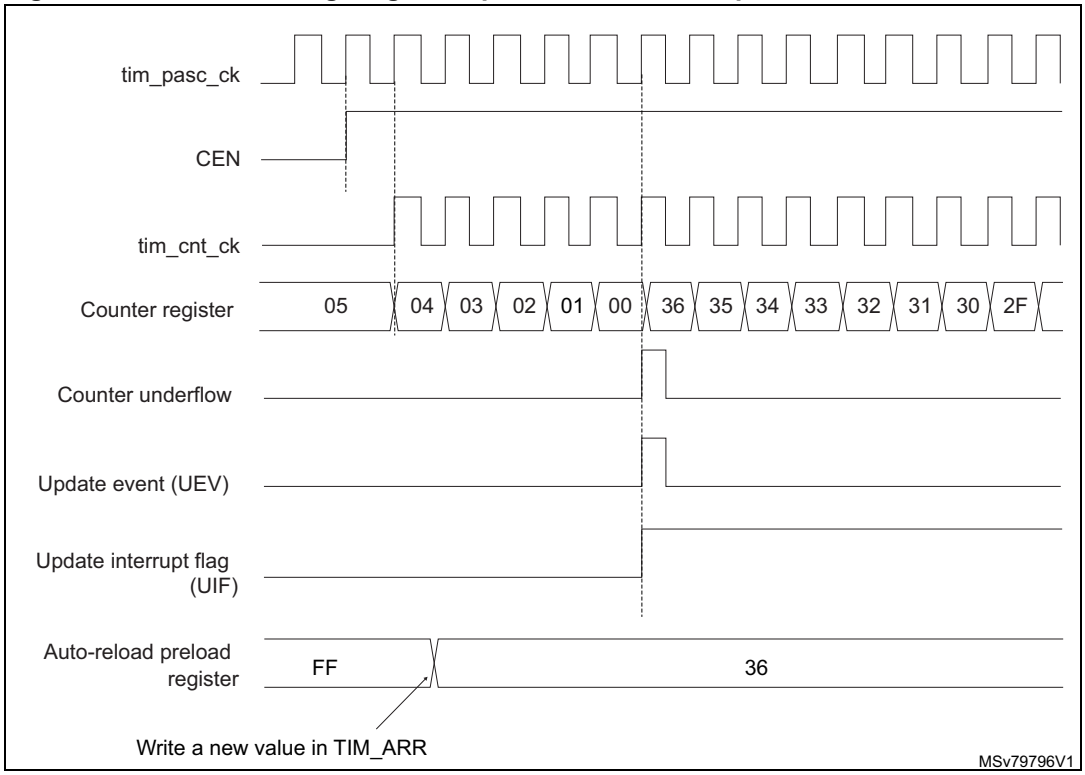


Figure 478. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIM_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

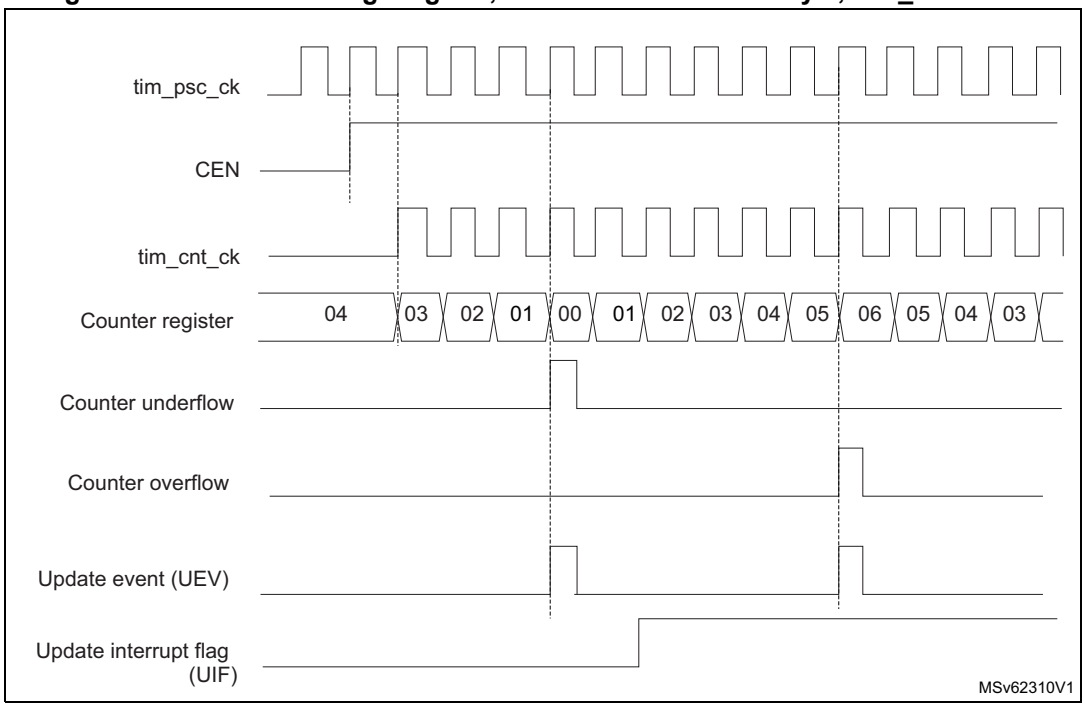
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register)
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 479. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 31.6: TIM1/TIM8 registers*).

Figure 480. Counter timing diagram, internal clock divided by 2

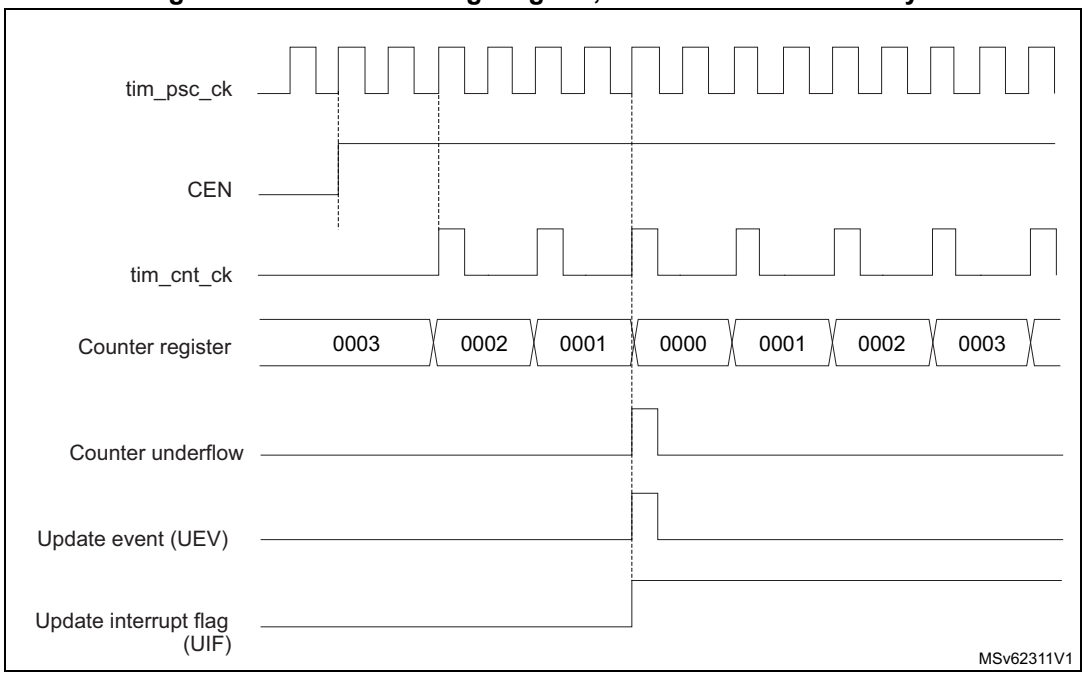


Figure 481. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36

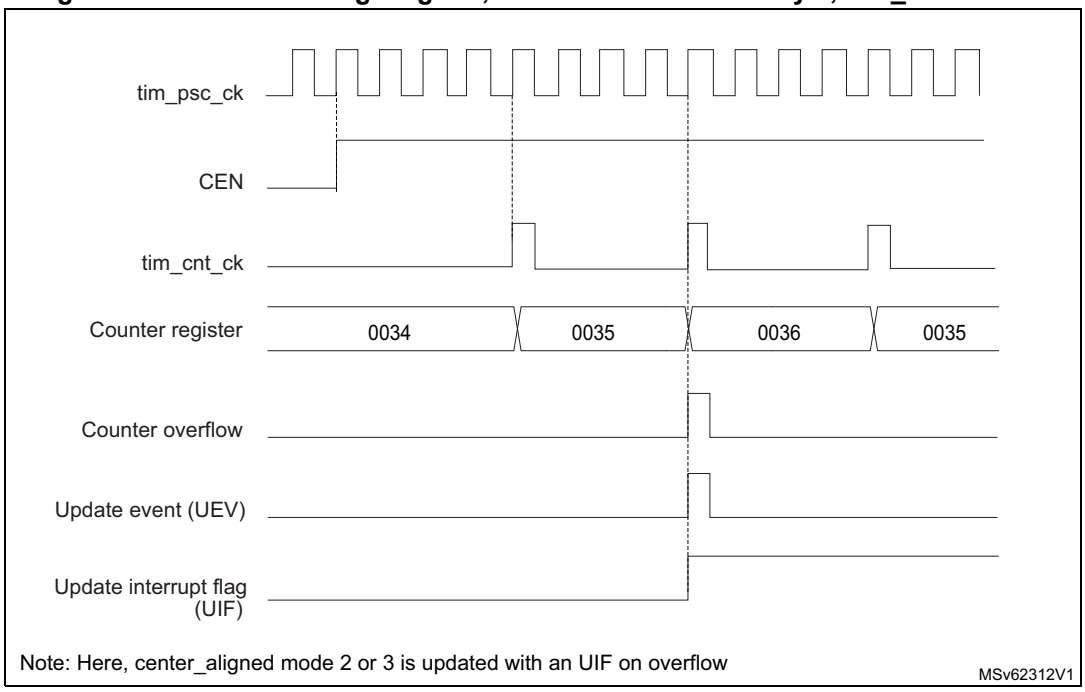


Figure 482. Counter timing diagram, internal clock divided by N

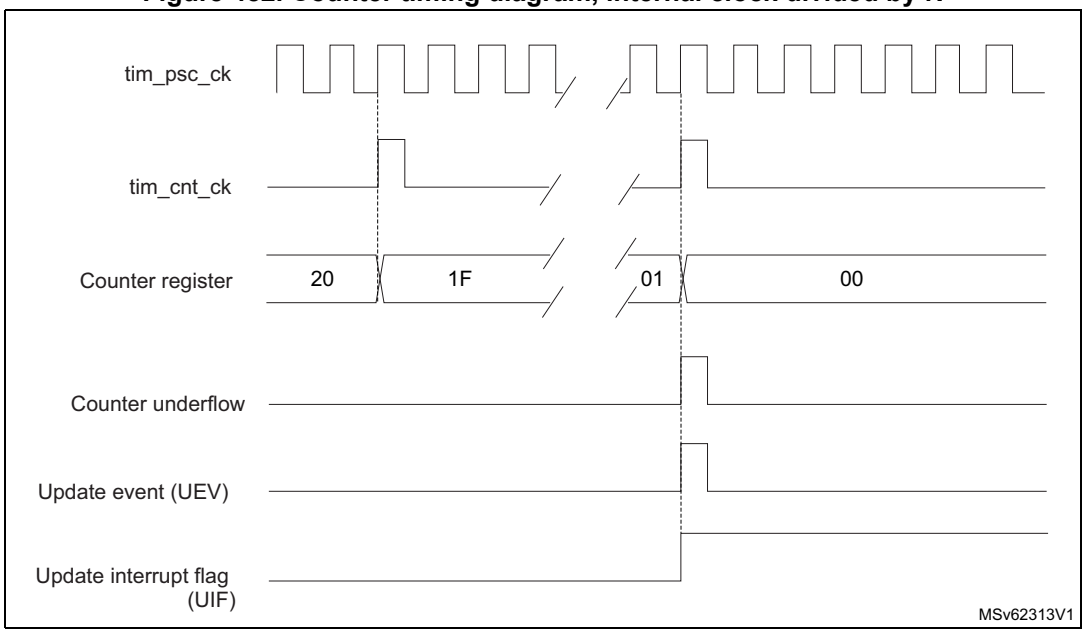


Figure 483. Counter timing diagram, update event with ARPE = 1 (counter underflow)

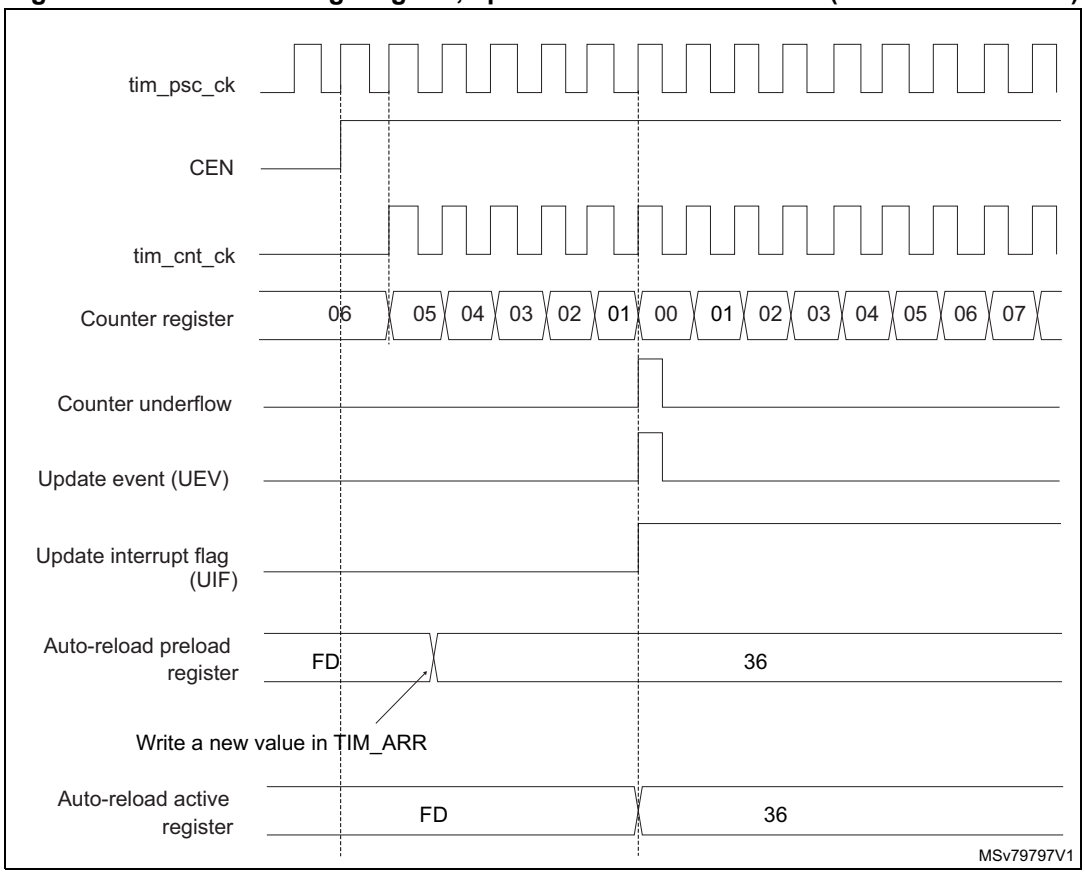
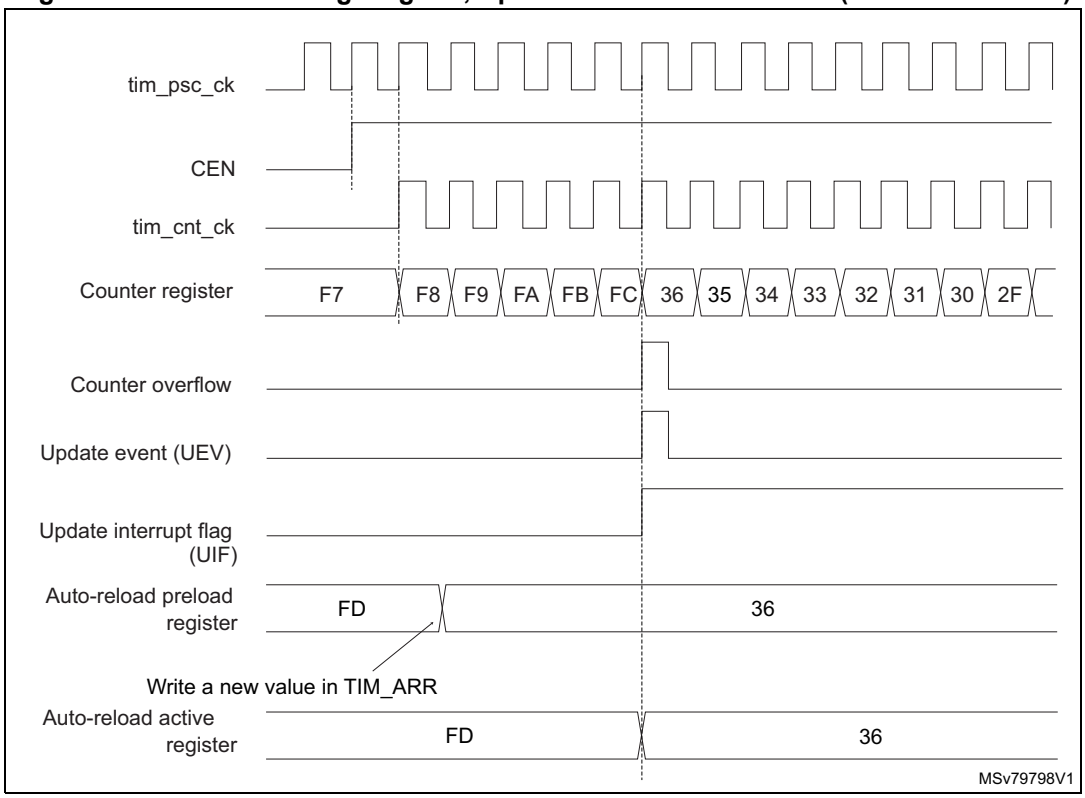


Figure 484. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



35.5.5 Repetition counter

Section 35.5.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIM_ARR autoreload register, TIM_PSC prescaler register, but also TIM_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIM_RCR repetition counter register.

The repetition counter is decremented:

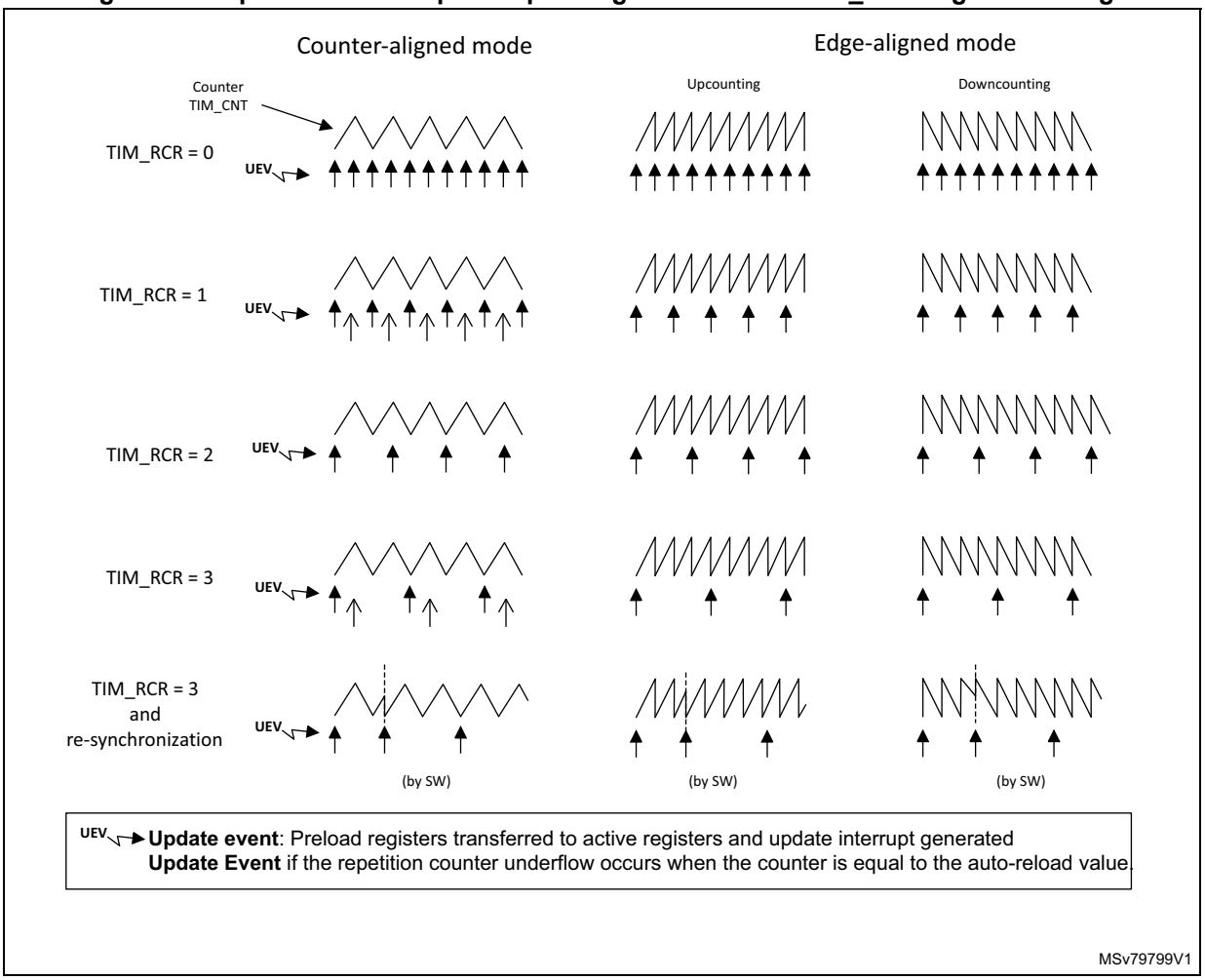
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIM_RCR register value (refer to *Figure 485*). When the update event is generated by software (by setting the UG bit in TIM_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIM_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 485. Update rate examples depending on mode and TIM_RCR register settings



35.5.6 Clock selection

The counter clock can be provided by the following clock sources:

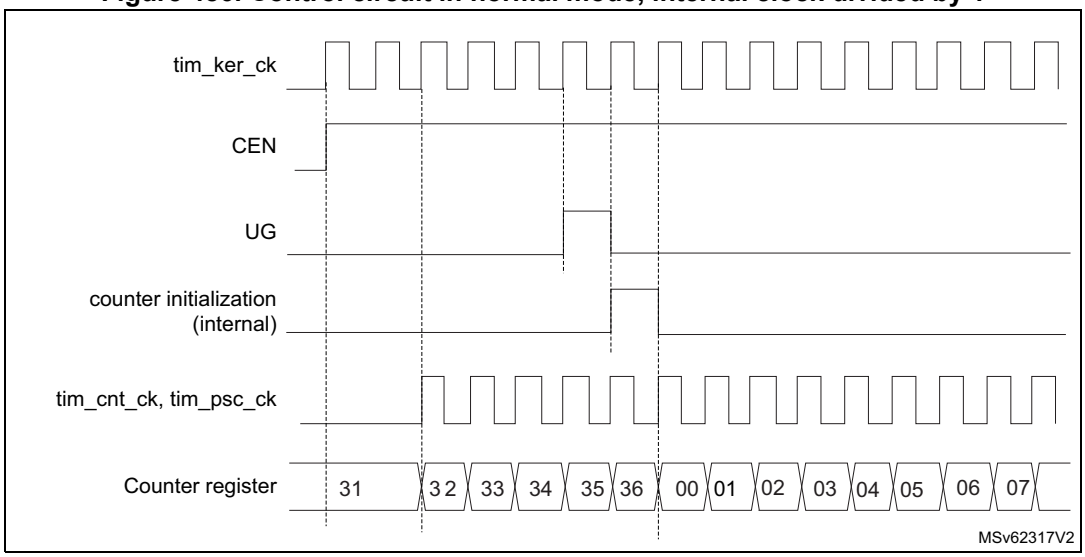
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer to enable another timer* for more details.
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 00000), then the CEN (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 486 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

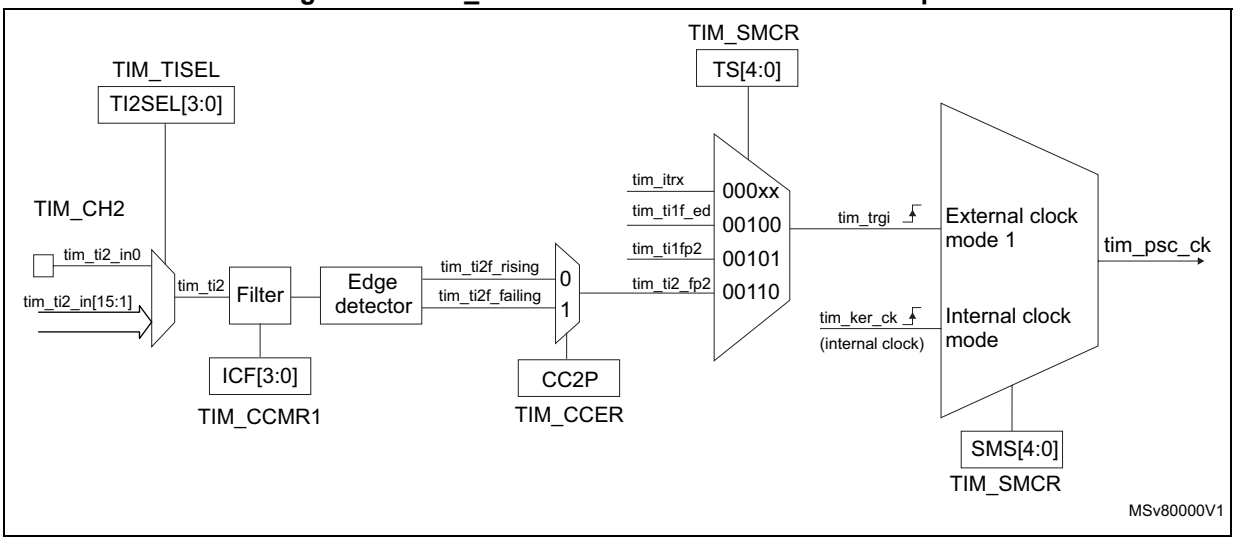
Figure 486. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 00111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 487. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

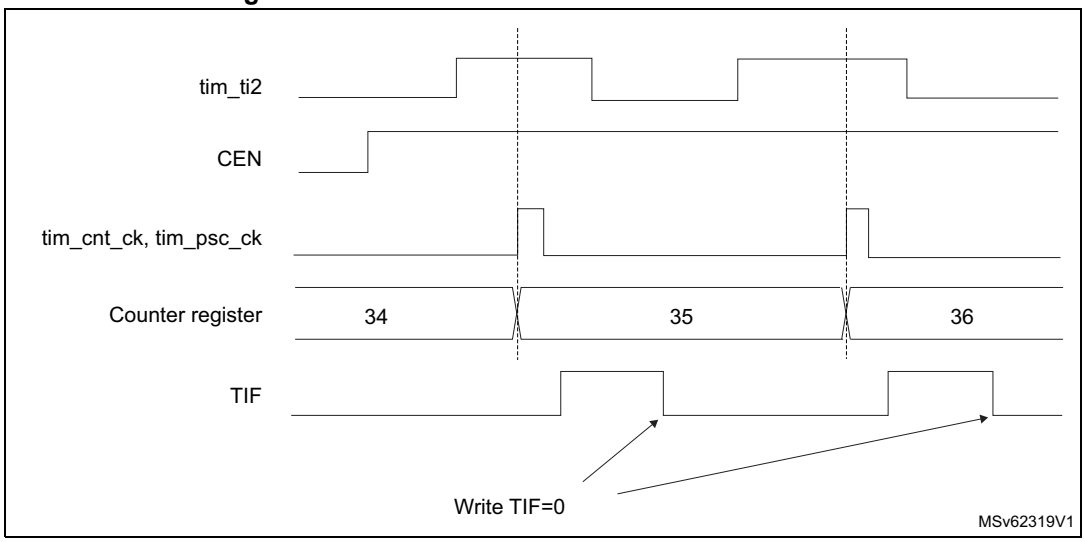
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 00111 in the TIM_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 488. Control circuit in external clock mode 1



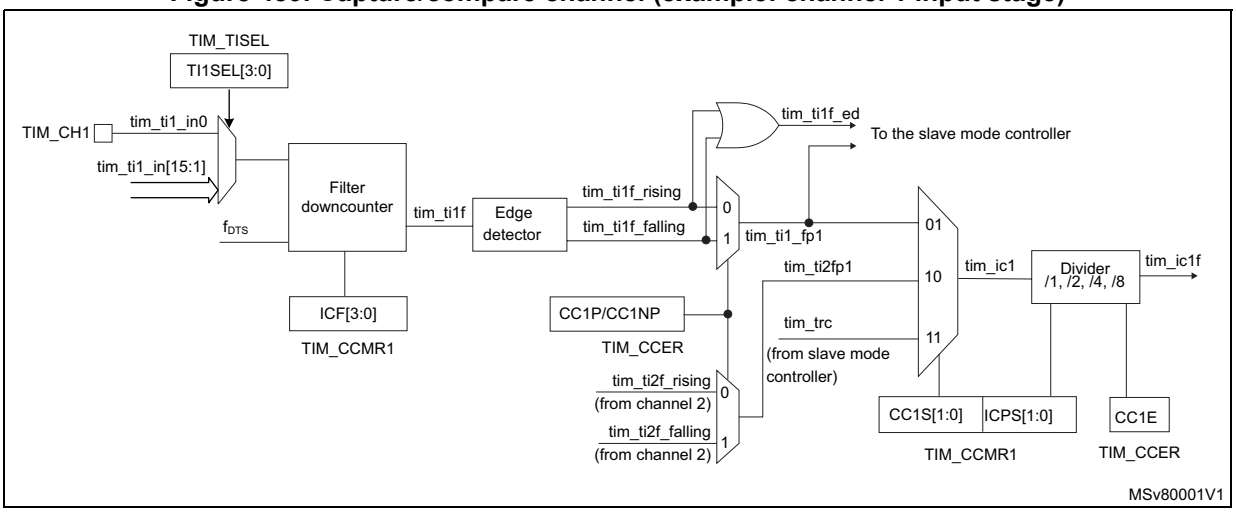
35.5.7 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 489 to *Figure 492* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 489. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 490. Capture/compare channel 1 main circuit

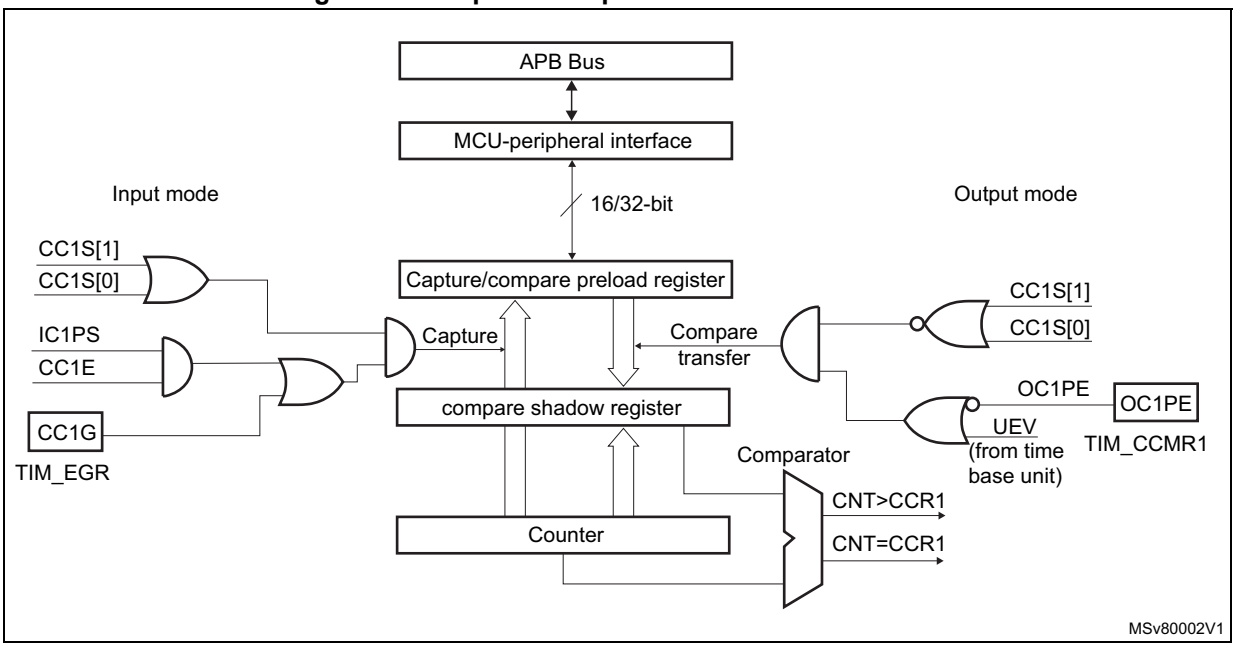


Figure 491. Output stage of capture/compare channel (channel 1)

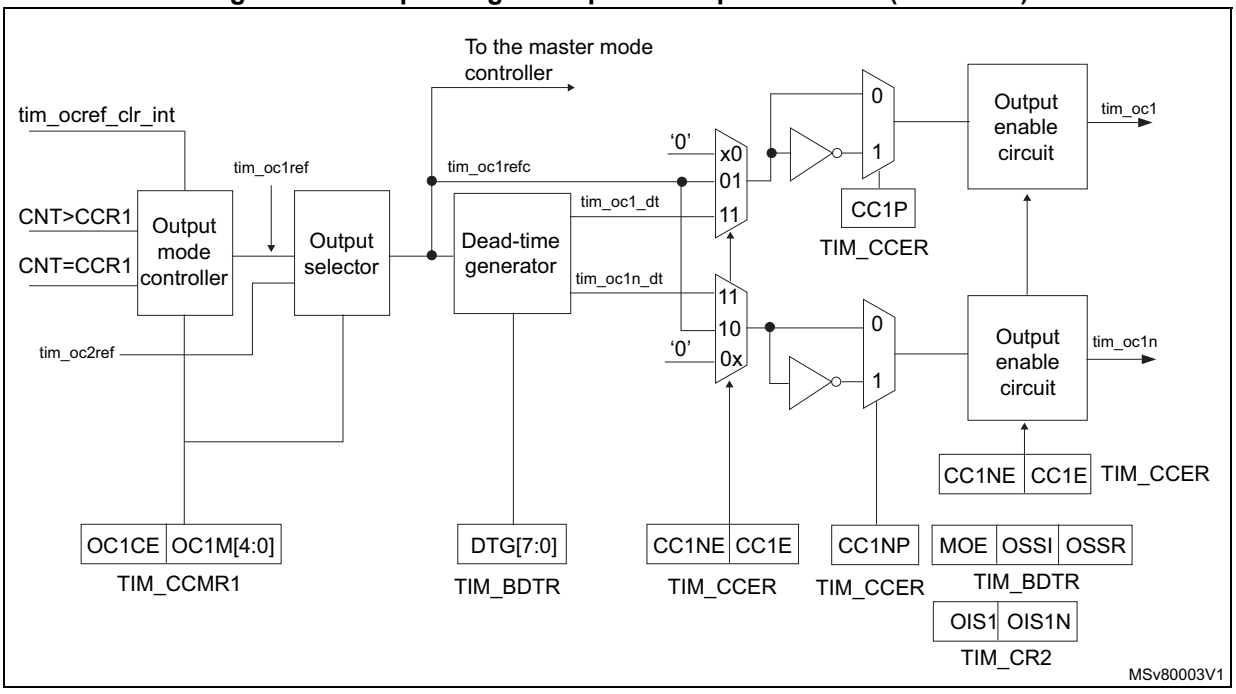
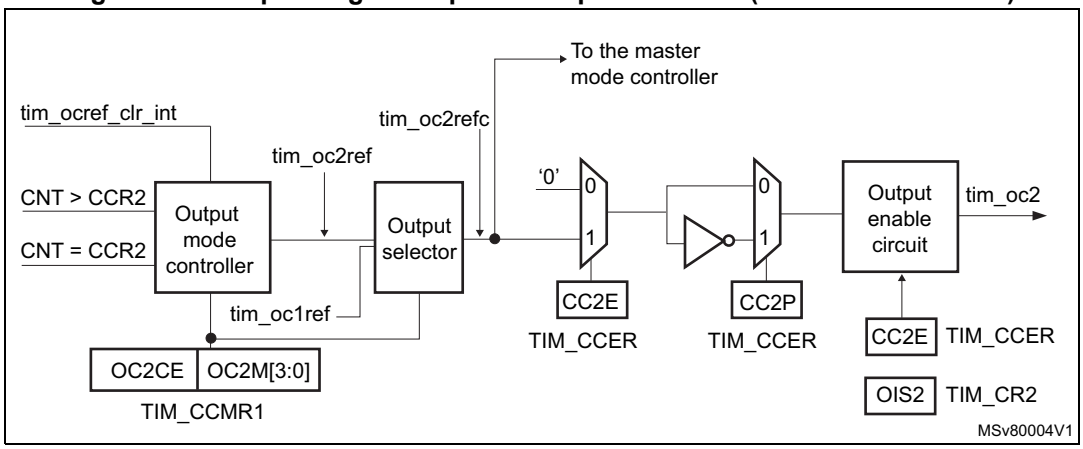


Figure 492. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

35.5.8 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIM_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new

level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.

4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

35.5.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

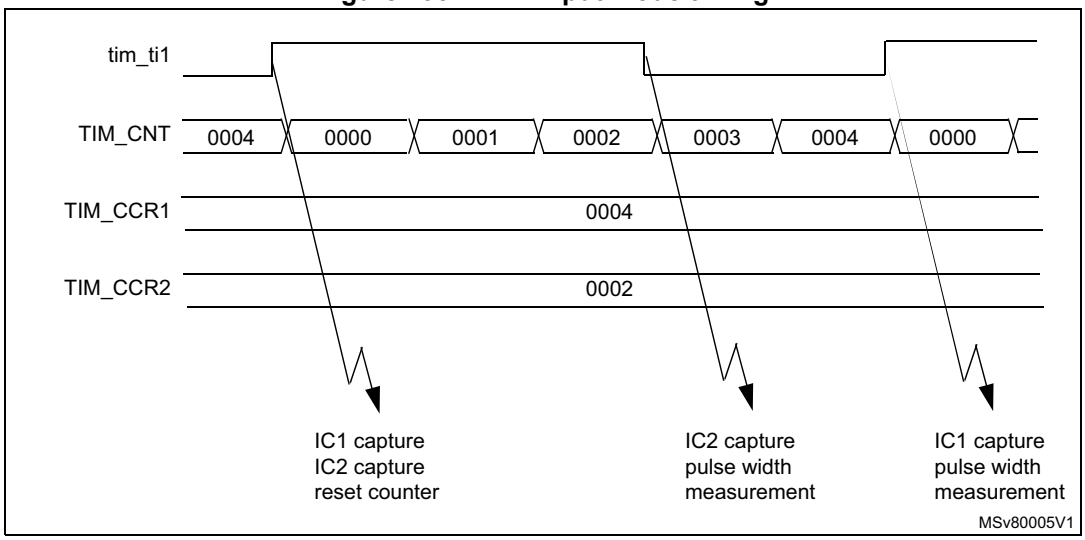
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIM_CCR1 register) and the duty cycle (in TIM_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 00100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 493. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

35.5.10 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

35.5.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCXM = 00000), be set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

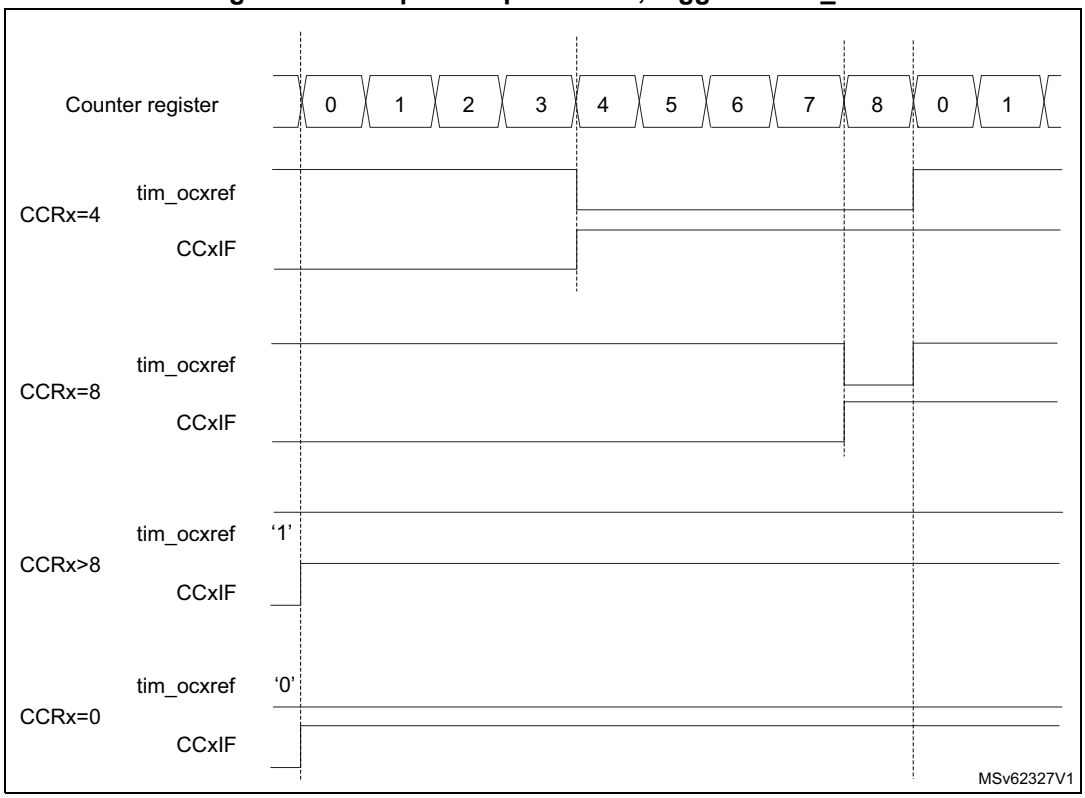
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 494*.

Figure 494. Output compare mode, toggle on tim_oc1



35.5.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIM_CCER and TIM_BDTR registers). Refer to the TIM_CCER register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

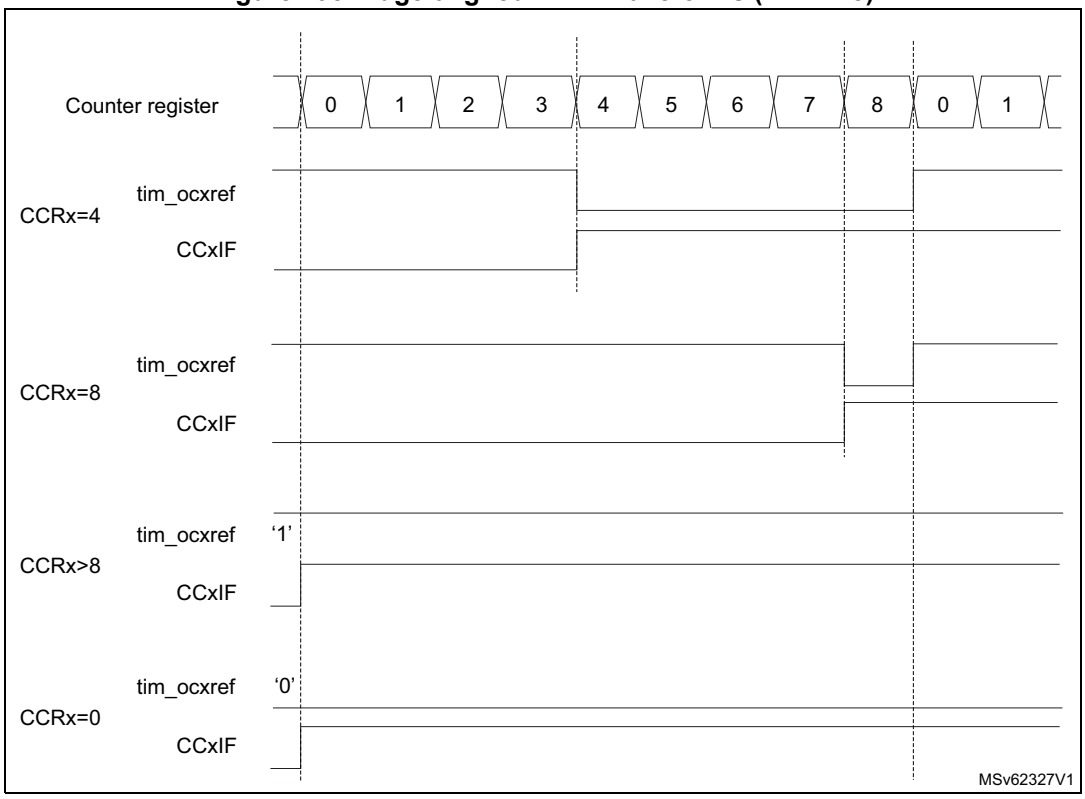
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 495* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 495. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIM_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT > TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

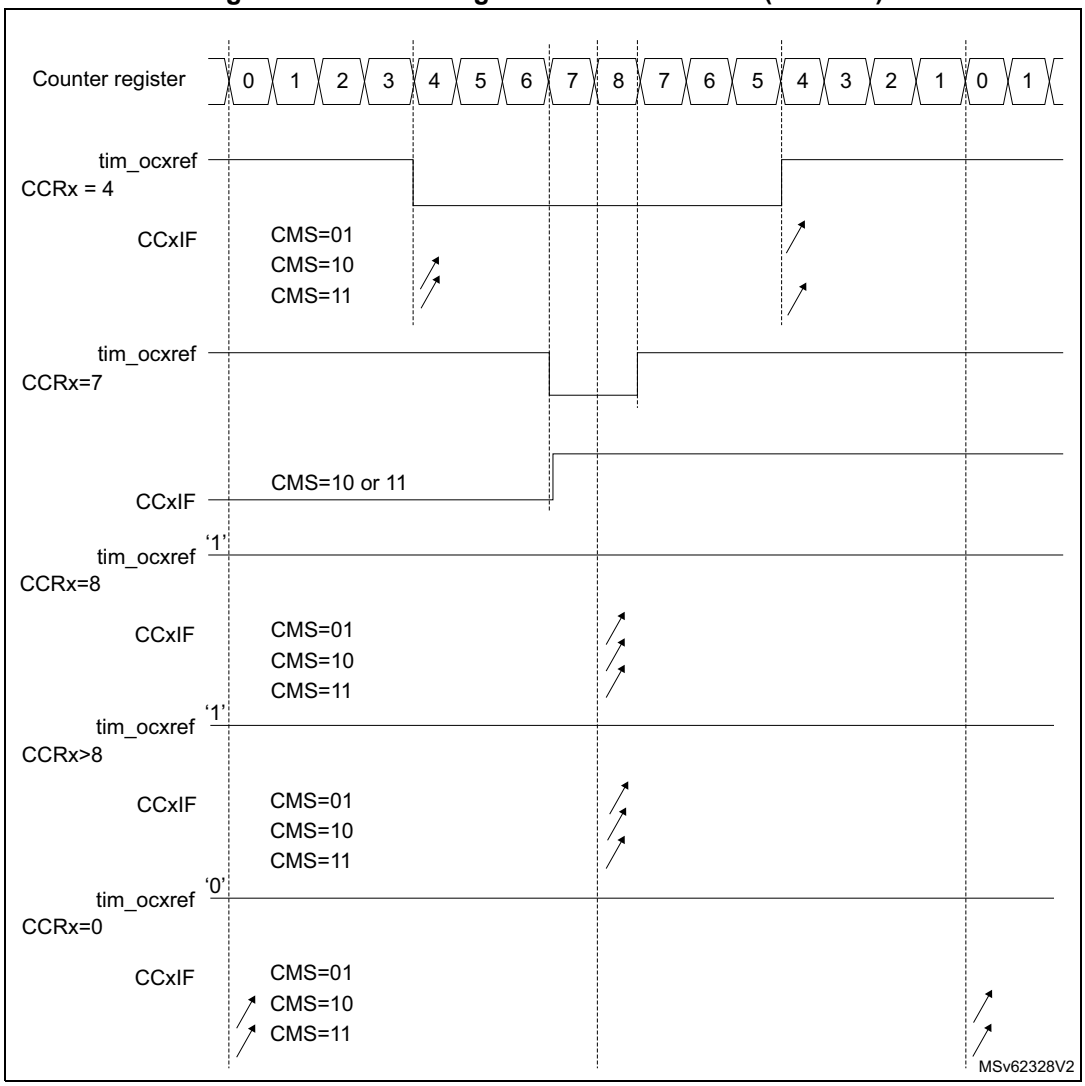
Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 496 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 496. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

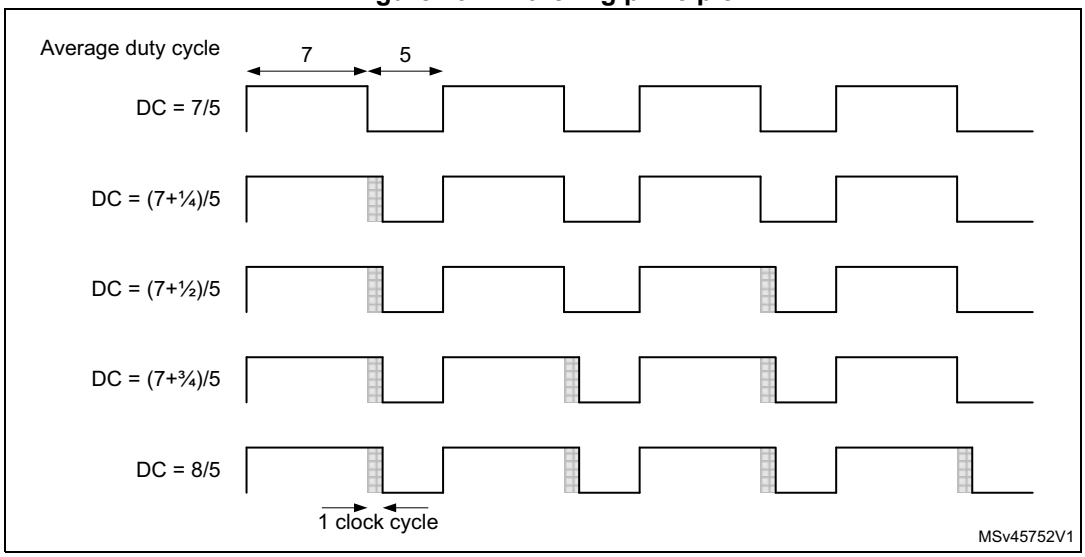
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT > TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 497* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 497. Dithering principle

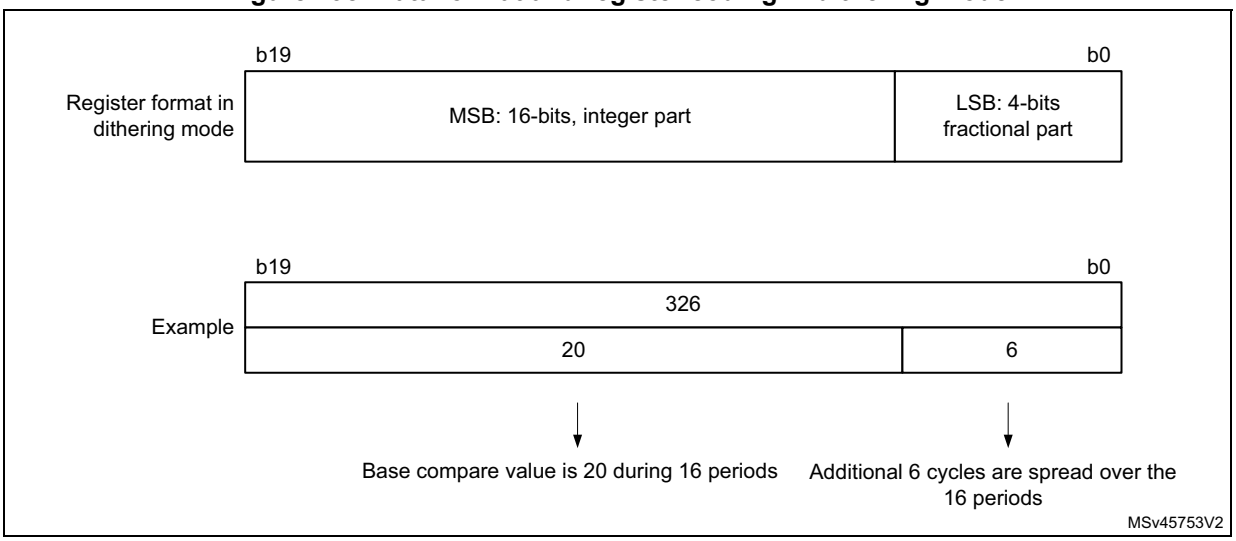


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 498* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 498. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

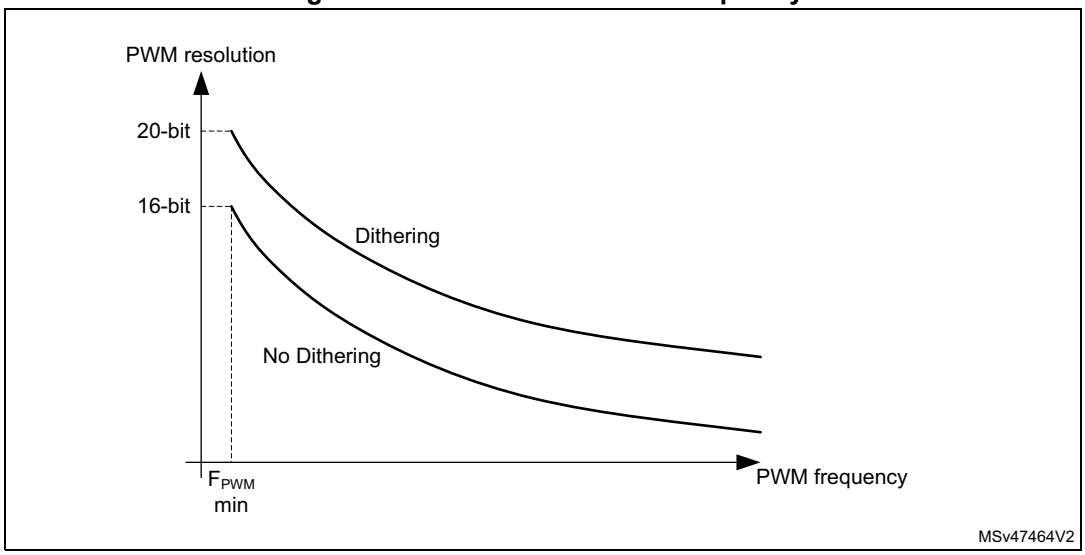
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

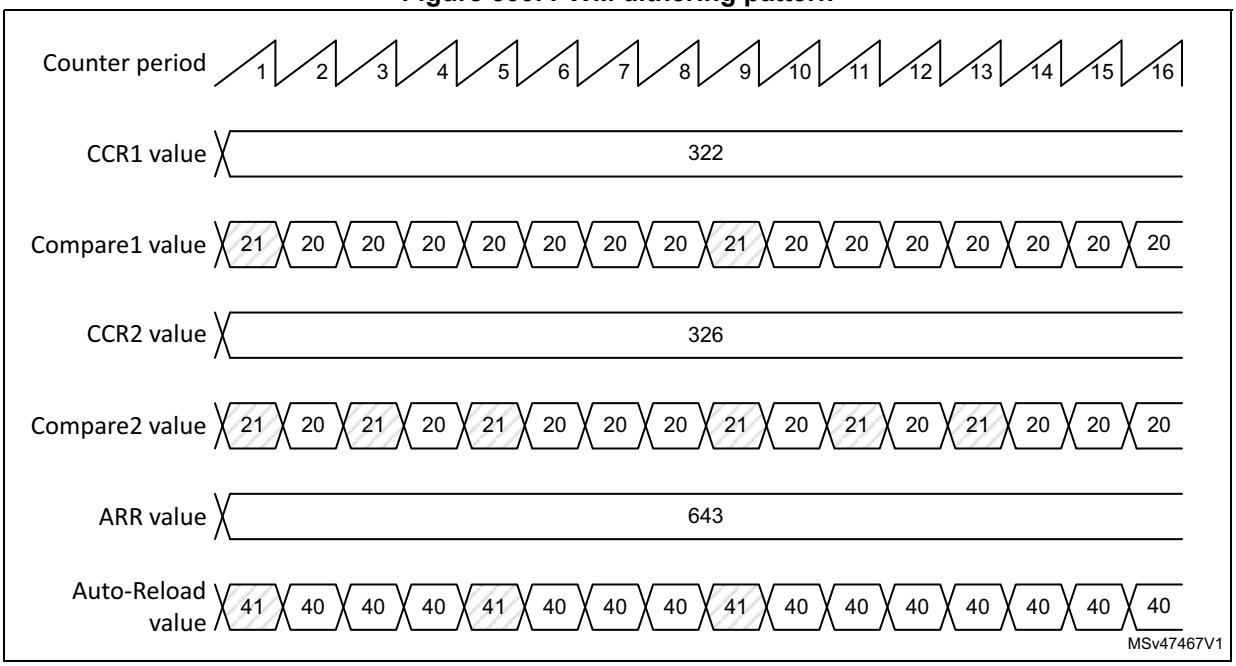
As shown on the *Figure 499* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 499. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 500* below.

Figure 500. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 353* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 353. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 501*.

Figure 501. Dithering effect on duty cycle in center-aligned PWM mode

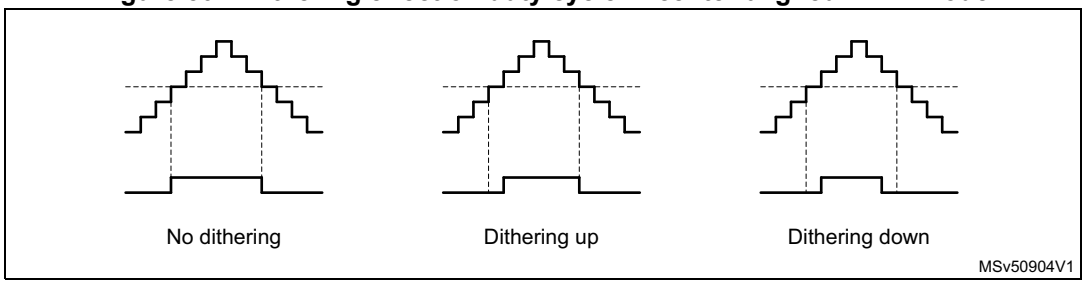


Table 354 shows how the dithering pattern is added in center-aligned PWM mode.

Table 354. CCR register change dithering pattern in center-aligned PWM mode

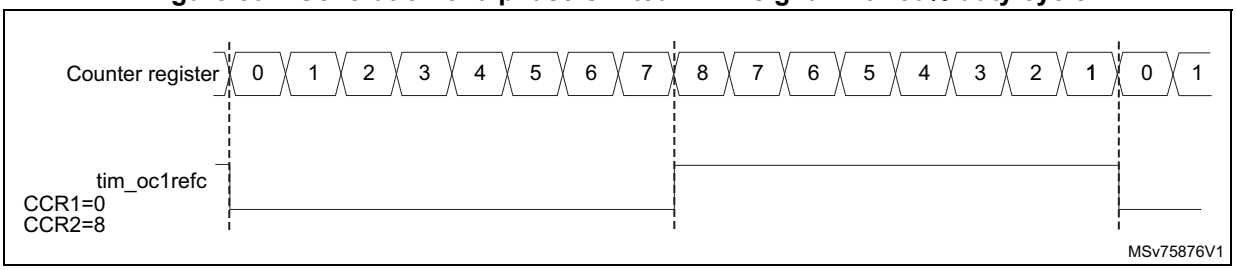
LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

35.5.13 Asymmetric PWM mode (TIM15 only)

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle. For instance, tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2.

Figure 502 represents an example of signal that can be generated using asymmetric PWM mode (channels 1 and 2 are configured in asymmetric PWM mode 2).

Figure 502. Generation of a phase-shifted PWM signal with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 355*. The selection is done with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (for example asymmetric PWM mode 3 and 4), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 and 2, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 01110 for the asymmetric PWM mode 1 on oc1refc.

The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if a tim_oc1refc signal is generated on channel 1 (asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2 or a tim_oc2refc signal resulting from asymmetric PWM mode 1.

Table 355. Asymmetric PWM modes

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	PWM Mode(1)
Asymmetric PWM mode 1	01110	oc1ref (up-counting)		1
Asymmetric PWM mode 2	01111	oc2ref (down-counting)		2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value.
PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the 2 channels used for the asymmetric mode.

35.5.14 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. Both signals, tim_oc1refc and tim_oc2refc, are made of an OR or AND logical combination of two reference PWMs controlled respectively by TIM_CCR1 and TIM_CCR2.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 01100 (Combined PWM mode 1) or 01101 (Combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

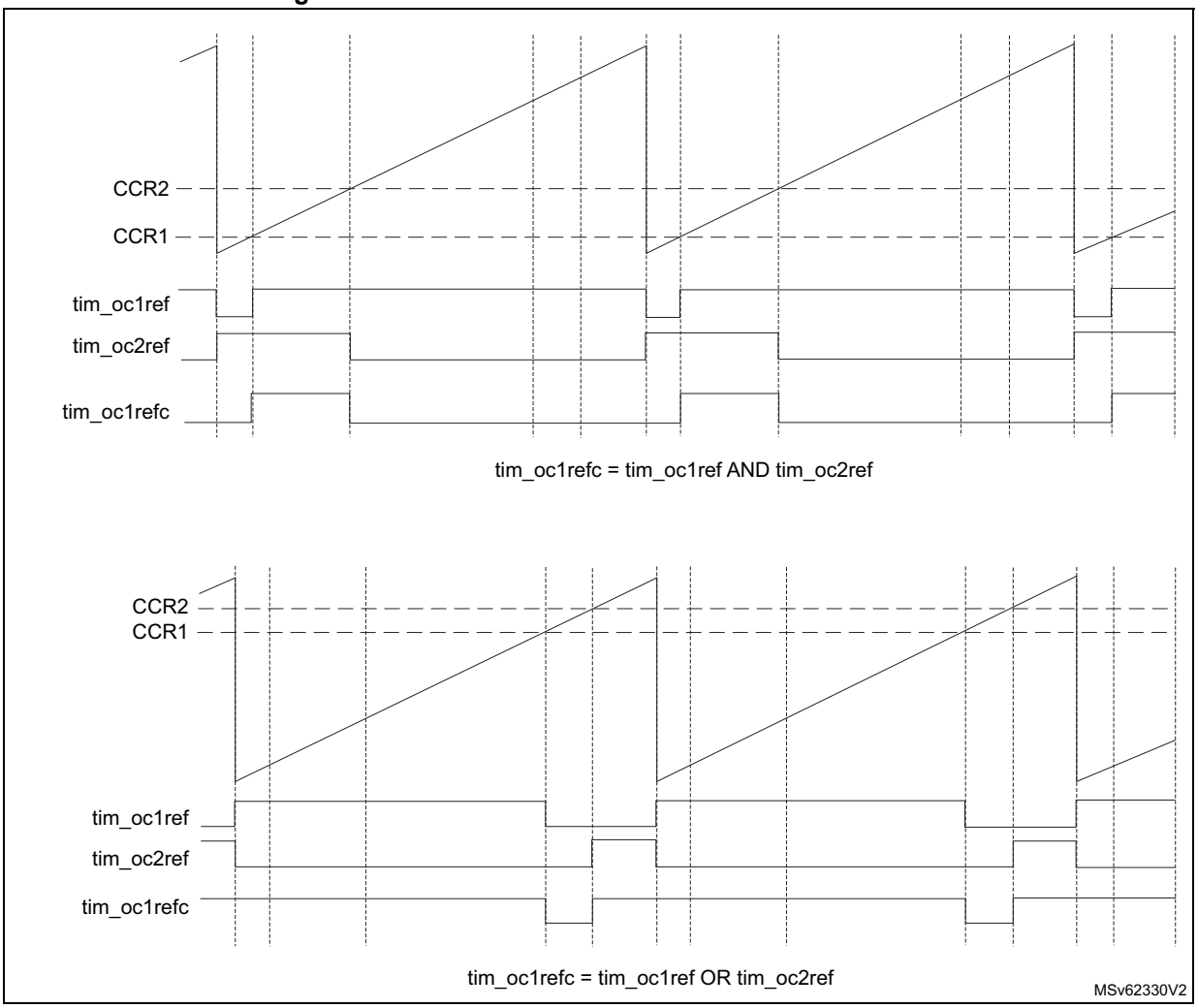
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 503 represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 503. Combined PWM mode on channel 1 and 2



35.5.15 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIM_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIM_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIM_BDTR and TIM_CR2 registers. Refer to *Table 362: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) on page 1431* and *Table 364: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 1463* for more details. In

particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

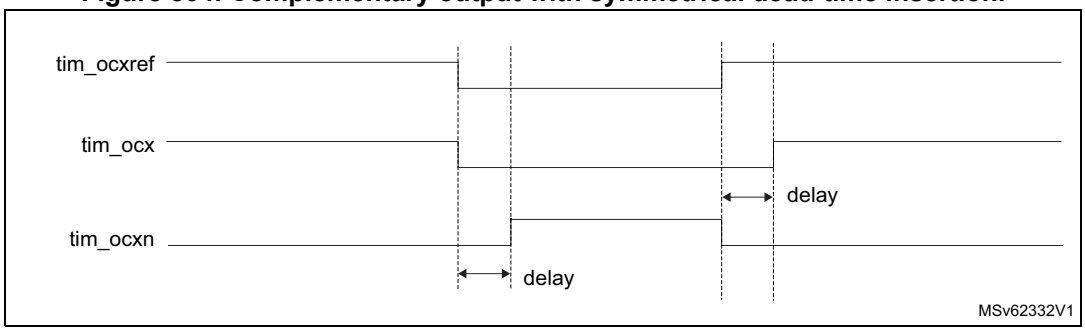
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 504. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIM_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 505*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIM_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIM_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIM_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 505. Asymmetrical deadtime

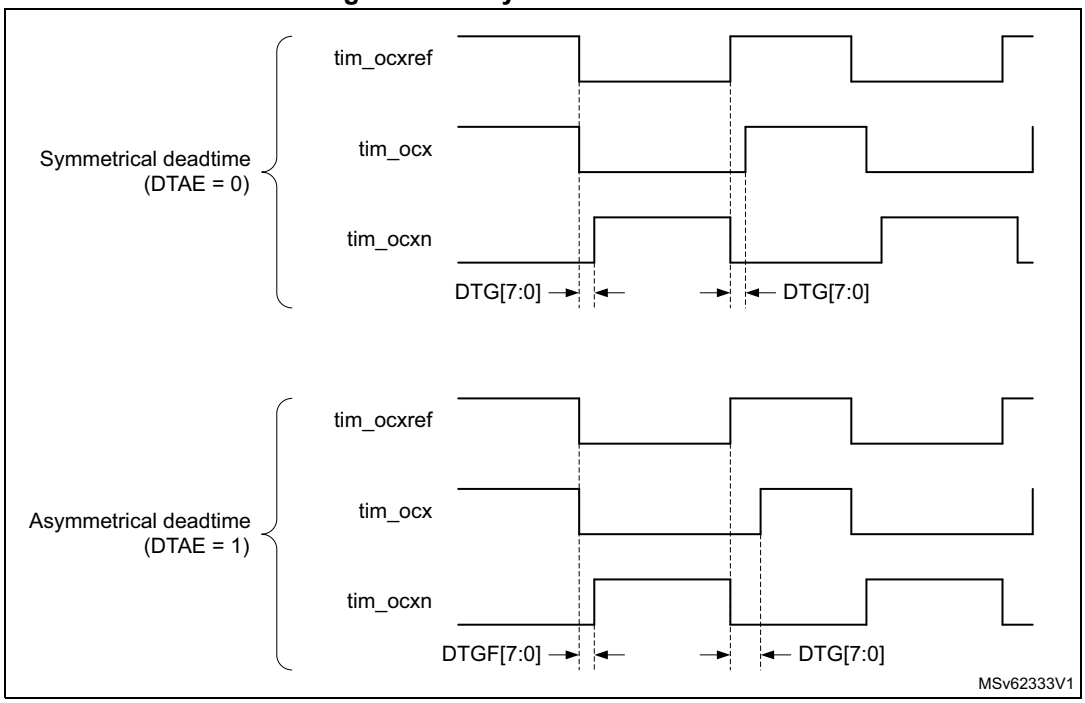


Figure 506. Dead-time waveforms with delay greater than the negative pulse.

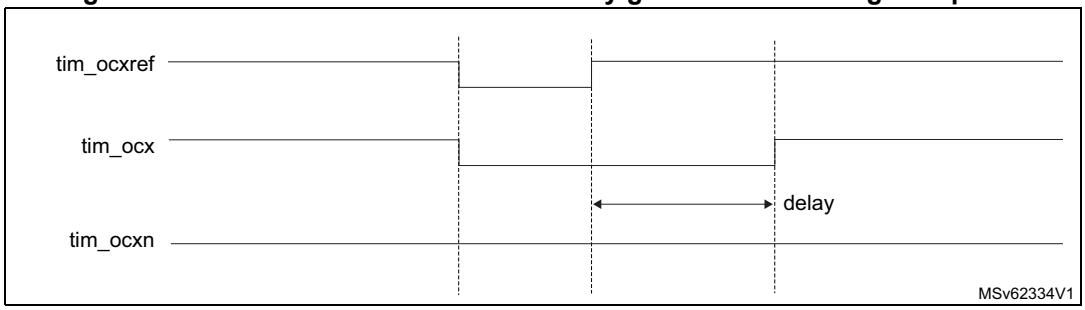
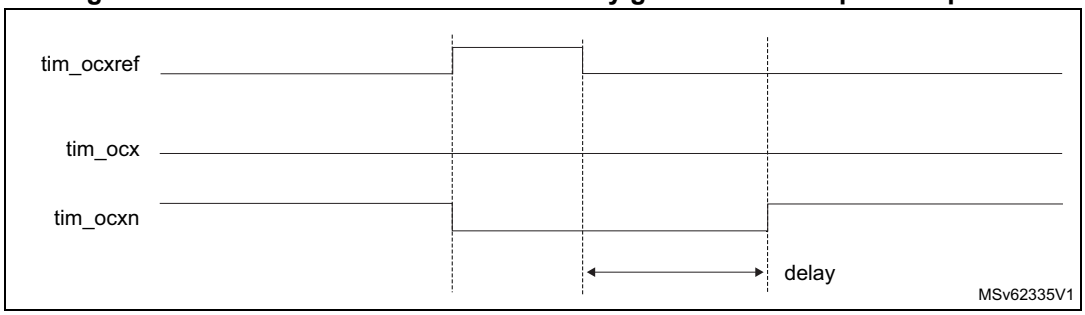


Figure 507. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIM_BDTR register. Refer to *Section 35.8.16: TIM break and dead-time register (TIM_BDTR) on page 1435* and *Section 35.9.14: TIM break and dead-time register (TIM_BDTR) on page 1467* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIM_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

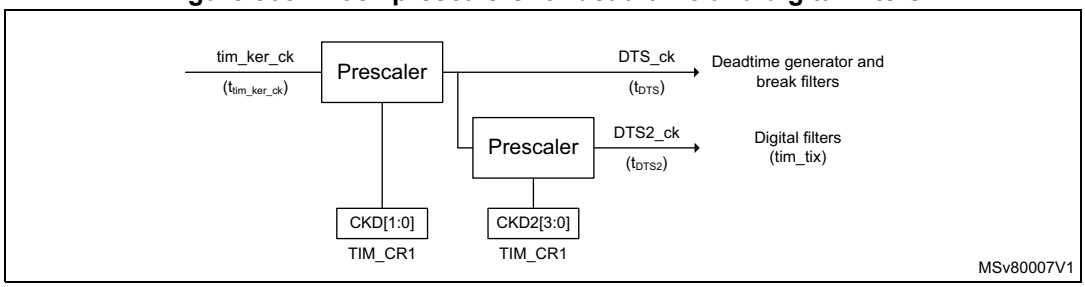
Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

35.5.16 Clock dividers for dead-time generators and digital filters

The dead-time generators and the digital filters (for tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the deadtime period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend solely the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 508. Clock prescalers for dead-time and digital filters



35.5.17 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIM_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIM_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIM_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 362: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) on page 1431* and *Table 364: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 1463* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIM_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIM_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIM_BDTR register).
- Programmable enable bit (BKE bit in the TIM_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIM_BDTR register) to avoid spurious events.

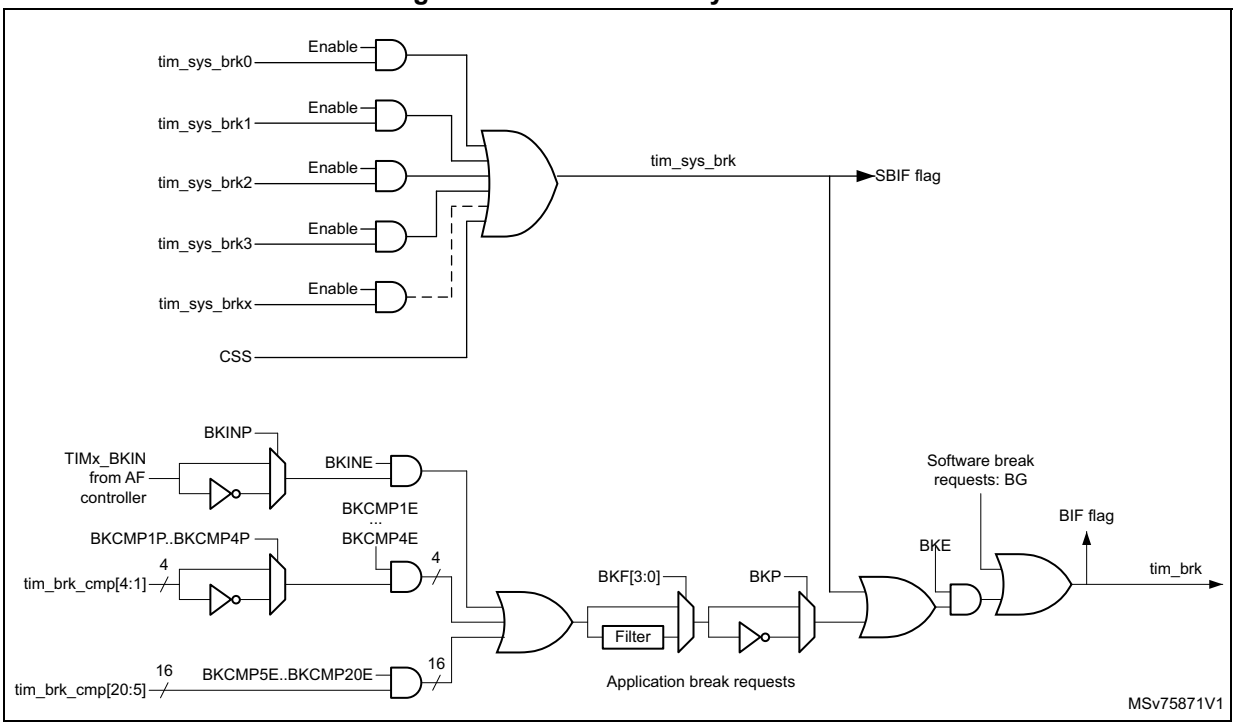
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIM_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIM_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 509* below.

Figure 509. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIM_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE,

the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).

– If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF and BGF bits in the TIM_SR register) is set. An interrupt can be generated if the BIE bit in the TIM_DIER register is set. A DMA request can be sent if the BDE bit in the TIM_DIER register is set.
- If the AOE bit in the TIM_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIM_CR2 register.

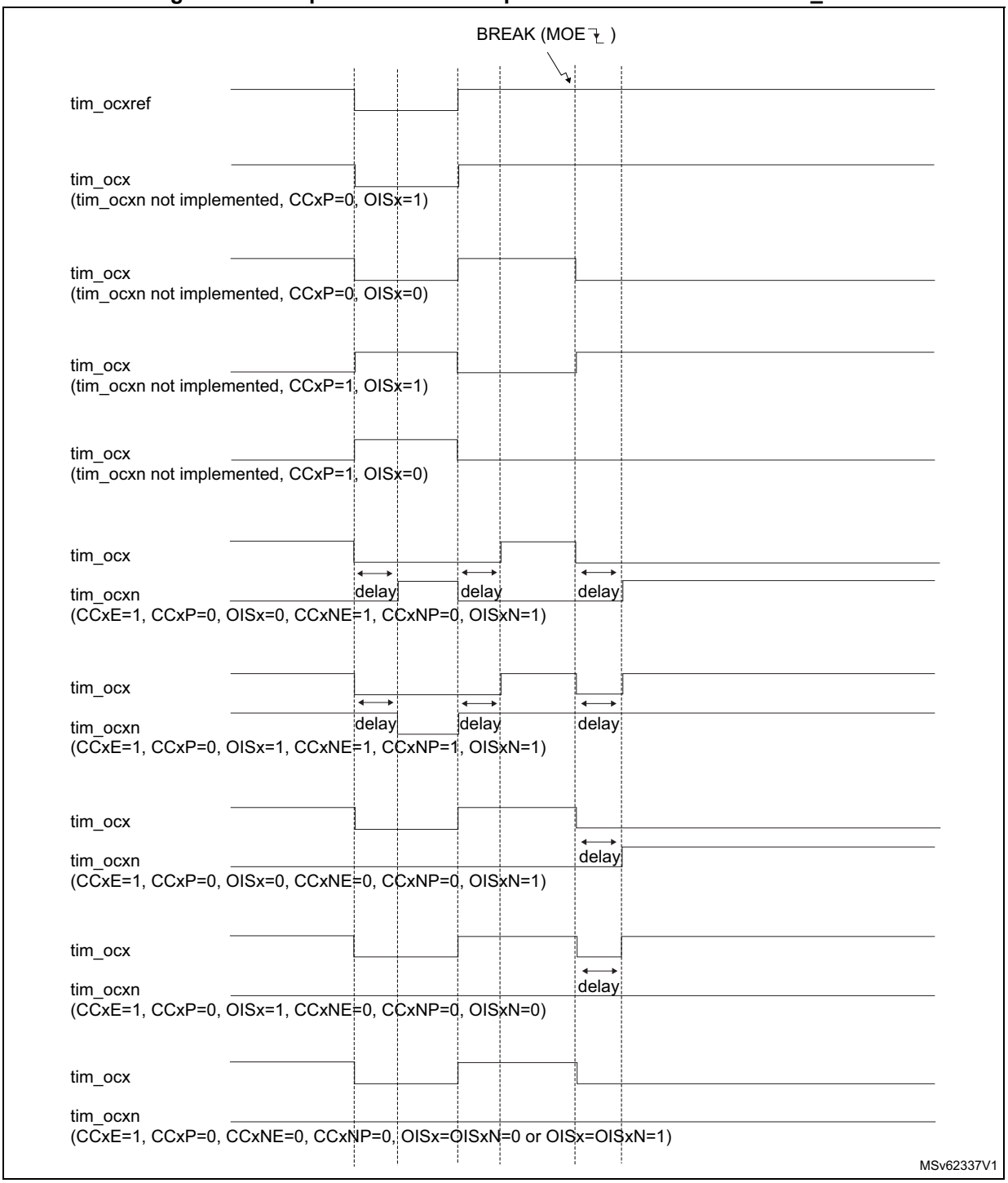
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIM_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIM_BDTR register. Refer to *Section 35.8.16: TIM break and dead-time register (TIM_BDTR)* and *Section 35.9.14: TIM break and dead-time register (TIM_BDTR)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 510* shows an example of behavior for the outputs in response to a break.

Figure 510. Output behavior in response to a break event on tim_brk



35.5.18 Delayed break

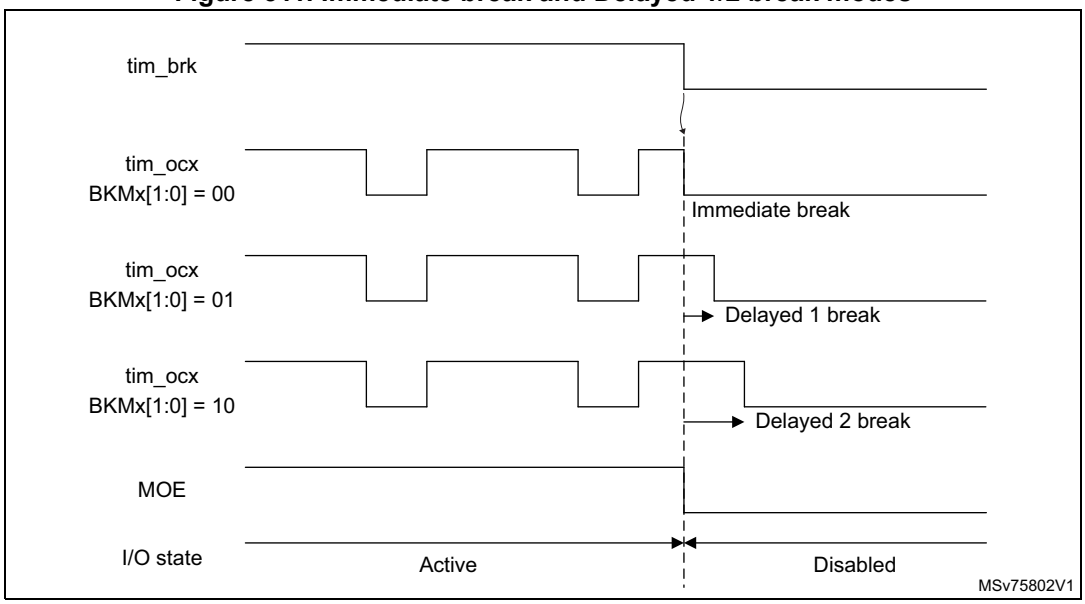
In Neutral-Point Clamped (NPC) multilevel converter topology, a break strategy with immediate outputs shut-down is not suitable, as it may damage the power switches with excessive voltage levels.

The delayed break features provide 2 options delaying the response to a break event, and implement a safe shut-down sequence.

BKMx[1:0] and BKMxN[1:0] bitfields in TIM_MPR1 provides 3 options for each complementary outputs (tim_oc1/tim_oc1n to tim_oc2/tim_oc2n), as show on *Figure 511* below:

- BKM[1:0] = 00: the break response is immediate
- BKM[1:0] = 01: Delayed 1 break. The break response is delayed with a delay defined in the DBK1[7:0] bitfield of the TIM_MPR2 register
- BKM[1:0] = 10: Delayed 2 break. The break response is delayed with a delay defined in the DBK2[7:0] bitfield of the TIM_MPR2 register

Figure 511. Immediate break and Delayed 1/2 break modes



The delayed break mode is triggered when:

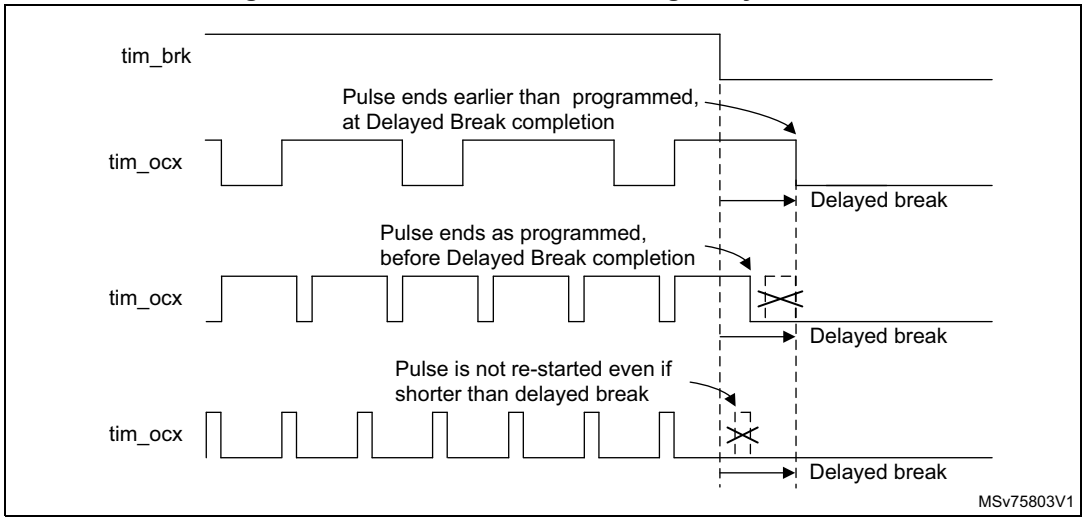
- A break event occurs
- The MOE bit is reset
- When the microcontroller enters debug mode with DBG_TIMx_STOP=1

When delayed break mode is triggered, the MOE bit is immediately cleared, and cannot be set during the delayed break duration. In bidirectional break mode, the break output is reset immediately to indicate the delayed break start to external components. Once the longest of the 2 delays is over, the deadtime generator is enabled to prevent any deadtime violation if OISx/OISxN bits are set.

If the tim_ocx/tim_ocxn signal is reset before the end of the delayed break (due to a compare match in PWM mode, typically), the signal is not maintained during the full delayed break period, to maintain the duty cycle, as shown on *Figure 512* below. Still, the Delayed

break circuitry guarantees that the tim_ocx/tim_ocxn signals cannot be set again once the break event occurred.

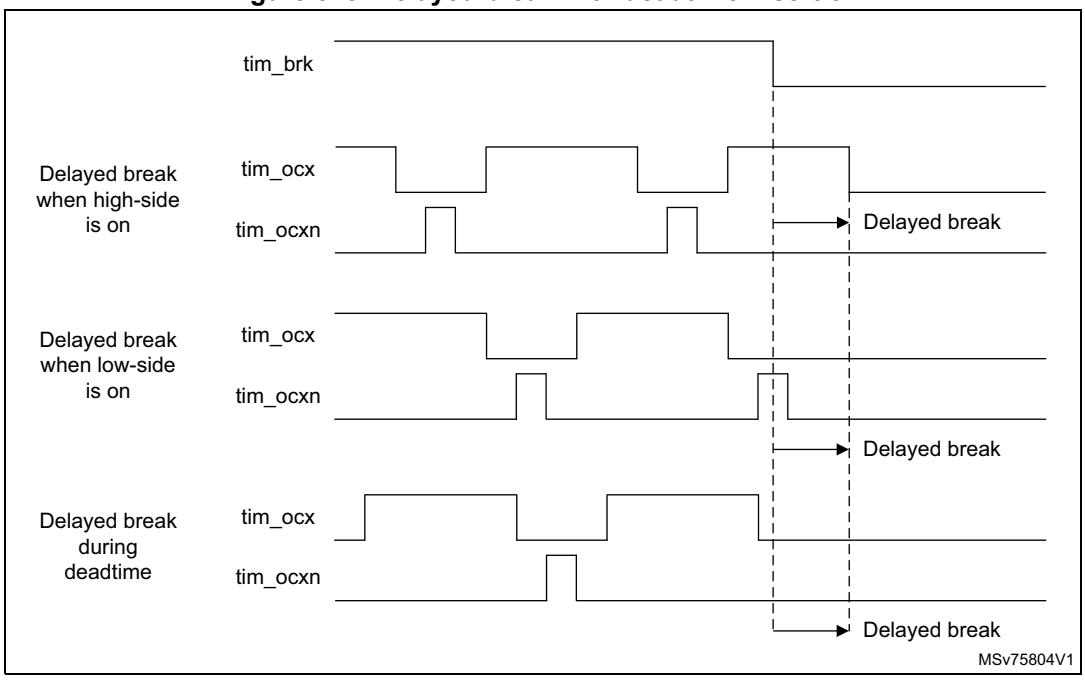
Figure 512. Pulse termination during delayed break



The *Figure 513* depicts the behavior when the deadtime generator is enabled, for the 3 cases:

- Delayed break triggered when tim_ocx is set
- Delayed break triggered when tim_ocxn is set
- Delayed break triggered during deadtime interval

Figure 513. Delayed break with deadtime insertion



The delay programming is identical to deadtime generator's, with 2 user-programmable durations in the TIM_MPR2 register: DBK1[7:0] bitfield for the Delayed 1 break and DBK2[7:0] bitfield for the Delayed 2 break.

DBK1[7:0] and DBK2[7:0] bitfields are preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

The bits coding for t_{DTG} selection are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBKx[6:0] if DBKx[7] = 0
- DBKx[5:0] if DBKx[7:6] = 10
- DBKx[4:0] if DBKx[7:5] = 110 or 111

The BKMx[1:0] and BKMxN[1:0] bits are static, they can only be modified when the counter is stopped (CEN=0 in TIM_CR1).

The content of the TIM_MPR1 and TIM_MPR2 register can be write protected using the LOCK[1:0] bits in the TIM_BDTR register.

35.5.19 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 514*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIM_BDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIM_BDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 356*).

Table 356. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

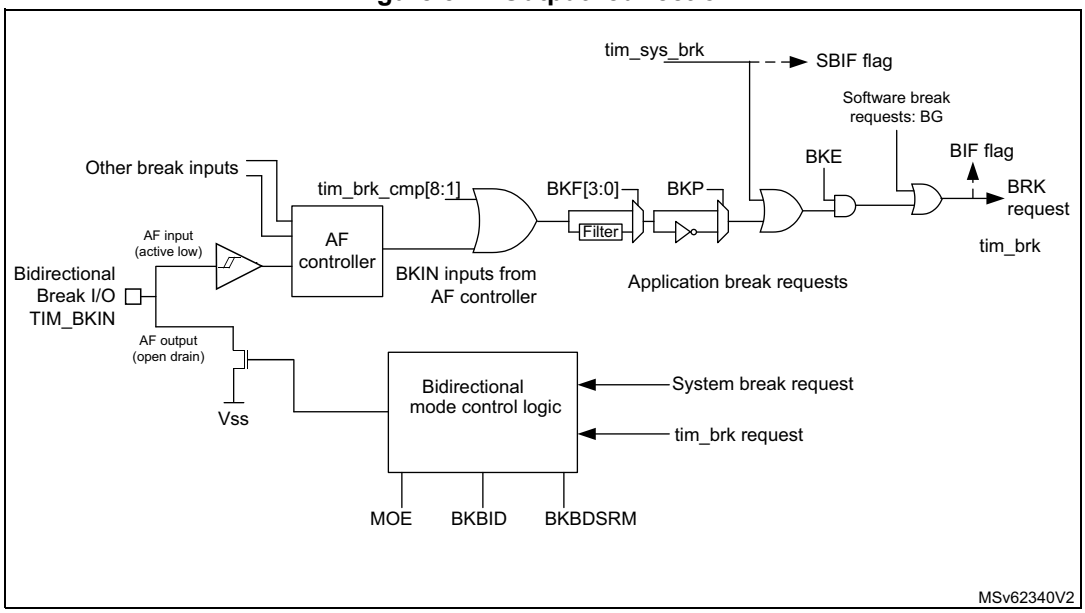
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 514. Output redirection



35.5.20 Output idle state override

It is possible to override the outputs states when they are disabled (ie MOE bit reset in TIM_BDTR). This feature can be used to apply specific patterns prior to start PWM operations, for instance to refresh bootstrap capacitors of power switch gate drivers.

The OOC control bit in the TIM_OOR register acts as following:

- OOC= 0: 0: tim_ocx and tim_ocxn outputs are in the default idle state
- OOC=1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits in the TIM_OOR register

The OOC bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared by hardware when the MOE bit is set, and the deadtime generator ensures there's no deadtime violation on complementary outputs.

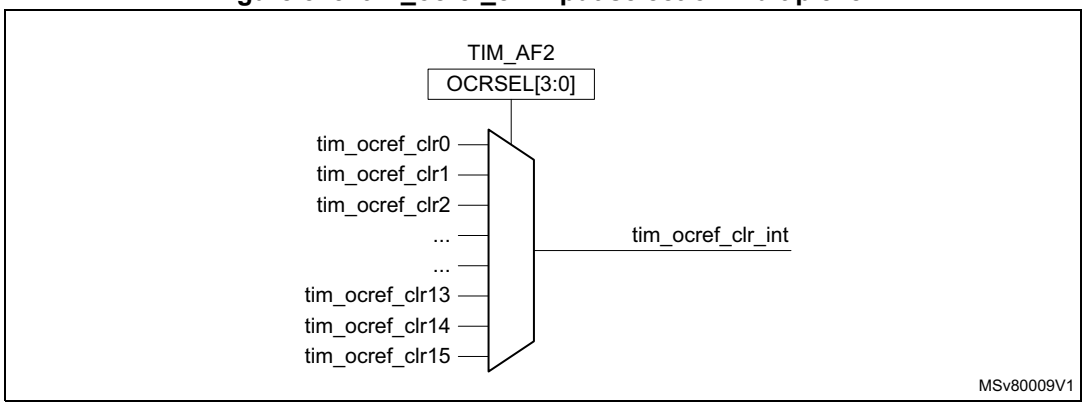
The OOSx/OOSxN bits in the TIM_OOR register cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

35.5.21 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It must not be used in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 515* below.

Figure 515. tim_ocref_clr input selection multiplexer



This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to have the OCxCE bit enabled for the channel, the OCxCE bit of the complementary channel (for example OC2CE = X for oc2ref in *Figure 516*) is ignored in this case.

Figure 516 to *Figure 518* are describing the output waveforms for some use cases, when the ocref_clr_int signal is asserted.

Figure 516. Asymmetric PWM mode 1

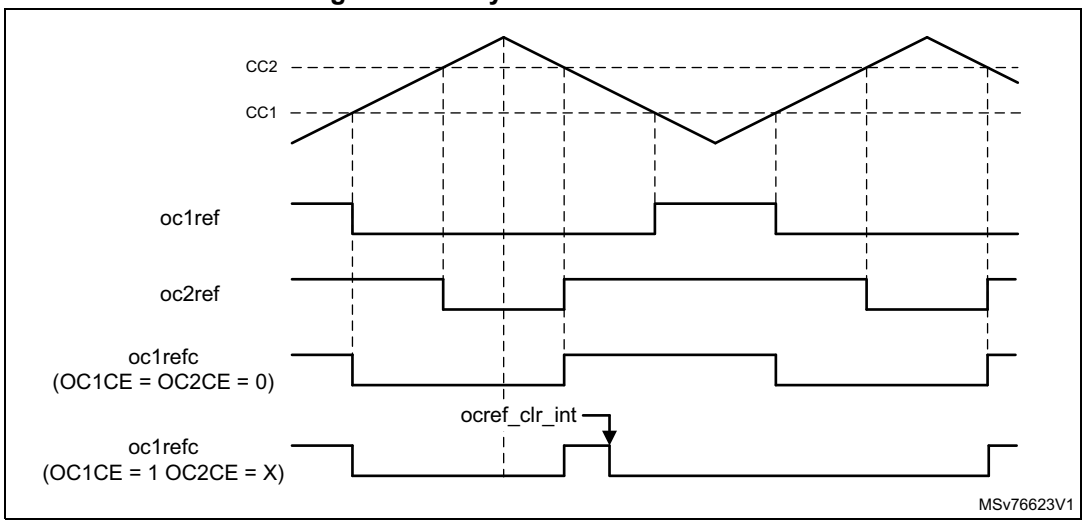


Figure 517. Asymmetric PWM mode 2

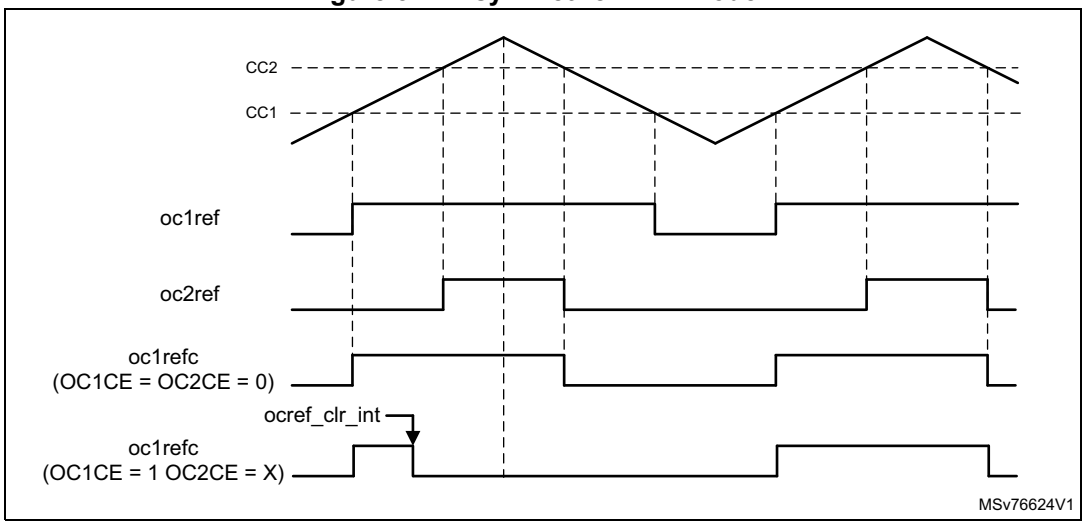
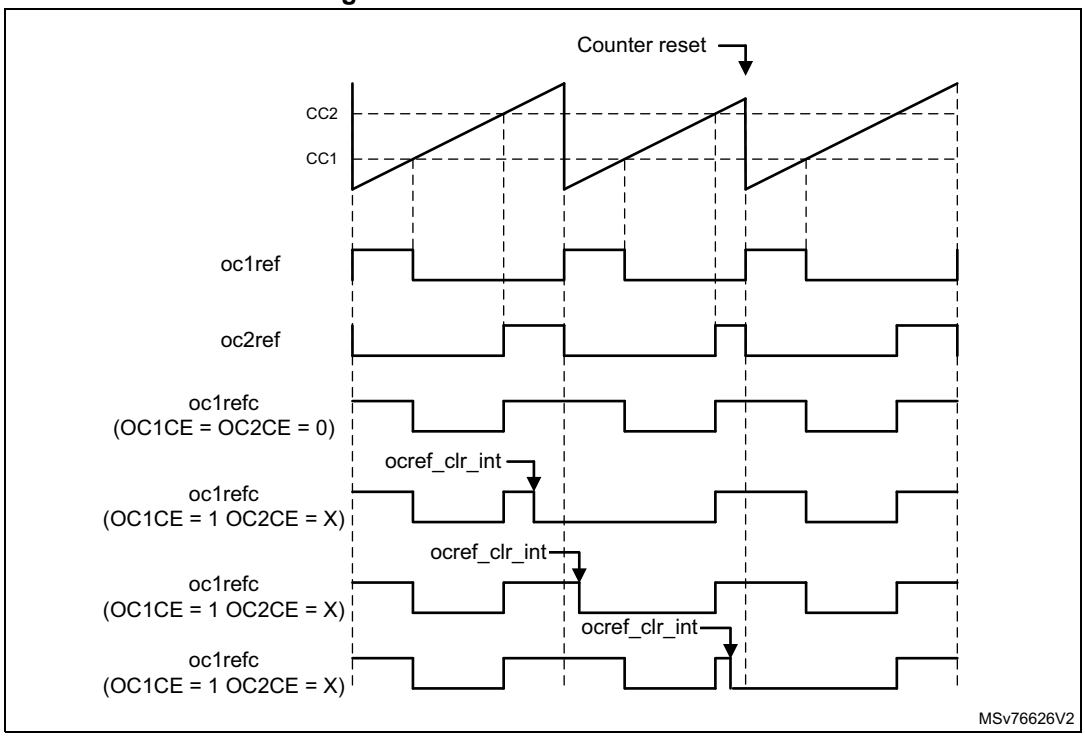


Figure 518. Combined PWM mode 1



Conditions of use

This feature is intended to support cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used for the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One pulse mode with PWM active on match (OPM=1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E)
- Combined PWM mode 1 in center aligned mode (CMS[1:0] = 1,2,3 and OCxM[4:0] = 0x0C)

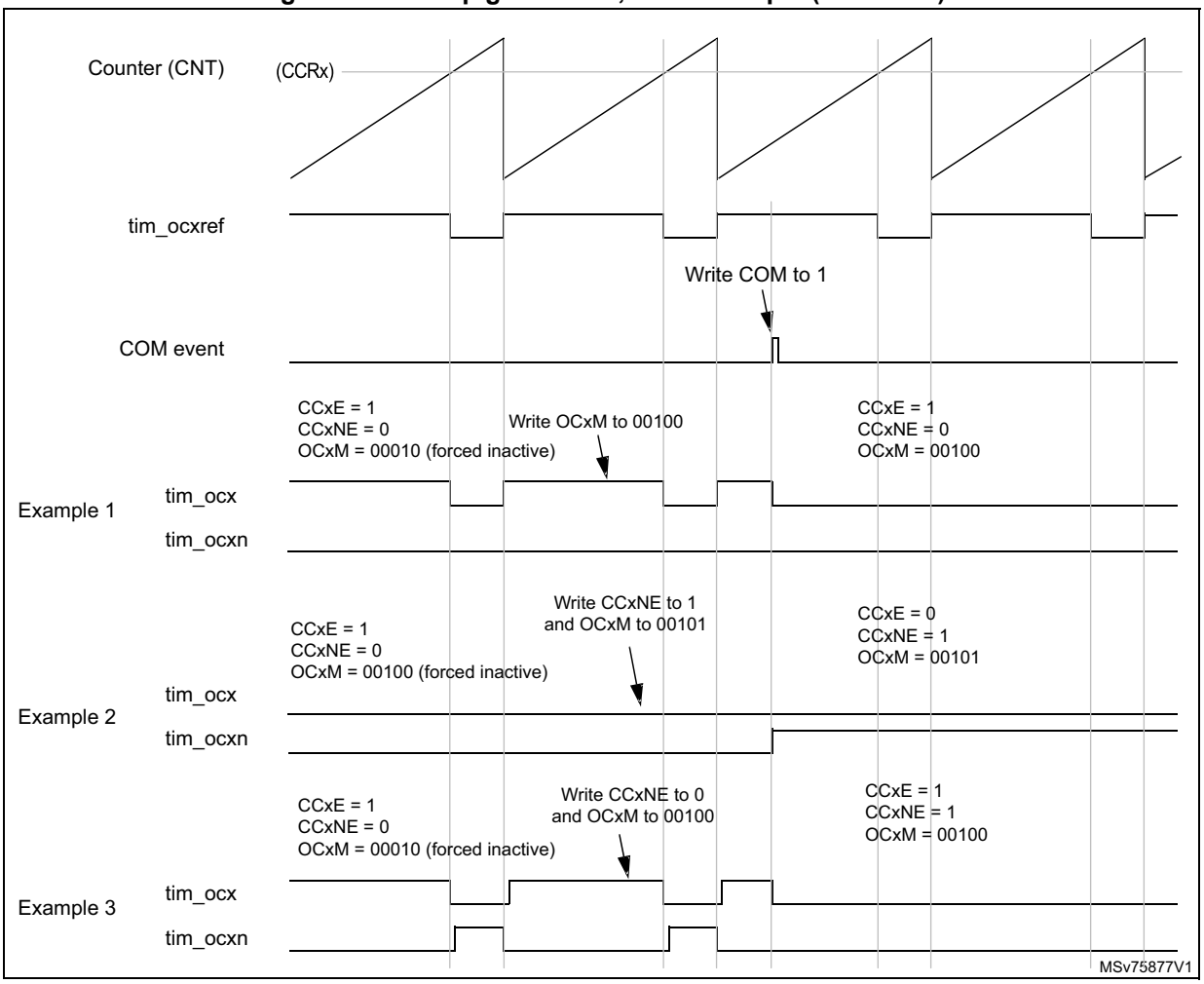
35.5.22 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIM_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIM_SR register), which can generate an interrupt (if the COMIE bit is set in the TIM_DIER register) or a DMA request (if the COMDE bit is set in the TIM_DIER register).

The *Figure 519* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 519. 6-step generation, COM example (OSSR = 1)



35.5.23 One-pulse mode

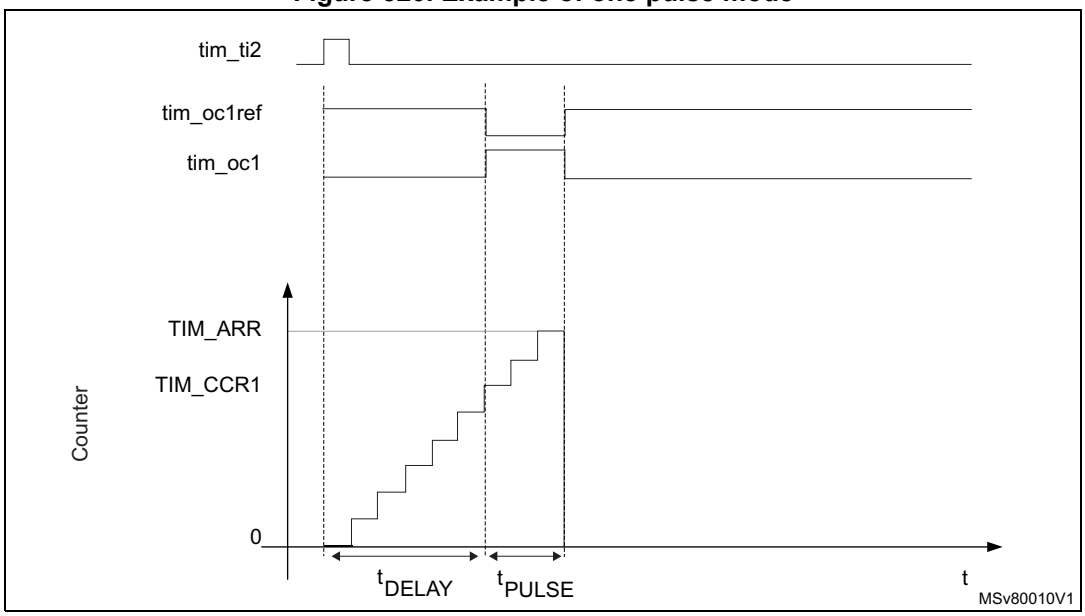
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 520. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 00110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 00111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

35.5.24 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 35.5.23*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

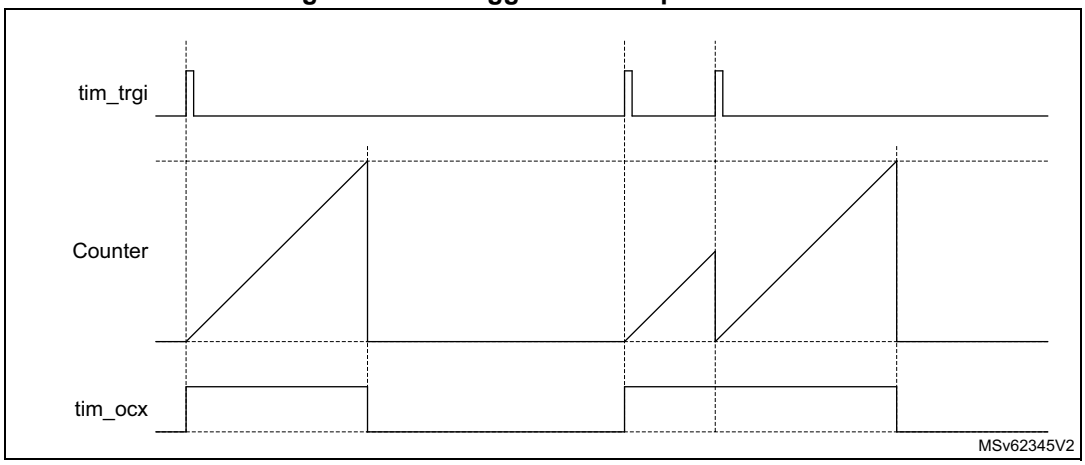
The timer must be in Slave mode, with the bits SMS[4:0] = 01000 (Combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[4:0] and SMS[4:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 521. Retriggerable one pulse mode



35.5.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 00001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 00010 if it is counting on tim_ti2 edges only and SMS = 00011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 357*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 357. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 522 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIM_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 00011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, Counter enabled).

Figure 522. Example of counter operation in encoder interface mode.

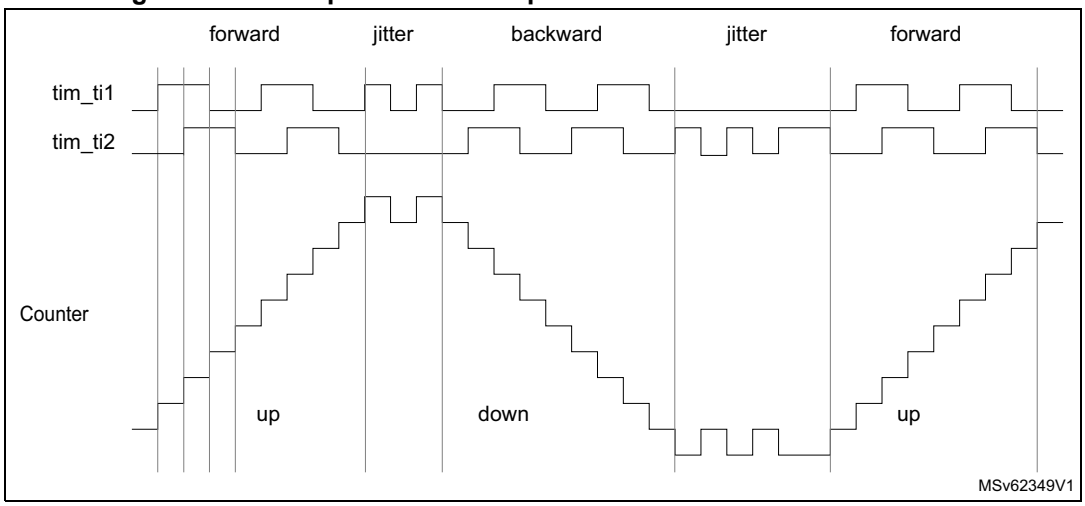


Figure 523 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 523. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

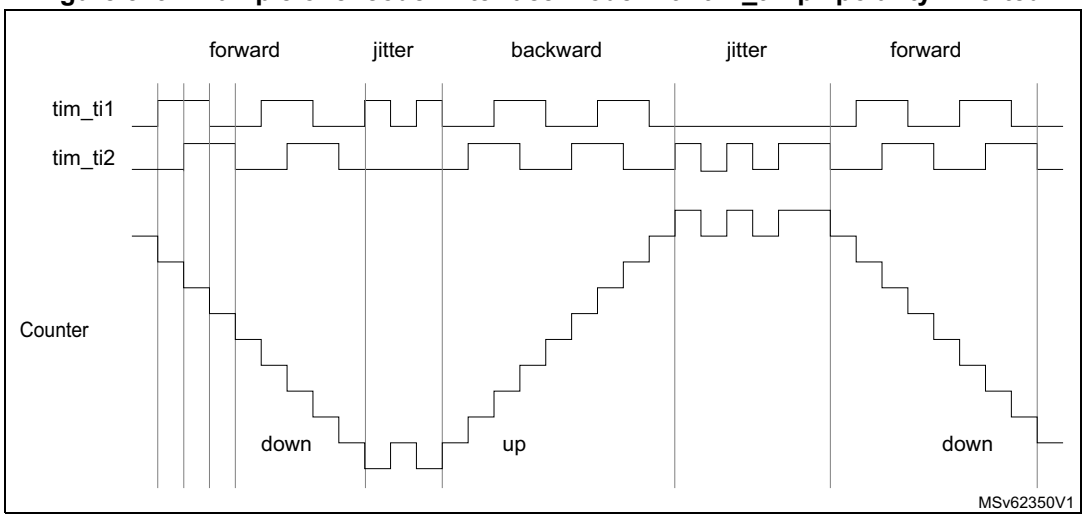
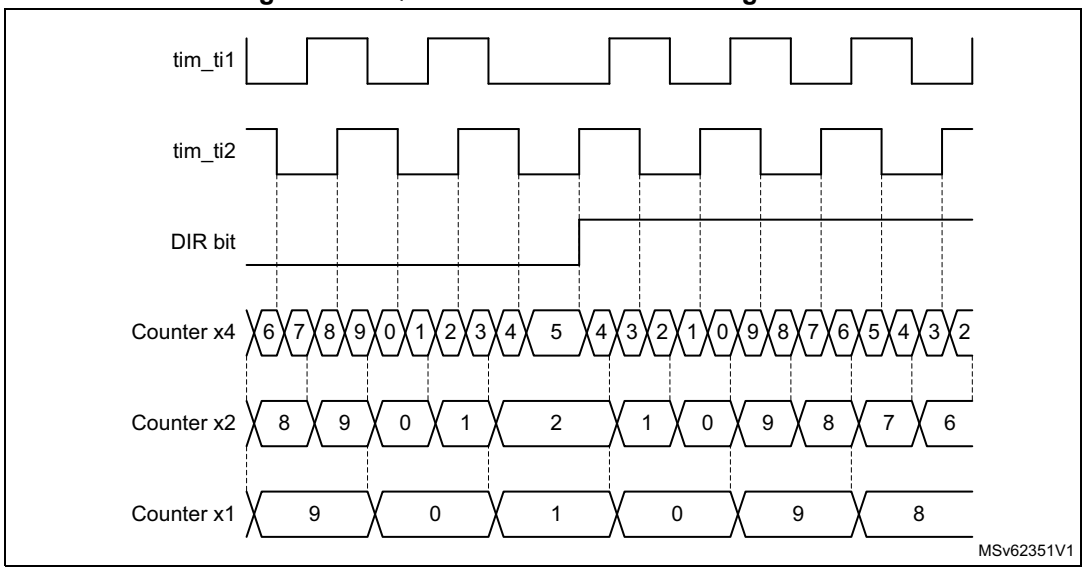


Figure 524 shows the timer counter value during a speed reversal, for various counting modes.

Figure 524. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

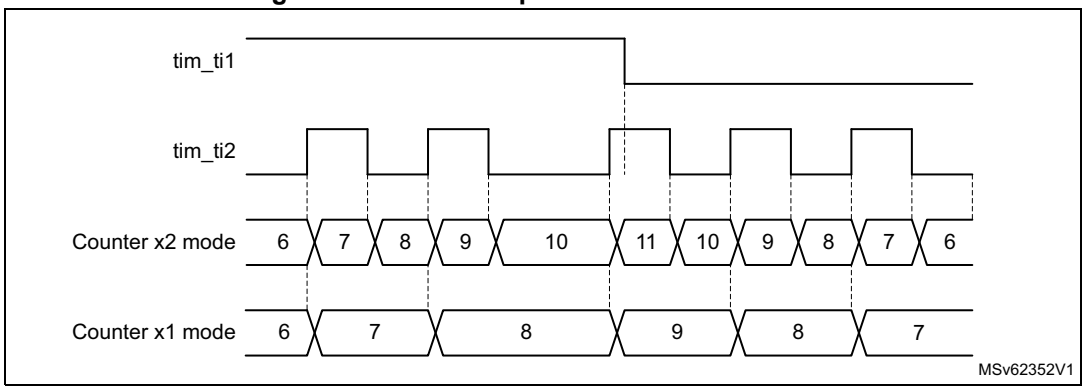
In the clock plus direction mode shown on *Figure 525*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 525. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 526*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 526*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 527*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 526*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 527*).

Figure 526. Directional clock encoder mode (CC1P = CC2P = 0)

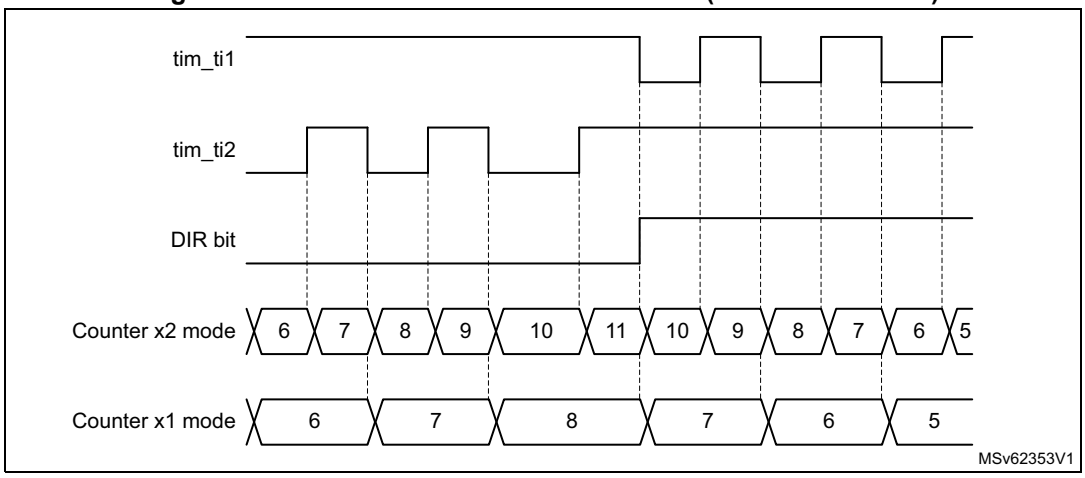


Figure 527. Directional clock encoder mode (CC1P = CC2P = 1)

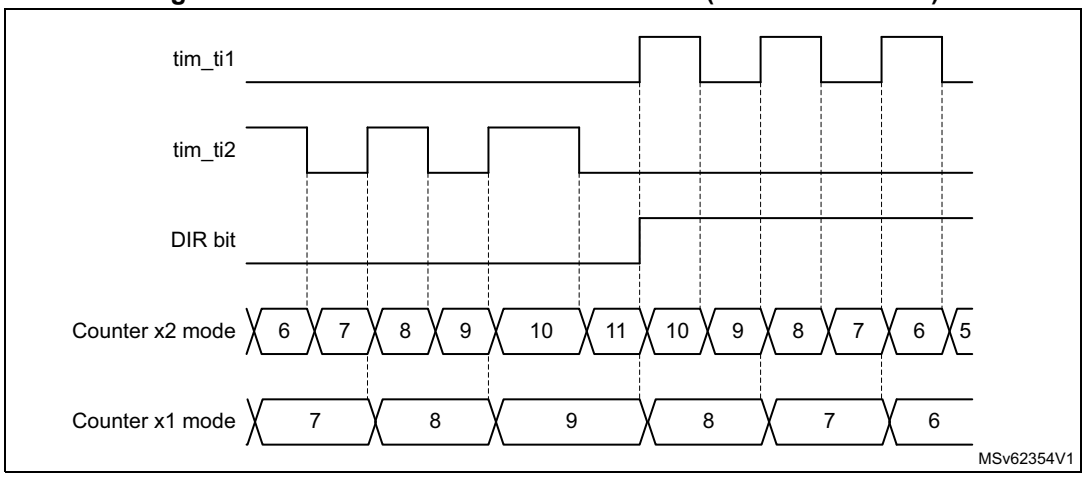


Table 358 here-below details how the directional clock mode operates, for any input transition.

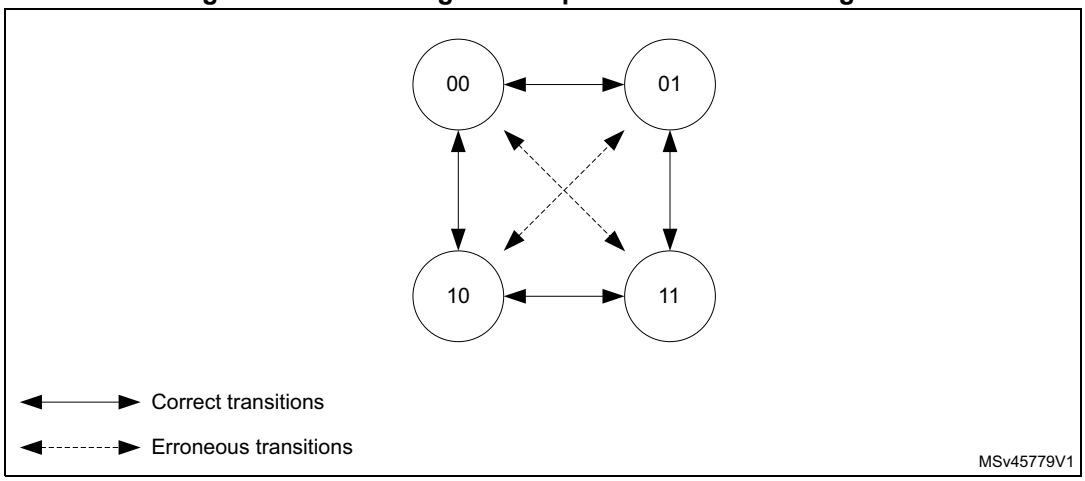
Table 358. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 528.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 528. State diagram for quadrature encoded signals



Functional encoder interrupts

The following interrupt is also available in encoder mode

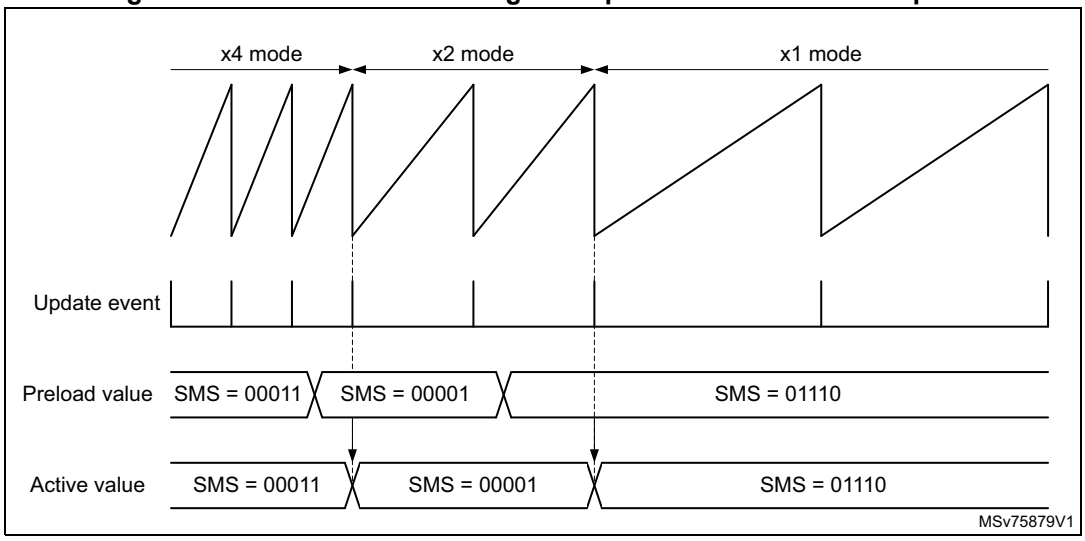
- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.

Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 529*.

For this purpose, the SMS[4:0] bit can be preloaded. The transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.

Figure 529. Encoder mode change with preload transferred on update



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register.It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 10101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

35.5.26 Encoder with built-in debouncer (TIM15 only)

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 530* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 531* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between 2 positions.

These 2 modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 530. Encoder with built-in debouncer, x2 mode

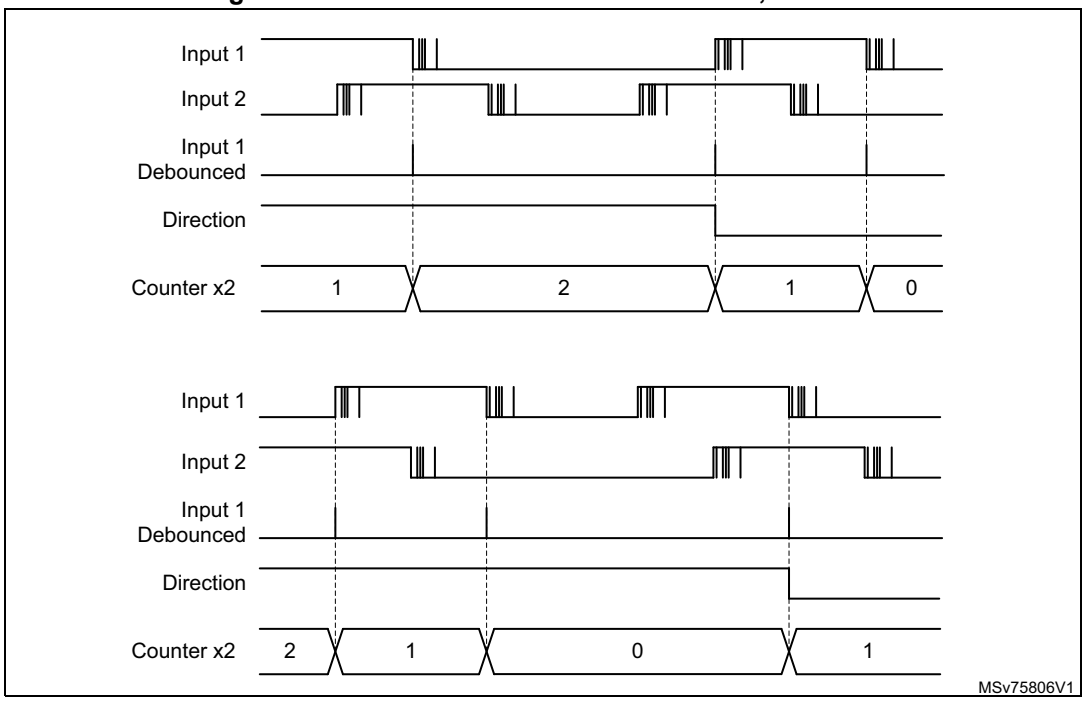
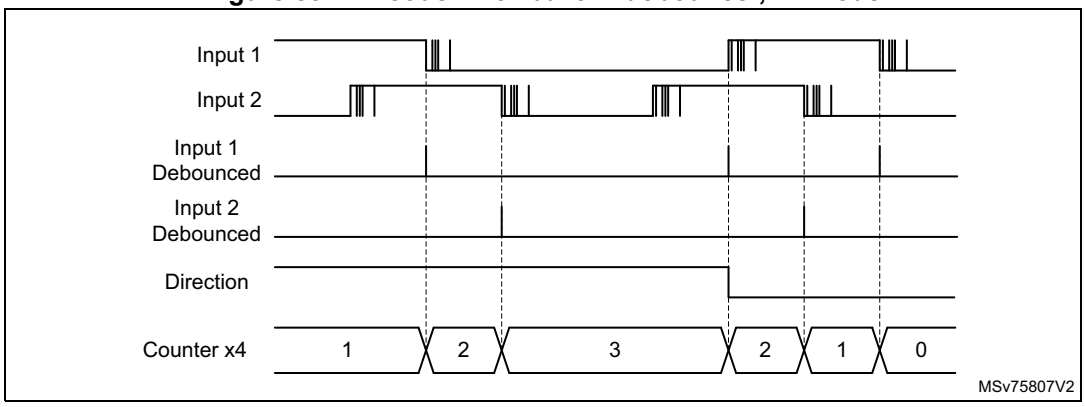


Figure 531. Encoder with built-in debouncer, x4 mode



35.5.27 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

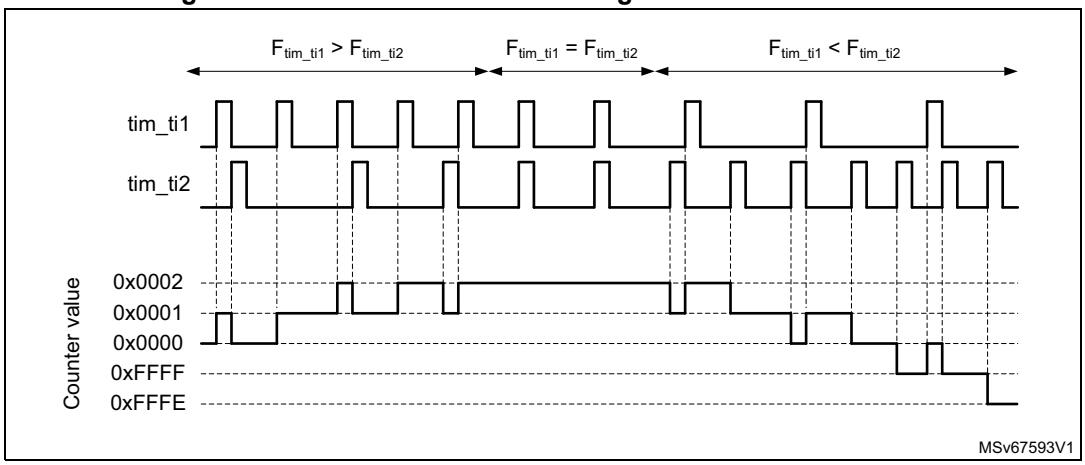
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 532*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value changes as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value dithers between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occuring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 532* below).

Figure 532. Clock drift measure using directional clock mode



35.5.28 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

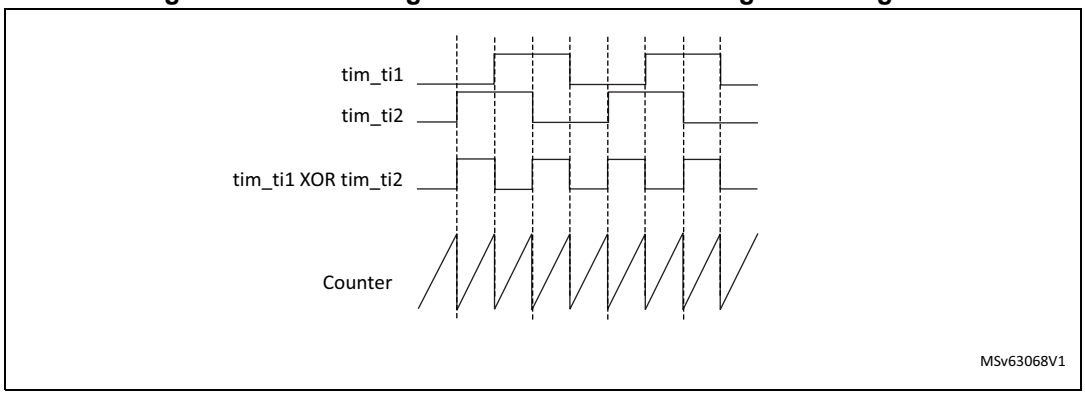
There is no latency between the assertions of the UIF and UIFCPY flags.

35.5.29 Timer input XOR function (TIM15 only)

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 533*.

Figure 533. Measuring time interval between edges on 2 signals



The XORPS control bit selects the position of the XOR gate, as shown in the *Figure 534* and *Figure 535* below, when the TI1S bit is set.

Figure 534. XOR gate connection with TI1S = 1 and XORPS = 0

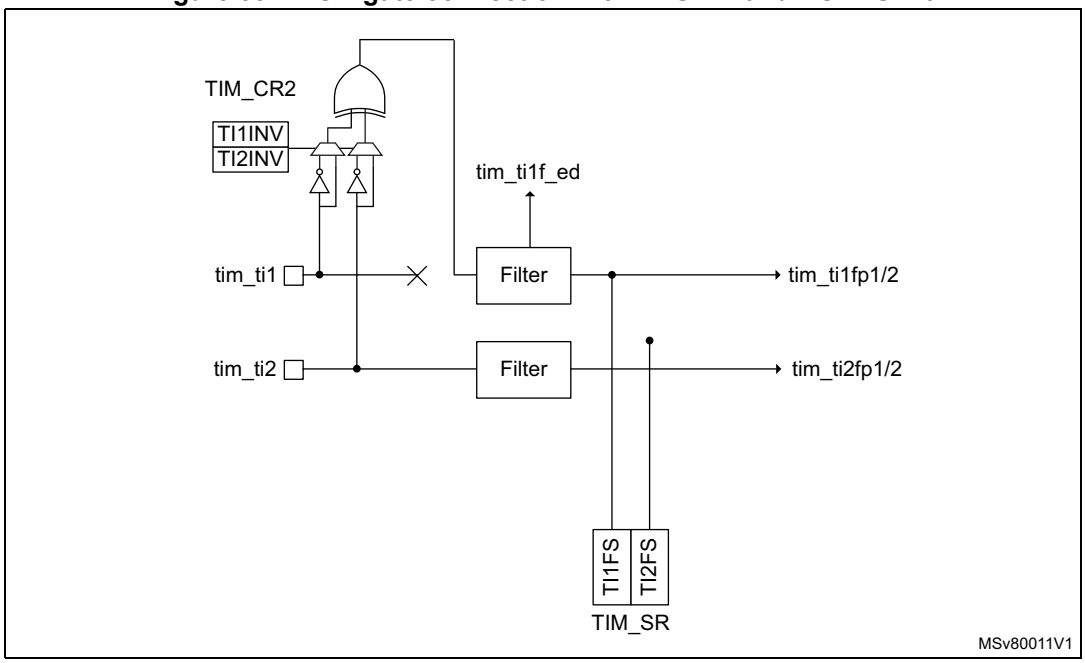
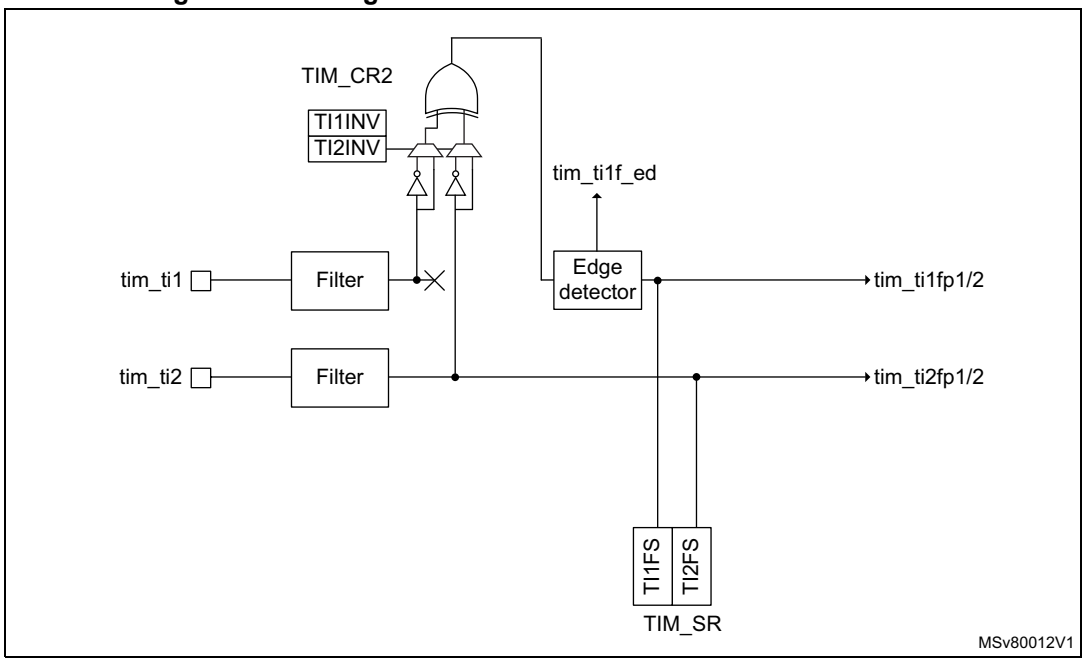


Figure 535. XOR gate connection with TI1S = 1 and XORPS = 1



Two control bits TI1INV and TI2INV in the TIM_CR2 register can invert the tim_in1 and tim_in2 signals prior entering the XOR gate.

The status of the tim_in1 and tim_in2 signals can be software polled using the TI1FS and TI2FS bits in the TIM_SR register.

35.5.30 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

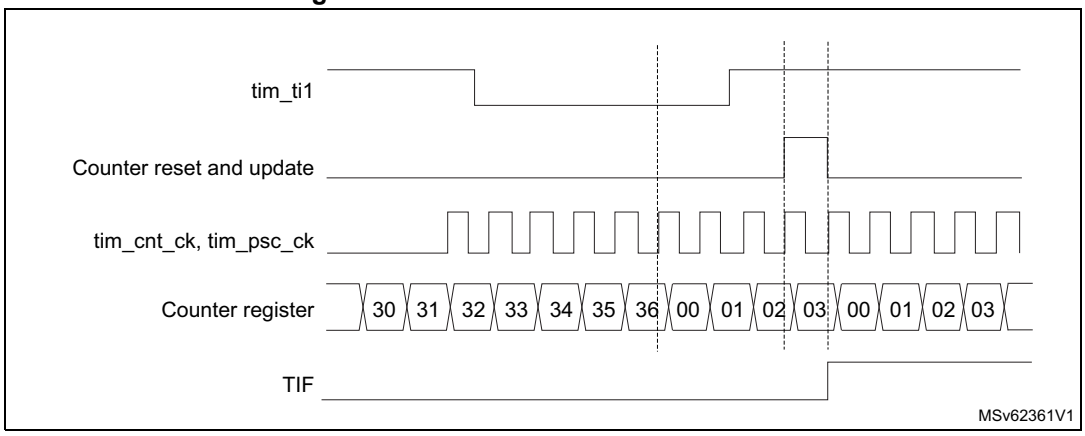
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 00100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 536. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

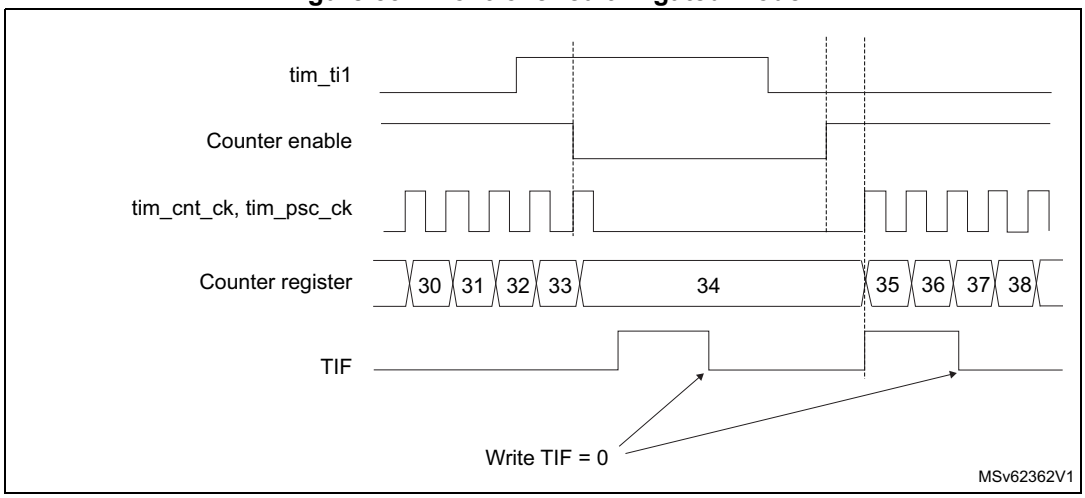
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 00101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 537. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

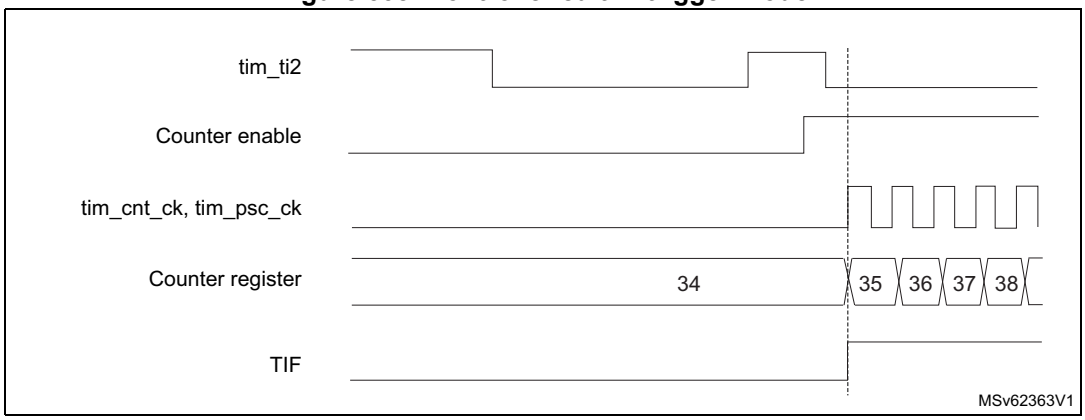
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 00110 in the TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 538. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value is the update event (UEV) occurring when the counter overflows.

35.5.31 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

35.5.32 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

35.5.33 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 32.4.27: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.5.34 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

35.5.35 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.5.36 ADC synchronization (TIM15 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semi-synchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the 2 requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CCRx} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

If none of the 2 above requirement are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

35.5.37 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIM_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 35.8.23: TIM output override register (TIM_OOR)* and *Section 35.9.21: TIM output override register (TIM_OOR)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 2, DBA = 0xE and DBSS[3:0] = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

35.5.38 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in *Table 359*.

Table 359. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

35.5.39 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

35.6 TIM15/TIM16/TIM17 low-power modes

Table 360. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

35.7 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 361*.

Table 361. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
TIM_TRG_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
	System Break	SBIF		write 0 in SBIF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

1. Available for TIM15 only.

35.8 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

35.8.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1).

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

35.8.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw		rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input

0: tim_ti2 non-inverted

1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input

0: tim_ti1 non-inverted

1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization
0: The timer operates independently from the ADC
1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 35.5.36* for requirements.
The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection
This bit defines the position of the XOR gate with respect to input filters.
0: XOR gate placed before TI1 filter
1: XOR gate placed after TI1 and TI2 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)
0: tim_oc2 = 0 when MOE = 0
1: tim_oc2 = 1 when MOE = 0
Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 after a dead-time when MOE = 0
1: tim_oc1 = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Others: reserved.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

35.8.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
							rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE:** SMS preload enable

This bit selects whether the SMS[4:0] bitfield is preloaded.

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)
Others: Reserved

See *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 17:16, 2:0 **SMS[4:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010:Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011:Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111:External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010:Encoder mode: Clock plus direction, x2 mode.

01011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

01100:Encoder mode: Directional Clock, x2 mode.

01101:Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000:Quadrature encoder with built-in debouncer: x2 mode.

10001:Quadrature encoder with built-in debouncer: x4 mode.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.8.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.
								rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 Reserved, must be kept at reset value.

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

35.8.5 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	Res.	DIRF	Res.	Res.	Res.	Res.	Res.
		r	r				rc_w0	rc_w0		rc_w0					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BGF	SBIF	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	rc_w0	rc_w0			rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 Reserved, must be kept at reset value.

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **BGF**: Break generation flag
This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break event occurred.
1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 35.8.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

35.8.6 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1 A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

35.8.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

35.8.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable
Refer to OC1CE description on bit 7.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode
Refer to OC1M description on bits 6:4.

Bit 11 **OC2PE**: Output compare 2 preload enable
Refer to OC1PE description on bit 3.

Bit 10 **OC2FE**: Output compare 2 fast enable
Refer to OC1FE description on bit 2.

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.
10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input.
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.
Note: This mode is not available for all PWM modes. Refer to Section 35.5.21: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - Channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive.

00111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active.

01000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Reserved

01011: Reserved

01100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

01101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

01110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

01111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

35.8.9 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 362* for details.

Table 362. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0 tim_ocxn = 0	
		1	0	0	Off-State (output enabled with inactive state) tim_ocx=CCxP, tim_ocxn=CCxNP	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP Then if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

35.8.10 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM_SR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

35.8.11 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

35.8.12 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 35.5.3: Time-base unit on page 1341* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

35.8.13 TIM repetition counter register (TIM_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

35.8.14 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

35.8.15 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

35.8.16 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 35.8.9: TIM capture/compare enable register (TIM_CCER) on page 1429*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1: Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 35.8.9: TIM capture/compare enable register (TIM_CCER) on page 1429*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 35.8.9: TIM capture/compare enable register (TIM_CCER) on page 1429*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. tim_ocx/tim_ocxn enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

35.8.17 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

35.8.18 TIM input selection register (TIM_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIM15_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

1111: tim_ti2_in15

Refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIM15_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for interconnects list.

35.8.19 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for product specific implementation.

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable

This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp20 input disabled

1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable

This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp13 input disabled

1: tim_brk_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable

This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp12 input disabled

1: tim_brk_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable

This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp11 input disabled

1: tim_brk_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable

This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp10 input disabled

1: tim_brk_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable

This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp9 input disabled

1: tim_brk_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

35.8.20 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

0000: tim_ocref_clr0

0001: tim_ocref_clr1

...

1111: tim_ocref_clr15

Refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

35.8.21 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **BK2M[1:0]**: Break mode for tim_oc2 output

This bit must be modified only when CEN=0 in TIM_CR1.

00: Immediate break

01: Delayed 1 break

10: Delayed 2 break

11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output

This bit must be modified only when CEN=0 in TIM_CR1.

00: Immediate break

01: Delayed 1 break

10: Delayed 2 break

11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output

This bit must be modified only when CEN=0 in TIM_CR1.

00: Immediate break

01: Delayed 1 break

10: Delayed 2 break

11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

35.8.22 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for Tdtg selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK2[6:0] if DBK2[7] = 0
- DBK2[5:0] if DBK2[7:6] = 10
- DBK2[4:0] if DBK2[7:5] = 110 or 111

DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for Tdtg selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK1[6:0] if DBK1[7] = 0
- DBK1[5:0] if DBK1[7:6] = 10
- DBK1[4:0] if DBK1[7:5] = 110 or 111

DBK1[7:5]=0xx => Delay1=DBK1[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK1[7:5]=10x => Delay1=(64+DBK1[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK1[7:5]=110 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK1[7:5]=111 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

35.8.23 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS2	OOS1N	OOS1
			rw										rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **OOS2**: Output override state for output tim_oc2

This bit defines the state for output tim_oc2 when OOC = 1

0: tim_oc2 = 0

1: tim_oc2 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **OOS1N**: Output override state for output tim_oc1n

This bit defines the state for output tim_oc1n when OOC = 1

0: tim_oc1n = 0

1: tim_oc1n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1

This bit defines the state for output tim_oc1 when OOC = 1

0: tim_oc1 = 0

1: tim_oc1 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

35.8.24 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).
0000: Reserved
0001: Update
0010: CC1
0011: CC2
0110: COM
0111: Trigger
Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length
This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).
00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address
This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.
Example:
00000: TIM_CR1
00001: TIM_CR2
00010: TIM_SMCR
...

35.8.25 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

35.8.26 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 363. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0													0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM_CR2**	Res.	Res.	Res.	ADSYNC	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value				0			0															0	0	0	0	0	0	0	0	0		0
0x08	**TIM_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value								0			0	0			0	0									0	0	0	0		0	0	0
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	DIRIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value										0	0							0	0			0	0	0	0	0	0			0	0	0
0x10	**TIM_SR**	Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	Res.	DIRF	Res.	Res.	Res.	Res.	Res.	Res.	BGF	SBIF	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value			0	0				0	0		0							0	0			0	0		0	0	0			0	0	0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 363. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	**TIM_AF1**	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 363. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x64	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x68 - 0x08C	Reserved	Res.																															
0x090	**TIM_MPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
	Reset value																											0	0	0	0	0	0
0x094	**TIM_MPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x98 - 0x09C	Reserved	Res.																															
0x0A0	**TIM_OOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS2	OOS1N	OOS1
	Reset value																				0										0	0	0
0xA4 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

35.9 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

35.9.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1).

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

35.9.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

35.9.3 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

35.9.4 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BGF	SBIF	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	rc_w0	rc_w0				rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.

0: tim_ti1f signal is low

1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:15 Reserved, must be kept at reset value.

Bit 14 **BGF**: Break generation flag

This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.

0: No break event occurred.

1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag

This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.

This flag must be reset to re-start PWM operation.

0: No break event occurred.

1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

35.9.5 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

35.9.6 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

35.9.7 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:18 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 17:16, 6:4 **OC1M[4:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - Channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive.

00111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

35.9.8 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 364* for details.

Table 364. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0 tim_oc1n = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n = 0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1 = CC1P	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n = CC1NP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

35.9.9 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in TIM_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

35.9.10 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

35.9.11 TIM auto-reautoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 35.5.3: Time-base unit on page 1341* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

35.9.12 TIM repetition counter register (TIM_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

35.9.13 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

35.9.14 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIM_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 35.9.8: TIM capture/compare enable register (TIM_CCER) on page 1461*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 35.9.8: TIM capture/compare enable register (TIM_CCER) on page 1461*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 35.9.8: TIM capture/compare enable register (TIM_CCER) on page 1461*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$

Example if t_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

35.9.15 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

35.9.16 TIM input selection register (TIM_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for interconnects list.

35.9.17 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for product specific implementation.

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable
This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp20 input disabled
1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable

This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp13 input disabled

1: tim_brk_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable

This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp12 input disabled

1: tim_brk_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable

This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp11 input disabled

1: tim_brk_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable

This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp10 input disabled

1: tim_brk_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable

This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp9 input disabled

1: tim_brk_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

35.9.18 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: tim_ocref_clr source selection
These bits select the tim_ocref_clr input source.
0000: tim_ocref_clr0
0001: tim_ocref_clr1
...
1111: tim_ocref_clr15
Refer to *Section 35.5.2: TIM15/TIM16/TIM17 pins and internalsignals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

35.9.19 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK1NM[1:0]		BK1M[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

35.9.20 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.
It is preloaded and can be updated during timer operation (new value taken into account after an update event).
DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.
It is preloaded and can be updated during timer operation (new value taken into account after an update event).
DBK2[7:5]=0xx => Delay1=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay1=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay1=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay1=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

35.9.21 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS1N	OOS1
			rw											rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **OOS1N**: Output override state for output tim_oc1n

This bit defines the state for output tim_oc1n when OOC = 1

0: tim_oc1n = 0

1: tim_oc1n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1

This bit defines the state for output tim_oc1 when OOC = 1

0: tim_oc1 = 0

1: tim_oc1 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

35.9.22 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.
Example:
00000: TIM_CR1,
00001: TIM_CR2,
00010: TIM_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIM_CR1. In this case the transfer is done to/from 7 registers starting from the TIM_CR1 address.

35.9.23 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

35.9.24 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 365. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0													0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value											0												0	0	0		0				0	0
0x10	**TIM_SR**	Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BGF	SBIF	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value				0				0										0	0				0		0		0				0	0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value															0	0									0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 365. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	**TIM_AF1**	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x6C - 0x08C	Reserved	Res.																															
0x090	**TIM_MPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK1NM[1:0]		BK1M[1:0]	
	Reset value																													0	0	0	0
0x094	**TIM1x_MPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x98 - 0x09C	Reserved	Res.																															

Table 365. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A0	**TIM_OOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS1N	OOS1
	Reset value																				0											0	0
0xA4 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2* for the register boundary addresses.

36 Low-power timer (LPTIM)

36.1 LPTIM introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

36.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

36.3 LPTIM implementation

Table 366 describes LPTIM implementation on STM32C5 devices.

Table 366. STM32C5 LPTIM features

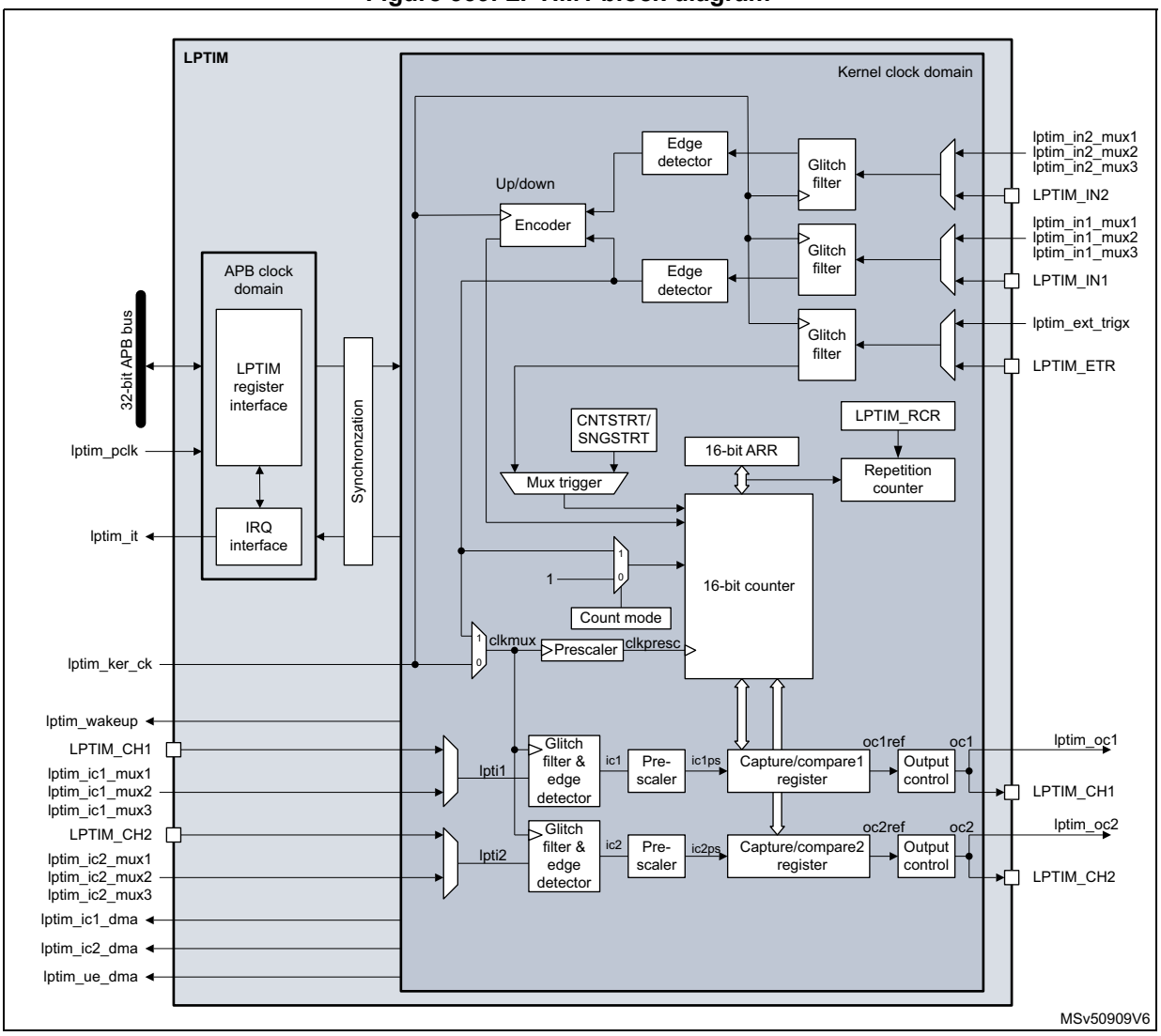
LPTIM modes/features[1]	LPTIM1
Encoder mode	X
PWM mode	X
Input Capture	X
Number of channels	2
Number of DMA requests	3
Wake-up in Stop mode	X

1. X = supported.

36.4 LPTIM functional description

36.4.1 LPTIM block diagram

Figure 539. LPTIM1 block diagram(1)



1. Some I/Os may not be available, refer to *Section 36.4.2: LPTIM pins and internal signals*.

36.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 367. LPTIM1 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0

Table 367. LPTIM1 input/output pins (continued)

Pin name	Pin type	Description
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

Table 368. LPTIM1 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

36.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 369. LPTIM1 external trigger connections

TRIGSEL	External trigger
	LPTIM1
lptim_ext_trig0	GPIO
lptim_ext_trig1	rtc_alra_trg
lptim_ext_trig2	rtc_alrb_trg
lptim_ext_trig3	tamp_trg1
lptim_ext_trig4	tamp_trg2
lptim_ext_trig5	lpdma1_ch1_tc
lptim_ext_trig6	comp1_out
lptim_ext_trig7	EVENTOUT

Table 370. LPTIM1 input 1 connections

lptim_in1_mux	LPTIM1 input 1 connected to
lptim_in1_mux0	GPIO
lptim_in1_mux1	comp1_out

Table 371. LPTIM1 input 2 connections

lptim_in2_mux	LPTIM1 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	comp2_out
lptim_in2_mux2	Reserved
lptim_in2_mux3	Reserved

Table 372. LPTIM1 input capture 1 connections

lptim_ic1_mux	LPTIM1 input capture 1 connected to
lptim_ic1_mux0	GPIO
lptim_ic1_mux1	comp1_out
lptim_ic1_mux2	EVENTOUT
lptim_ic1_mux3	RCC_MCO1

Table 373. LPTIM1 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	LSI
lptim_ic2_mux2	LSE
lptim_ic2_mux3	RCC_HSE_1MHZ

36.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

36.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

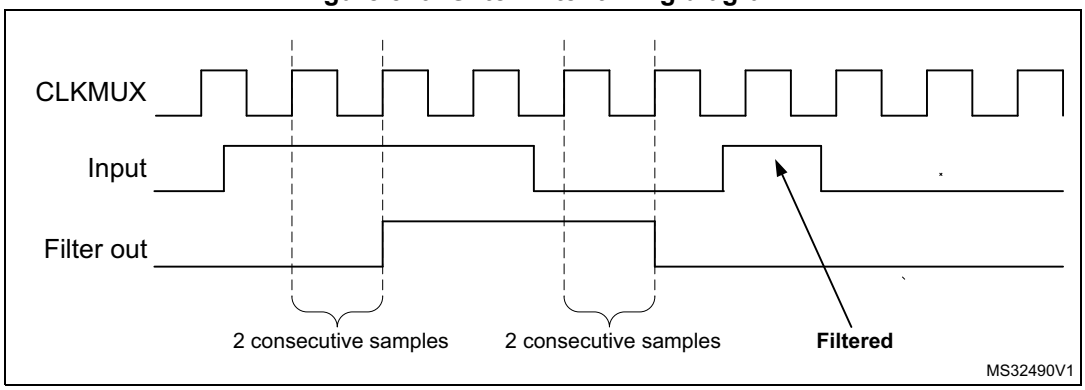
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 540* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 540. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

36.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 374. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

36.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible

values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.

- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

36.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

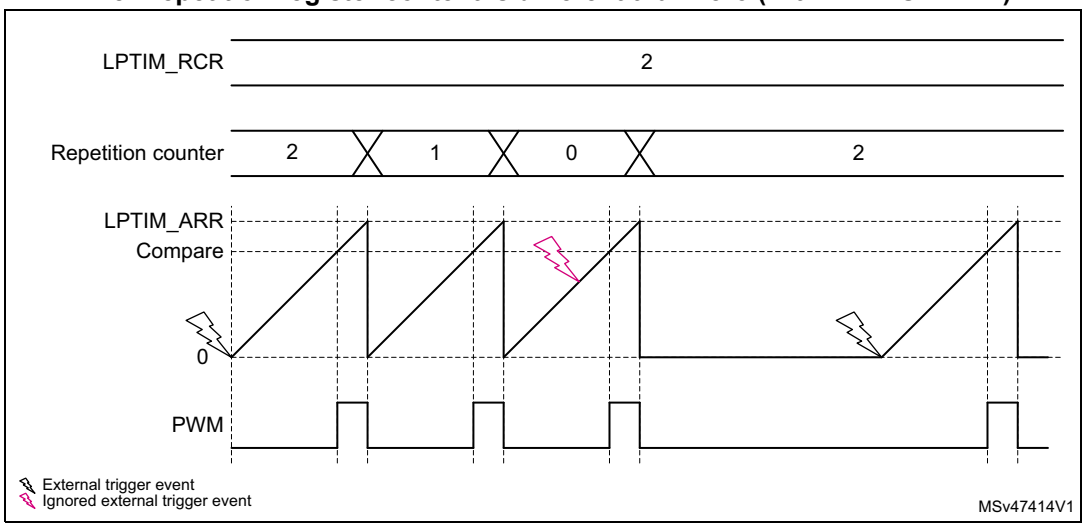
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 541*.

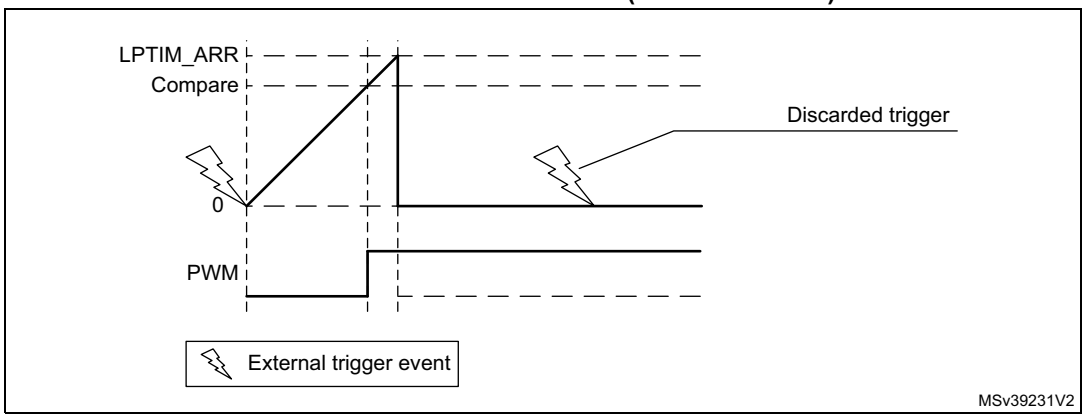
Figure 541. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 542*.

Figure 542. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

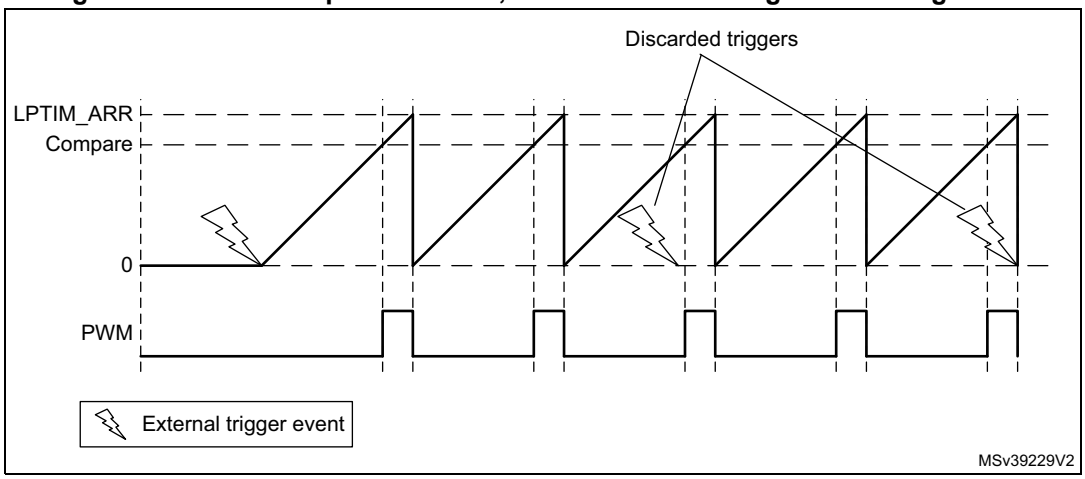
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 543*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 543. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

36.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

36.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 36.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

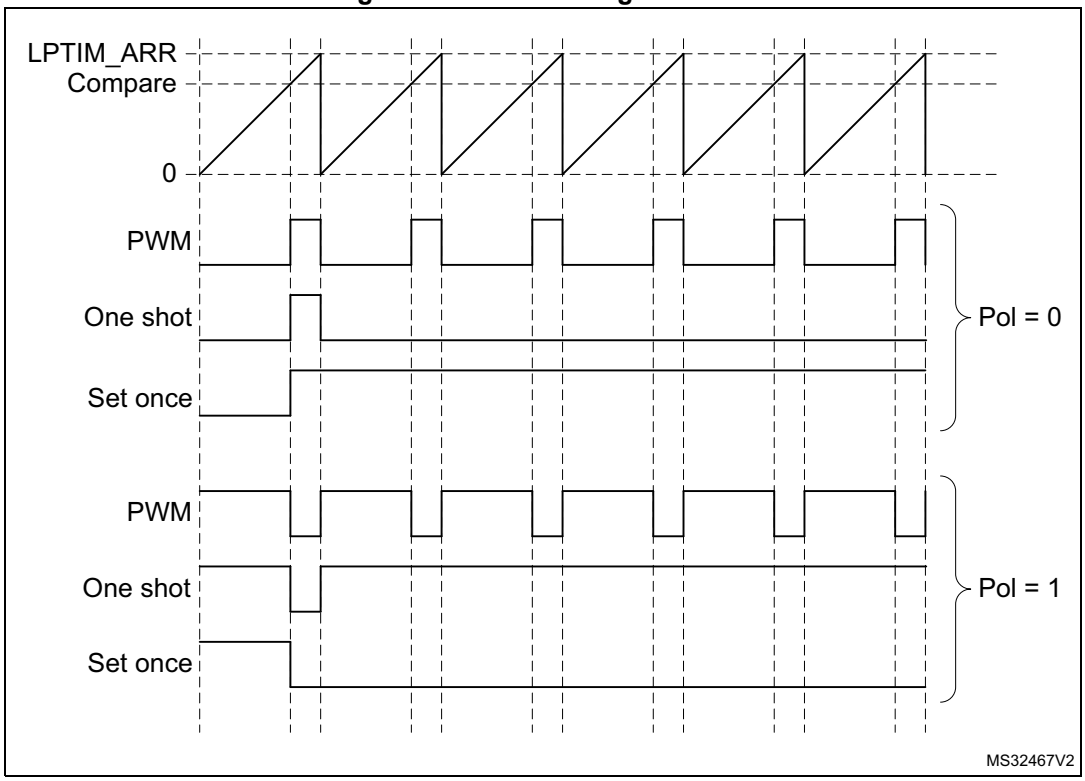
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 544* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the CCxP bit.

Figure 544. Waveform generation



36.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

36.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as

system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

36.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

36.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software ensures that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

36.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

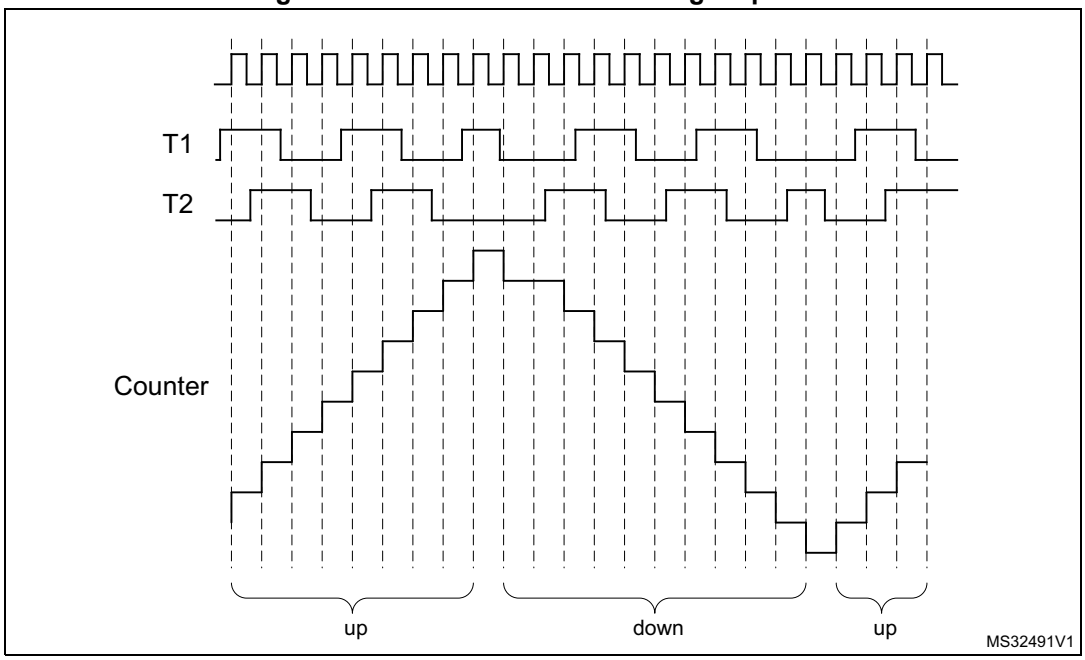
Table 375. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 545. Encoder mode counting sequence



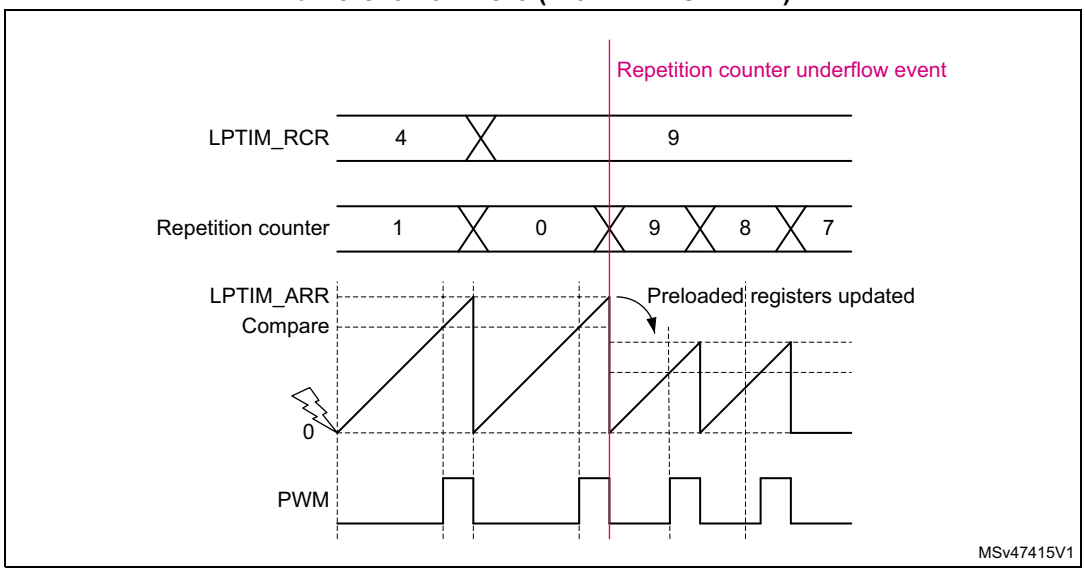
36.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 546*.

Figure 546. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 36.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

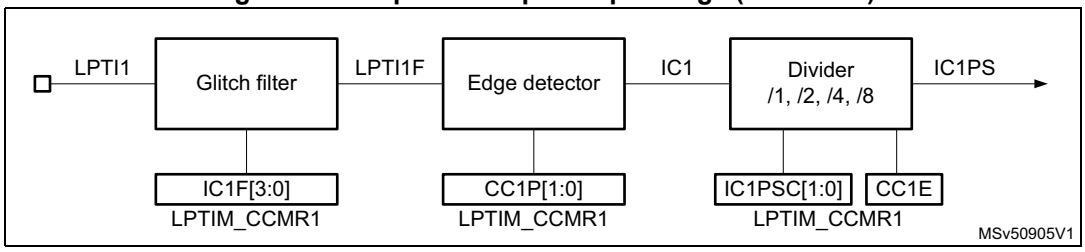
36.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

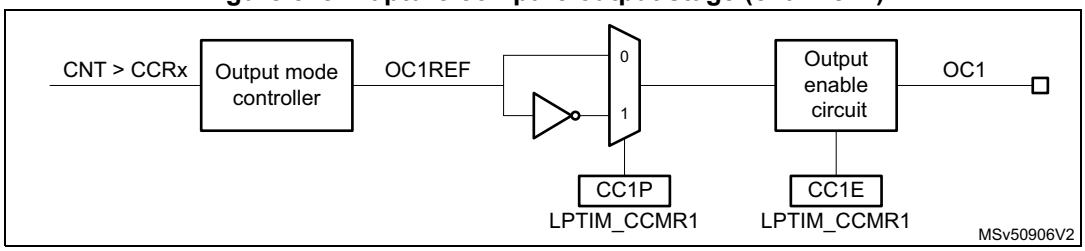
Figure 547. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 548. Capture/compare output stage (channel 1)



36.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 376*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 376*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 376. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 376. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

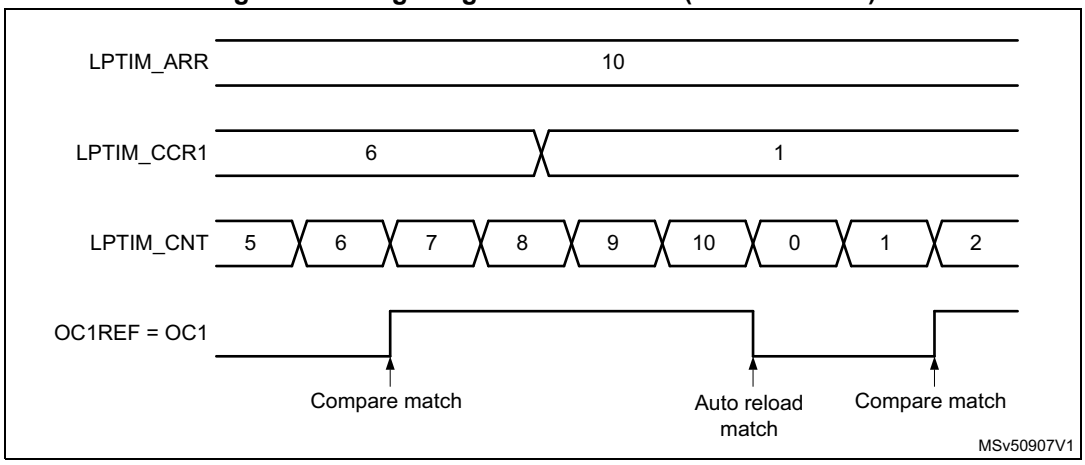
36.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 549 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

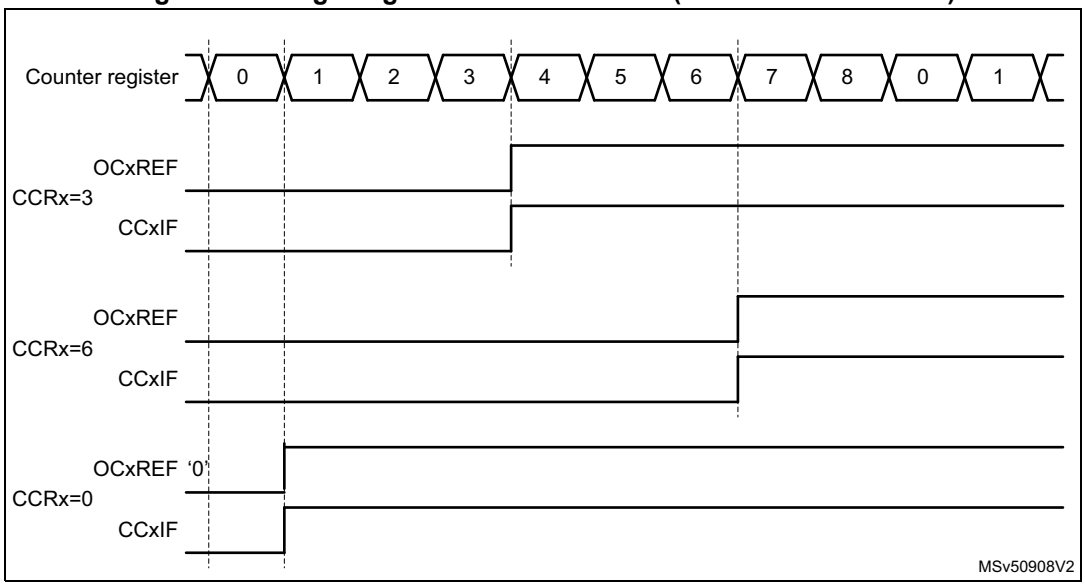
Figure 549. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 550 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 550. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

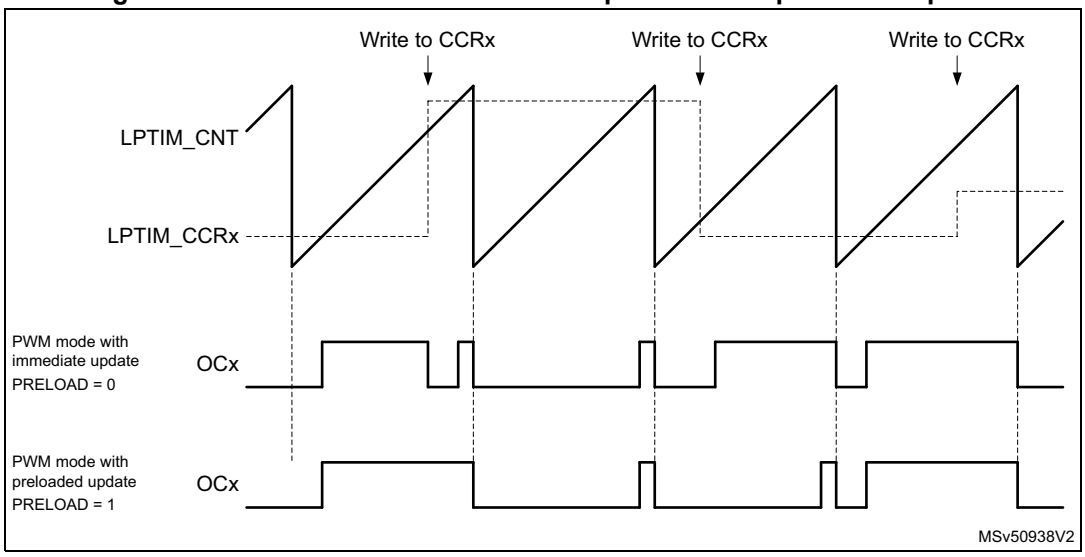
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 551* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 551. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 549 illustrates this behavior.

36.4.20 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

36.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

36.5 LPTIM low-power modes

Table 377. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

Note: All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

36.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 378. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(2)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(2)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 36.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 36.3: LPTIM implementation*.

36.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 85* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

36.7.1 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

36.7.2 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OF	CC1OF	Res.	Res.	CC2IF	REPOK	UE	DOWN	UP	ARROK	Res.	EXTTRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 36.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

36.7.3 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag

Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

36.7.4 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OCF	CC1 OCF	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARRO KCF	Res.	EXTTR IGCF	ARRM CF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 36.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

36.7.5 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 36.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

36.7.6 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 36.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 36.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 36.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 36.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

36.7.7 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	Res.	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw		rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 36.3.

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 Reserved, must be kept at reset value.

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started is ignored

1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: Software trigger (counting start is initiated by software)

01: Rising edge is the active edge

10: Falling edge is the active edge

11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0

001: lptim_ext_trig1

010: lptim_ext_trig2

011: lptim_ext_trig3

100: lptim_ext_trig4

101: lptim_ext_trig5

110: lptim_ext_trig6

111: lptim_ext_trig7

See *Section 36.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1

001: /2

010: /4

011: /8

100: /16

101: /32

110: /64

111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger

01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.

10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.

11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition

01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 36.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

36.7.8 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.

This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

36.7.9 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

36.7.10 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value

ARR is the autoreload value for the LPTIM.

This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

36.7.11 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value

When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

36.7.12 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 36.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 36.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection
The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 36.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection
The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 36.4.3: LPTIM input and trigger mapping*.

36.7.13 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

36.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3) kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

36.7.15 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 36.3: LPTIM implementation*.

36.7.16 LPTIM register map

The following table summarizes the LPTIM registers.

Table 379. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM1_ISR Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIM1_ISR Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF(1)	CC1OF	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIM1_ICR Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIM1_ICR Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF(1)	CC1OCF	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIM1_DIER Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIM1_DIER Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE(1)	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE(1)	CC1OIE	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0

Table 379. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	Res.	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0		0	0	0	0		0	0	0		0	0	0		0	0		0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	LPTIM_ CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(3)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 36.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 36.3: LPTIM implementation*.
3. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 36.3: LPTIM implementation*.

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

37 Independent watchdog (IWDG)

37.1 IWDG introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 37.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

37.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

37.3 IWDG implementation

Table 380. IWDG features (1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate an interrupt in system Stop	X
Capability to generate an interrupt in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the activity in Stop mode(4)	X
Option bytes to control the activity in Standby mode(5)	X
Option bytes to control the Hardware mode(6)	X

1. ‘X’ = supported, ‘-’ = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.

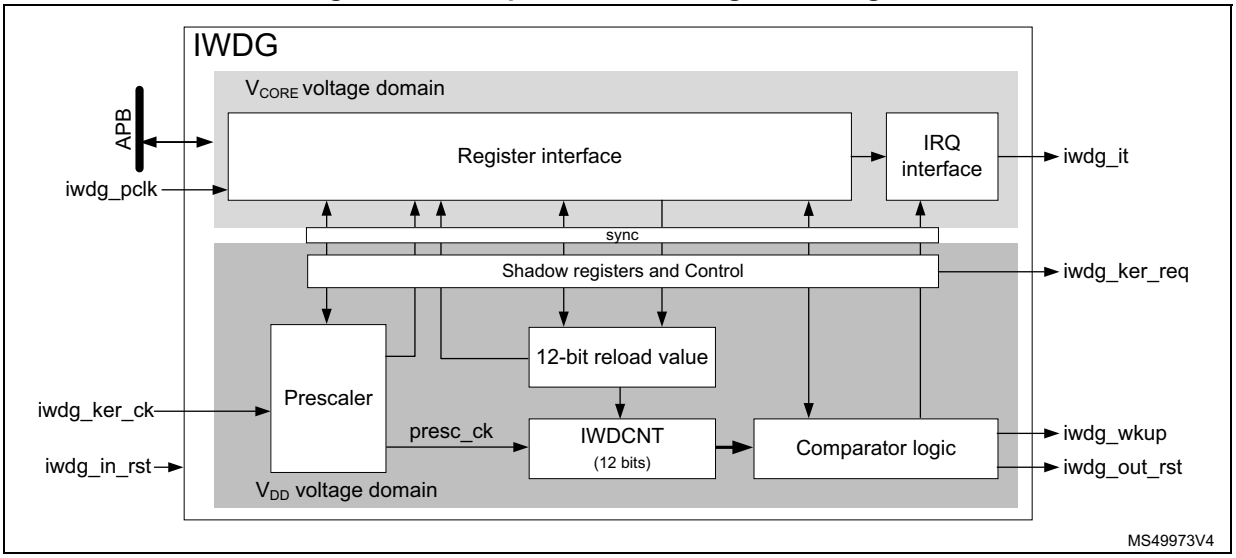
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

37.4 IWDG functional description

37.4.1 IWDG block diagram

Figure 552 shows the functional blocks of the independent watchdog module.

Figure 552. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 37.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 381. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

37.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 382*.

Table 382. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

37.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 37.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

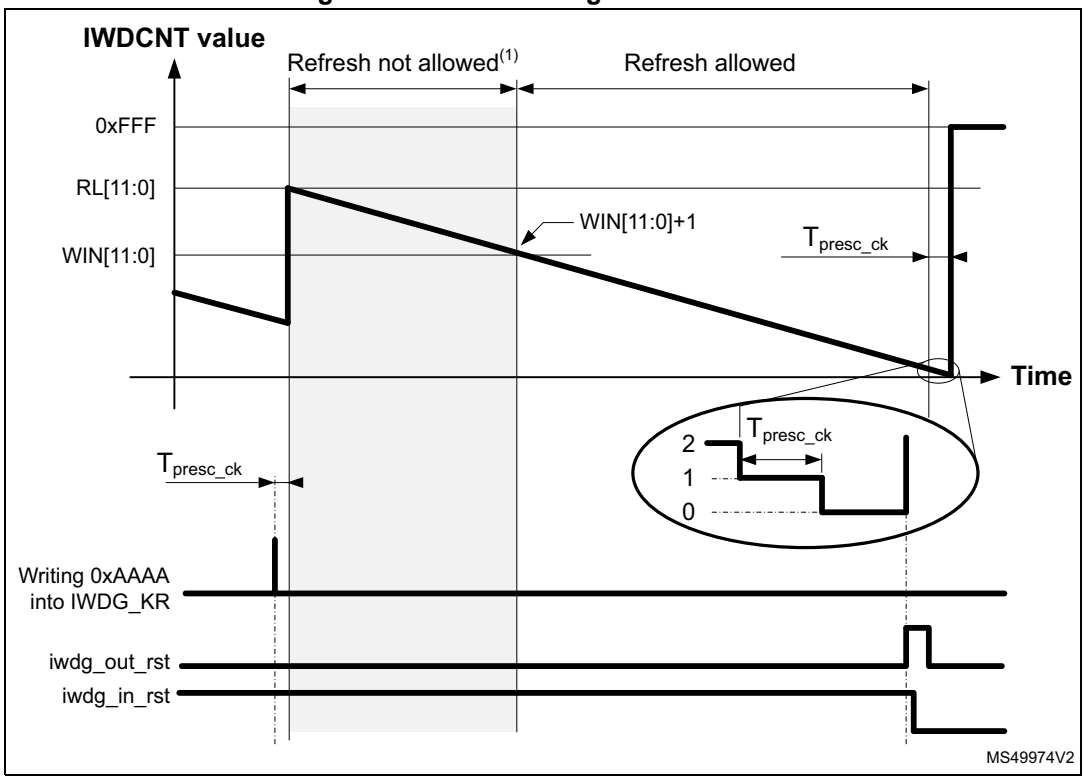
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 553*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 553. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

37.4.4 Window option

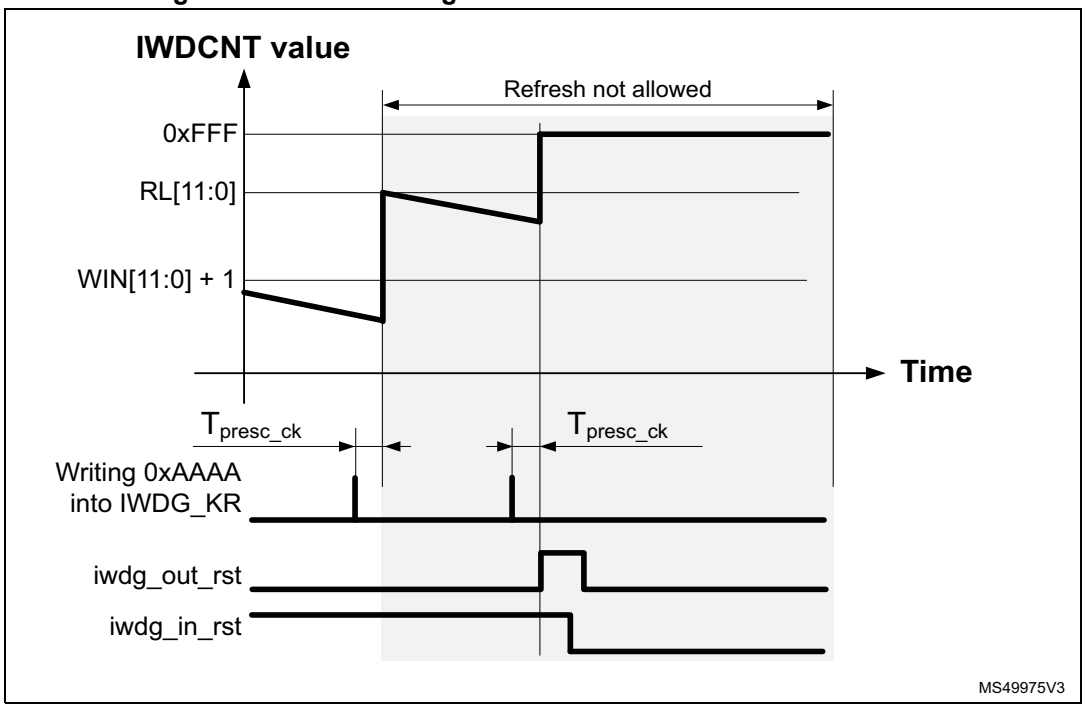
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 554*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 554. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

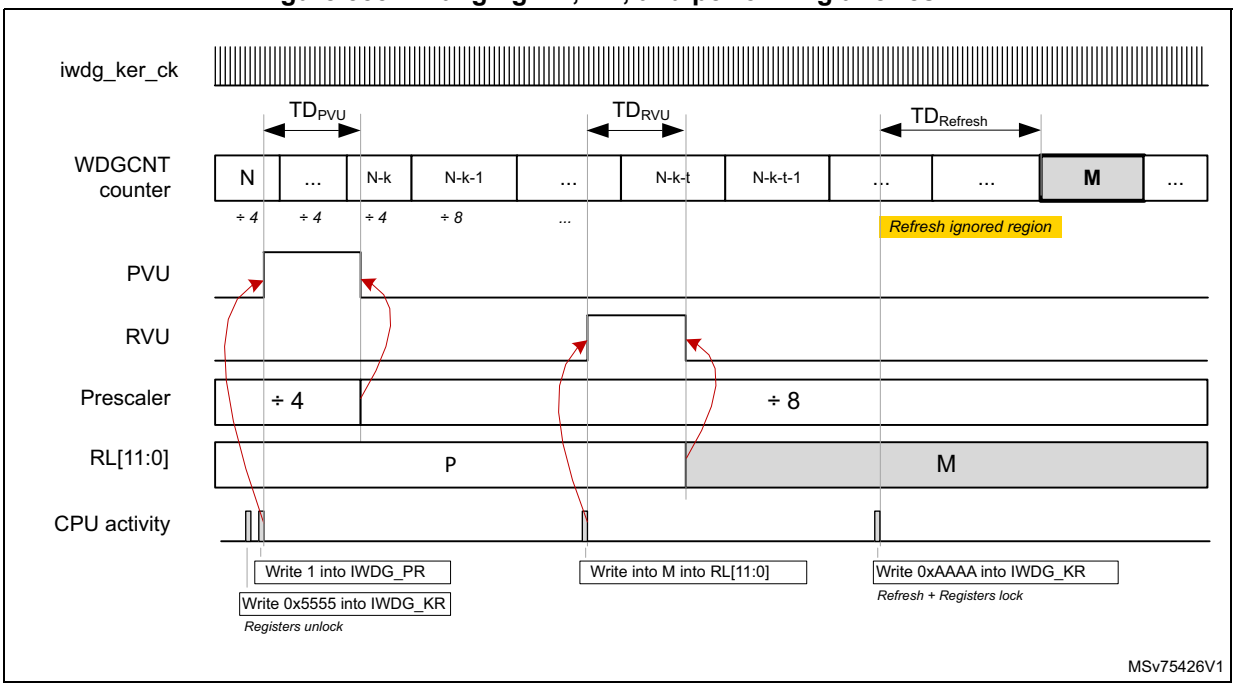
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 555. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 381: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

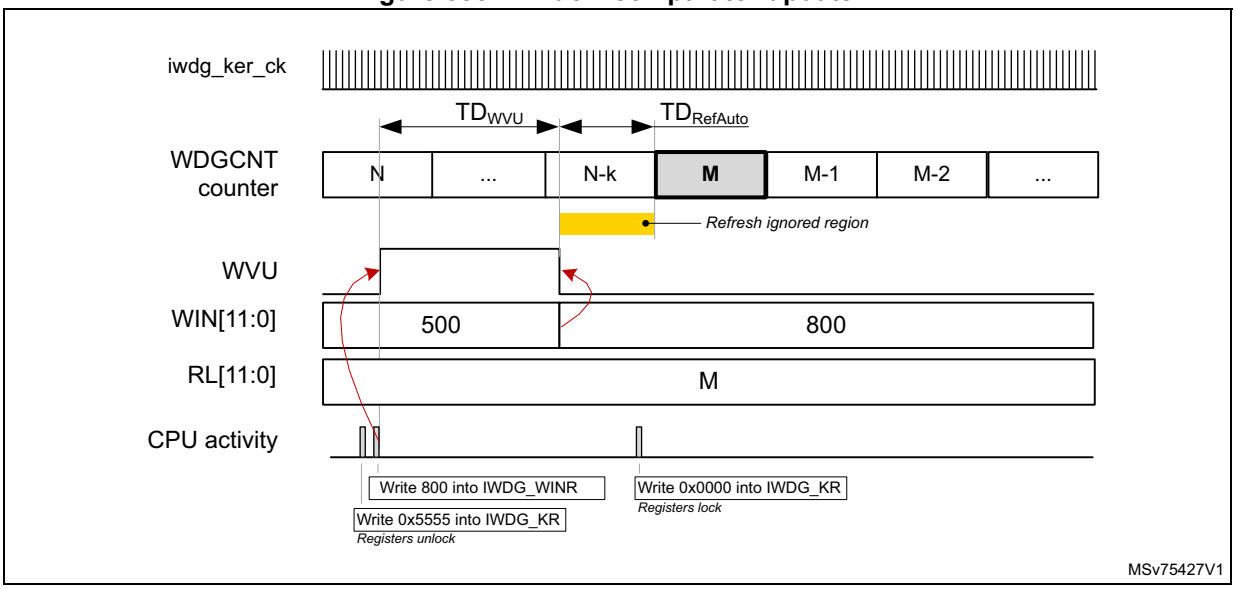
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 556 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 556. Window comparator update(1)



1. Refer to *Table 381: IWDG delays versus actions* for details on timing values.

37.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 37.3* for details on the capabilities of this product.

37.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the

sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

37.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 37.3* for details.

Table 383. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 37.3* for details. IWDG interrupts cause the device exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 37.3* for details. IWDG interrupts do not make the device to exit from Standby mode.

37.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

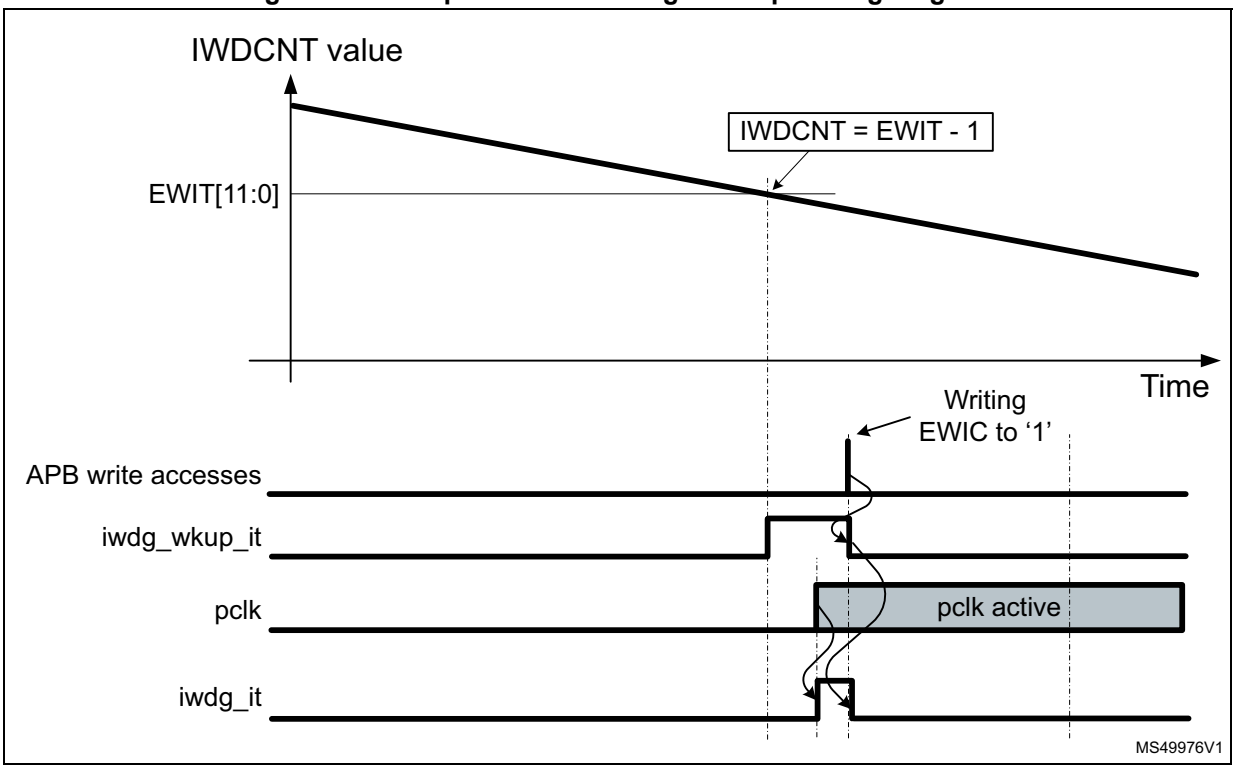
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG interrupt clear register (IWDG_ICR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 557. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

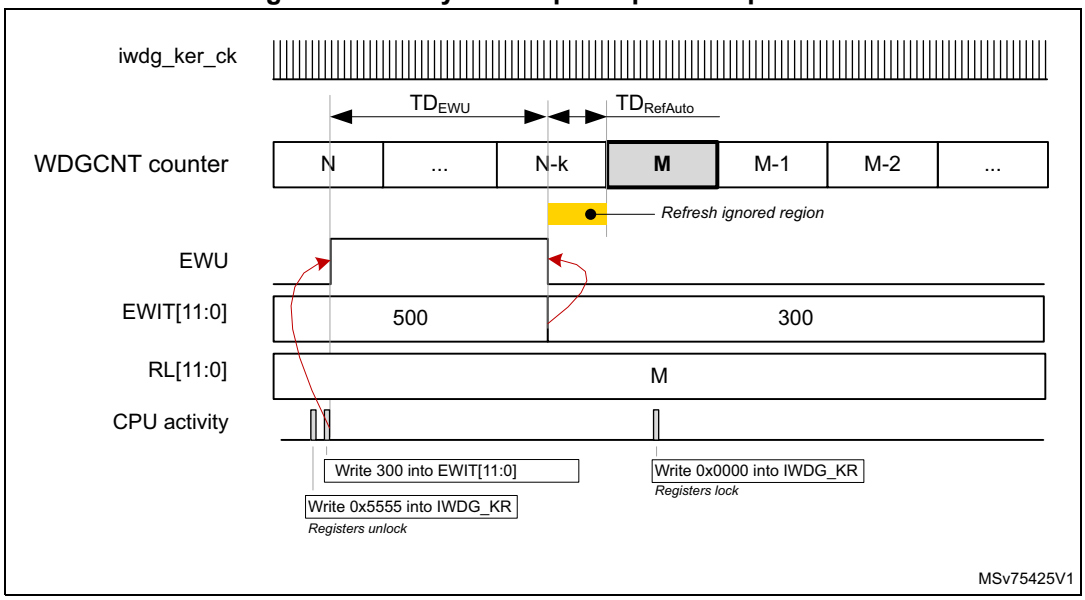
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 558 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 558. Early wake-up comparator update(1)



1. Refer to *Table 381: IWDG delays versus actions* for details on timing values.

Table 384 summarizes the IWDG interrupt request.

Table 384. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

37.7 IWDG registers

Refer to *Section 1.2 on page 85* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

37.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

37.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 37.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

37.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 37.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

37.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in

software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
r							r					r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_ICR register to ‘1’.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 381: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 381: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 381: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 381: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the

early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

37.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 37.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

37.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIE	Res.	Res.	Res.	EWIT[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter early wake-up interrupt value

These bits are write access protected (see *Section 37.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

37.7.7 IWDG interrupt clear register (IWDG_ICR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 14:0 Reserved, must be kept at reset value.

37.7.8 IWDG register map

Table 385. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													0	0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																	0							x					0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	Res.	Res.	Res.	EWIT[11:0]											
	Reset value																	0				0	0	0	0	0	0	0	0	0	0	0	0
0x18	IWDG_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0															

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

38 System window watchdog (WWDG)

38.1 WWDG introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

38.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 560*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

38.3 WWDG implementation

Table 386. WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the hardware mode(4)	X

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

38.4 WWDG functional description

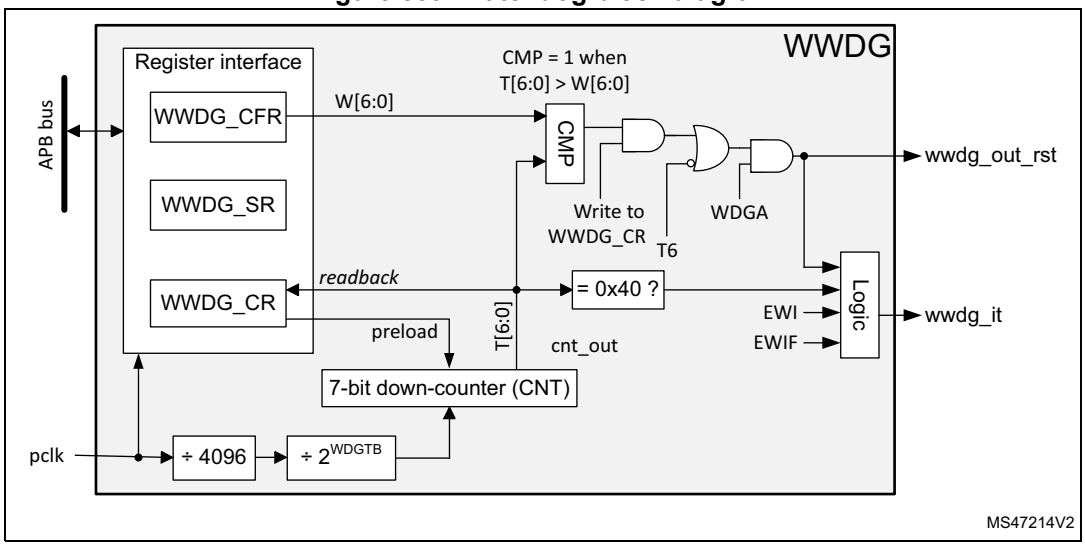
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 559* for the WWDG block diagram.

38.4.1 WWDG block diagram

Figure 559. Watchdog block diagram



38.4.2 WWDG internal signals

Table 387 gives the list of WWDG internal signals.

Table 387. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

38.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

38.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 560*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 560* describes the window watchdog process.

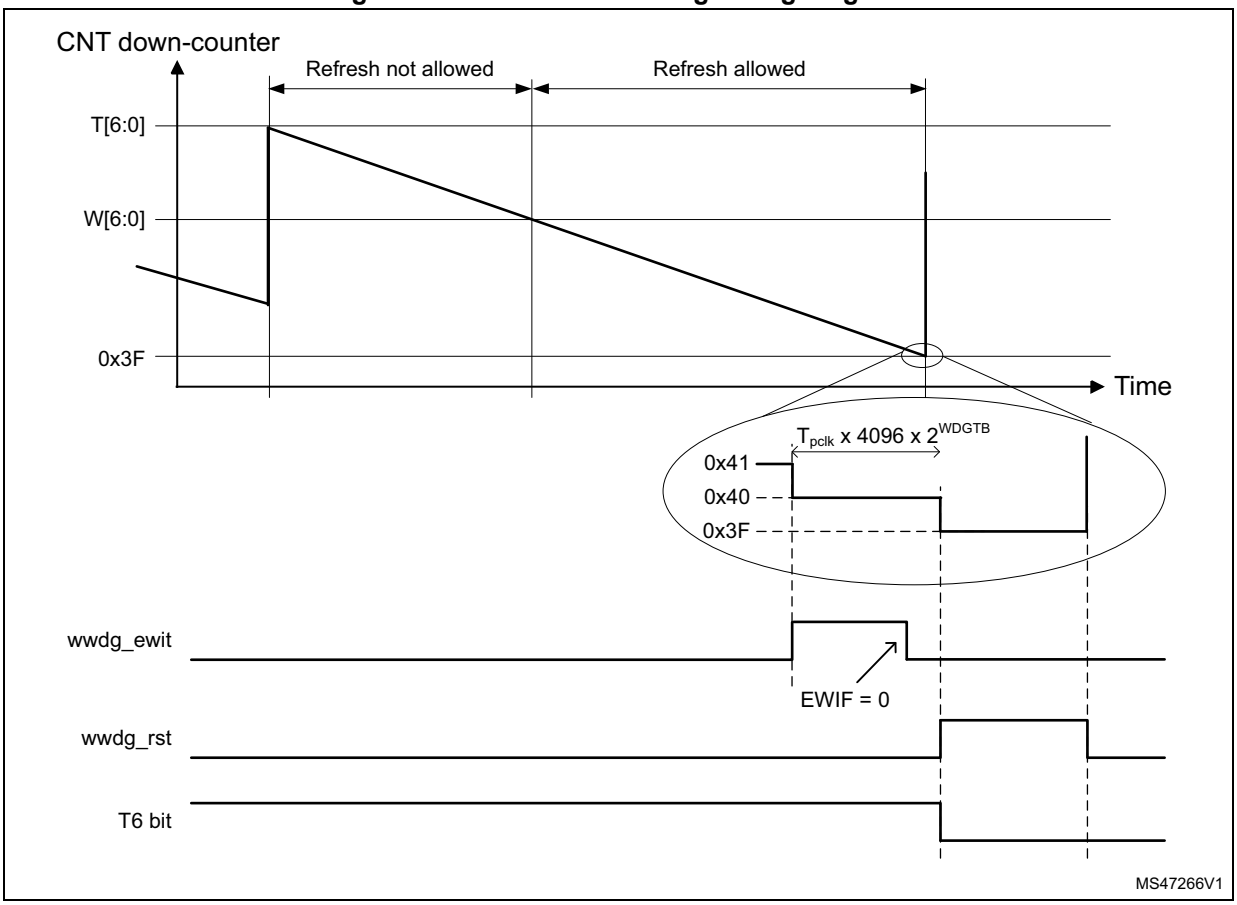
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

38.4.5 How to program the watchdog timeout

Use the formula in *Figure 560* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 560. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

38.4.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 71: Debug support (DBG)*.

38.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

Table 388. WWDG interrupt requests

Interrupt		Event flag	Enable control bit	Interrupt clearing method	Exit from mode		
Vector	Event				Sleep	Stop(1)	Standby(1)
WWDG(2)	Early wake-up interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 38.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

38.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

38.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

38.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

38.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

38.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 389. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

39 Real-time clock (RTC)

39.1 RTC introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

39.2 RTC main features

The RTC supports the following features (see *Figure 561: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content.This function can be triggered by an event on the timestamp pin, or by a tamper event.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

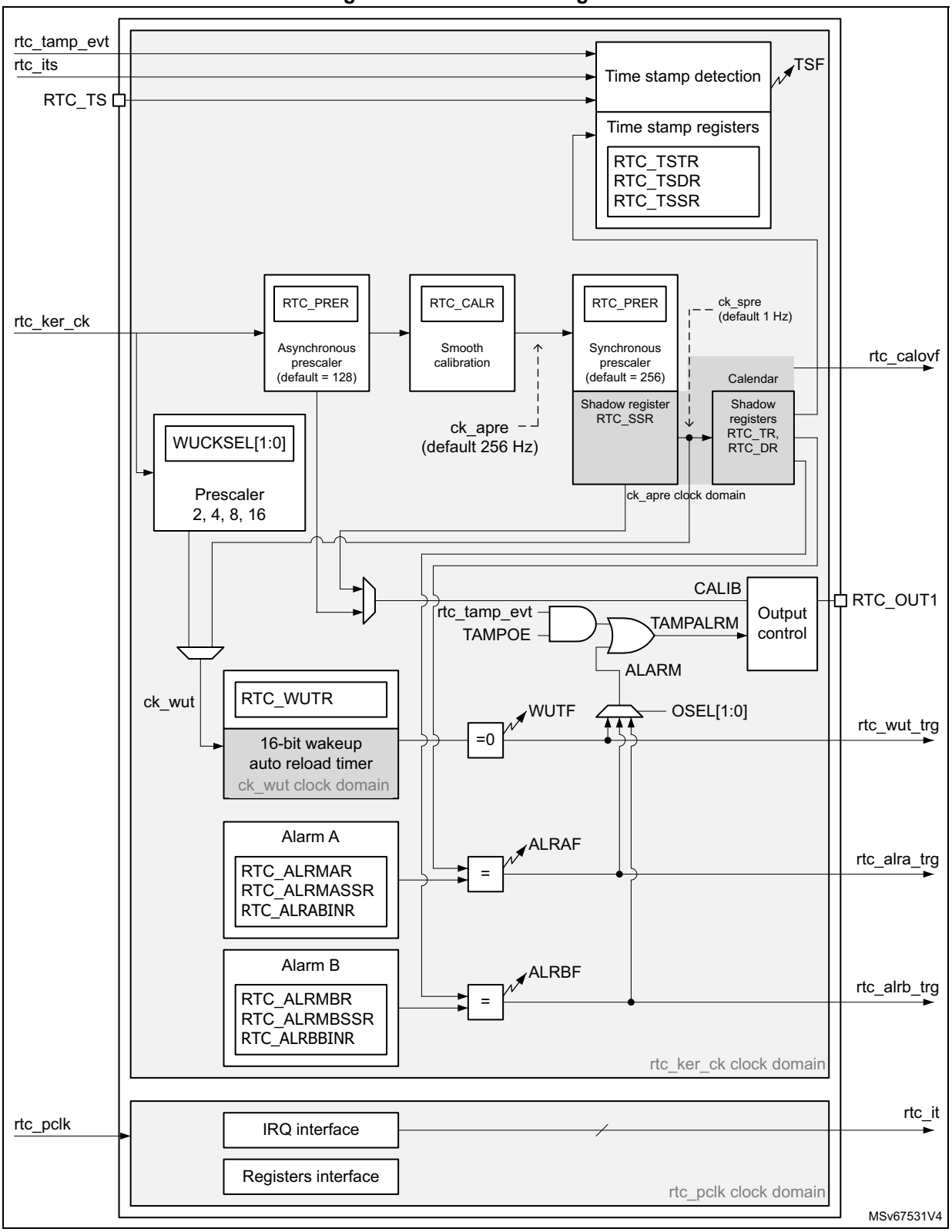
The RTC is functional in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

39.3 RTC functional description

39.3.1 RTC block diagram

Figure 561. RTC block diagram



39.3.2 RTC pins and internal signals

Table 390. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 391. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_it	Output	RTC interrupts (refer to *Section 39.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes when the selected clock is not LSE. Refer to *Section 39.4: RTC low-power modes* for more details.

Table 392. RTC interconnection

Signal name	Source/destination
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

39.3.3 GPIOs controlled by the RTC and TAMP

RTC_OUT1, RTC_TS and TAMP_IN1 are mapped on the same pin. The RTC and TAMP functions mapped on this pin are available in all power modes. This pin output mechanism follows the priority order shown in *Table 393*.

Table 393. RTC pin configuration(1)

Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care

Table 393. RTC pin configuration(1) (continued)

Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMP_IN1 input floating	00	0	0	Don't care	Don't care	Don't care	1	0
	00	0	1	1				
	Don't care	Don't care	0					
RTC_TS and TAMP_IN1 input floating	00	0	0	Don't care	Don't care	Don't care	1	1
	00	0	1	1				
	Don't care	Don't care	0					
RTC_TS input floating	00	0	0	Don't care	Don't care	Don't care	0	1
	00	0	1	1				
	Don't care	Don't care	0					
Wake-up pin or Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0
	00	0	1	1				
	Don't care	Don't care	0					

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

In addition, it is possible to output RTC_OUT2 on PB2 pin thanks to OUT2EN bit. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in *Table 394: RTC_OUT mapping*.

Table 394. RTC_OUT mapping

RTC_OUT mapping				
OSEL[1:0] bits ALARM output enable)	**COE bit (CALIB output enable)**	**OUT2EN bit**	**RTC_OUT1 on PC13**	**RTC_OUT2 on PB2**
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

39.3.4 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and unprivileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and unprivilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and unprivilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in

the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR can be read and written only in privilege mode.

- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE, ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_MISR can be read and written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR can be read and written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, TSE, TSIE, TSEDGE control bits in the RTC_CR, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF in RTC_SR, and TSMF, TSOVMF in RTC_MISR can be read and written only in privilege mode.

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

39.3.5 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 561: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 39.3.9: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

39.3.6 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 39.6.11: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the

BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

39.3.7 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

39.3.8 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

39.3.9 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 1563*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 39.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop, and Standby) have no influence on the wake-up timer.

39.3.10 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After a system reset, the RTC registers are protected against parasitic write access by the DRTCP bit in the power control peripheral (refer to the PWR power control section). DRTCP bit must be set to enable RTC registers write access.

After Backup domain reset, the RTC registers related to RTC initialization and RTC calibration are write-protected: Writing the unlock sequence in RTC_WPR is required to gain write access to these registers.

The following steps are required to unlock the write protection on the registers related to RTC initialization that is RTC_TR, RTC_DR, RTC_PRER, the bits INIT, BIN and BCDU in RTC_ICSR, and the bit FMT in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be performed with the privilege mode defined by PRIV, INITPRIV configuration

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with incorrect privilege settings has no effect and is ignored without any illegal access notification.

The following steps are required to unlock the write protection of the registers related to RTC calibration, that is RTC_CALR, RTC_SHIFTR, and the bits SUB1H, ADD1H, REFCKON in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be done with the privilege mode defined by PRIV, CALPRIV configuration.

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with the incorrect privilege settings has no effect and is ignored without any illegal access notification.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   – If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   – If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   – If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   – If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: There is no hardware protection against incorrect calendar value initialization. Software must ensure that the initialization is performed correctly using allowed calendar BCD values.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its RTC domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

39.3.11 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than two times the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller

than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 1562*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 39.3.13: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the two readings. If the values differ, repeat the process until matching results are obtained. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

39.3.12 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a RTC domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the RTC domain reset one (refer to RCC for details about RTC clock

sources not affected by system reset). When a RTC domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

39.3.13 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

39.3.14 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

39.3.15 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

39.3.16 Timestamp function

Timestamp is enabled by setting the TSE bit of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and refer to *Section : Timestamp on tamper event*.

When a timestamp event occurs, the timestamp flag bit (TSF) in RTC_SR register is set.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin occurs, due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

39.3.17 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

39.3.18 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

39.4 RTC low-power modes

Table 395. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 396. RTC pins functionality over modes

Functions	Functional in Stop mode	Functional in Standby mode
RTC_TS	Yes	Yes
RTC_REFIN	Yes	No
RTC_OUT1	Yes	Yes
RTC_OUT2	Yes	No

39.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 397. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

39.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

39.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1562* and *Reading the calendar on page 1563*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1561*.

This register can be write-protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x00

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

39.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1562* and *Reading the calendar on page 1563*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1561*.

This register can be write-protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x04

RTC domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

39.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values
When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):
SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.
When BCD mode is selected (BIN=00):
SS[31:16] are forced by hardware to 0x0000.
SS[15:0]: Subsecond value/synchronous binary counter LSB values
When Binary mode is selected (BIN = 01 or 10 or 11):
SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.
When BCD mode is selected (BIN=00):
SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:
Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)
SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

39.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1561*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x0C

RTC domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed
1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized
1: Calendar shadow registers synchronized

Note: RSF is not set when the frequency of the APB clock is less than twice the frequency of RTCCLK.

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (RTC domain reset state).

0: Calendar has not been initialized
1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wake-up timer configuration update not allowed

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

39.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 1562*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1561*.

This register can be write-protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x10

RTC domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

39.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x14

RTC domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUTOCLR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

39.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1561.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x18

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.
1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved

1: Save RTC timestamp on tamper detection event

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.

Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **COE**: Calibration output enable

This bit enables the CALIB output

0: Calibration output disabled

1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to TAMPALRM output.

00: Output disabled

01: Alarm A output enabled

10: Alarm B output enabled

11: Wake-up output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of TAMPALRM output.

0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection

When COE = 1, this bit selects which signal is output on CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 39.3.17: Calibration clock output*.

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Note: ADD1H is ignored when SUB1H is set.

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp interrupt disable

1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable
0: Wake-up timer interrupt disabled
1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable
0: Wake-up timer disabled
1: Wake-up timer enabled
Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable
0: SSR underflow interrupt disabled
1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

39.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged.

Address offset: 0x1C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV	INIT PRIV	CAL PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS PRIV	WUT PRIV	ALRB PRIV	ALRA PRIV
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or unprivileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or unprivileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, daylight saving, calibration and reference clock privilege protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is privileged or unprivileged.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection

0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 39.3.4: RTC privilege protection modes for details on the read protection.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 39.3.4: RTC privilege protection modes*).

39.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key
This byte is written by software.
Reading this byte always returns 0x00.
Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

39.6.10 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1561*.

This register can be write-protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x28

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.

Refer to *Section 39.3.15: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 39.3.15: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 39.3.15: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 39.3.15: RTC smooth digital calibration on page 1566*.

39.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1561*.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

39.6.12 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

39.6.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

39.6.14 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x38

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values
SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

39.6.15 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x40

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don't care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

39.6.16 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x44

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.
2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.
...
31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.
From 32 to 63: All 32 SS bits are compared and must match to activate alarm.
Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value
This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

39.6.17 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x48

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don't care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don't care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don't care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

39.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x4C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:4]		MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.

39.6.19 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x50

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload. SSRUF is set only in binary and mixed modes. It is not set in BCD mode.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

39.6.20 RTC masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x54

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow masked flag
This flag is set by hardware when the SSR underflow interrupt occurs.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVMF**: Timestamp overflow masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp masked flag
This flag is set by hardware when a timestamp interrupt occurs.

Bit 2 **WUTMF**: Wake-up timer masked flag
This flag is set by hardware when the wake-up timer interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B masked flag
This flag is set by hardware when the alarm B interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

39.6.21 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x5C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w		w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing ‘1’ in this bit clears the SSRUF in the RTC_SR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.

Bit 2 **CWUTF**: Clear wake-up timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

39.6.22 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x70

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSR, and so can also be read or written through RTC_ALRMASSR.

39.6.23 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 39.3.4: RTC privilege protection modes*.

Address offset: 0x74

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSR, and so can also be read or written through RTC_ALRMBSSR.

39.6.24 RTC register map

Table 398. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	Res.	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0

Table 398. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0		0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0		0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0		0	0	0	0	0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

40 Tamper and backup registers (TAMP)

40.1 TAMP introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 3 tamper pins and 6 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering.

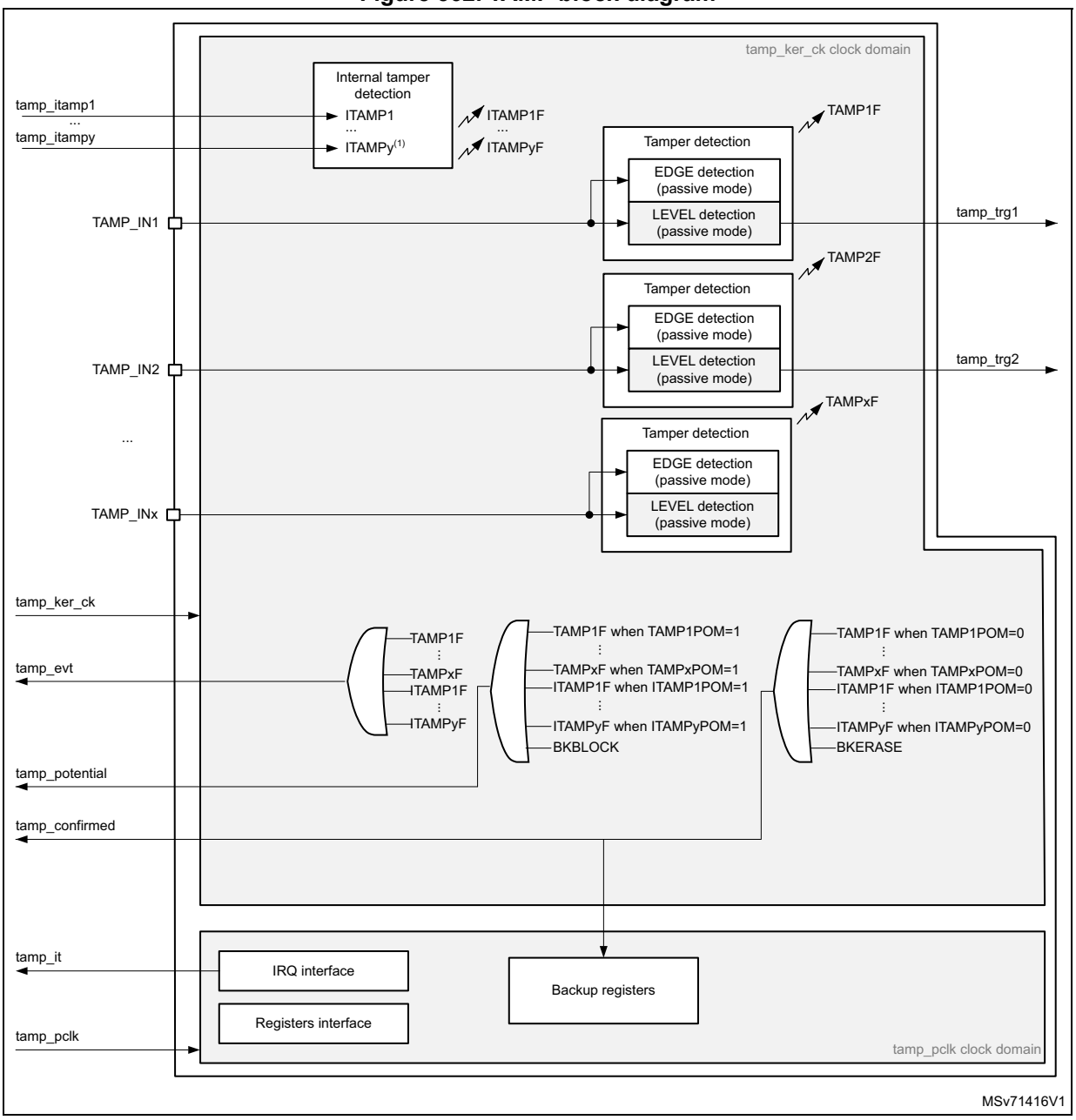
40.2 TAMP main features

- A tamper detection can optionally erase the backup registers, SRAM2, ICACHE, and cryptographic peripherals. The device resources protected by tamper are named “device secrets”.
- 32×32-bit backup registers
- Up to 3 tamper pins for 3 external tamper detection events:
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 6 internal tamper events to protect against transient attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- Tamper configuration and backup registers privilege protection

40.3 TAMP functional description

40.3.1 TAMP block diagram

Figure 562. TAMP block diagram



1. The number of external and internal tampers depends on products.

40.3.2 TAMP pins and internal signals

Table 399. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin

Table 400. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 40.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger

1. Refer to *Table 401: TAMP interconnection*.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). The TAMP kernel clock is required only for external tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock.

Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes depending on the selected clock (refer to for more details.

Table 401. TAMP interconnection

Signal name	Source/Destination
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_potential	The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: • Backup registers • SRAM2 The RHUK (root hardware unique key) in system flash memory hardware bus to SAES is blocked. The tamp_potential signal is used to erase the device secrets listed hereafter: • ICACHE content • SAES, AES, HASH, CCB peripherals • PKA including PKA SRAM The device secrets access is blocked when erase is ongoing.
tamp_confirmed	The tamp_confirmed signal is used to erase the device secrets listed hereafter: • Backup registers • SRAM2 • ICACHE content • SAES, AES, HASH, CCB peripherals • PKA including PKA SRAM The device secrets access is blocked when erase is ongoing. The RHUK (root hardware unique key) in system flash memory hardware bus to SAES is blocked.
tamp_itamp3	LSE monitoring (LSECSS)(1)
tamp_itamp4	HSE monitoring (CSS)(2)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	Unexpected debug activation
tamp_itamp9	Cryptographic peripherals fault (AES or SAES or PKA or TRNG)(3)
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)

1. This monitoring must be enabled by setting LSECSSON in RTC domain control register (RCC_RTCCR).
2. This monitoring must be enabled by setting HSECSSON in RCC control register (RCC_CR).
3. Only TRNG on STM32C53x/C54x/C55x/C56x.

40.3.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 39.3.3: GPIOs controlled by the RTC and TAMP*.

40.3.4 TAMP register write protection

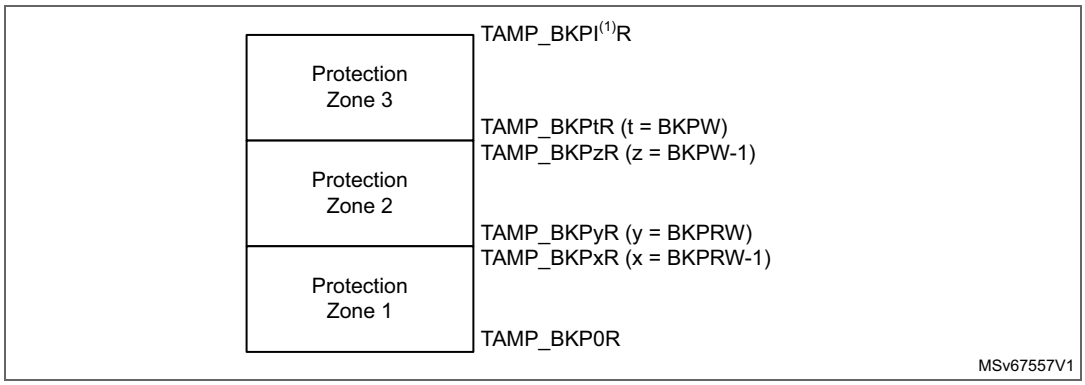
After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DRTCP bit in the power control peripheral (refer to the PWR

power control section). DRTCP bit must be set in order to enable TAMP registers write access.

40.3.5 Backup registers protection zones

The backup registers protection is configured thanks to BKPRW[7:0] and BKPW[7:0] (refer to the figure below):

Figure 563. Backup registers protection zones



1. I = last backup register index

40.3.6 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and unprivileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:
  - Writing the TAMP registers is possible only in privilege mode, except for the backup registers which have their own protection setting.
  - Reading TAMP_CFGR, TAMP_PRIVCFGR is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode, and CNT1SEC can be written only in privilege mode.
- When the BKPRWPRIV is set in the TAMP_PRIVCFGR register: the BKPW can be written only in privilege mode.
- When the BKPWPRIV is set in the TAMP_PRIVCFGR register: the BKPRW and BKPW can be written only in privilege mode.

The backup registers protection is configured thanks to BKPRW[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRW[7:0], BKPW[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 563*).

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

40.3.7 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 401: TAMP interconnection*
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 401: TAMP interconnection*
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or level.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

40.3.8 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed signal in the *Table 401: TAMP interconnection*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential signal in the *Table 401: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 401: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

40.3.9 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ≠ 00). Refer to *Section : Level detection with filtering on tamper inputs (passive mode)*.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 564. Tamper sampling with precharge pulse

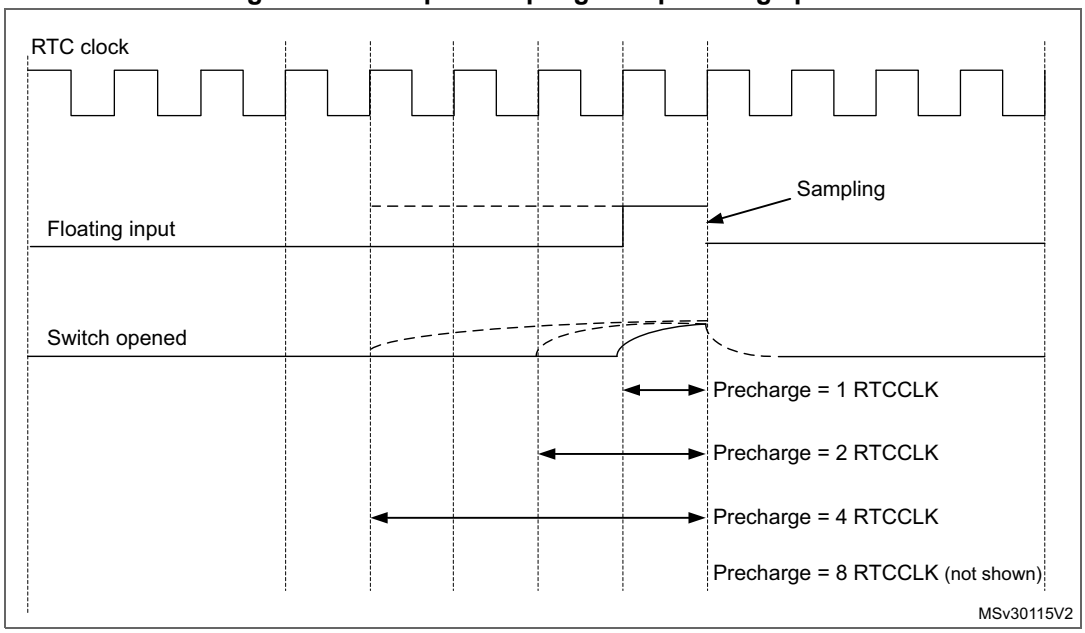
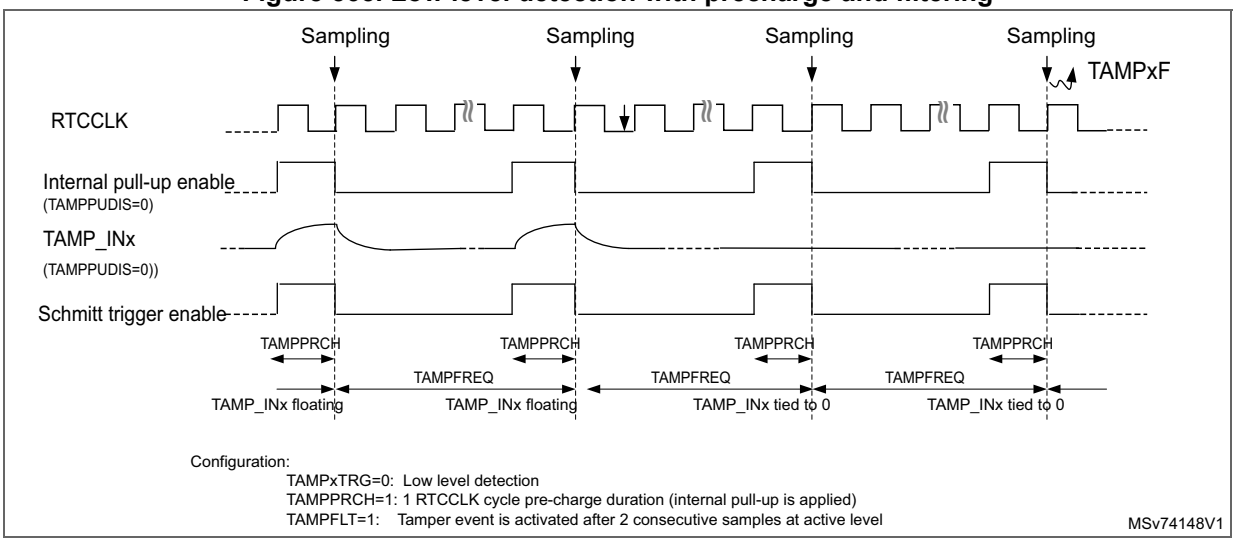


Figure 565. Low level detection with precharge and filtering



Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

40.4 TAMP low-power modes

Table 402. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby mode.

Table 403. TAMP pins functionality over modes

Pin name	Functional in all low-power modes
TAMP_IN[3:1]	Yes

40.5 TAMP interrupts

Table 404. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 402: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

40.6 TAMP registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

40.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP*

privilege protection modes.

Address offset: 0x00

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11E	Res.	ITAMP9E	Res.	Res.	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	Res.	Res.
					rw		rw			rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3E	TAMP2E	TAMP1E
													rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR register and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

40.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x04

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
					rw	rw	rw	w	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3 POM	TAMP2 POM	TAMP1 POM
													rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input
0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.
1: If TAMPFLT ≠ 00 tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input

0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input

0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase

Writing '1' to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked

0: backup registers and device secrets(1) can be accessed if no tamper flag is set

1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TAMP3POM**: Tamper 3 potential mode
0: Tamper 3 event detection is in confirmed mode(1).
1: Tamper 3 event detection is in potential mode(2).

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode(1).
1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode(1).
1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed signal in *Table 401: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential signal in *Table 401: TAMP interconnection*.

40.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ITAMP 11POM	Res.	ITAMP9 POM	Res.	Res.	ITAMP6 POM	ITAMP5 POM	ITAMP4 POM	ITAMP3 POM	Res.	Res.
					rw		rw			rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 Reserved, must be kept at reset value.

Bit 0 Reserved, must be kept at reset value.

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed signal in *Table 401: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential signal in *Table 401: TAMP interconnection*.

40.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x0C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT [1:0]		TAMPFREQ [2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.

0: Precharge TAMP_INx pins before sampling (enable internal pull-up)

1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

40.6.5 TAMP configuration register (TAMP_CFGR)

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 40.3.6: TAMP privilege protection modes*).

Address offset: 0x20

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPW[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRW[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPW[7:0]**: Backup registers write protection offset
BKPW value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRW) to TAMP_BKPzR (z = BKPW-1, with BKPW > BKPRW):
If BKPW = 0 or if BKPW ≤ BKPRW: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPW if BKPW ≥ BKPRW, else t = BKPRW).
If BKPW = 32: there is no protection zone 3.

Refer to *Figure 563: Backup registers protection zones*.

Note: If BKPWPRIV is set, BKPW[7:0] can be written only in privileged mode.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRW[7:0]**: Backup registers read/write protection offset
BKPRW value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRW-1, with BKPRW ≥ 1).
If BKPRW = 0: there is no protection zone 1.

Refer to *Figure 563: Backup registers protection zones*.

Note: If BKPRWPRIV or BKPWPRIV is set, BKPRW[7:0] can be written only in privileged mode.

40.6.6 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

Address offset: 0x24

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding monotonic counters and backup registers)
0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
1: Tamper configuration and interrupt can be written only with privileged access.
Note: Refer to Section 40.3.6: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection
0: Backup registers zone 2 can be written with privileged or unprivileged access.
1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection
0: Backup registers zone 1 can be read and written with privileged or unprivileged access.
1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 Reserved, must be kept at reset value.

40.6.7 TAMP interrupt enable register (TAMP_IER)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11IE	Res.	ITAMP9IE	Res.	Res.	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	Res.	Res.
					rw		rw			rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3IE	TAMP2IE	TAMP1IE
													rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

40.6.8 TAMP status register (TAMP_SR)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11F	Res.	ITAMP9F	Res.	Res.	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	Res.	Res.
					r		r			r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F
													r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11F**: Internal tamper 11 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6F**: Internal tamper 6 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

40.6.9 TAMP masked interrupt status register (TAMP_MISR)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11MF	Res.	ITAMP9MF	Res.	Res.	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	Res.	Res.
					r		r			r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
													r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11MF**: internal tamper 11 interrupt masked flag
This flag is set by hardware when the internal tamper 11 interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 interrupt masked flag
This flag is set by hardware when the internal tamper 9 interrupt is raised.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6MF**: Internal tamper 6 interrupt masked flag
This flag is set by hardware when the internal tamper 6 interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 interrupt masked flag
This flag is set by hardware when the internal tamper 5 interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 interrupt masked flag
This flag is set by hardware when the internal tamper 4 interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 interrupt masked flag
This flag is set by hardware when the internal tamper 3 interrupt is raised.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TAMP3MF**: TAMP3 interrupt masked flag
This flag is set by hardware when the tamper 3 interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 interrupt masked flag
This flag is set by hardware when the tamper 2 interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 interrupt masked flag
This flag is set by hardware when the tamper 1 interrupt is raised.

40.6.10 TAMP status clear register (TAMP_SCR)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CITAMP11F	Res.	CITAMP9F	Res.	Res.	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	Res.	Res.
					w		w			w	w	w	w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP3F	CTAMP2F	CTAMP1F
													w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

40.6.11 TAMP option register (TAMP_OR)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x50

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN3_RMP	Res.	IN2_RMP
													rw		rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IN3_RMP**: TAMP_IN3 pin select
0x0: TAMP_IN3 is mapped on PA1
0x1: TAMP_IN3 is mapped on PA2

Bit 1 Reserved, must be kept at reset value.

Bit 0 **IN2_RMP**: TAMP_IN2 pin select
0x0: TAMP_IN2 is mapped on PA0
0x1: TAMP_IN2 is mapped on PC1

40.6.12 TAMP backup x register (TAMP_BKPxR)

This register can be protected against unprivileged access. Refer to *Section 40.3.6: TAMP privilege protection modes*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 **BKP[31:0]:**

The application can write or read data to and from these registers.
In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

40.6.13 TAMP register map

Table 405. TAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TAMP_CR1	Res.	Res.	Res.	Res.	Res.	ITAMP11E	Res.	ITAMP9E	Res.	Res.	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3E	TAMP2E	TAMP1E
	Reset value						0		0			0	0	0	0																0	0	0
0x04	TAMP_CR2	Res.	Res.	Res.	Res.	Res.	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value						0	0	0	0	0				0	0	0													0	0	0
0x08	TAMP_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP11POM	Res.	ITAMP9POM	Res.	Res.	ITAMP6POM	ITAMP5POM	ITAMP4POM	ITAMP3POM	Res.	Res.
	Reset value																						0		0			0	0	0	0		
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]			TAMPFREQ[2:0]		
	Reset value																								0	0	0	0	0	0	0	0	0
0x20	TAMP_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRW[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x24	TAMP_ PRIVCFGR	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0																													
0x2C	TAMP_IER	Res.	Res.	Res.	Res.	Res.	ITAMP11IE	Res.	ITAMP9IE	Res.	Res.	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value						0		0			0	0	0	0																0	0	0
0x30	TAMP_SR	Res.	Res.	Res.	Res.	Res.	ITAMP11F	Res.	ITAMP9F	Res.	Res.	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F
	Reset value						0		0			0	0	0	0																0	0	0
0x34	TAMP_MISR	Res.	Res.	Res.	Res.	Res.	ITAMP11MF	Res.	ITAMP9MF	Res.	Res.	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value						0		0			0	0	0	0																0	0	0
0x3C	TAMP_SCR	Res.	Res.	Res.	Res.	Res.	CITAMP11F	Res.	CITAMP9F	Res.	Res.	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res.	Res.	Res.	Res.	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value						0		0			0	0	0	0																0	0	0

Table 405. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	TAMP_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN3_RMP	Res.	IN2_RMP
	Reset value																														0		0
0x100 + 0x04*x, (x= 0 to 31)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

41 Inter-integrated circuit interface (I2C)

41.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

41.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to *Section 41.3: I2C implementation*.

41.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 406. I2C implementation

I2C features(1)	I2C1	I2C2(2)
7-bit addressing mode	X	X
10-bit addressing mode	X	X
Standard-mode (up to 100 kbit/s)	X	X
Fast-mode (up to 400 kbit/s)	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X
Independent clock	X	X
Wake-up from Stop mode only (no autonomous mode)	X	X
SMBus/PMBus	X	X

1. X = supported.
2. I2C2 is not available on STM32C53x/542 devices.

41.4 I2C functional description

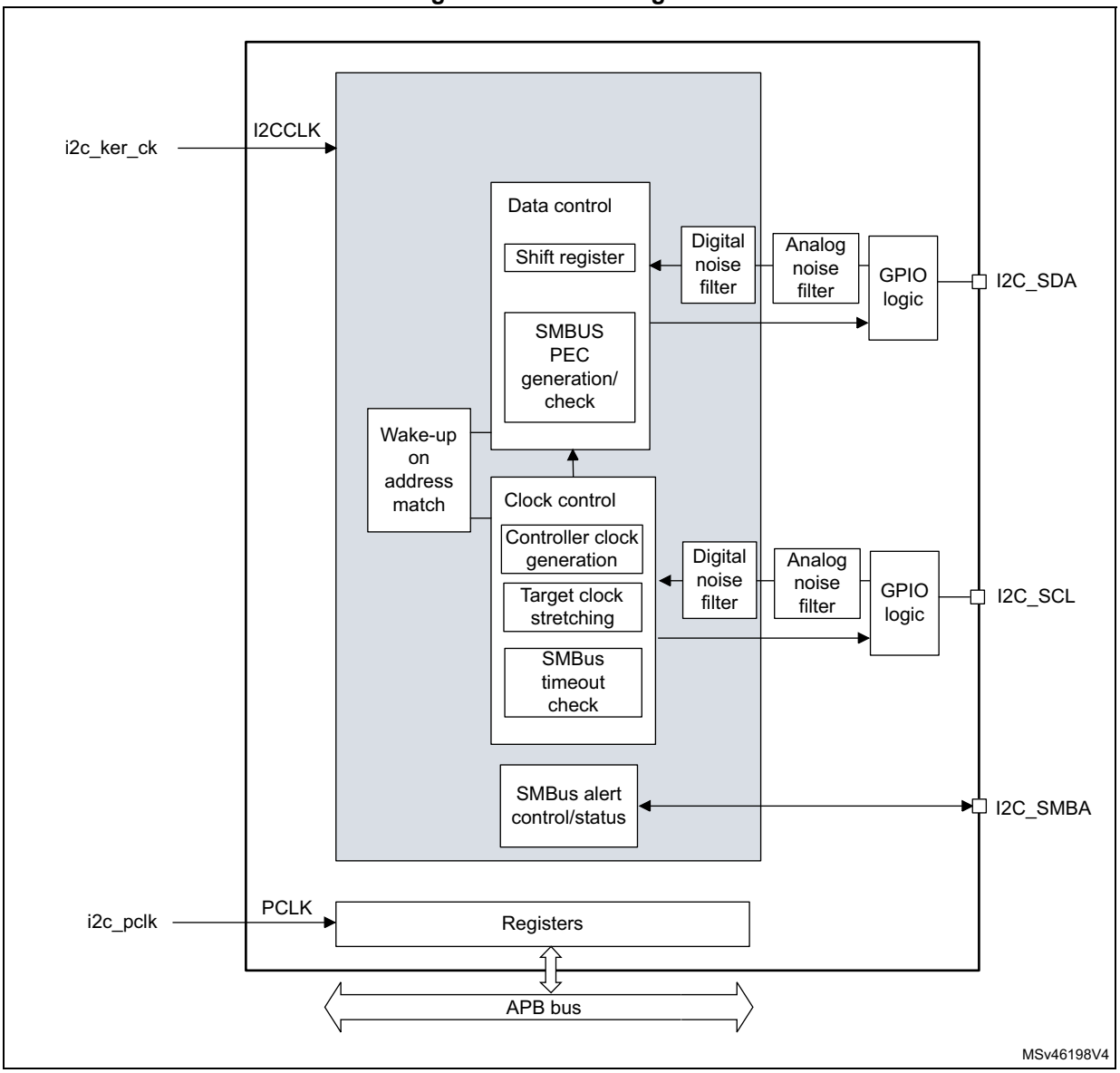
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

41.4.1 I2C block diagram

Figure 566. Block diagram



41.4.2 I2C pins and internal signals

Table 407. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 408. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 421* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

41.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

41.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

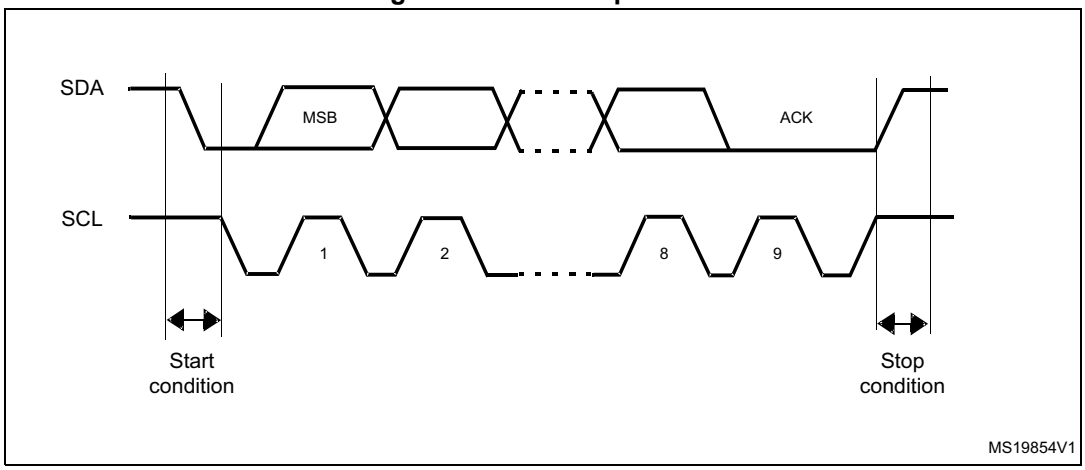
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 567. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

41.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 41.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 409. Comparison of analog and digital filters

Item	Analog filter	Digital filter
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

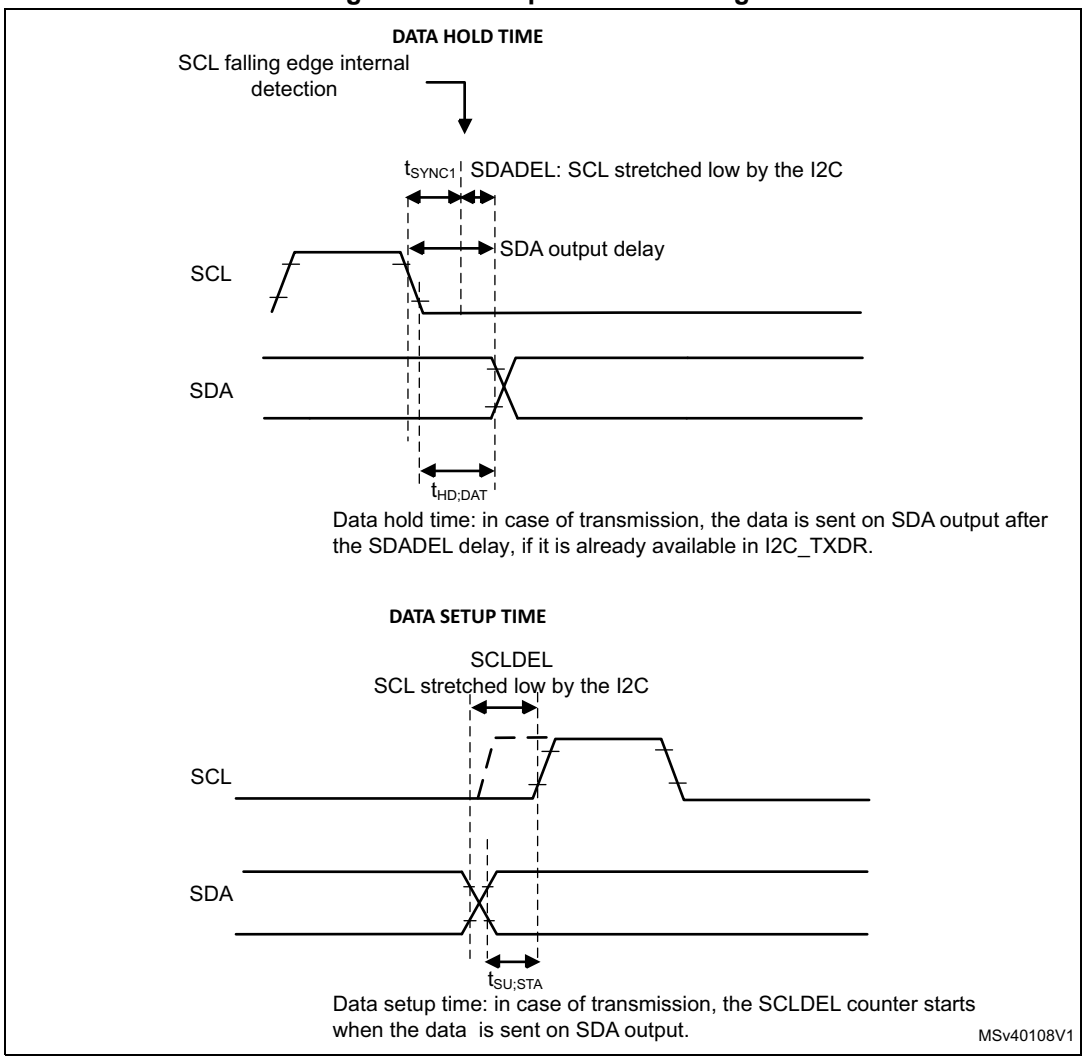
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 568. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

$\{t_{f(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\,(max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 410. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

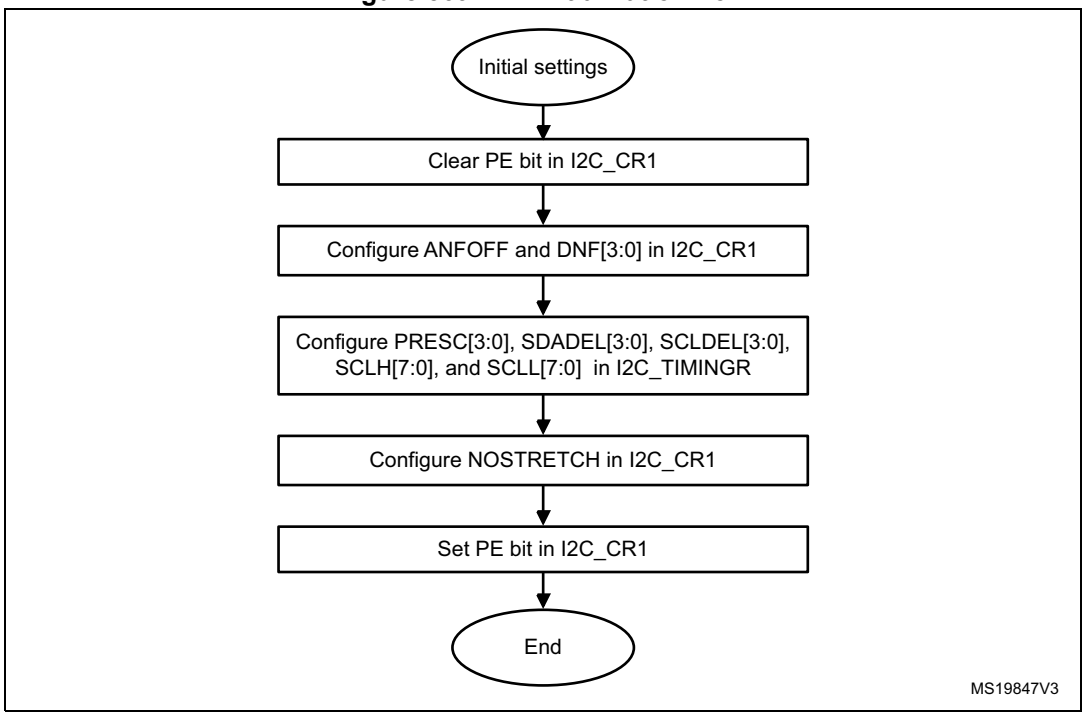
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 569. I2C initialization flow



41.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

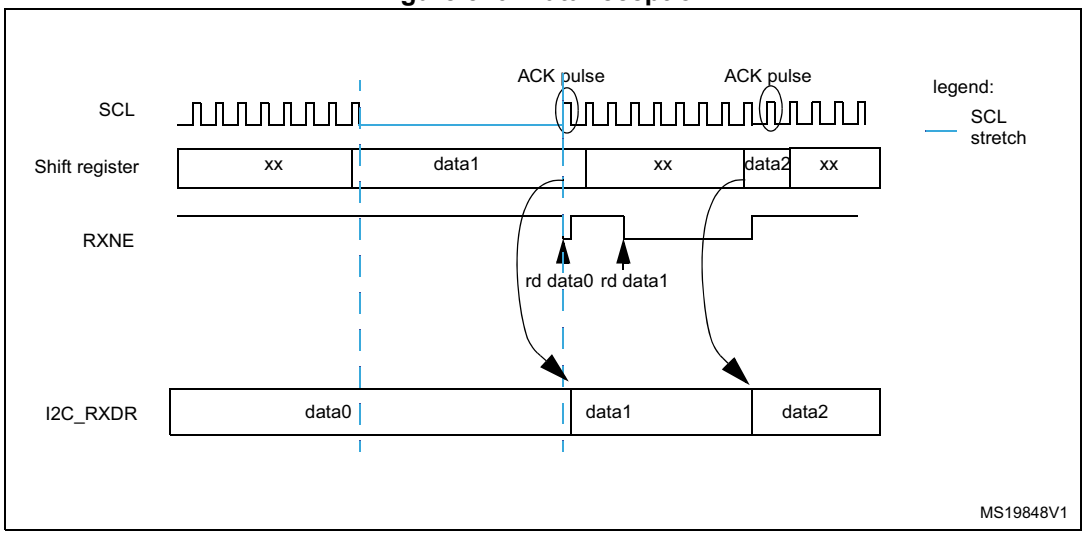
41.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

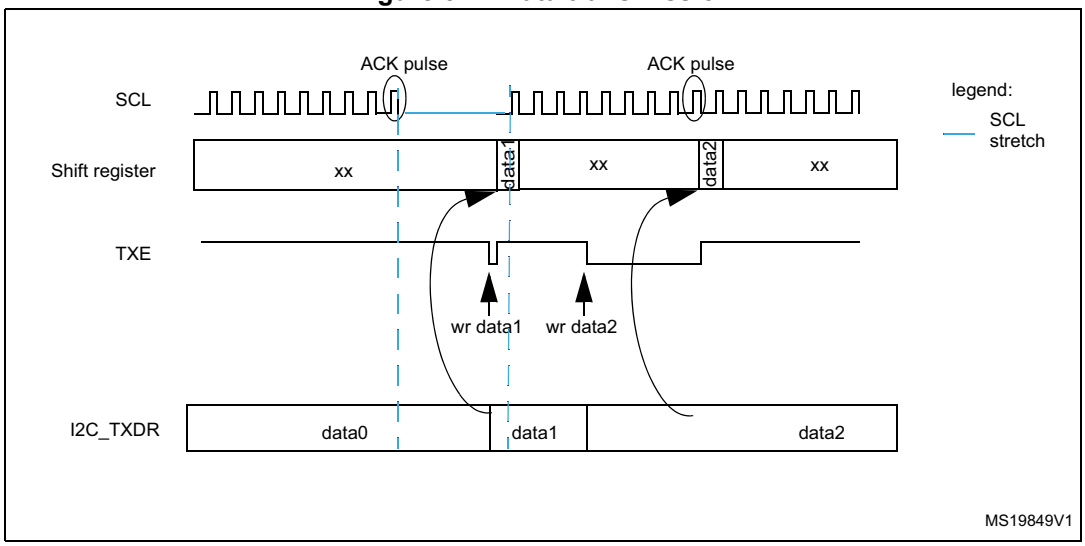
Figure 570. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 571. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 411. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

41.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

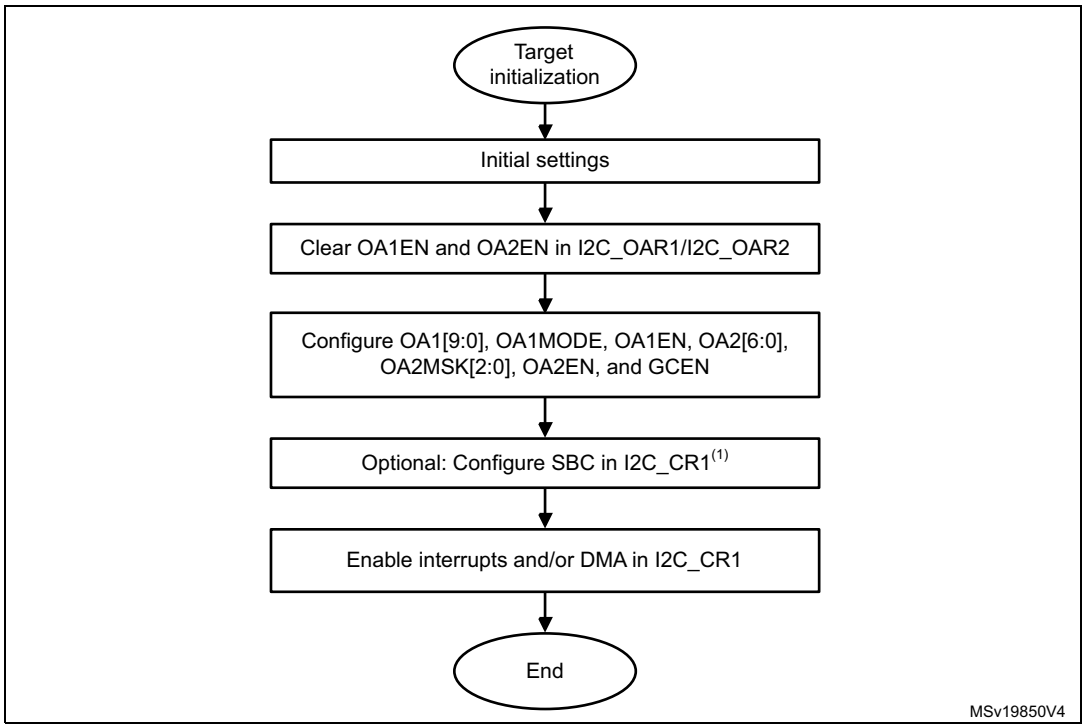
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 572. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 573. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

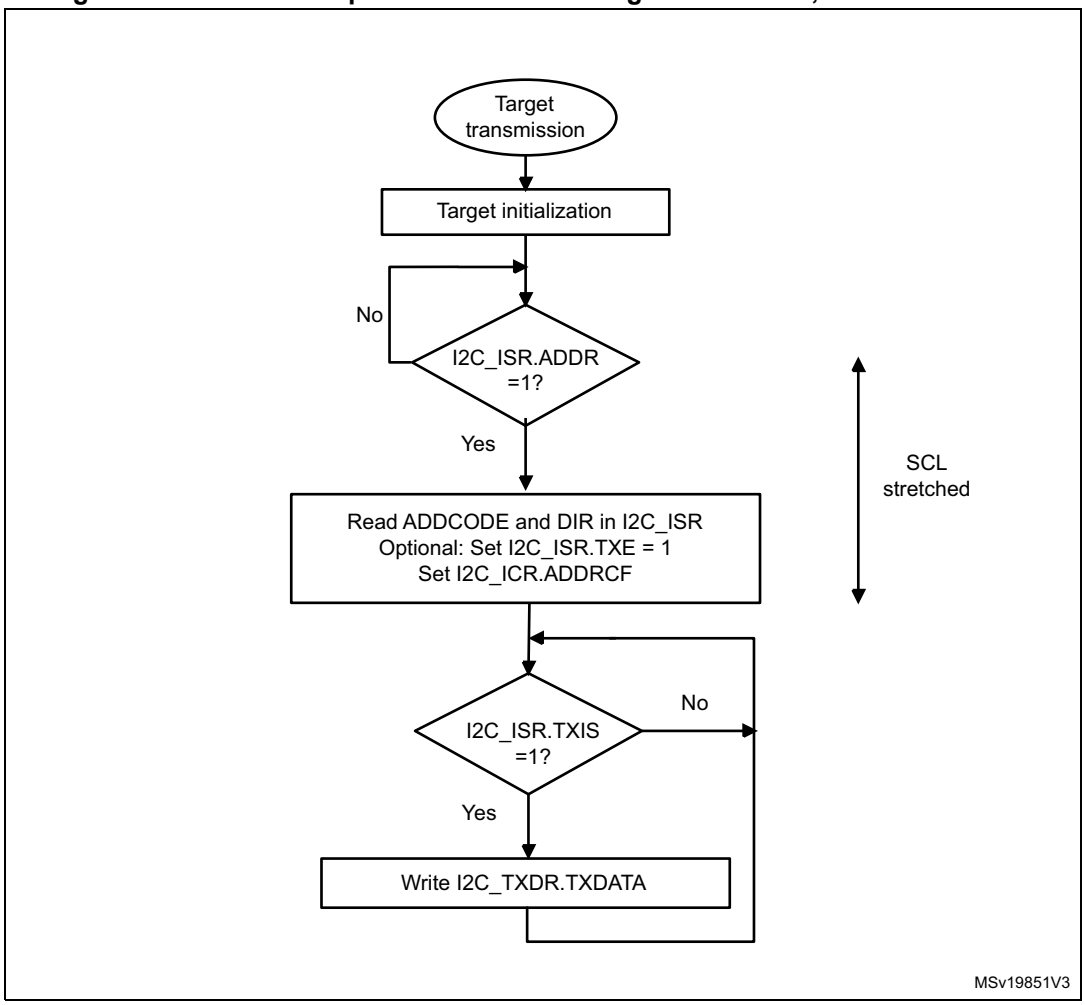


Figure 574. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

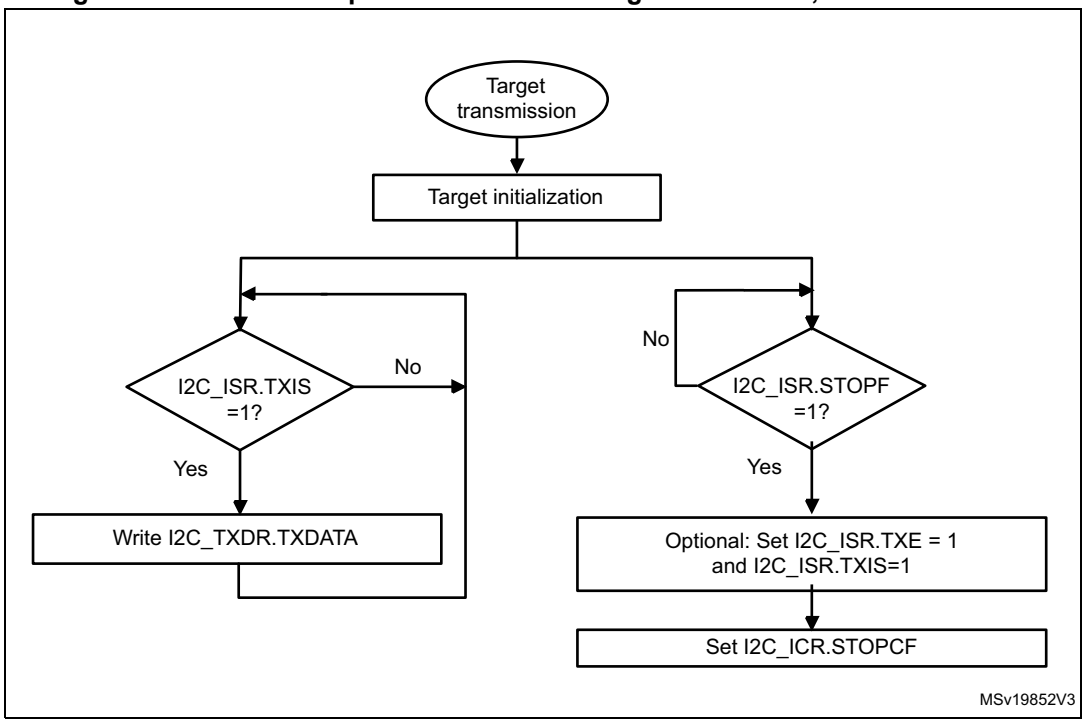
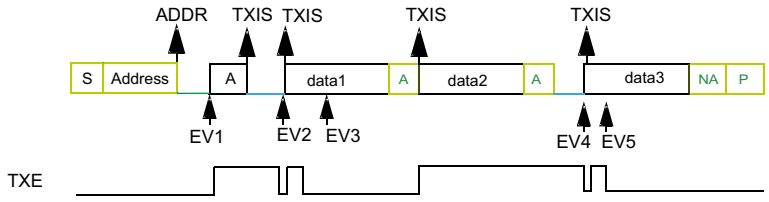


Figure 575. Transfer bus diagrams for I2C target transmitter (mandatory events only)

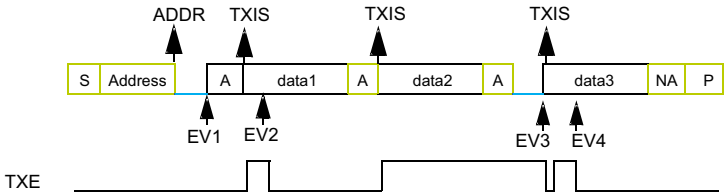
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

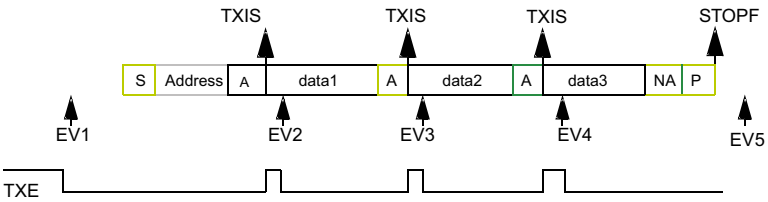
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 576. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

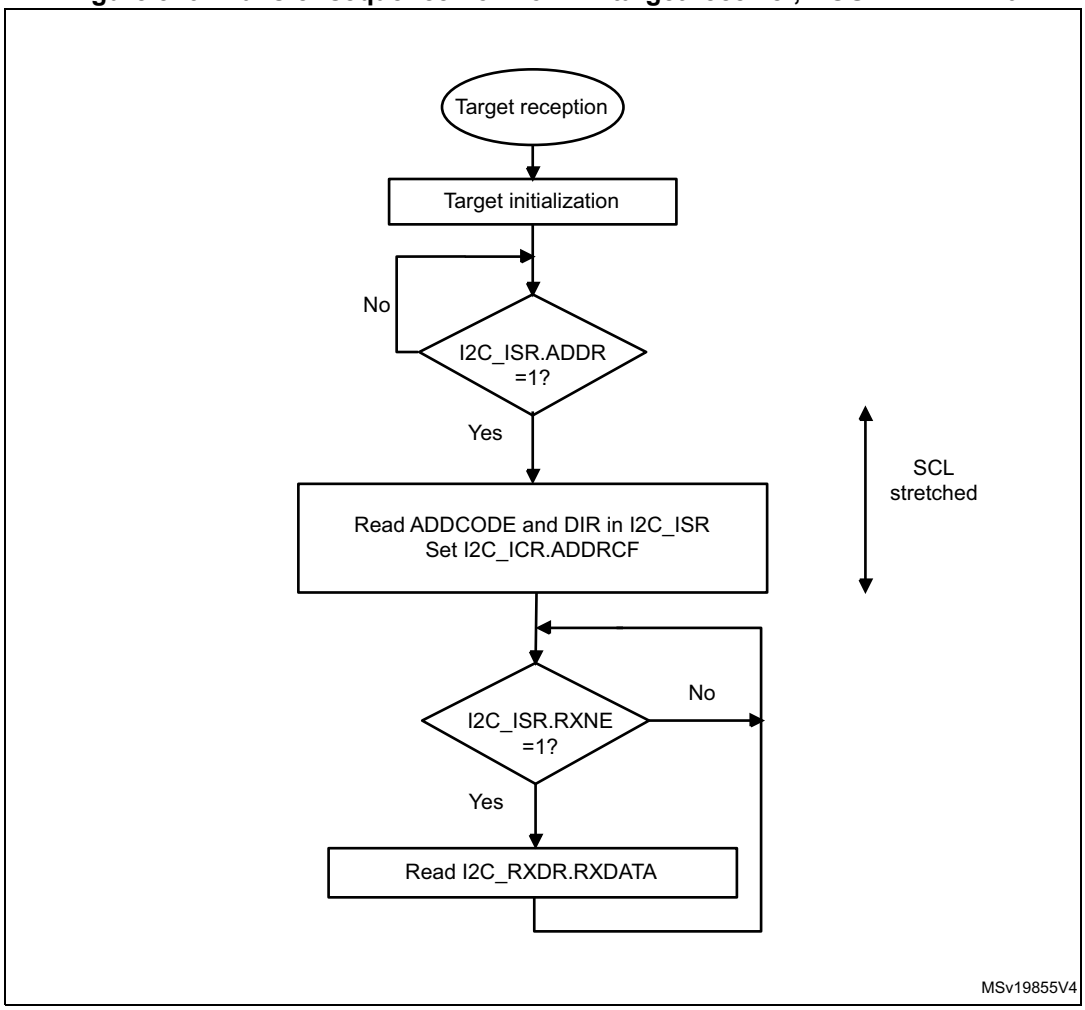


Figure 577. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

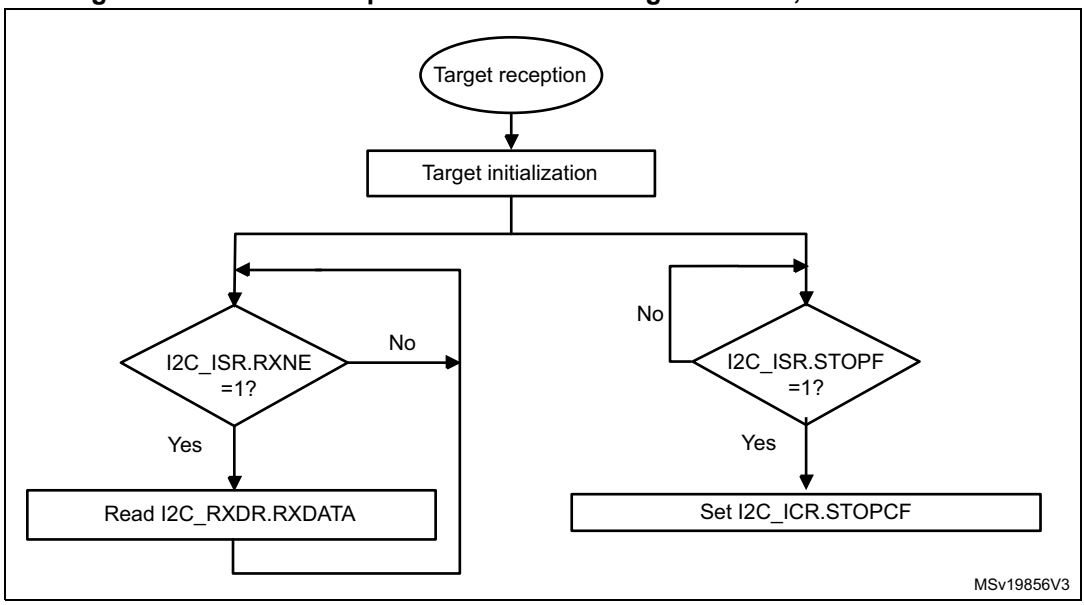
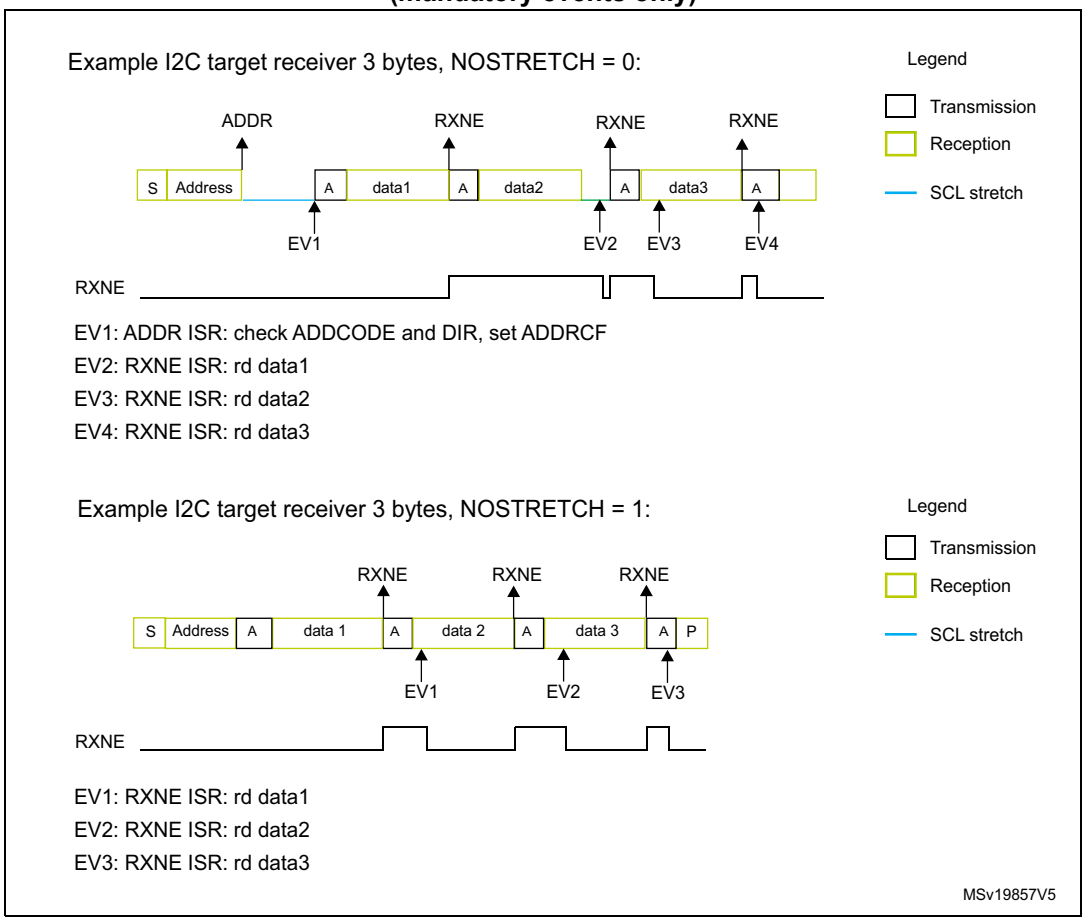


Figure 578. Transfer bus diagrams for I2C target receiver (mandatory events only)



41.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

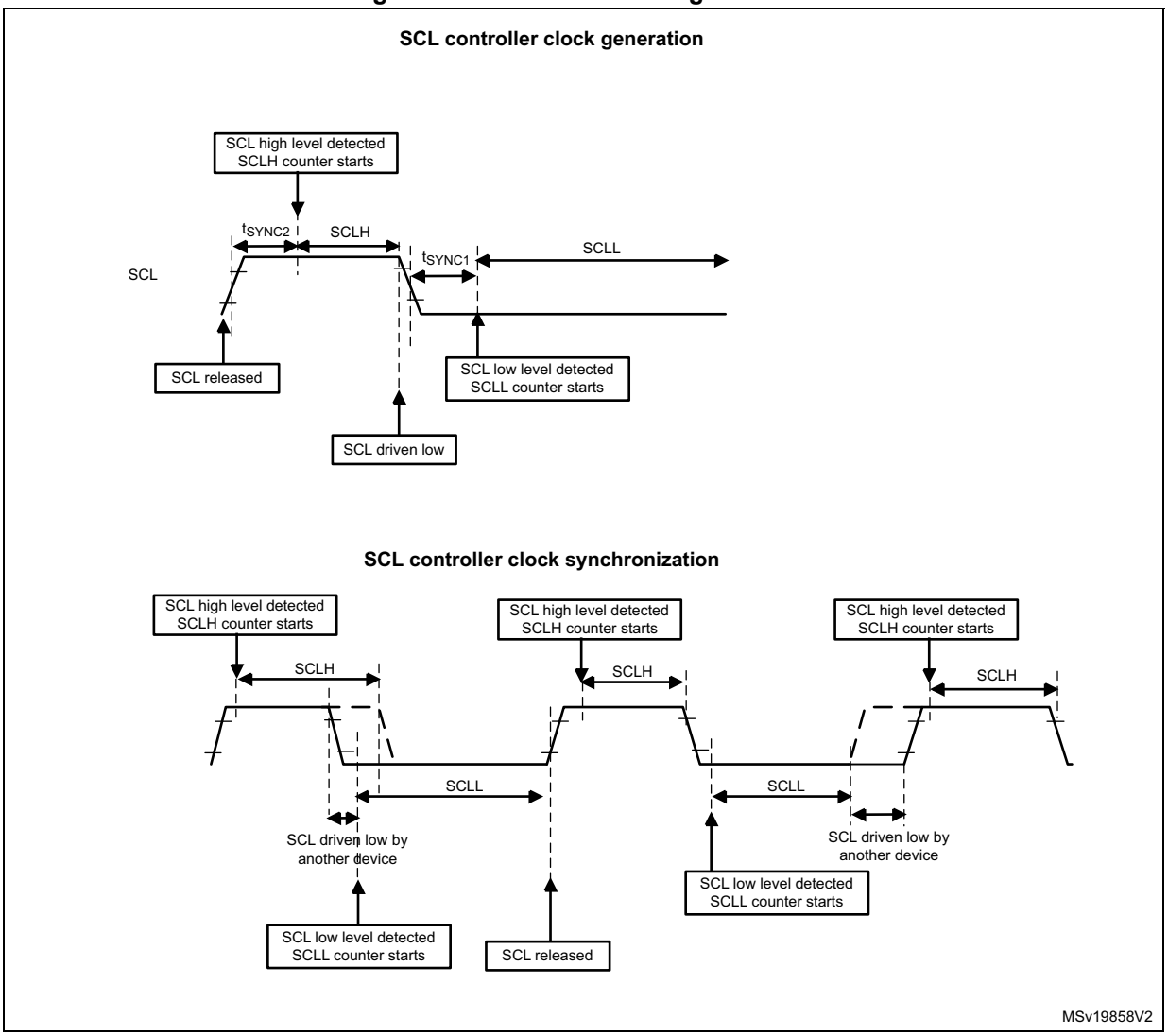
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 579. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 412. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 41.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

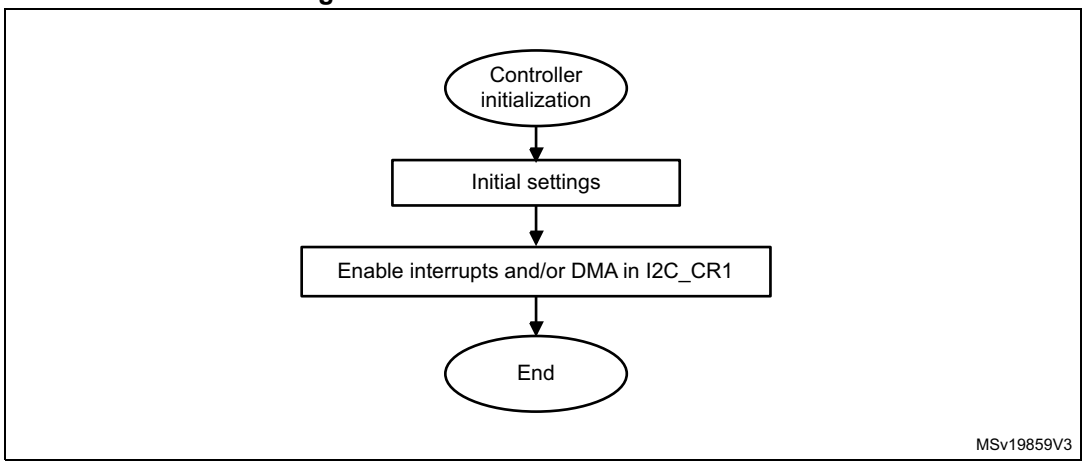
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 580. Controller initialization flow

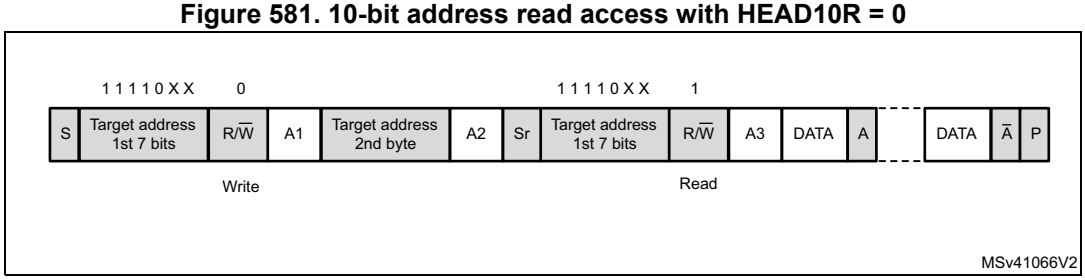


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

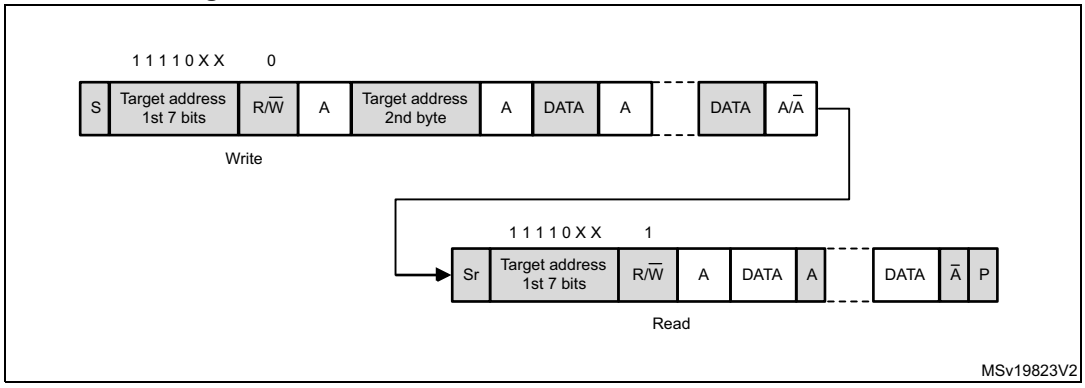
Figure 581. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 582. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 583. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

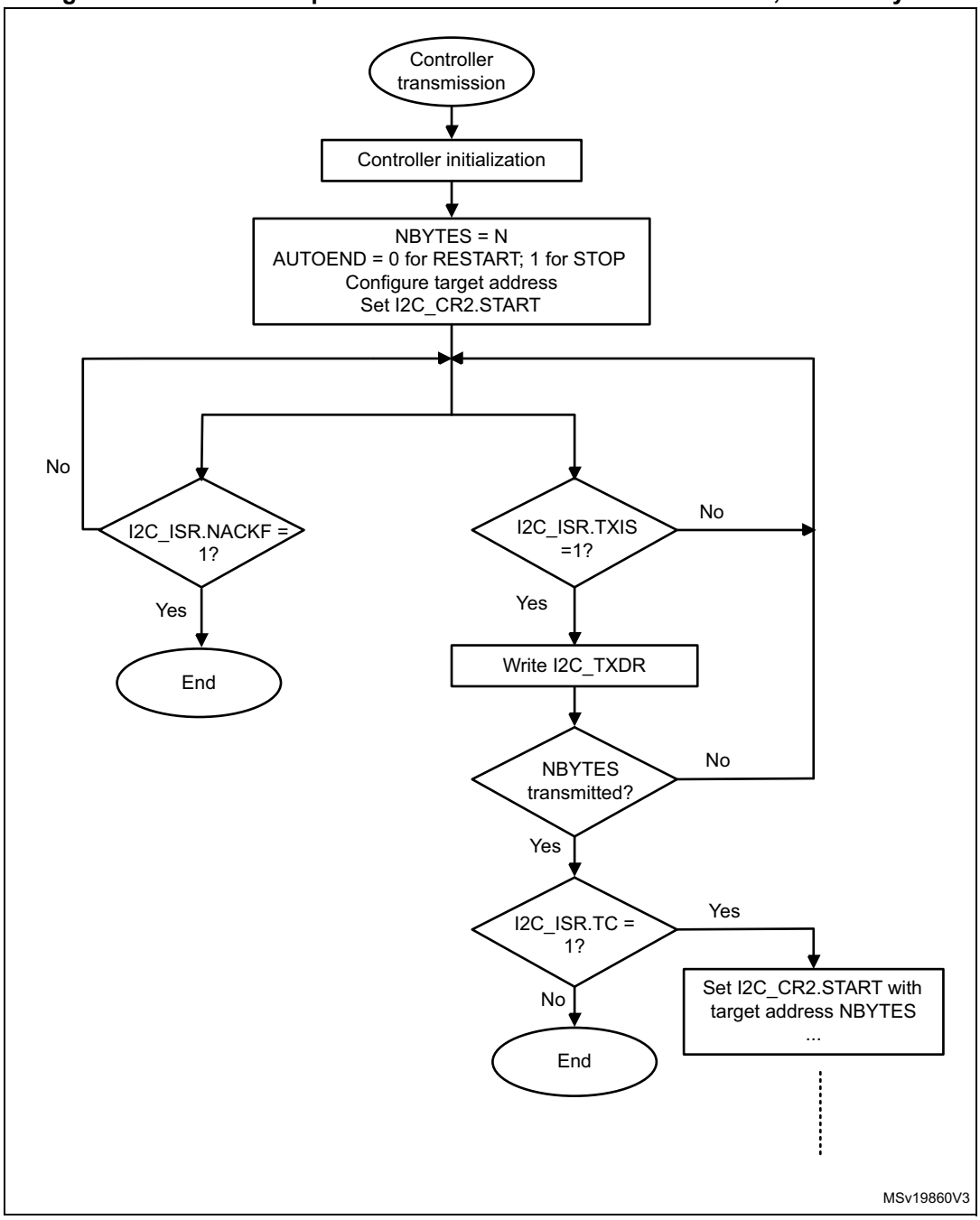


Figure 584. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

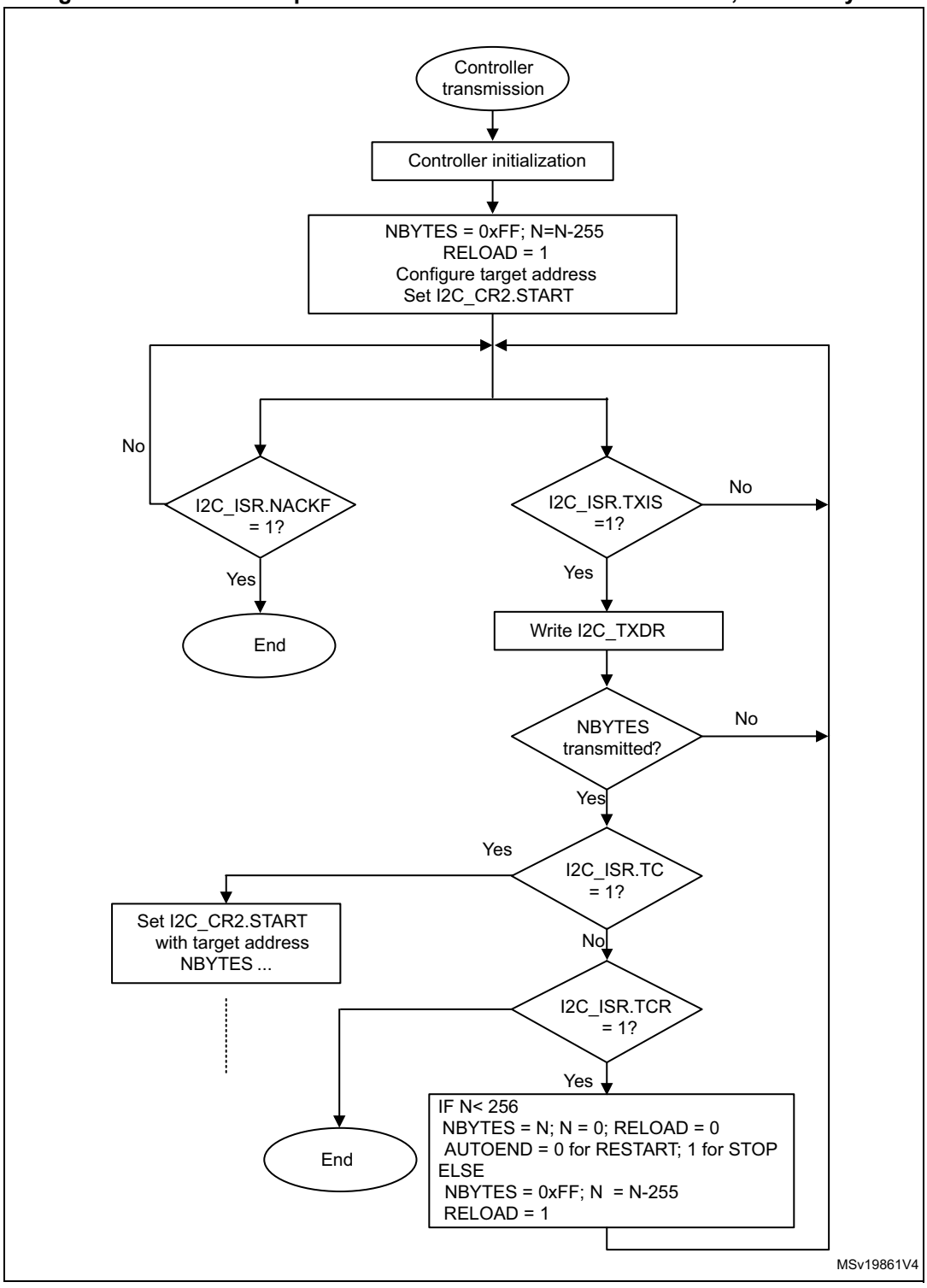
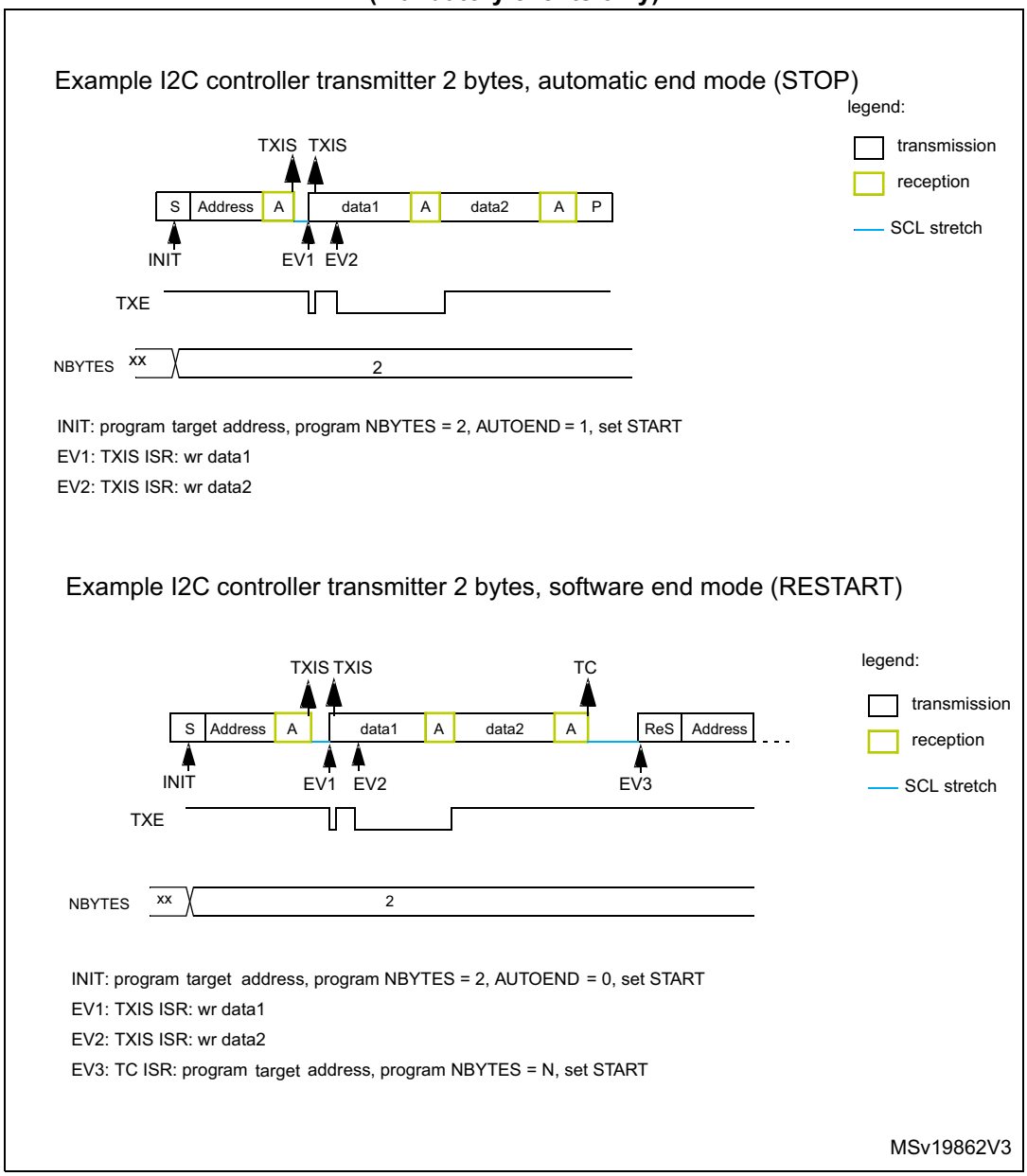


Figure 585. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 586. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

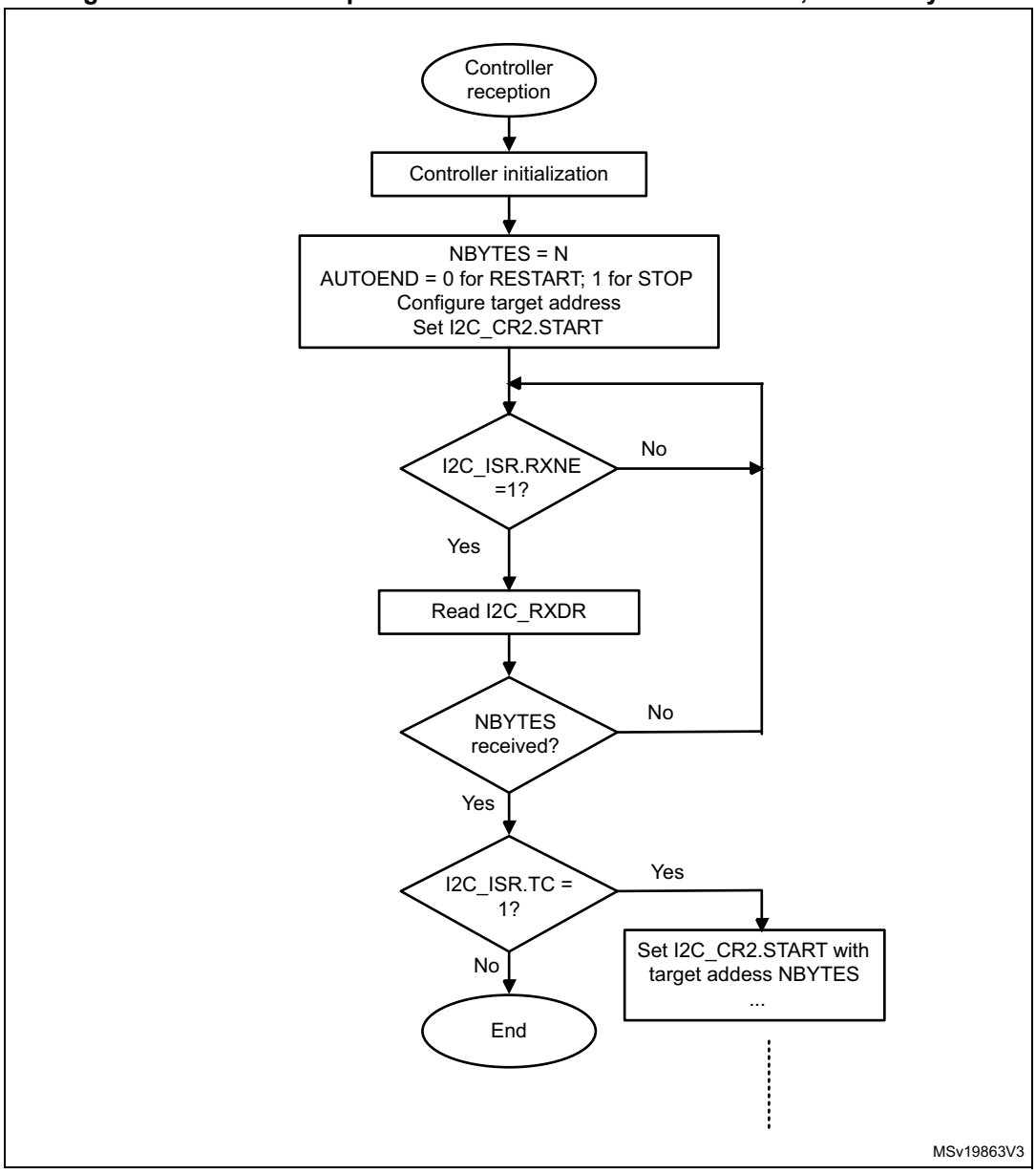


Figure 587. Transfer sequence flow for I2C controller receiver, N > 255 bytes

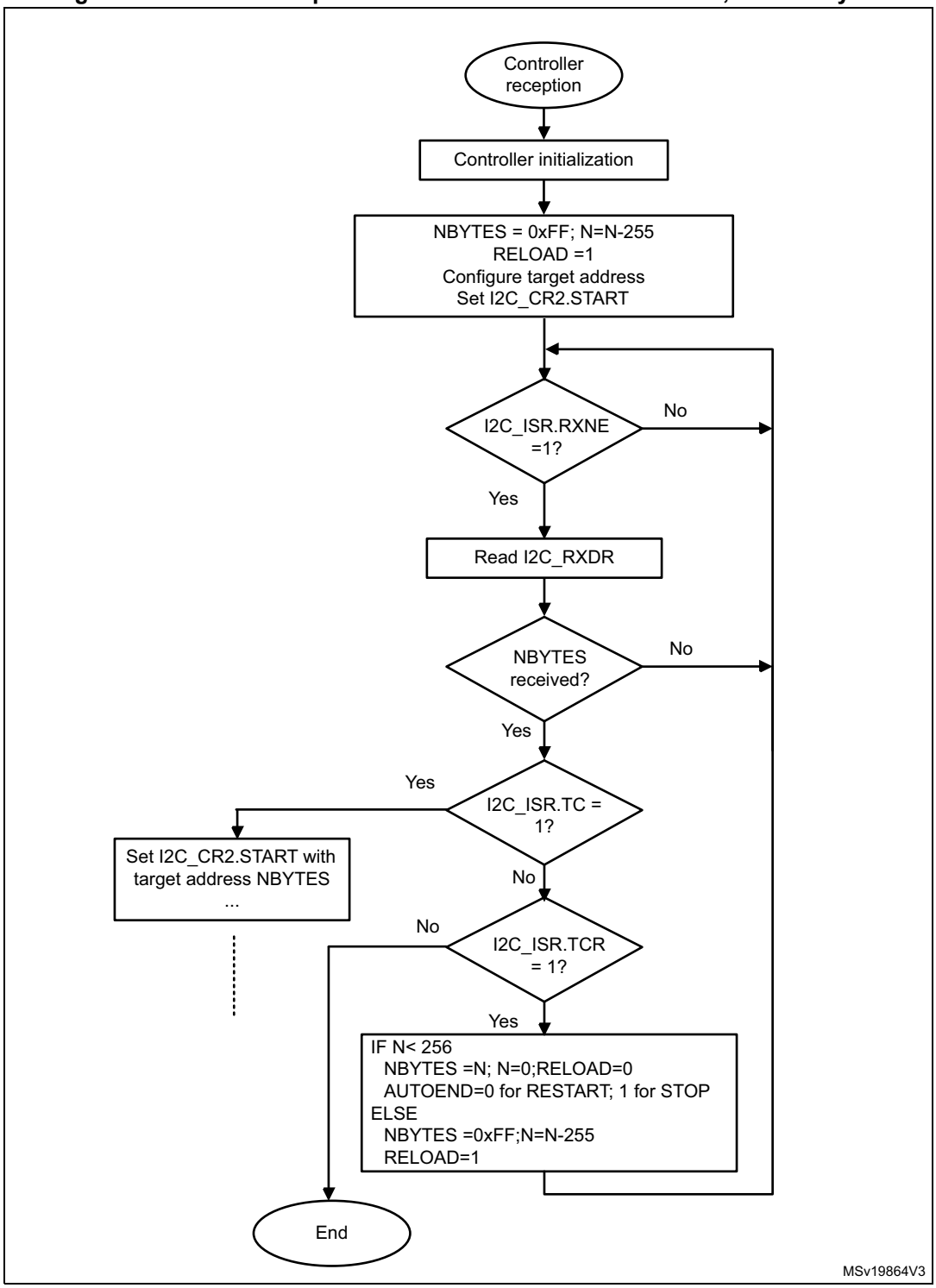
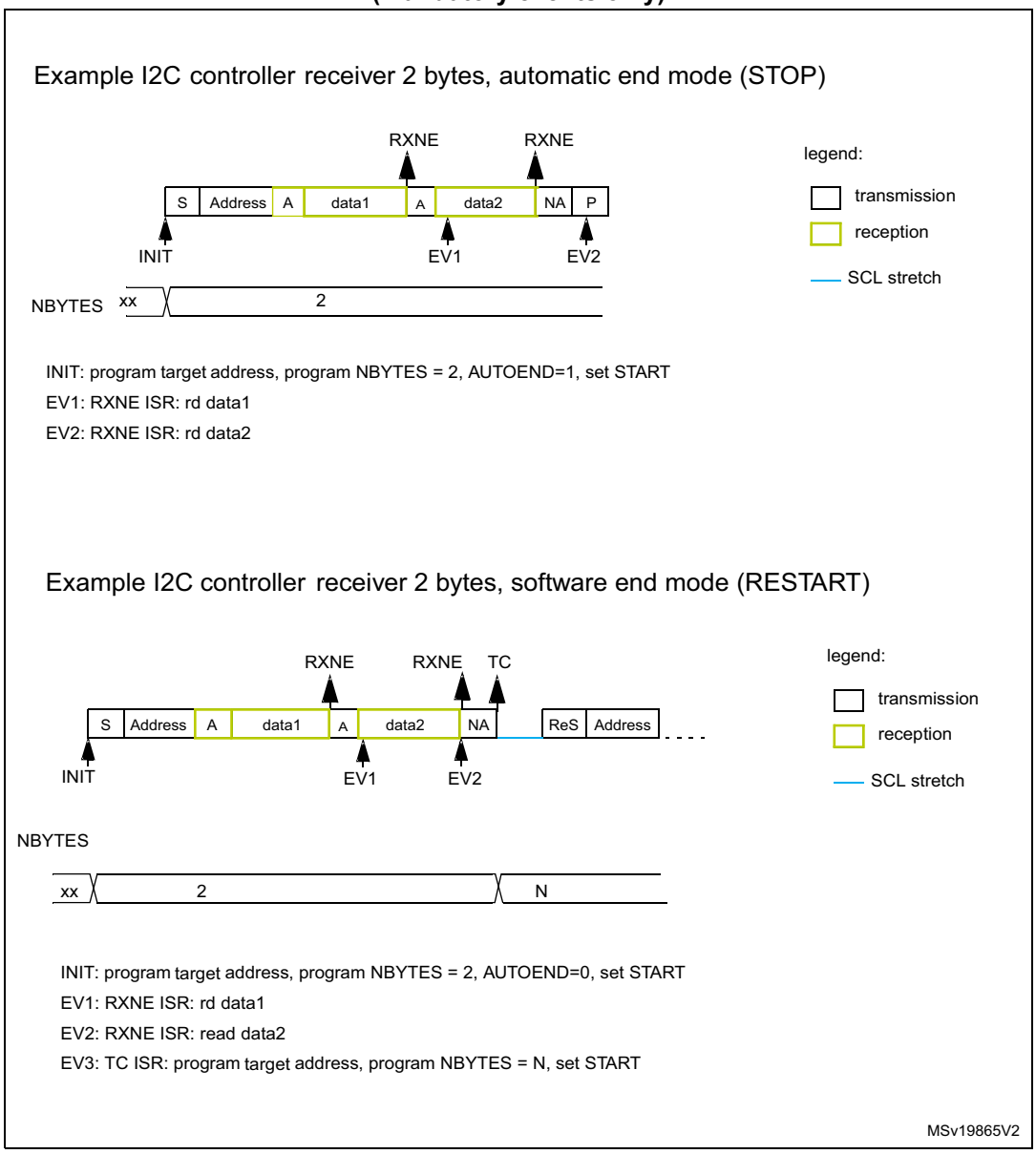


Figure 588. Transfer bus diagrams for I2C controller receiver (mandatory events only)



41.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 413. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 414. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

41.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

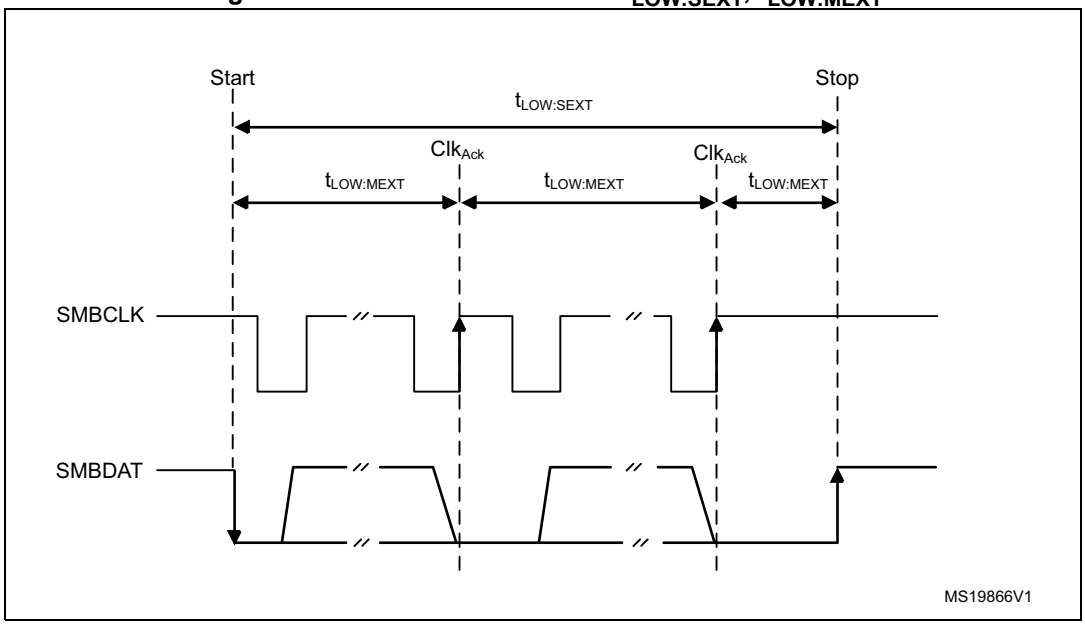
Table 415. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 589. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 41.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

41.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 416. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 417*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 418*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 419*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

41.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 417. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

Table 418. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

Table 419. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

41.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 41.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 590. Transfer sequence flow for SMBus target transmitter N bytes + PEC

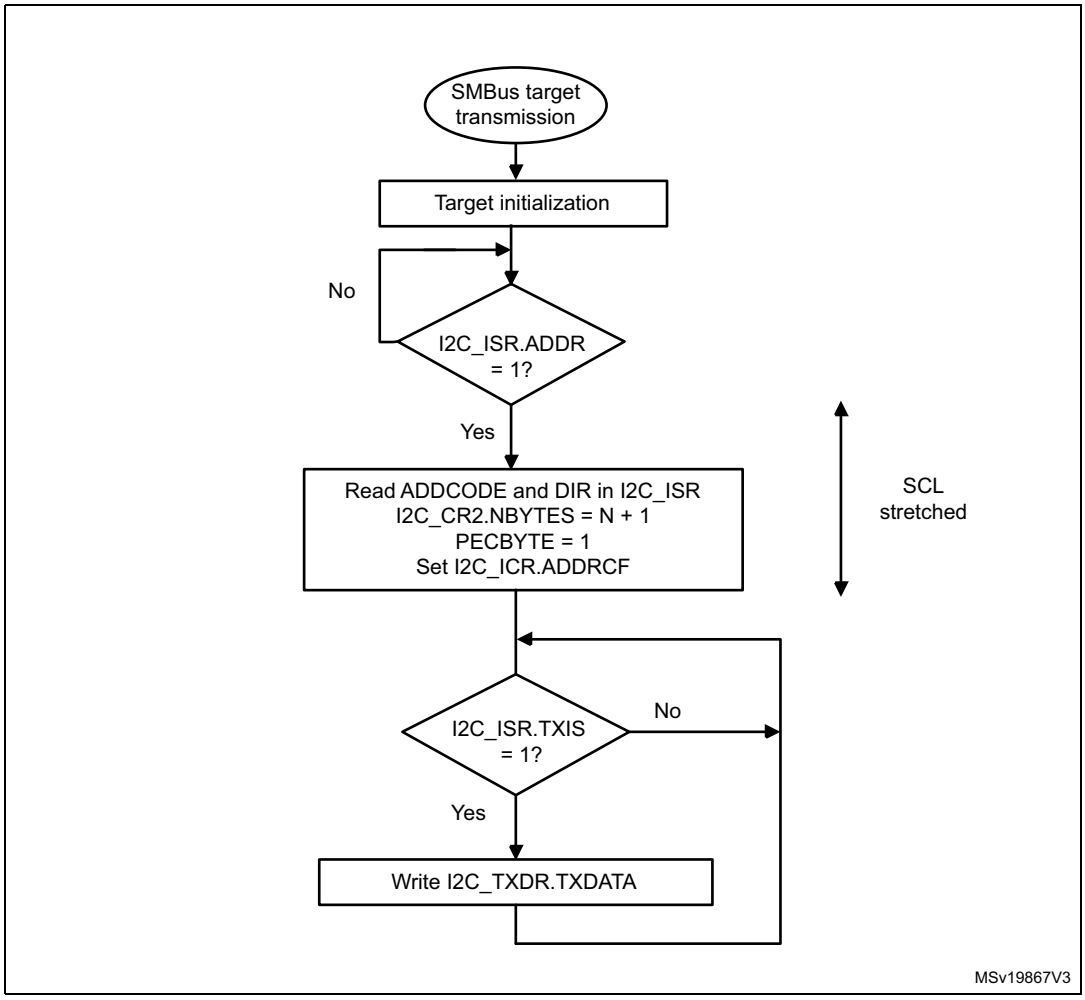
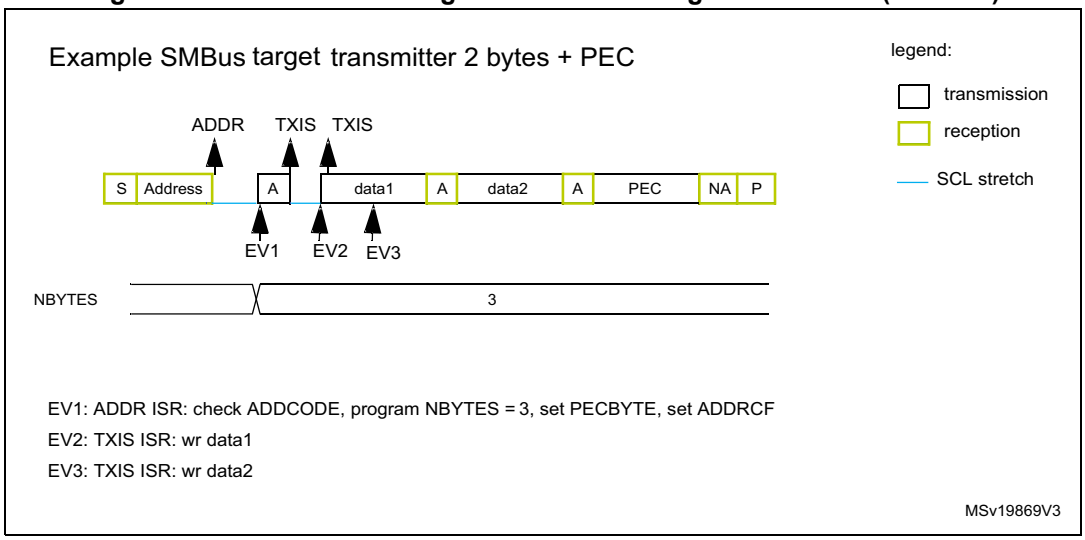


Figure 591. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 592. Transfer sequence flow for SMBus target receiver N bytes + PEC

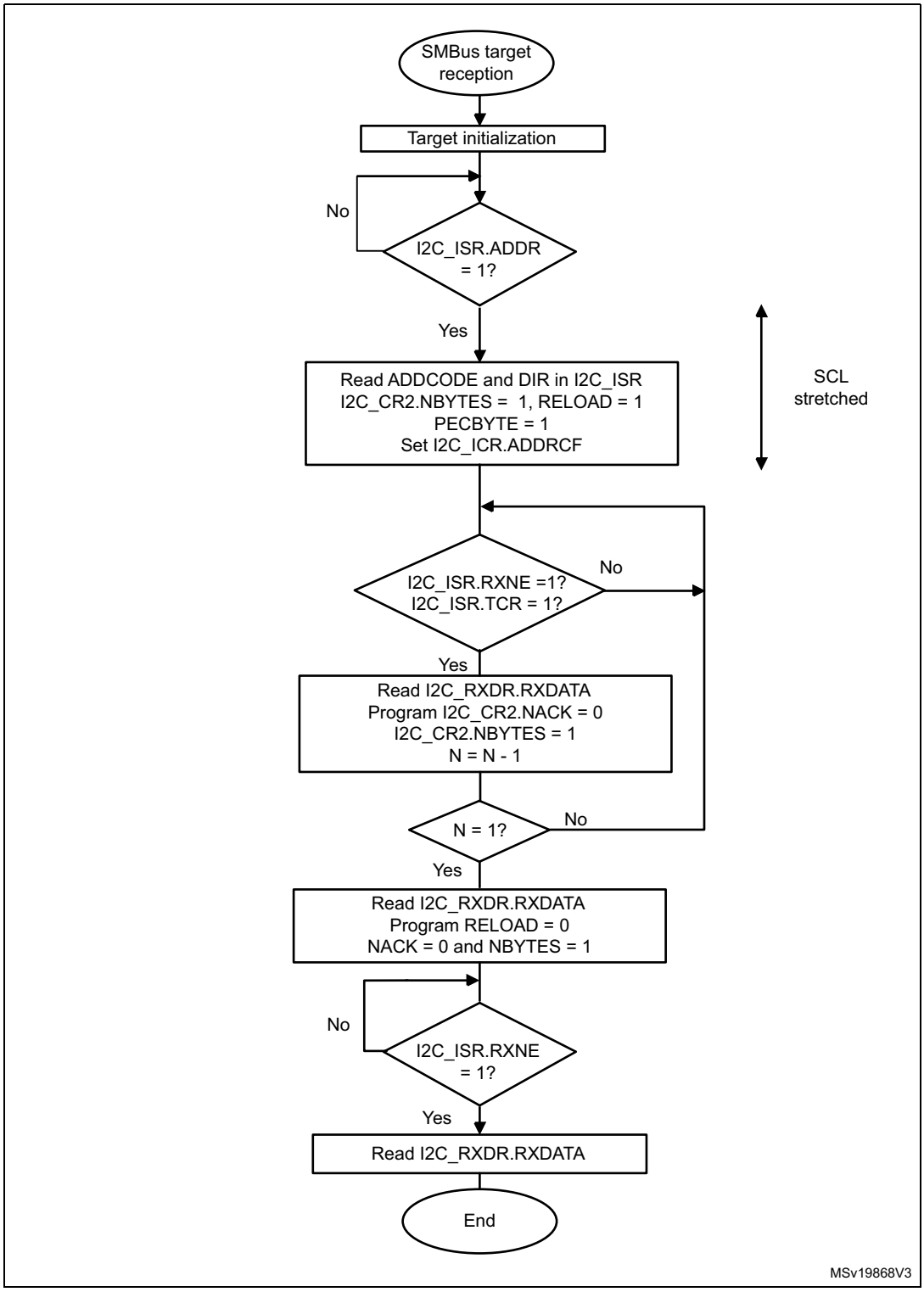
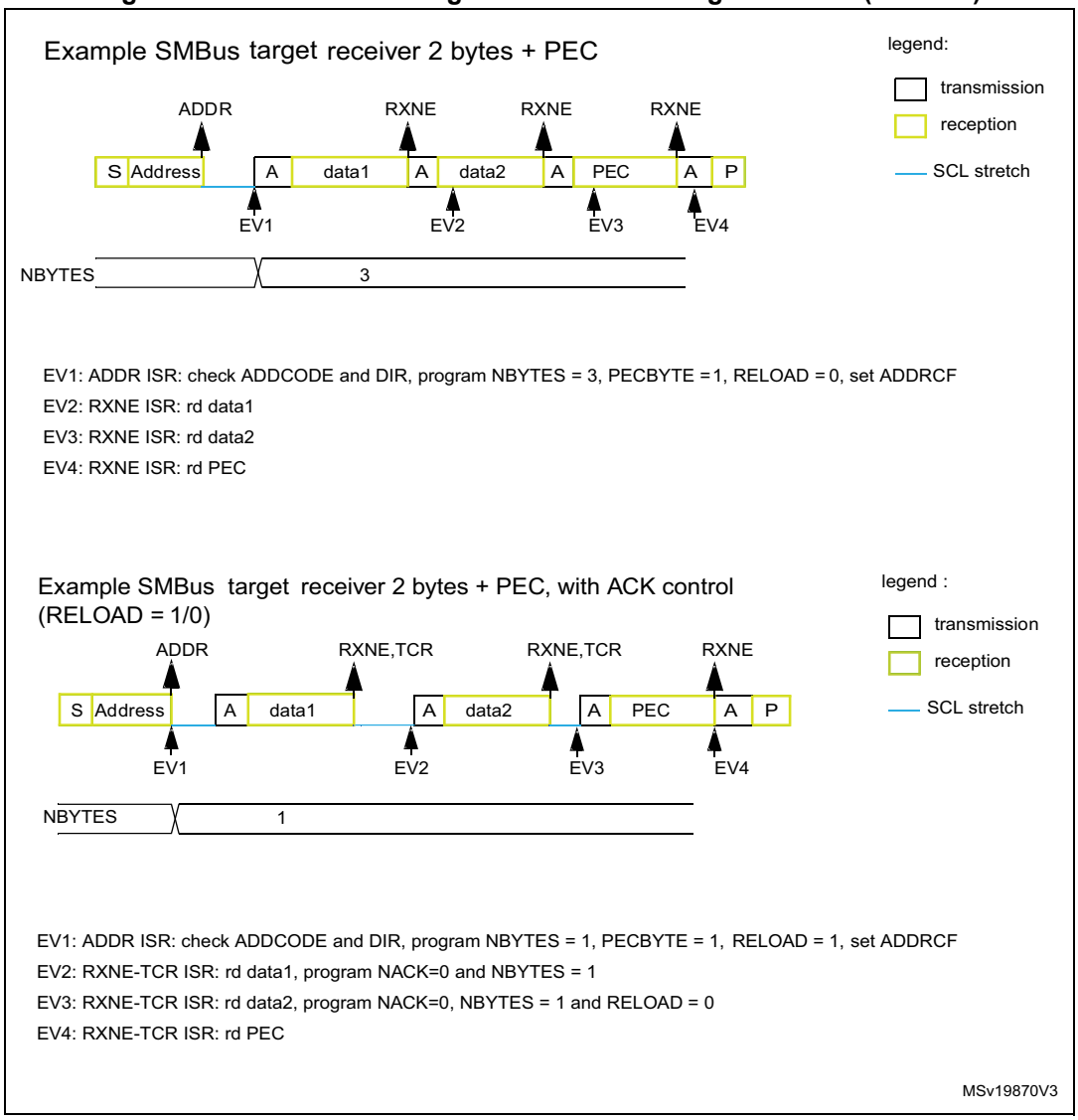


Figure 593. Bus transfer diagrams for SMBus target receiver (SBC = 1)



41.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 41.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

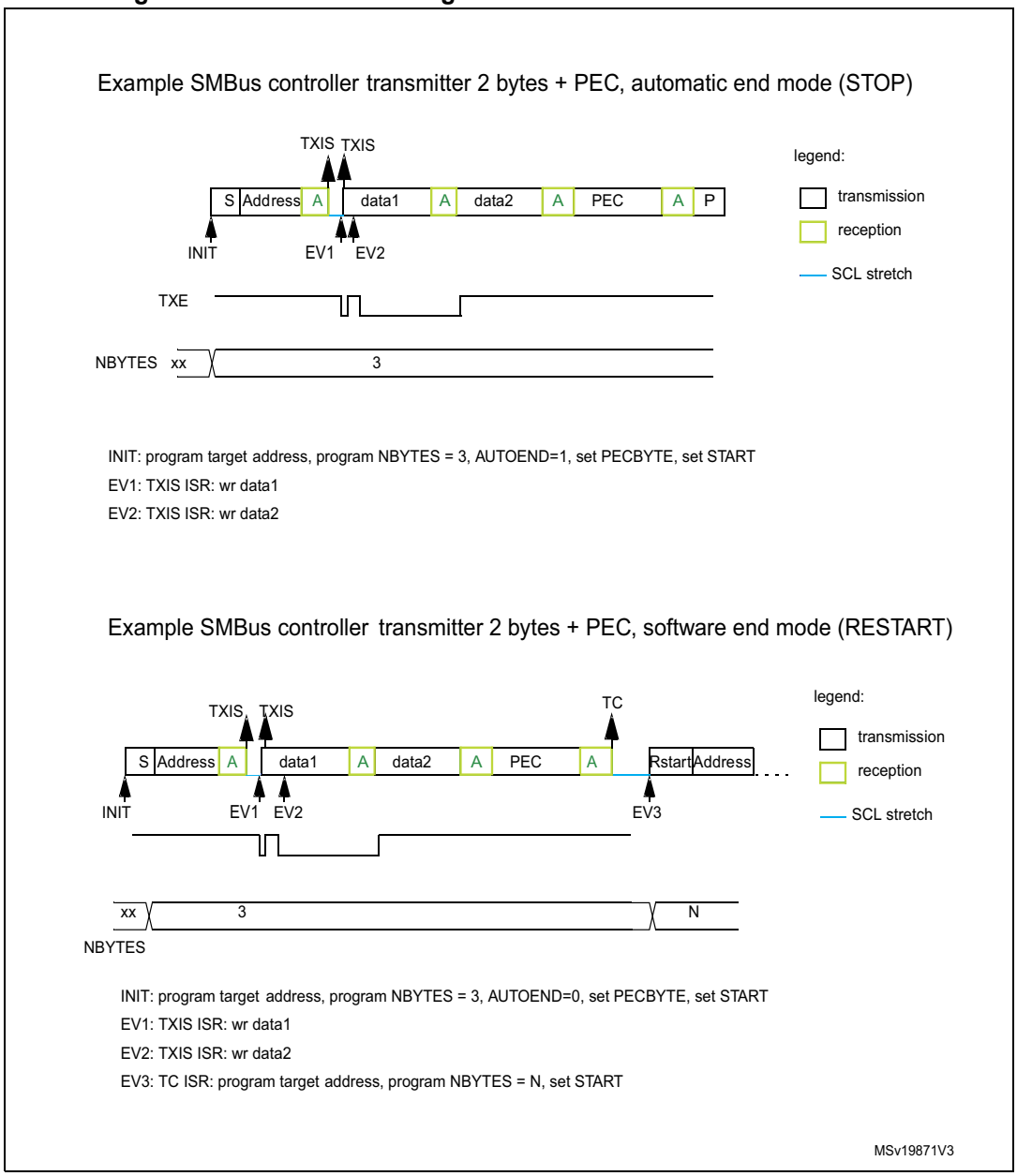
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 594. Bus transfer diagrams for SMBus controller transmitter



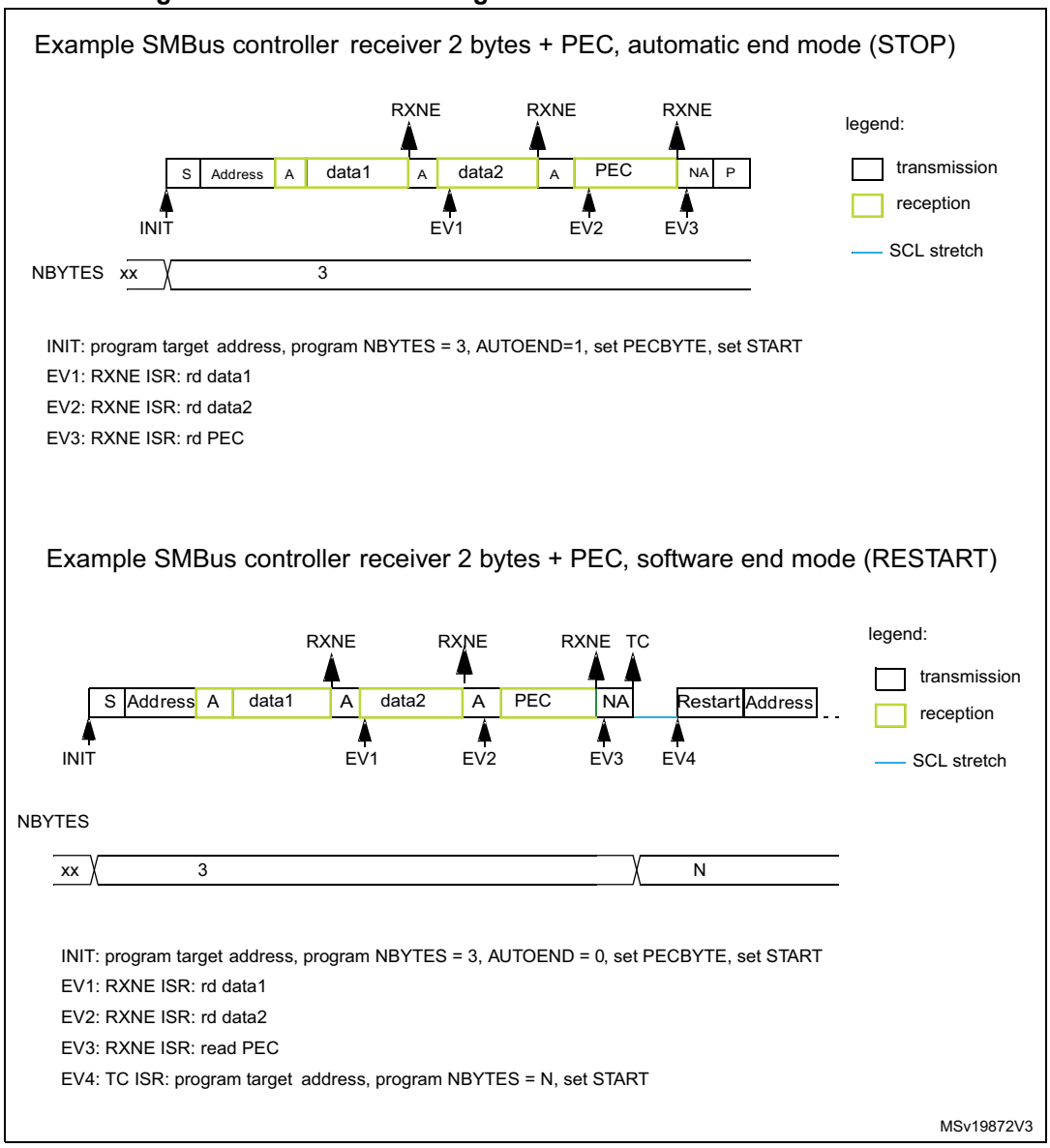
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 595. Bus transfer diagrams for SMBus controller receiver



41.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI oscillator and stretches SCL low until the oscillator wakes up.

HSI is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: The feature is only available when the HSI oscillator is selected as the I2C clock.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

41.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

41.5 I2C in low-power modes

Table 420. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 41.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

41.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 421. I2C interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop 0 modes	Exit Stop and Standby modes
I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
	Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
	STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
	Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
	Transfer complete	TC		Write START = 1 or STOP = 1			
	Address matched	ADDR	ADDRIE	Write ADDRCF = 1		Yes(1)	
	NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
	Arbitration loss	ARLO		Write ARLOCF = 1			
	Overrun/underrun	OVR		Write OVRCF = 1			
I2C_ERR	PEC error	PECERR	ERRIE	Write PECERRCF = 1	Yes	No	No
	Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF = 1			
	SMBus alert	ALERT		Write ALERTCF = 1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 41.3: I2C implementation*.

41.7 I2C DMA requests

41.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 14: Direct memory access controller (DMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

41.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 14: Direct memory access controller (DMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

41.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

41.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

41.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus drive enable

0: FMP I/O drive disabled

1: FMP I/O drive enabled

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).

1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable
0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable
0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable
0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.
Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control
This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}
Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generates an interrupt:
- arbitration loss (ARLO)
- bus error detection (BERR)
- overrun/underrun (OVR)
- timeout detection (TIMEOUT)
- PEC error detection (PECERR)
- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable
0: Transfer complete interrupt disabled
1: Transfer complete interrupt enabled
Note: Any of these events generates an interrupt:
Transfer complete (TC)
Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable
0: STOP detection (STOPF) interrupt disabled
1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disabled
1: Peripheral enabled
Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

41.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer

1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.

This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

41.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

41.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable
0: Own address 2 disabled. The received target address OA2 is NACKed.
1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.
Note: These bits can be written only when OA2EN = 0.
As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address
Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

41.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

41.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

– Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

41.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

41.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

41.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

41.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

41.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

41.9.12 I2C register map

The table below provides the I2C register map and the reset values.

Table 422. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

42 Improved inter-integrated circuit (I3C)

42.1 I3C introduction

The I3C interface handles communication with other devices such as sensor(s) and host processor.

An I3C bus is a two-wire, serial single-ended, multi-drop bus, intended to improve the legacy I^2C bus. The I3C interface improves the features of the I^2C interface (adding especially in-band interrupts, dynamic address assignment, energy-efficient data rates, power management) meanwhile preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C standard-mode (Sm), fast-mode (Fm), or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C SDR-only peripheral implements all the required features and many optional features of the MIPI® I3C specification v1.1 (see *Table 427: I3C peripheral controller/target features versus MIPI v1.1*). It can control all I3C bus-specific sequencing, protocol, arbitration, and timing.

After reset, the software can initialize the I3C peripheral as primary controller or as target:

- As I3C controller, the software can enable the I3C peripheral and initialize the I3C bus.
- As I3C target, the software can enable the I3C peripheral to operate on an I3C bus.

The I3C peripheral can be used with DMA, to off-load the CPU.

42.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in *Table 427*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages to legacy I^2C targets in Sm/Fm/Fm+
  - I3C SDR read/write private messages

  - I3C SDR broadcast CCC messages (see details in *Table 433*)
  - I3C SDR read/write direct CCC messages (see details in *Table 433*)
- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four I3C targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 42.3.2*):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 42.3.2*):
  - On a reset pattern
  - On a missed start on the serial bus
- Multi-clock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 42.6.2*)

42.3 I3C implementation

42.3.1 I3C instantiation

There is a single I3C instance in the device.

42.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 424*. For more details about the wake-up capabilities, refer to *Section 42.13*.

Table 424. I3C wake-up

Wake-up
From Stop 0 mode

42.3.3 I3C FIFOs

The FIFOs are implemented as defined in *Table 425*.

Table 425. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

42.3.4 I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

42.3.5 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 426*.

Table 426. I3C interrupt(s)

Signal name	Signal type	Description
i3c_err_it	O	Error interrupt line
i3c_evt_it	O	Event interrupt line

42.3.6 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 427*.

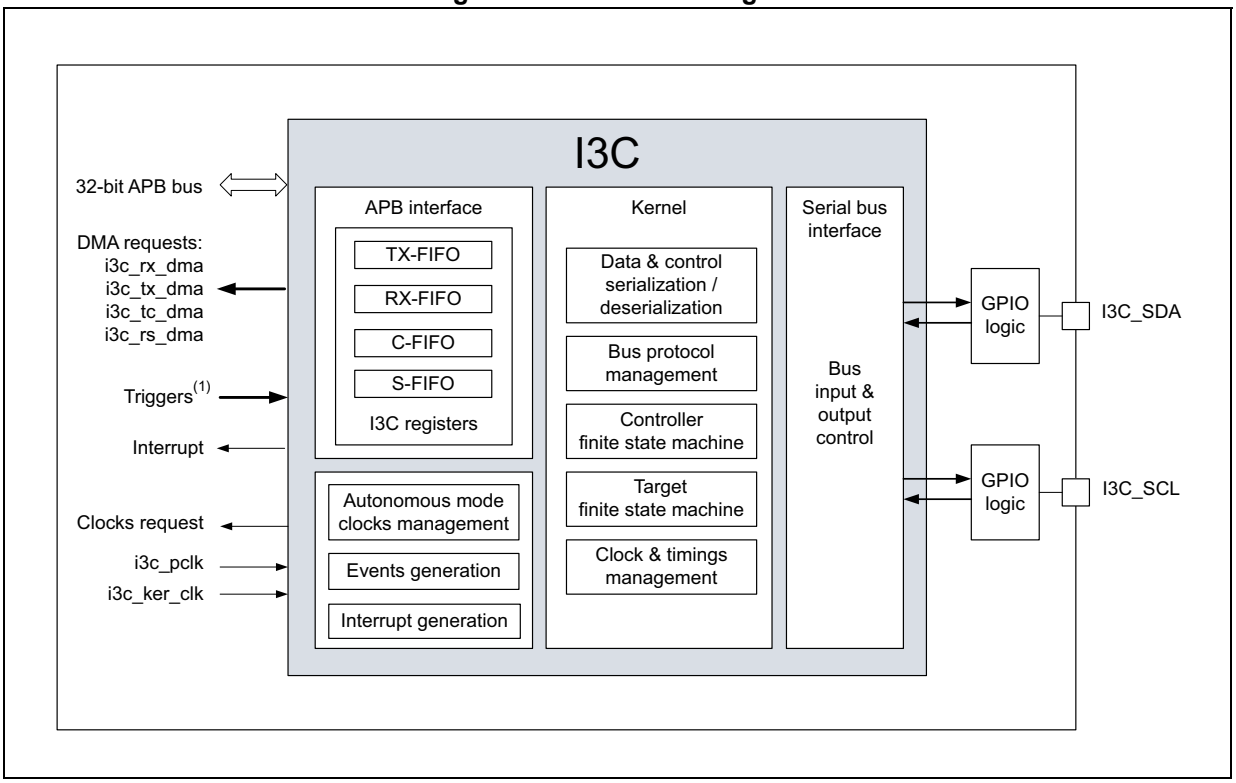
Table 427. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Sm/Fm/Fm+)	X	X	-	Mandatory when controller if the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	Supported broadcast ENTDAA when controller and target. Supported broadcast SETAASA and direct SETDASA when controller.
7-bit static I^2C address	X	X	-	No support of the I3C peripheral as target on an I^2C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 433* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller. Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

42.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 596*.

Figure 596. I3C block diagram



1. Refer to *Section 42.3.4: I3C triggers*.

42.5 I3C pins and internal signals

Table 428. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	Serial bus data line
I3C_SCL	Input/output	Serial bus clock line

Note: During initialization, the selected GPIOs for I3C_SCL and I3C_SDA should be configured as alternate function and no pull.

Table 429. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request

Table 429. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller

1. Refer to *Section 42.3.5: I3C interrupt(s)*.

42.6 I3C reset and clocks

42.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 433* and *Section 42.16.17* for more details.

This reset interrupt notification can be used to wake up from a low power mode.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

42.6.2 I3C clocks and requirements

As indicated in *Figure 596*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE} for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to section Reset and clock control (RCC) for more details).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

42.7 I3C peripheral state and programming

42.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 597* and *Figure 598*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different initialized configuration registers, and switches to the (enabled and) idle state. The peripheral is initialized as I3C target or I3C controller.
  - As I3C controller, then software can enable the I3C peripheral and initialize the I3C bus.
  - As I3C target, then the software can enable the I3C peripheral to operate on an I3C bus.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

    As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

    As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF).

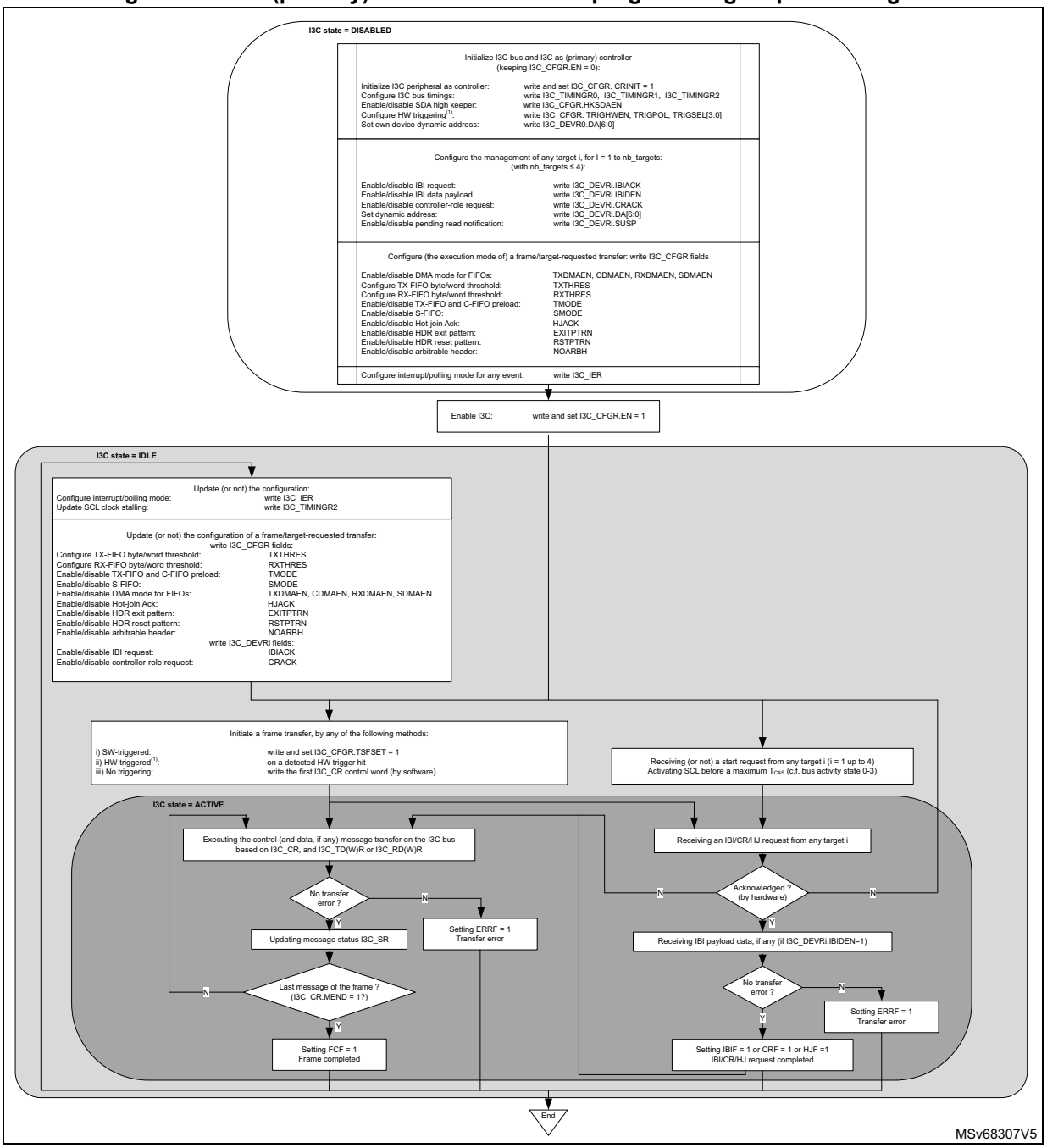
Note: The software can disable the I3C peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

Note: The software can disable the generation of the FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 42.3), to stream a series of frames.

42.7.2 I3C controller state and programming sequence

Figure 597 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 597. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 42.3.4: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:

  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG_H_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG_OD_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
  - FCFDIS: FCF generation enable/disable
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 42.9*:

- *Figure 609: IBI transfer, as controller/target*
- *Figure 610: Hot-join request transfer, as controller/target*
- *Figure 611: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 42.9*:

- *Figure 599: I3C CCC messages, as controller*
- *Figure 600: I3C broadcast ENTDAA CCC, as controller*
- *Figure 601: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 606: I3C private read/write messages, as controller*
- *Figure 608: Legacy I2C read/write messages, as controller*

Figure 597 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 42.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 42.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH
  - FCFDIS

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

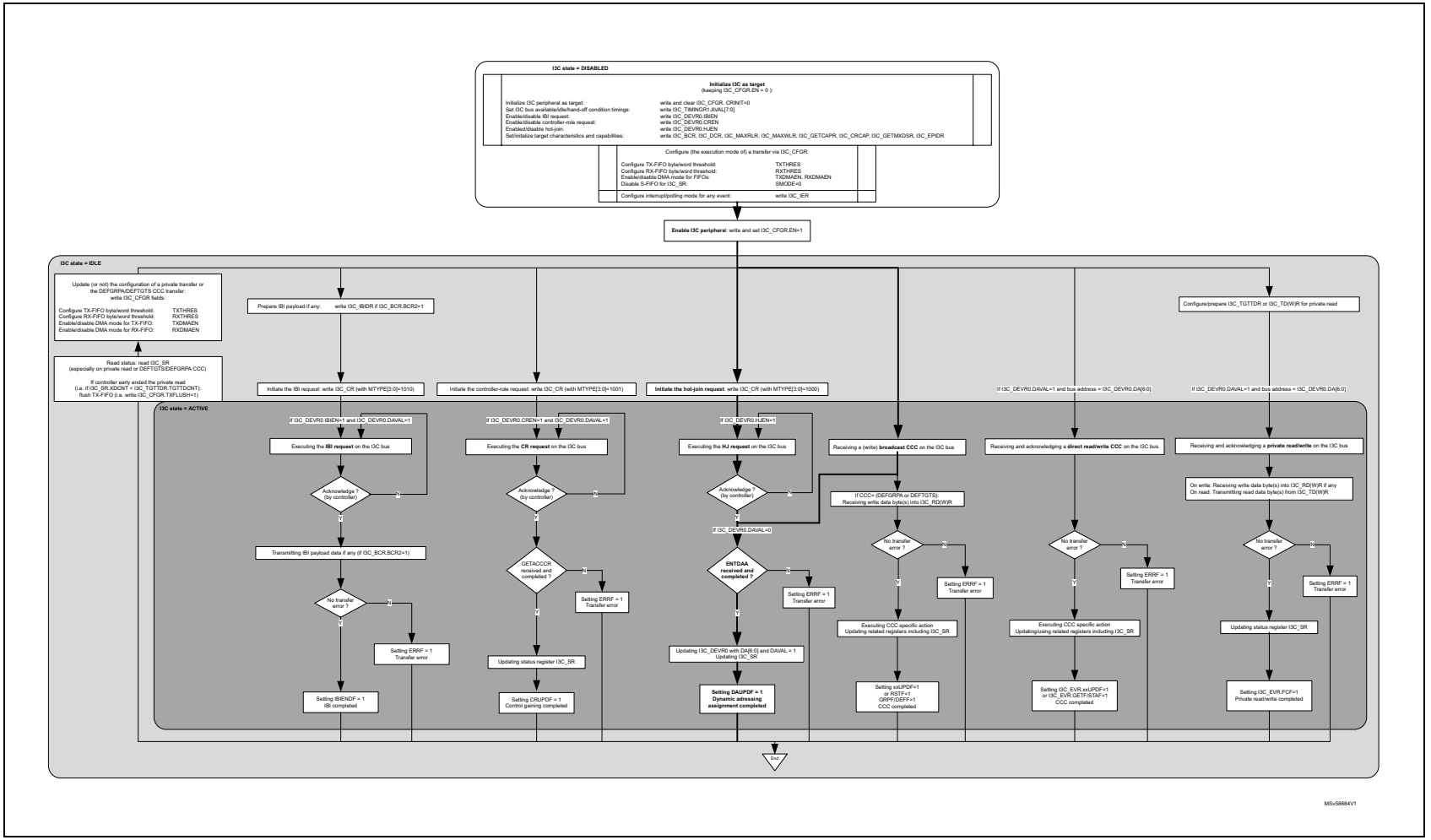
The registers usage versus the I3C peripheral role as controller is summarized in *Section 42.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 431*.

42.7.3 I3C target state and programming sequence

Figure 598 illustrates the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 598. I3C target state and programing sequence diagram



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  - Bus available condition time (t_{AVAL}) for IBI or controller-role request
  - Bus idle condition time (t_{IDLE}) for hot-join request
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - *I3C bus characteristics register (I3C_BCR)*
  - *I3C device characteristics register (I3C_DCR)*
  - *I3C maximum read length register (I3C_MAXRLR)*
  - *I3C maximum write length register (I3C_MAXWLR)*
  - *I3C get capability register (I3C_GETCAPR)*
  - *I3C controller-role capability register (I3C_CRCAPR)*
  - *I3C get max data speed register (I3C_GETMXDSR)*
  - *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message on the I3C bus

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 42.9*:

- *Figure 602: I3C CCC messages, as target*
- *Figure 603: I3C broadcast ENTDAA CCC, as target*
- *Figure 604: I3C broadcast DEFTGTS CCC, as target*
- *Figure 605: I3C broadcast DEFGRPA CCC, as target*
- *Figure 607: I3C private read/write messages, as target*

Figure 598 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 42.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer on the I3C bus

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 42.9*:

- *Figure 609: IBI transfer, as controller/target*
- *Figure 610: Hot-join request transfer, as controller/target*
- *Figure 611: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC)

The registers usage vs. the I3C peripheral role as target is summarized in *Section 42.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 432*.

42.8 I3C registers and programming

42.8.1 I3C register set, as controller/target

Table 430 lists the registers and their usage versus the I3C peripheral role.

Table 430. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_MISR	X	X

Table 430. I3C register usage (continued)

Register	Used as controller	Used as target
I3C_DEVR0	X	X
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

42.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 431* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 431. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X

Table 431. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CFGR	X	CRINIT HKSDAEN EN(1)	Write: any used field except {CRINIT, HKSDAEN}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) FCFDIS CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 431. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_MISR	X	-	-	Read (controller-role fields): HJMIS CRMIS IBIMIS FCMIS ERRMIS RXTGTENDMIS RXFNEMIS TXFNFMIS SFNEMIS CFNFMIS
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			

Table 431. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

42.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 432* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 432. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 432. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_MISR	X			Read (target-role fields): GRPMIS DEFMIS INTUPDMIS ASUPDMIS RSTMIS MRLUPDMIS MWLUPDMIS DAUPDMIS STAMIS GETMIS WKPMIS CRUPDMIS IBIENDMIS ERRMIS FCMIS RXFNEMIS TXFNFMIS

Table 432. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

42.9 I3C bus transfers and programming

42.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 433*.

Table 433. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF if DAVAL = 0	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 604*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	-
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X	-	-
SETAASA	0x29		No defining/sub-command byte	No data byte	X	-	-
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 605*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 433. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	-	-
SETNEWDA	0x88			With one data byte	X	X, DAUPDF if DAVAL = 1	Update I3C_DEVR0: DA[6:0] (and keep DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 42.16.20*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 42.16.19*.

Table 433. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 42.16.29*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 42.16.24*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 42.16.25*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 42.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 42.16.28*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 42.16.26*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 42.16.27*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

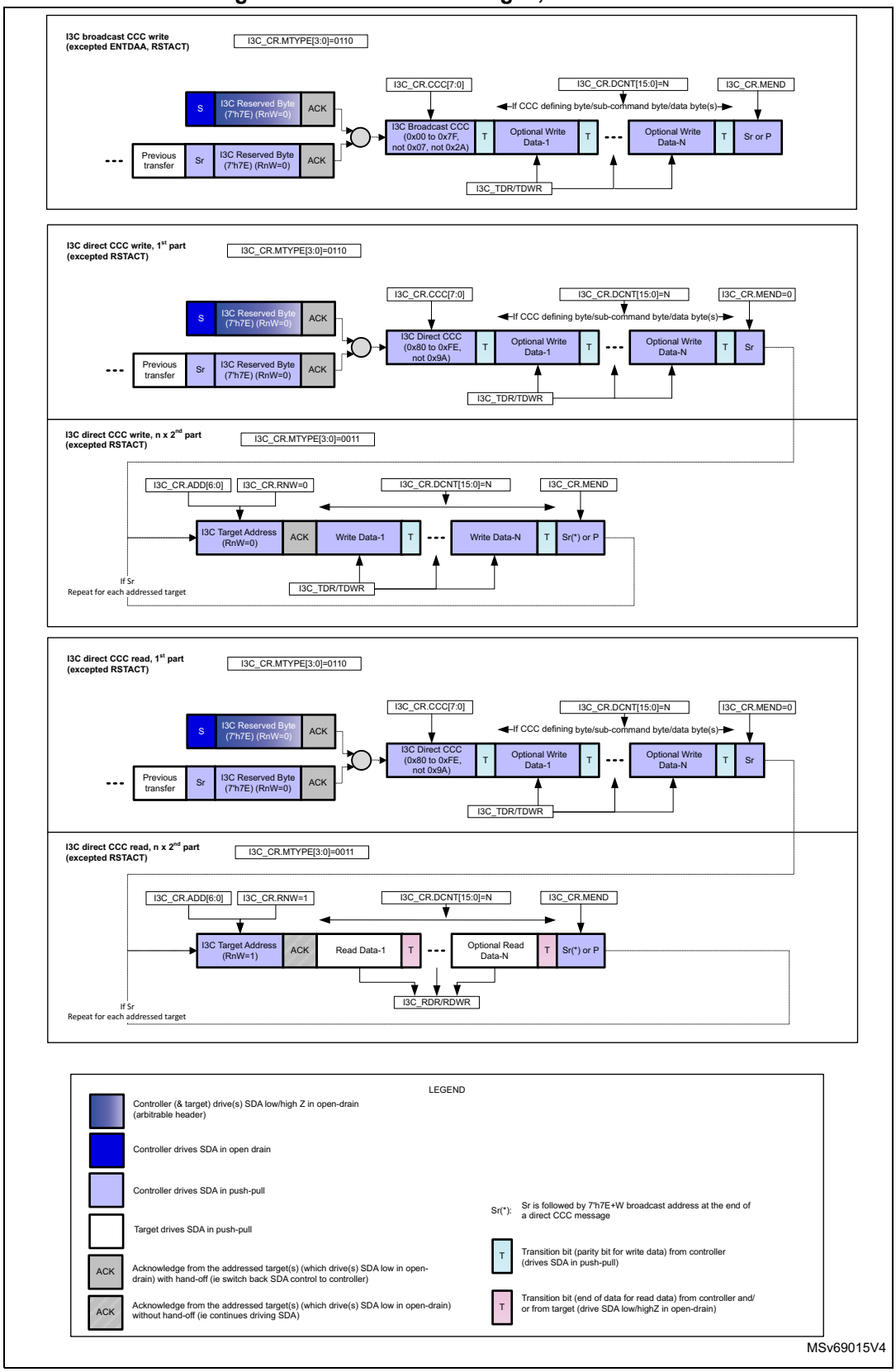
Table 433. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

42.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 599 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

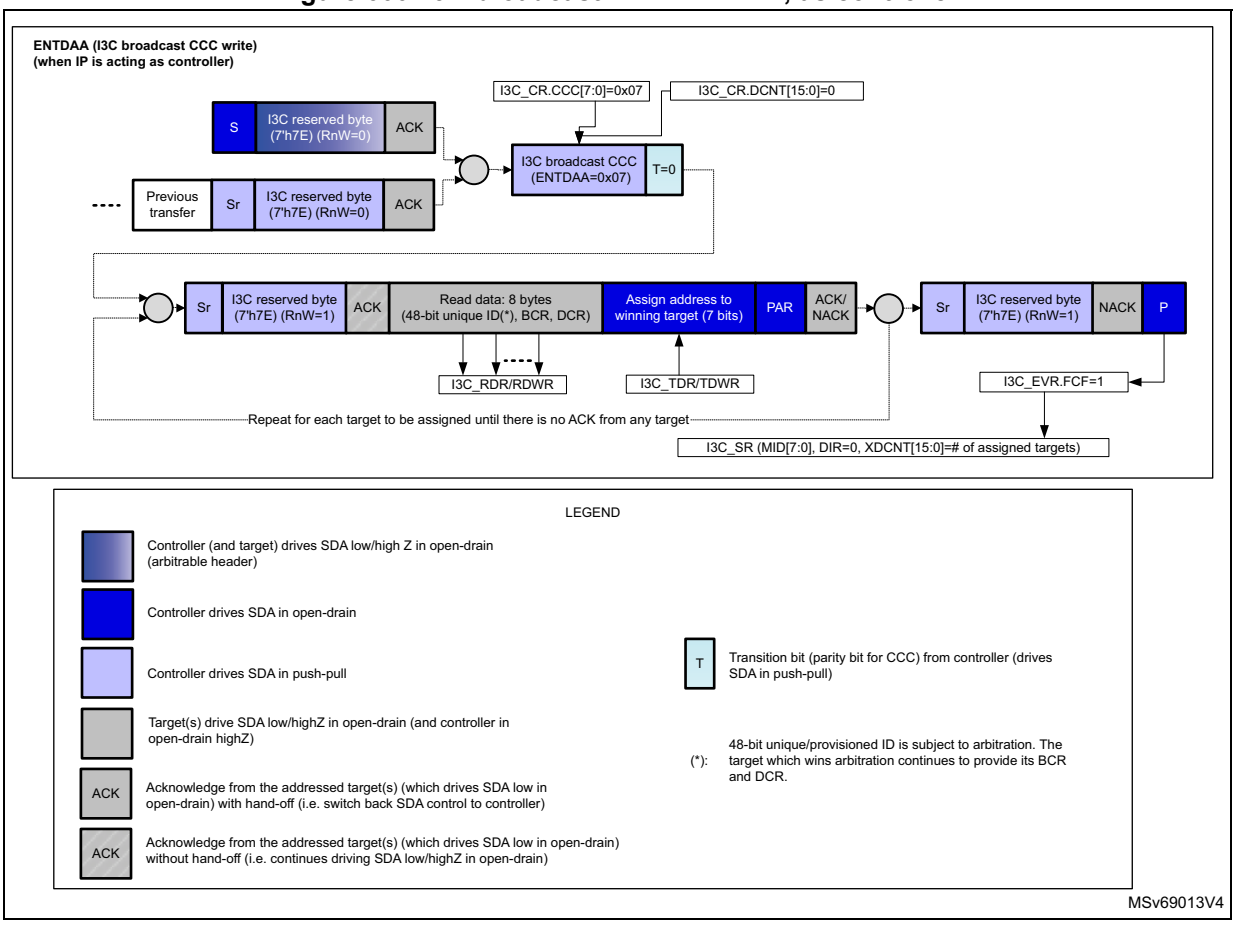
Figure 599. I3C CCC messages, as controller



42.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 600 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

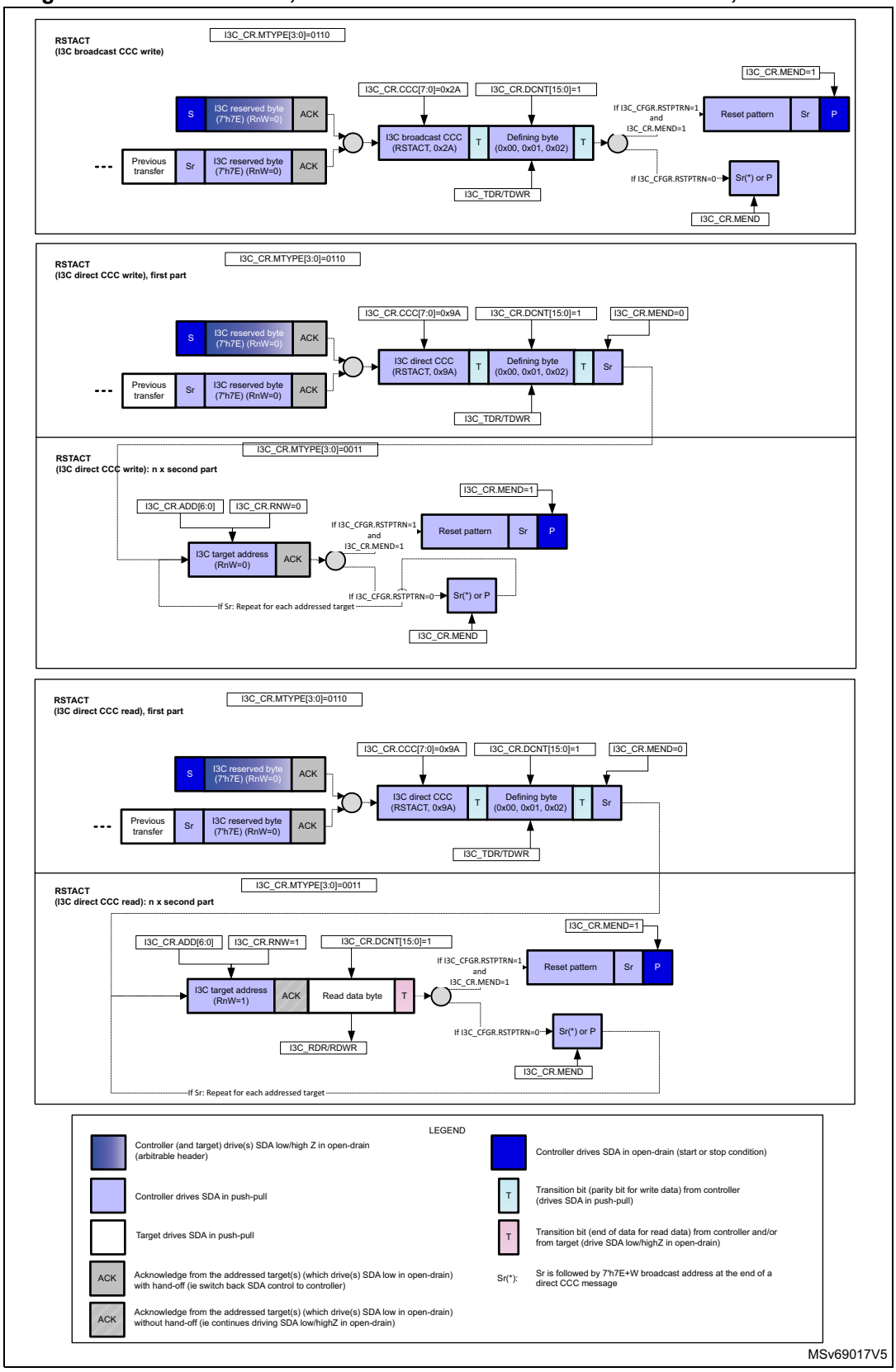
Figure 600. I3C broadcast ENTDAA CCC, as controller



42.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 601 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

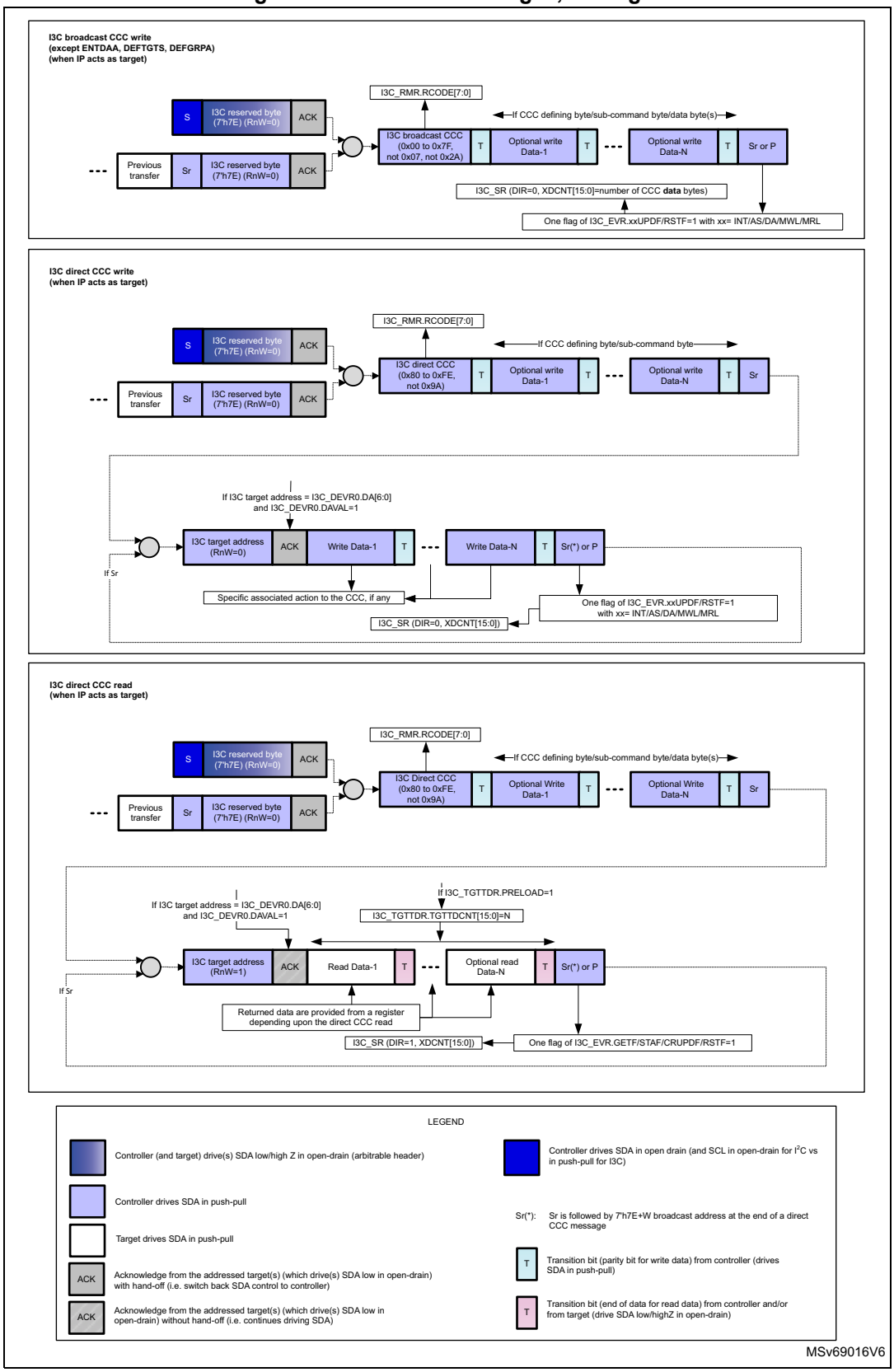
Figure 601. I3C broadcast, direct read and direct write RSTACT CCC, as controller



42.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 602 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

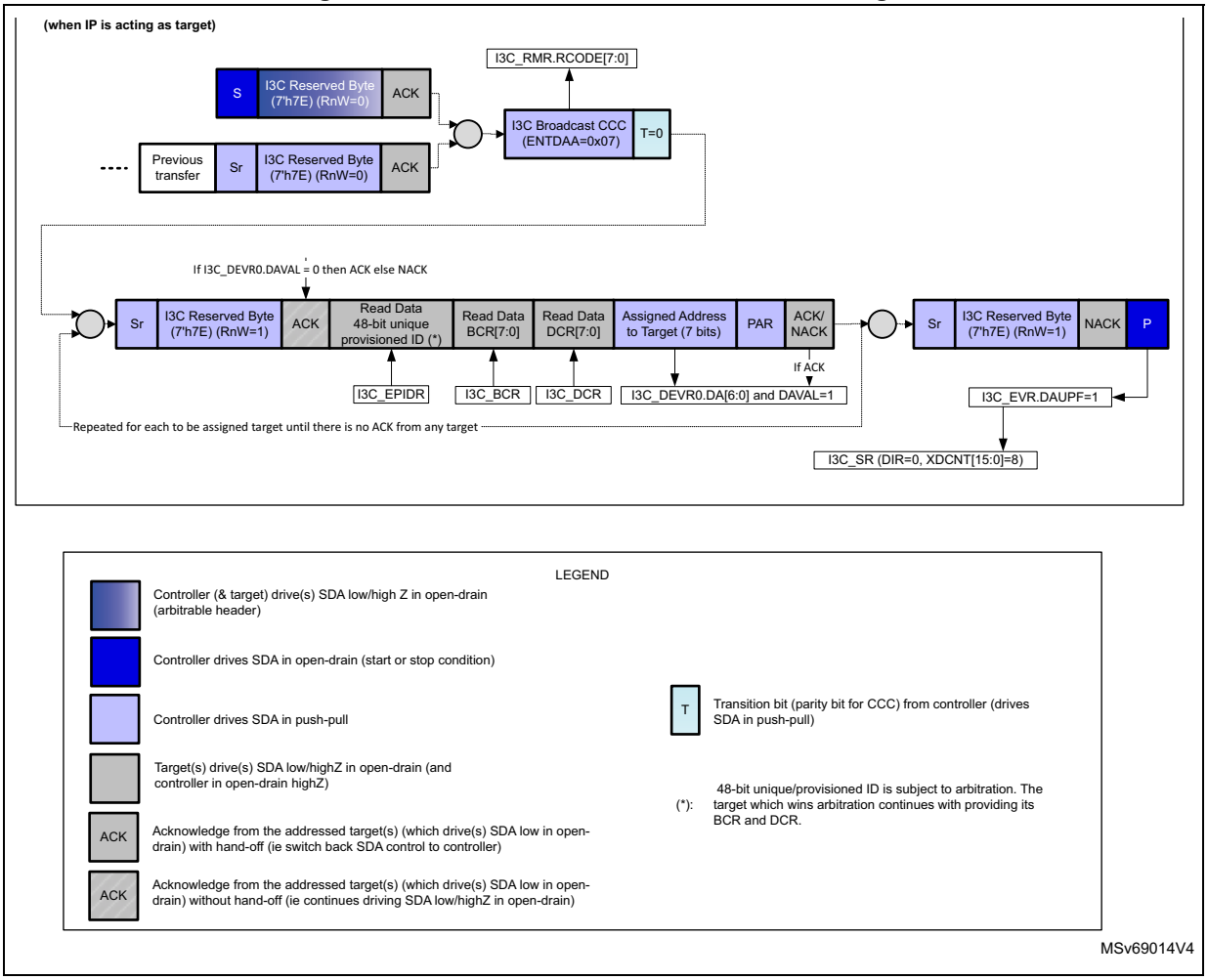
Figure 602. I3C CCC messages, as target



42.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 603 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

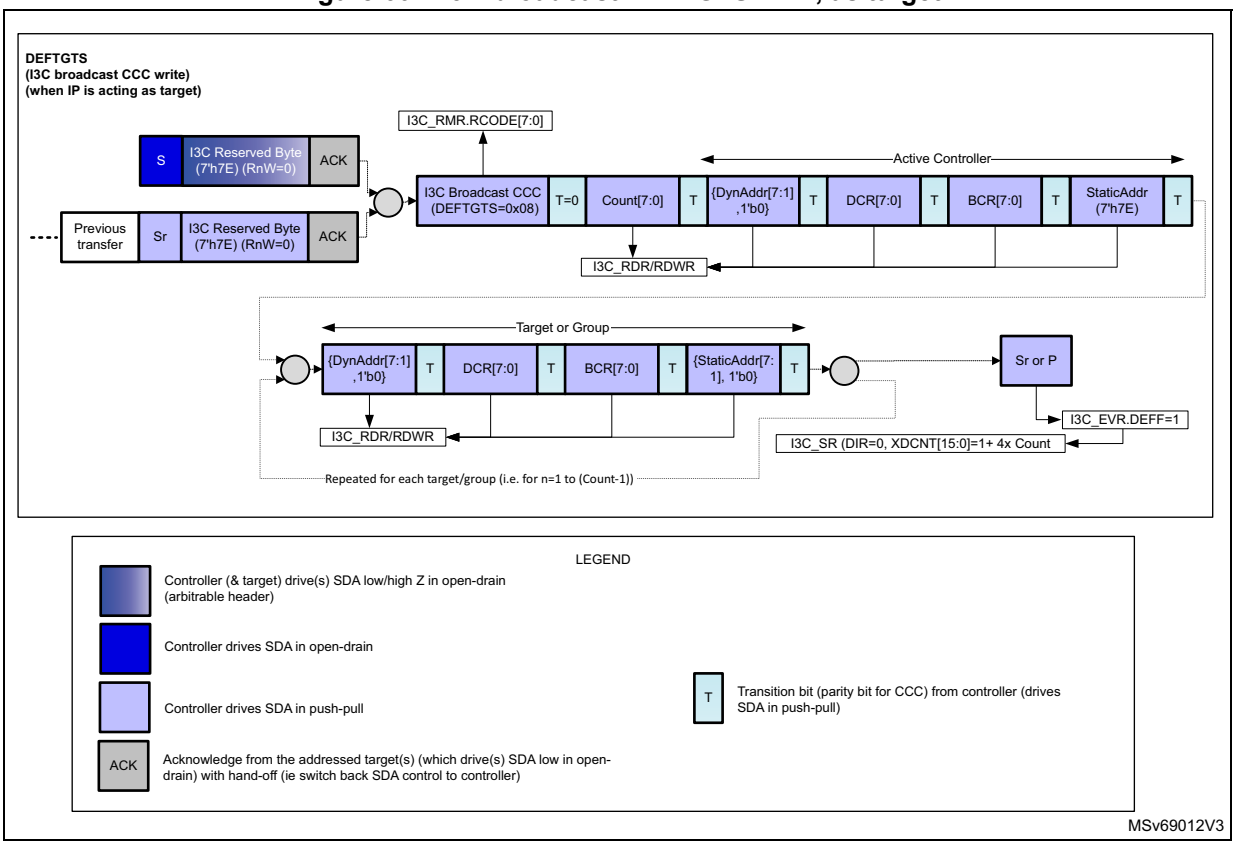
Figure 603. I3C broadcast ENTDAA CCC, as target



42.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 604 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

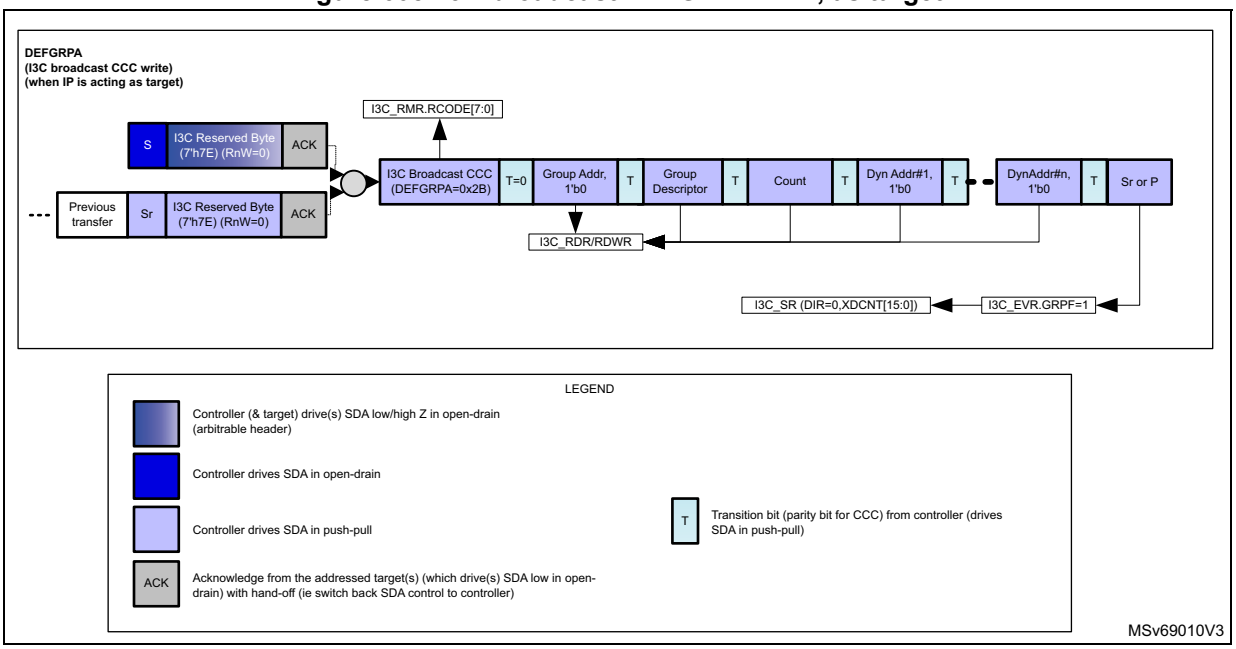
Figure 604. I3C broadcast DEFTGTS CCC, as target



42.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 605 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 605. I3C broadcast DEFGRPA CCC, as target



42.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

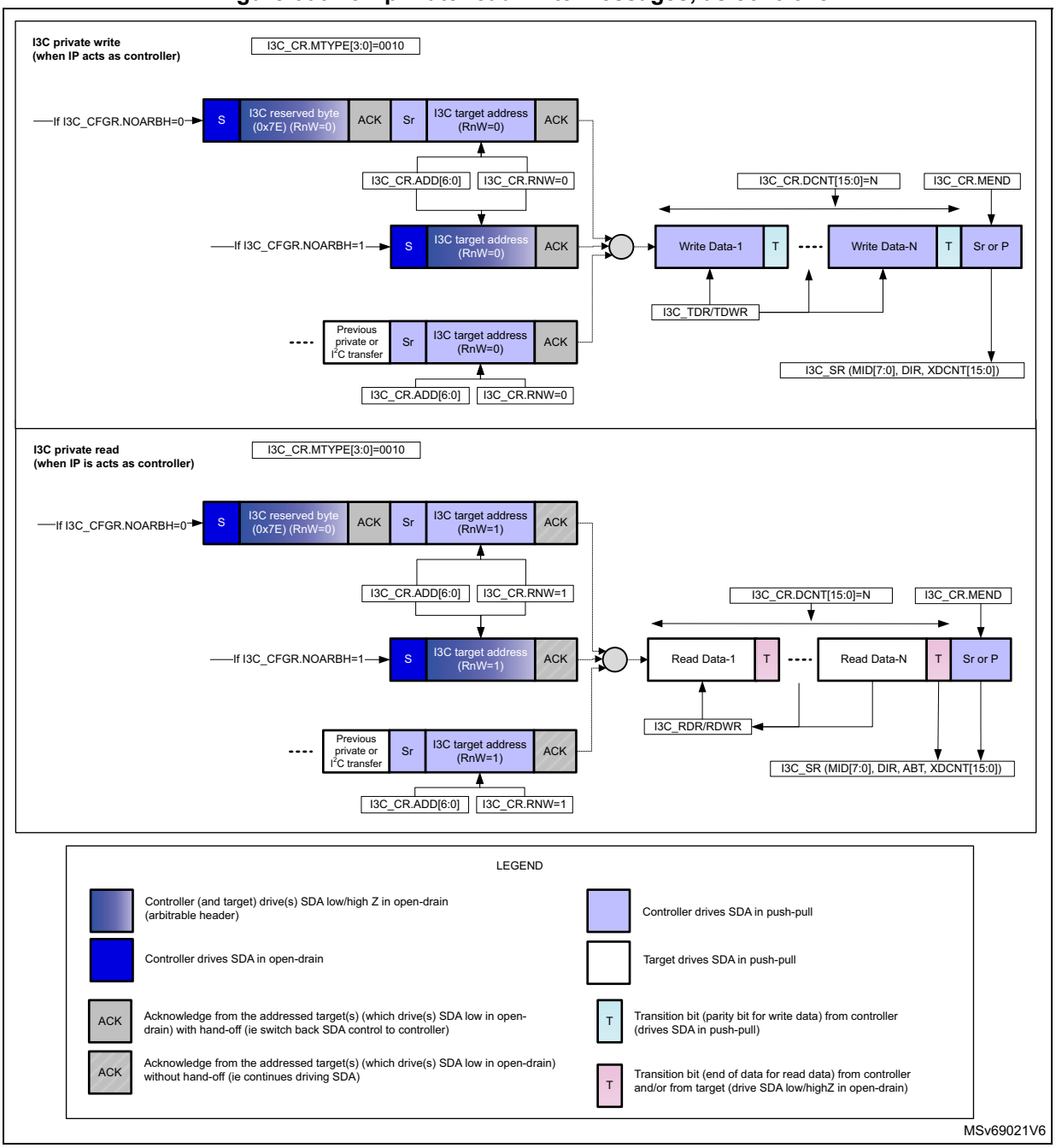
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

42.9.10 I3C private read/write transfer, as controller

Figure 606 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

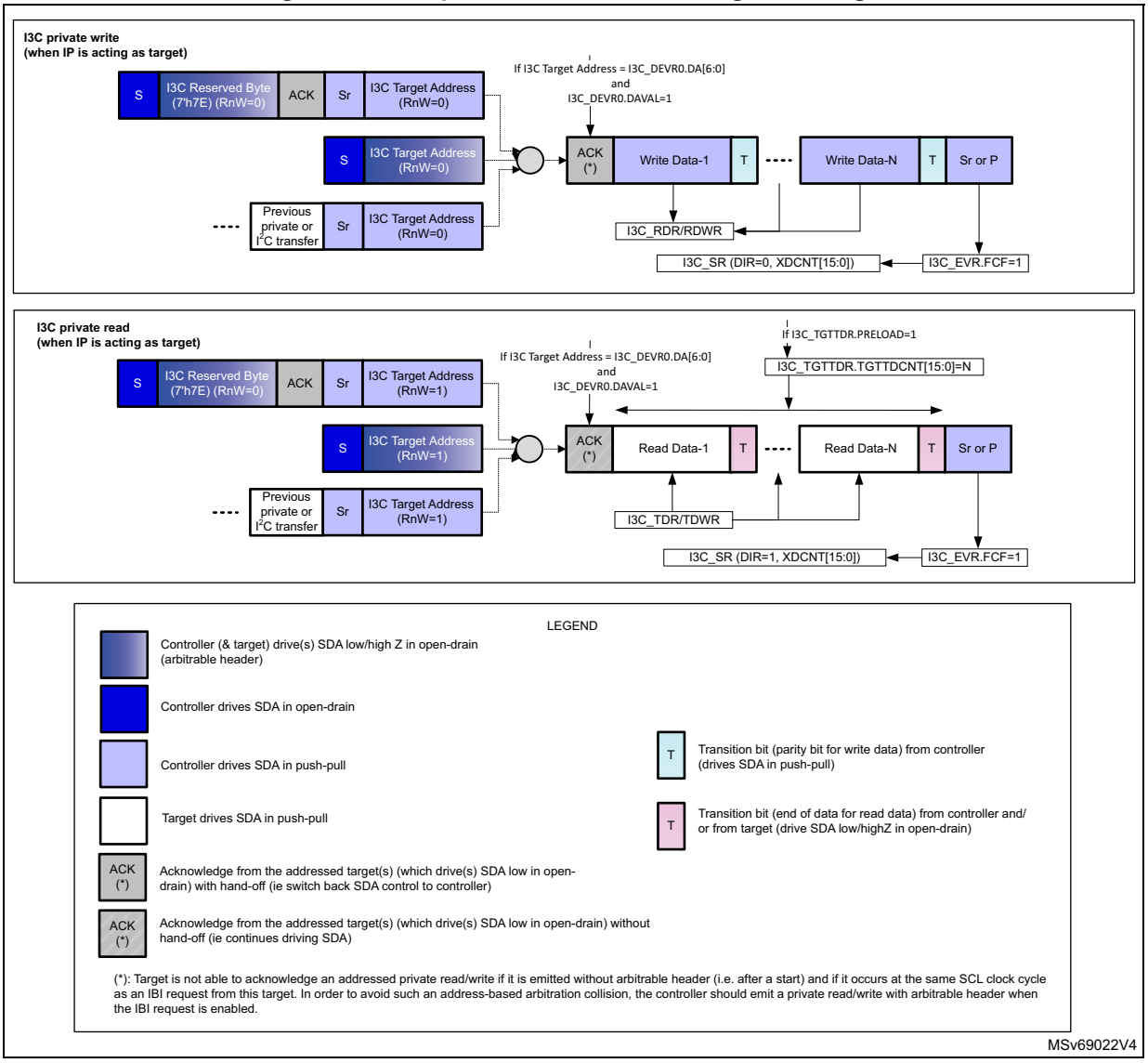
Figure 606. I3C private read/write messages, as controller



42.9.11 I3C private read/write transfer, as target

Figure 607 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

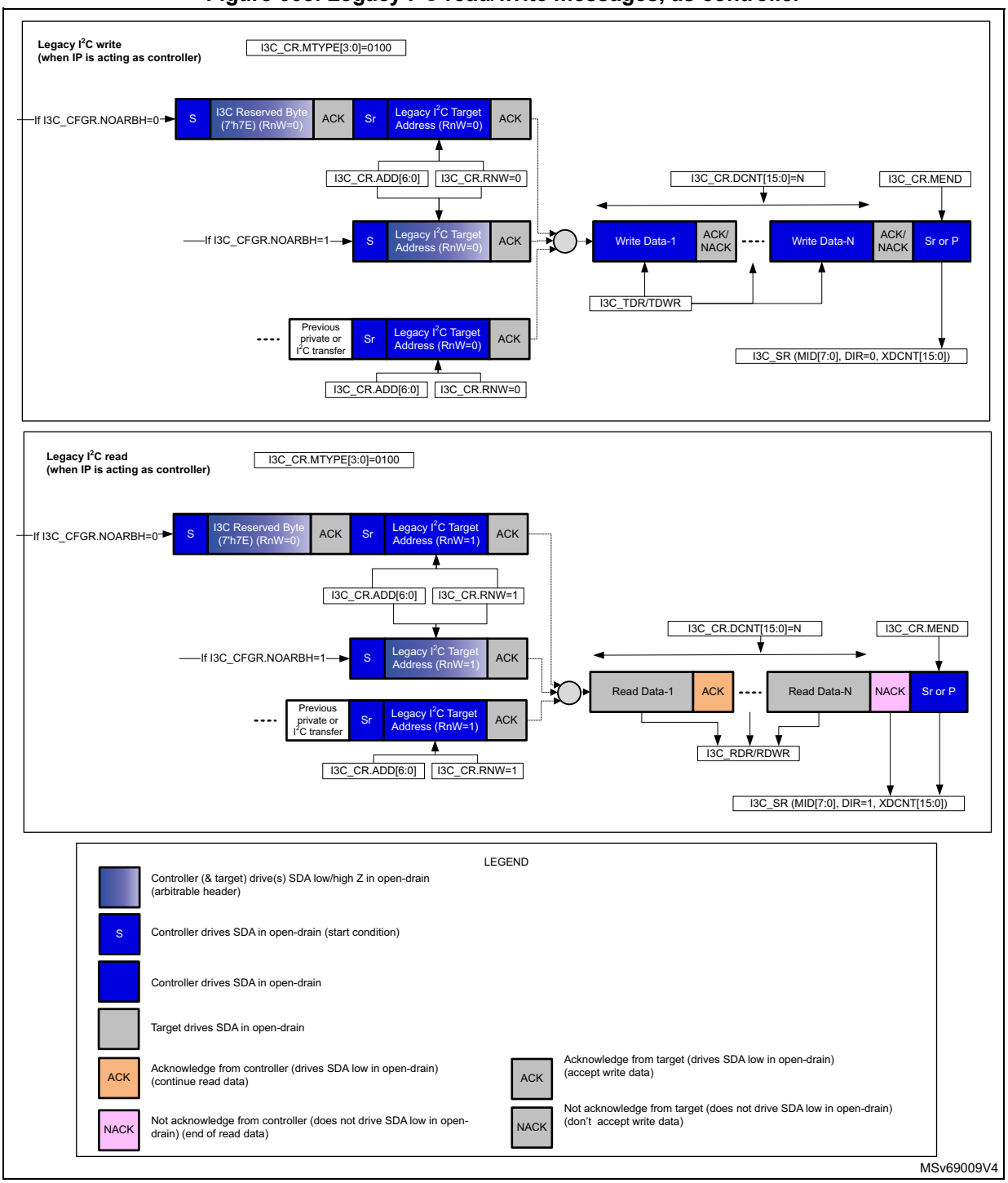
Figure 607. I3C private read/write messages, as target



42.9.12 Legacy I²C read/write transfer, as controller

Figure 608 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

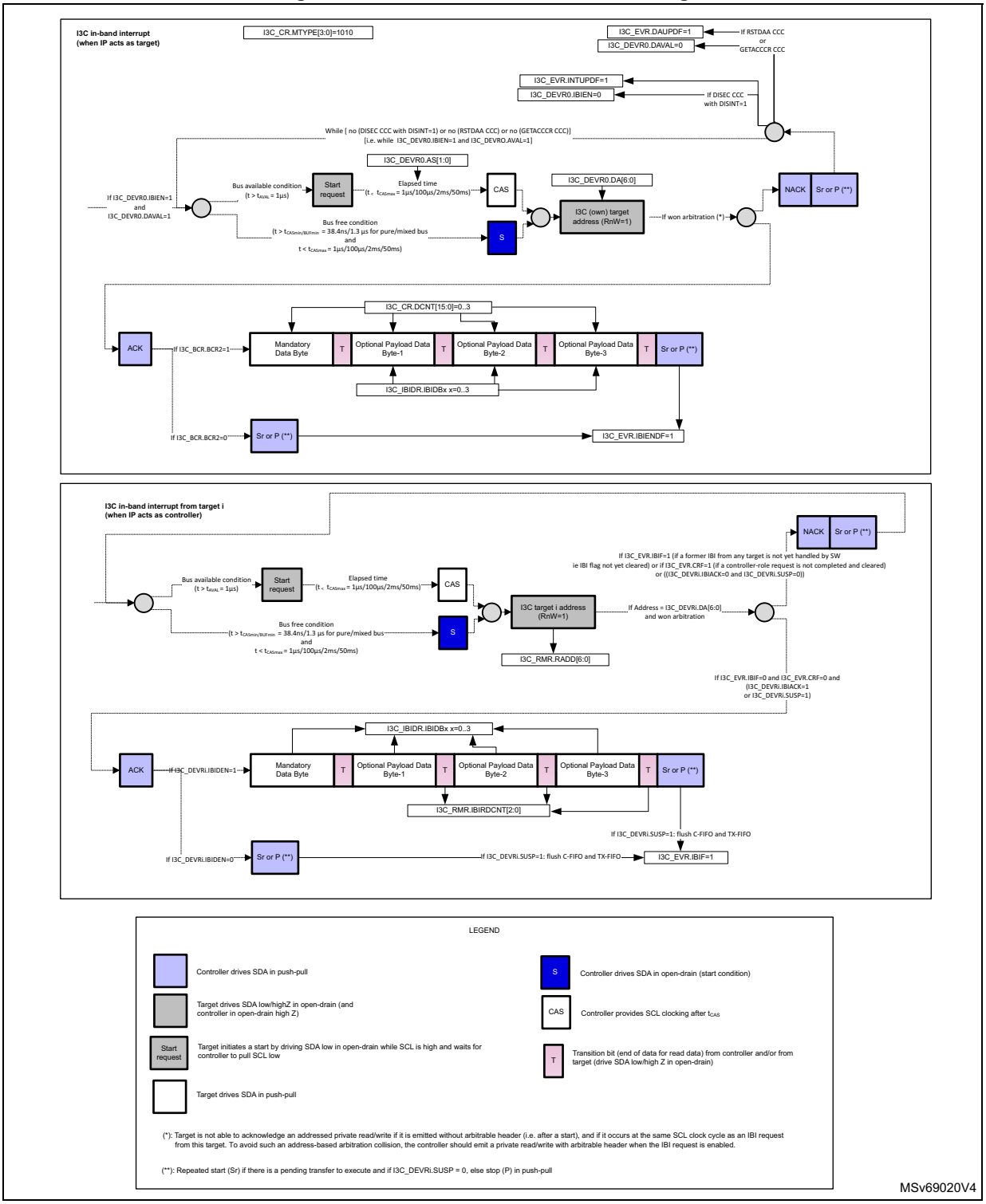
Figure 608. Legacy I²C read/write messages, as controller



42.9.13 I3C IBI transfer, as controller/target

Figure 609 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 609. IBI transfer, as controller/target

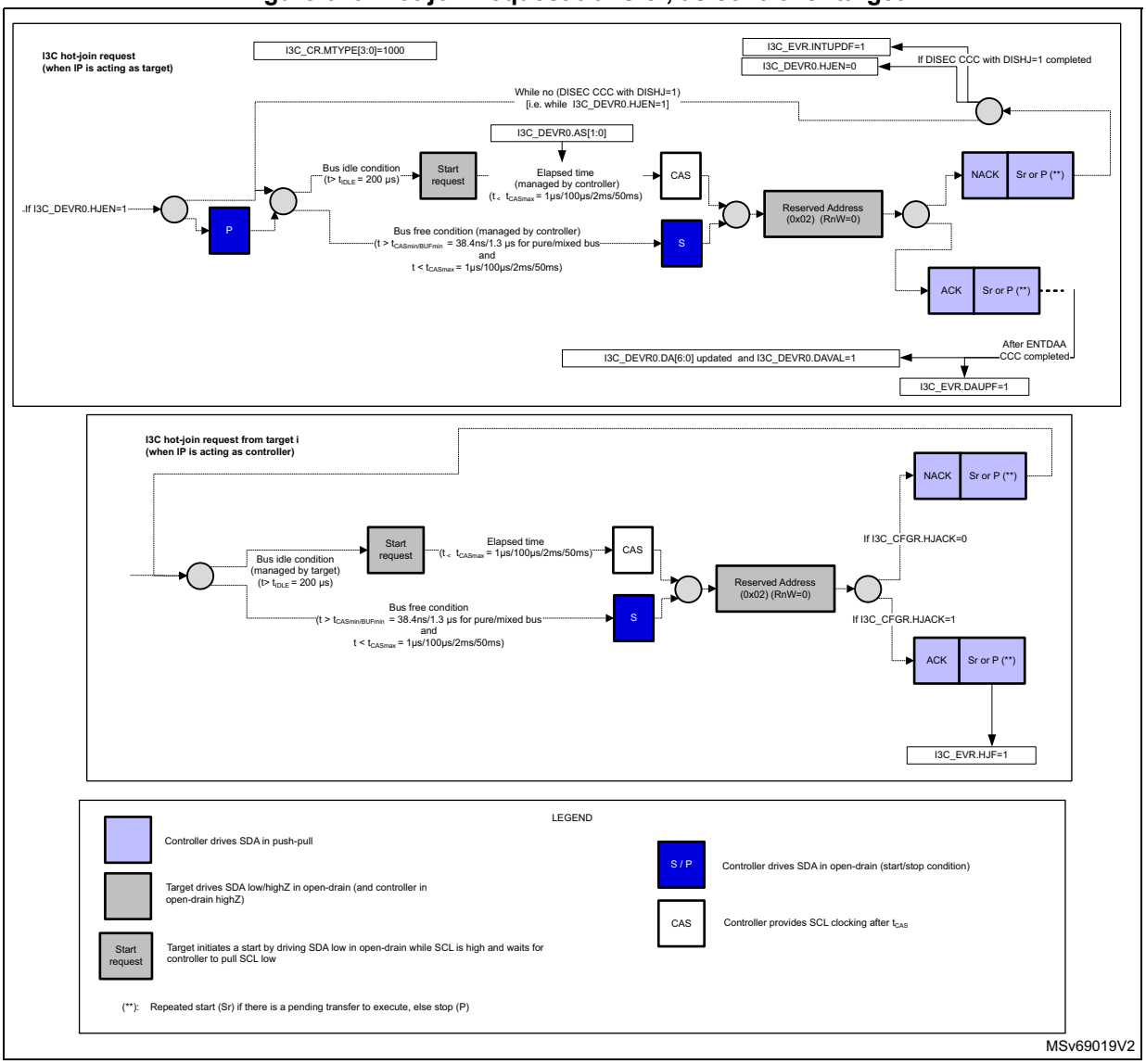


When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

42.9.14 I3C hot-join request transfer, as controller/target

Figure 610 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

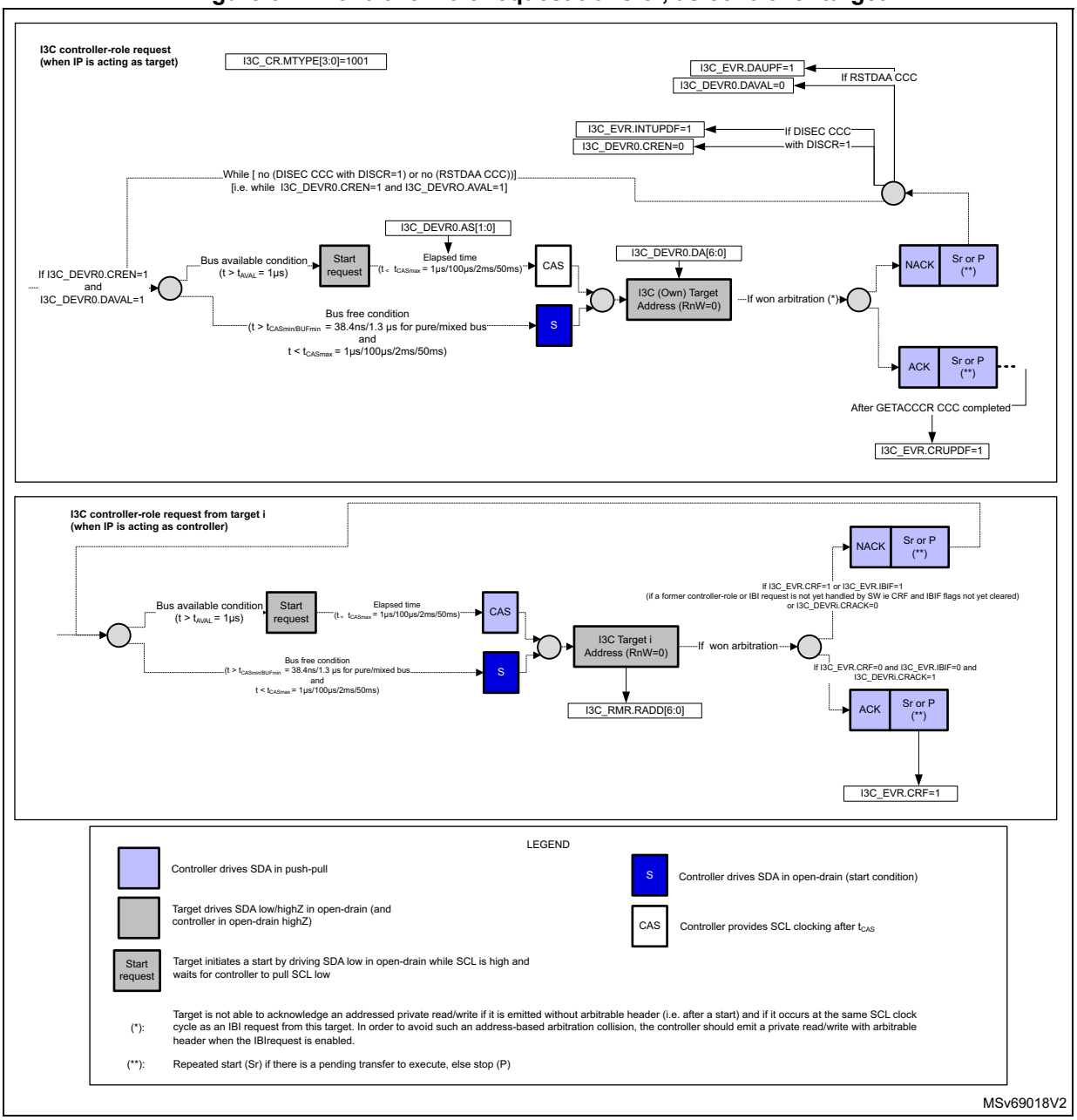
Figure 610. Hot-join request transfer, as controller/target



42.9.15 I3C controller-role request transfer, as controller/target

Figure 611 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 611. Controller-role request transfer, as controller/target



42.10 I3C FIFOs management, as controller

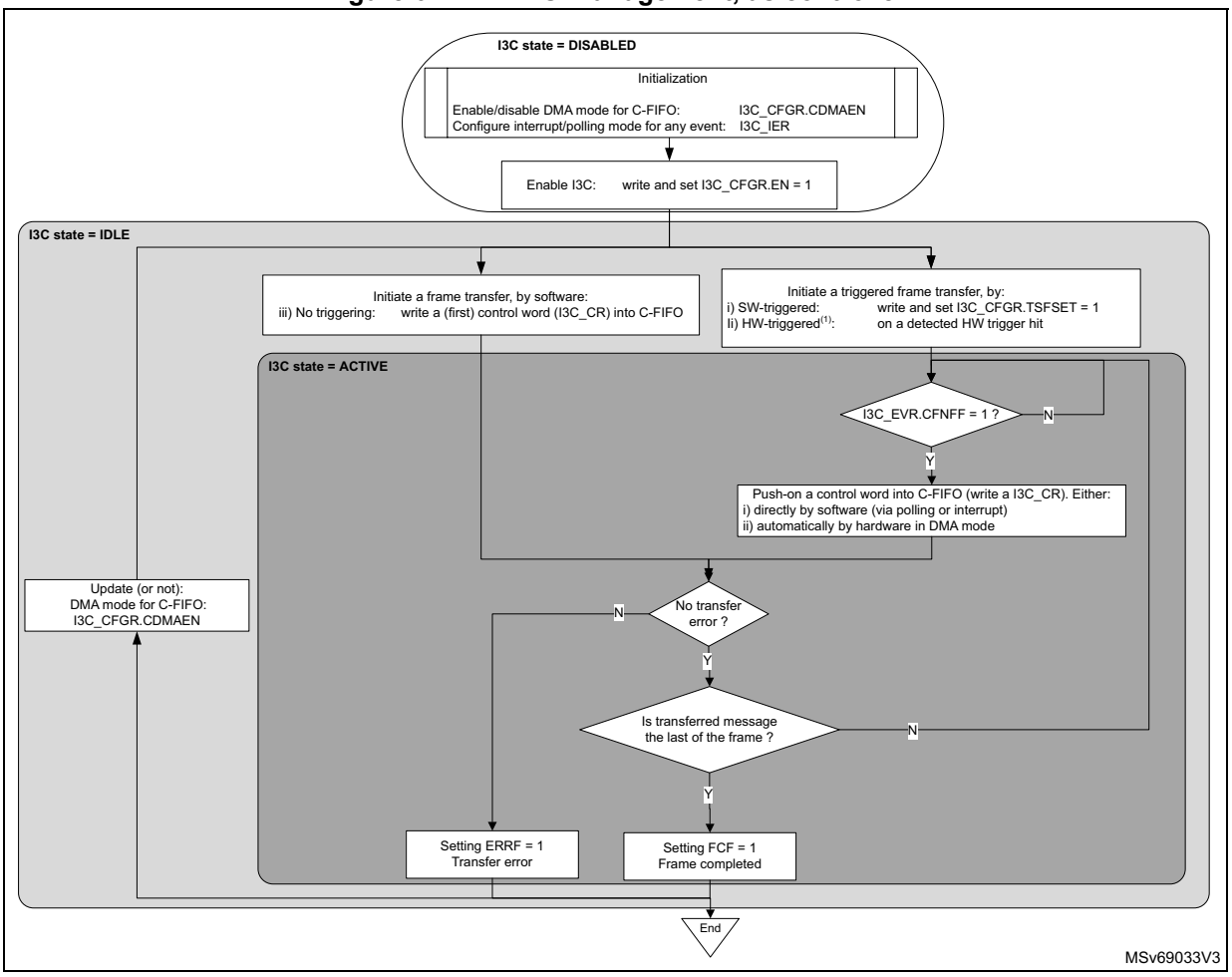
42.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 42.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 599*, *Figure 600*, *Figure 601*)
- direct read/write CCC (*Figure 601*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 606*)
- legacy I2C read/write (*Figure 608*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 612 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 612. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 42.3.4*.

Note: The software can disable the generation of FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 42.3) to stream a series of frames.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

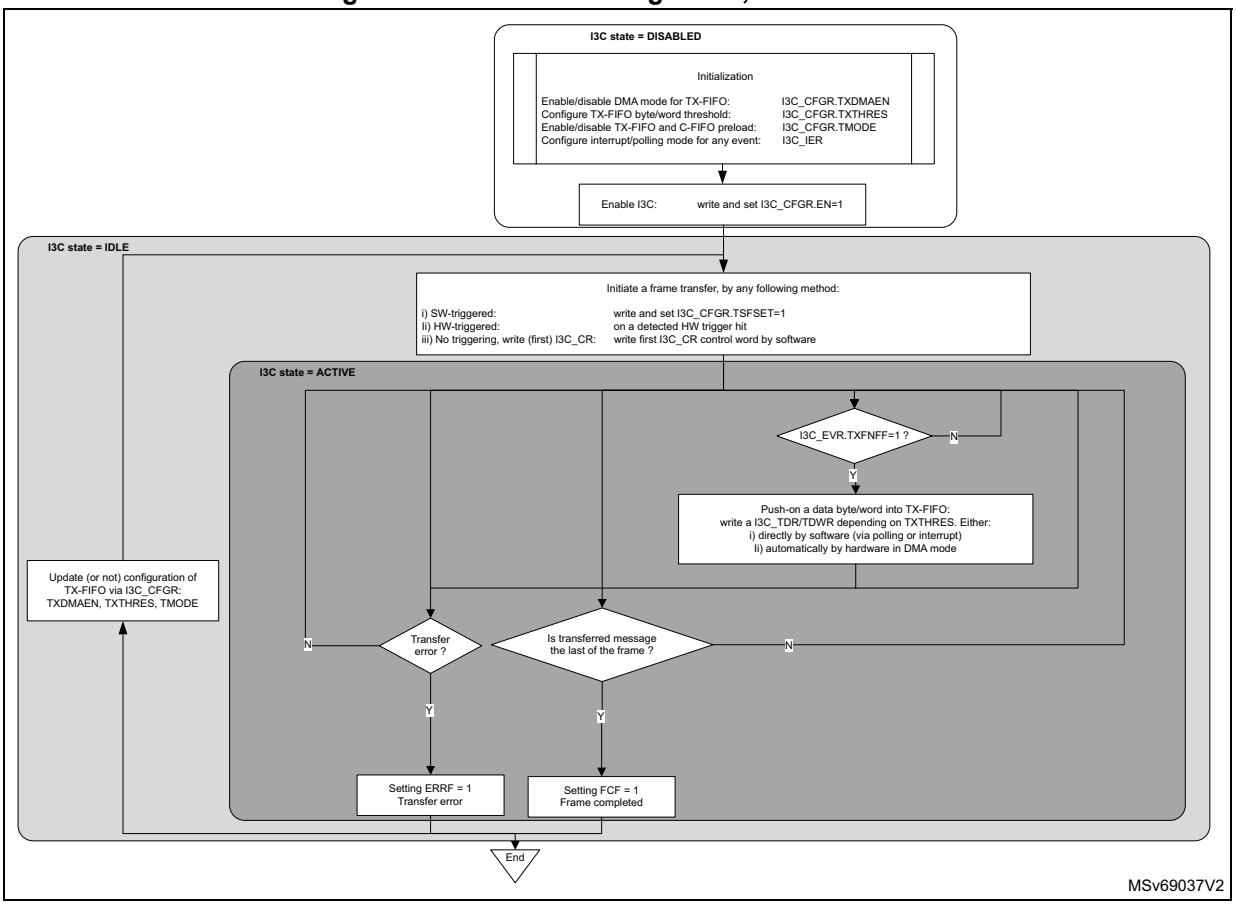
42.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 42.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 599*, *Figure 600*, *Figure 601*)
- private write (*Figure 606*)
- legacy I2C write (*Figure 608*)

Figure 613 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 613. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 425*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

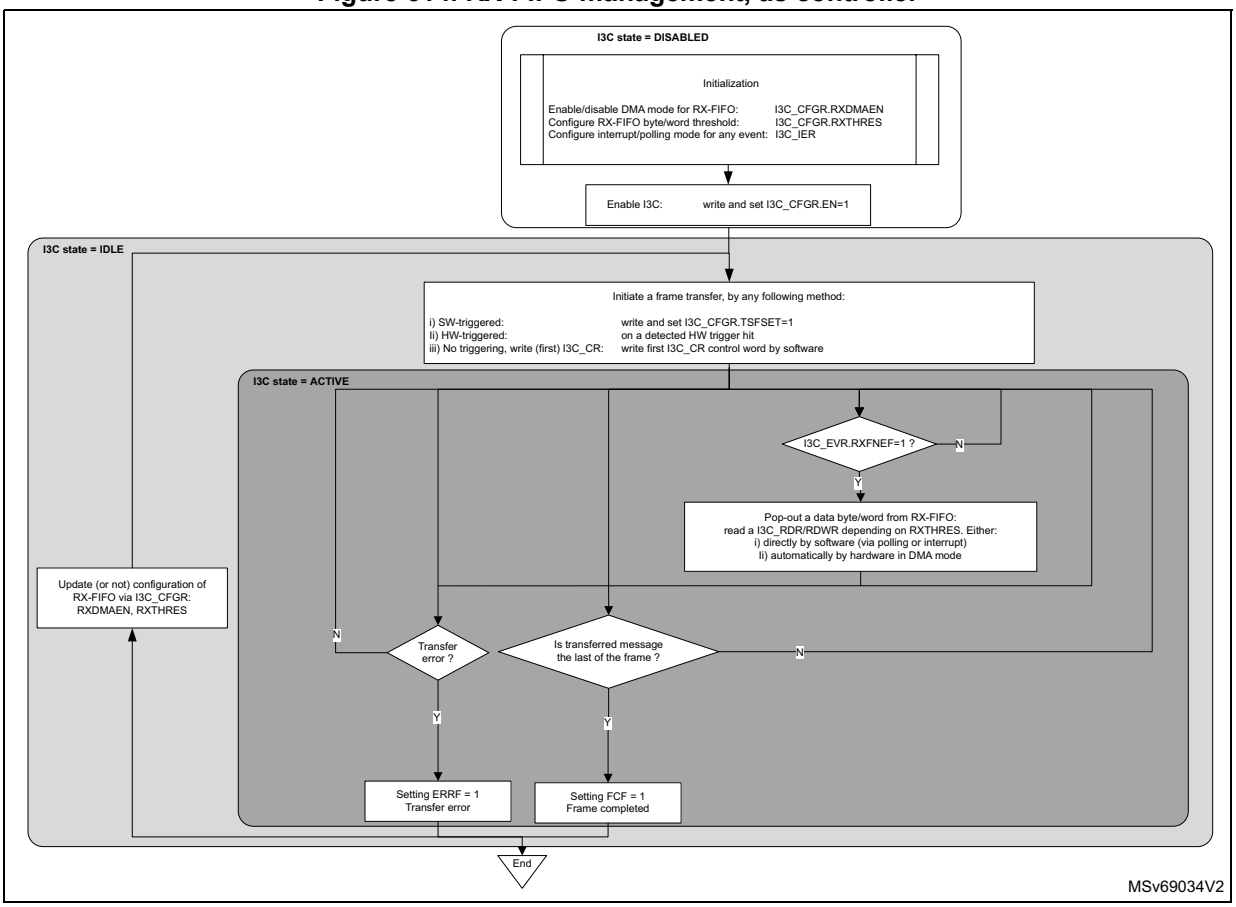
42.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 42.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 600*)
- direct CCC read (*Figure 599*), including direct RSTACT CCC read (*Figure 601*)
- private read (*Figure 606*)
- legacy I2C read (*Figure 608*)

Figure 614 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 614. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 42.10.4*.

42.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

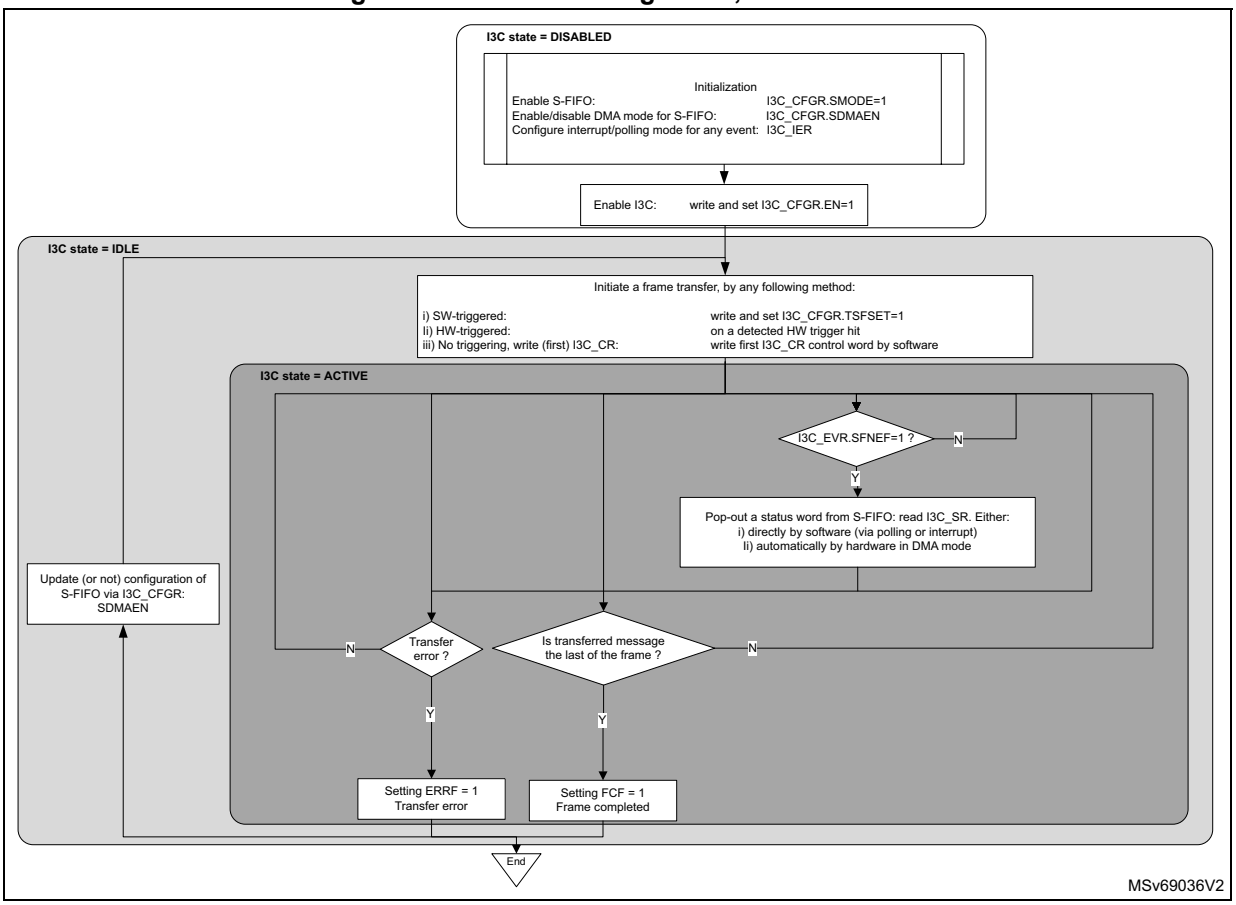
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 42.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 615* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 615. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

42.11 I3C FIFOs management, as target

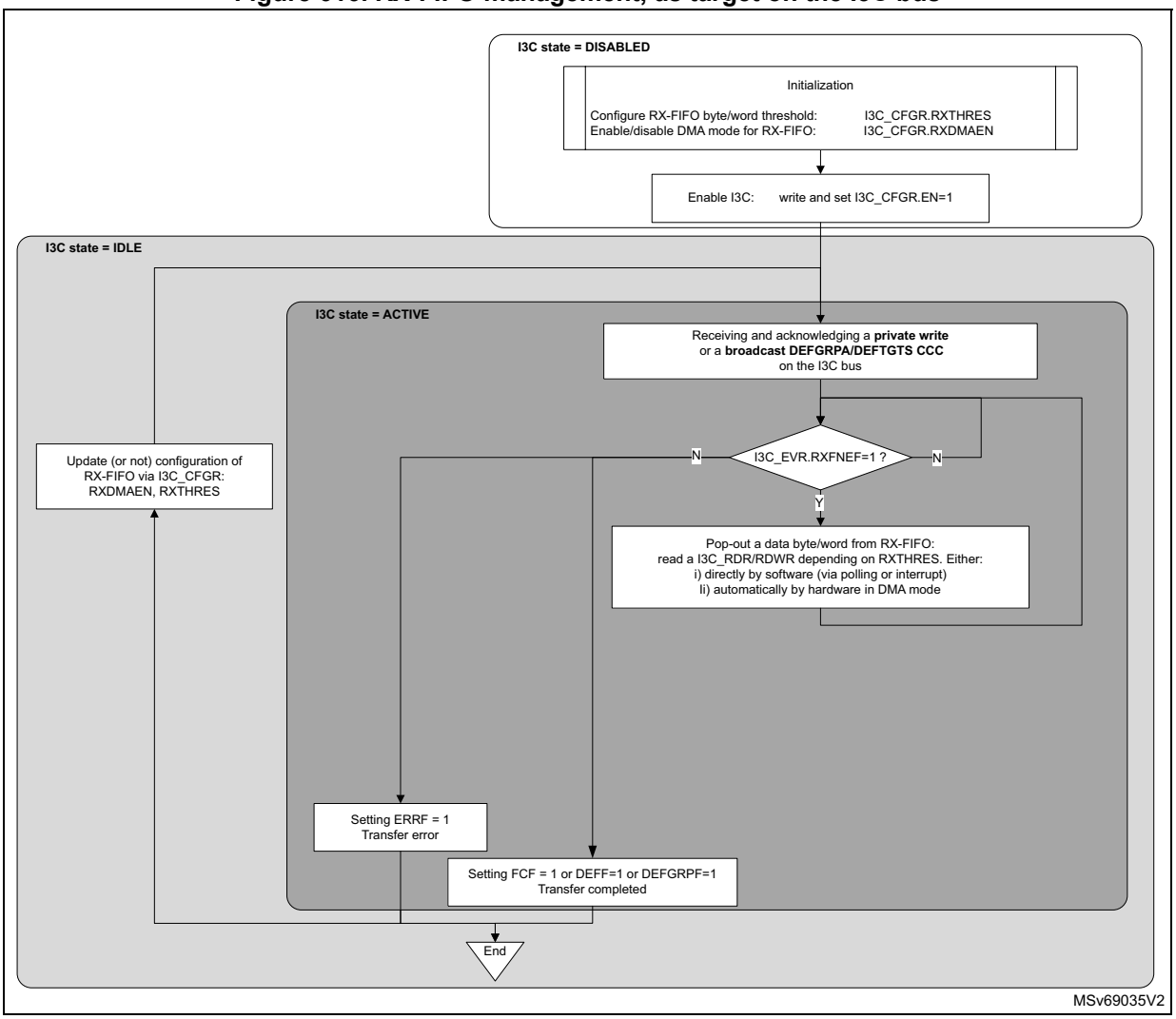
42.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 42.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 604*)
- broadcast DEFGRPA CCC (*Figure 605*)
- private write (*Figure 607*)

Figure 616 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received on the I3C bus, when the I3C peripheral acts as target.

Figure 616. RX-FIFO management, as target on the I3C bus



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed or private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

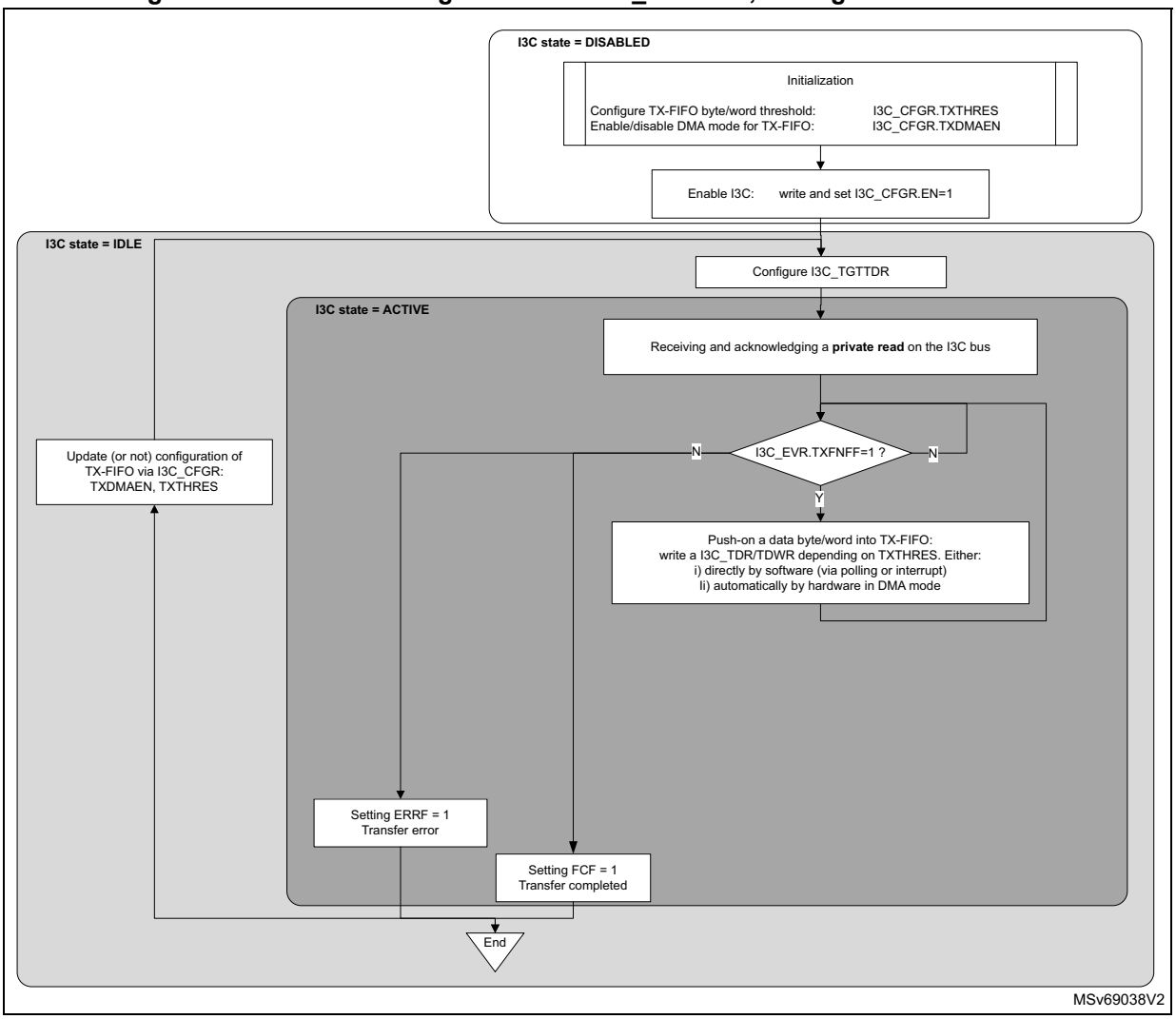
42.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 42.9*, the TX-FIFO is used only during a private read (*Figure 607*).

Figure 617 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.

Figure 617. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus , the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the serial bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 425).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the serial bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the serial bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the serial bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another read, depending upon the user application

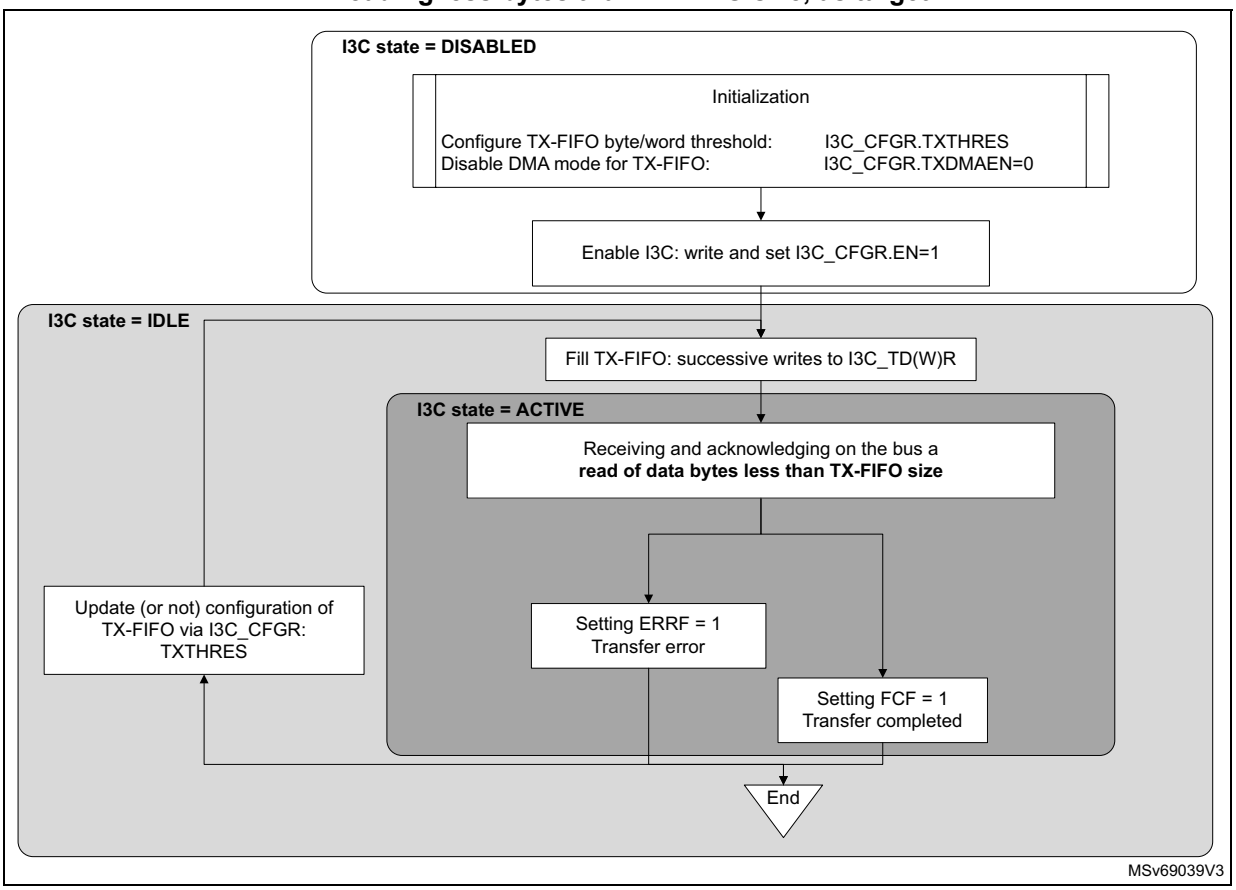
If TX-FIFO is empty and a data byte must be transmitted on the serial bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 618*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the serial bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 618. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



42.12 I3C error management

42.12.1 Controller error management

Table 434 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 434. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 434. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 434. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

42.12.2 Target error management

Table 435 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 435. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 435. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error of I3C bus	SDA is detected as driven at unexpected value on an data read (during a CCC direct read or a private read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C bus timings	Data to be transmitted not written by software or DMA in due time on a private read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

42.13 I3C wake-up from low-power mode(s)

42.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x = 1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
   - On a detected hardware trigger to initiate an I3C transfer on the I3C bus (refer to *Section 42.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 609*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 610*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 611*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as

the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.

7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a I3C reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start on the serial bus

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the serial bus (SDA line is detected to be driven low while SCL is high)
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a bus start is missed and waits for the kernel clock to be provided. A controller message that is specifically addressed to this low-power device may have been missed (and NAcked). However the I3C peripheral can also detect a missed start if the controller started a CCC or private or legacy I^2C message to another addressed target, or if another target initiated a start for an IBI/CR/HJ request.
   - On a detected hardware trigger to initiate an I3C IBI/CR/HJ transfer on the I3C bus (refer to *Section 42.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in

Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).

3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   – The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that a serial bus transaction was missed, but not whether the target was addressed or not. The software uses a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

42.14 I3C in low-power modes

Table 436. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 42.3.2: I3C wake-up from low-power mode(s)* to see which, if any, Stop modes are supported.

42.15 I3C interrupts

42.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

42.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 42.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0]							RNW
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: **SCL clock is forced to stop** until a next control word is executed

Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 434*.

0001: **header message**

Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 434* and MIPI specification about escalation handling.

If the fields of the I3C_CFGR register are programmed with EXITPTRN= 0 and RSTPTRN=1, a target reset pattern is sent on the I3C bus after the header message, whatever the header is ACK-ed or NACK-ed. This can be used to generate and broadcast a target reset pattern, especially in the case of an I3C bus topology with a single I3C target in deep sleep. Alternatively, and more generally, a target reset pattern can be encapsulated within a RSTACT CCC command (refer to *Figure 601*).

0010: **private message** (refer to *Figure 606*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 599*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: **legacy I^2C message** (refer to *Figure 608*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: **hot-join request** (W) (refer to *Figure 610*)

The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: **controller-role request** (W) (refer to *Figure 611*)

The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 609*)

Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.

0: write message

1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.

When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).

Linear encoding up to 64 Kbytes - 1

0x0000: no data to transfer

0x0001: 1 byte

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

42.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	CCC[7:0]							
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 433*, *Figure 599*, *Figure 600*, *Figure 601*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr
– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 600*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P
– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 600).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 601).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

42.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH, FCFDIS)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w							rw		w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **FCFDIS**: FCF generation disable (when I3C acts as controller)

When the IP acts as controller, the generation of the FCF flag in the register I3C_EVR can be disabled. The software can avoid clearing this flag between two generated I3C frames (by writing 1 into CFCF in the I3C_CEVR register), especially in hardware-triggered autonomous mode.

0: enable the generation of the flag FCF in the I3C_EVR register

1: disable the generation of the flag FCF in the I3C_EVR register

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 42.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Additionally, if RSTPTRN = 1 and EXITPTRN = 0, a reset pattern is sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register), whatever the message header is ACK-ed or NACK-ed.

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).
- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.
- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)

42.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

42.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

42.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

42.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

42.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

42.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE LOAD
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGTTDCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

42.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
r	r	r	r	r	r	r	r						r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

42.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

42.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							Res.
								r	r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

42.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

42.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
rw				rw	rw	rw				rw	rw	rw	rw		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bits 1:0 Reserved, must be kept at reset value.

42.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
w	w	w	w	w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

42.16.16 I3C masked interrupt status register (I3C_MISR)

Address offset: 0x05C

Reset value: 0x0000 0000

This is a read register, containing the masked interrupt status for each individual event flag bit. Each bit is the logical AND of the corresponding interrupt enable bit of the I3C_IER register and the event flag bit of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPMIS**: DEFGRPA CCC masked interrupt status (when the I3C acts as target)

Bit 30 **DEFMIS**: DEFTGTS CCC masked interrupt status (when the I3C acts as target)

Bit 29 **INTUPDMIS**: ENEC/DISEC CCC masked interrupt status (when the I3C acts as target)

Bit 28 **ASUPDMIS**: ENTASx CCC masked interrupt status (when the I3C acts as target)

Bit 27 **RSTMIS**: reset pattern masked interrupt status (when the I3C acts as target)

Bit 26 **MRLUPDMIS**: SETMRL CCC masked interrupt status (when the I3C acts as target)

Bit 25 **MWLUPDMIS**: SETMWL CCC masked interrupt status (when the I3C acts as target)

Bit 24 **DAUPDMIS**: ENTDAA/RSTDAA/SETNEWDA CCC masked interrupt status (when the I3C acts as target)

Bit 23 **STAMIS**: format 1 GETSTATUS CCC masked interrupt status (when the I3C acts as target)

Bit 22 **GETMIS**: GETxxx CCC masked interrupt status, except GETSTATUS of format 1 (when the I3C acts as target)

Bit 21 **WKPMIS**: Wake-up/missed start masked interrupt status (when the I3C acts as target)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJMIS**: Hot-join masked interrupt status (when the I3C acts as controller)

Bit 18 **CRUPDMIS**: Controller-role update masked interrupt status (when the I3C acts as target)

Bit 17 **CRMIS**: Controller-role request masked interrupt status (when the I3C acts as controller)

Bit 16 **IBIENDMIS**: IBI end masked interrupt status (when the I3C acts as target)

Bit 15 **IBIMIS**: IBI request masked interrupt status (when the I3C acts as controller)

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRMIS**: error masked interrupt status (whatever the I3C acts as controller/target)

Bit 10 **RXTGTENDMIS**: target-initiated read end masked interrupt status (when the I3C acts as controller)

Bit 9 **FCMIS**: frame complete masked interrupt status (whatever the I3C acts as controller/target)

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEMIS**: RX-FIFO not empty masked interrupt status (whatever the I3C acts as controller/target)

Bit 4 **TXFNFMIS**: TX-FIFO not full masked interrupt status (whatever the I3C acts as controller/target)

Bit 3 **SFNEMIS**: S-FIFO not empty masked interrupt status (when the I3C acts as controller)

Bit 2 **CFNFMIS**: C-FIFO not full masked interrupt status (when the I3C acts as controller)

Bits 1:0 Reserved, must be kept at reset value.

42.16.17 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)

This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.

This bit is cleared by hardware when the target receives a frame start.

When the device is not in Stop mode:

– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.

– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset:

– the software writes and sets the SYSRESETREQ control bit of the AIRCR register, when the device is controlled by a Cortex®-M.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: IBI request disabled

1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

42.16.18 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK
r												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed

1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.
- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.
- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;
- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

42.16.19 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

42.16.20 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MWL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

42.16.21 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:

$t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK}$

Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.

Note: With I^2C Fm+ device t_{DIG_Hmin} = 260 ns, with I^2C Fm+ device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

$t_{SCLL_OD} = (SCLL_OD + 1) \times t_{I3CCLK}$

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG_OD_L}$(I3C) timings.

Note: With I^2C Fm+ device t_{DIG_Lmin} = 500 ns, with I^2C Fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG_OD_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

$t_{SCLH_I3C} = (SCLH_I3C + 1) \times t_{I3CCLK}$

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG_H_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG_H_MIXEDmin}$ = 32 ns and $t_{DIG_H_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

$t_{SCLL_PP} = (SCLL_PP + 1) \times t_{I3CCLK}$

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

42.16.22 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SDA_HD[1:0]		Res.	Res.	Res.	Res.	Res.	FREE[6:0]						
		rw	rw						rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **SDA_HD[1:0]**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD[1:0] + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C Fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C Fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: t_{CASr} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: t_{CBP} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: t_{CBSr} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].

This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

42.16.23 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bit 7 Reserved, must be kept at reset value.

Bit 6 **STALLL:** Controller clock stall enable in the address ACK/NACK phase (before the ninth bit) of a legacy I2C read/write message. This allows the I2C target to prepare the next data byte to send/receive.

0: no stall

1: stall enabled

Bit 5 **STALLS:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C write message. This allows the I2C target to prepare the next data byte to receive.

0: no stall

1: stall enabled

Bit 4 **STALLR:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C read message. This allows the I2C target to prepare the next data byte to send.

0: no stall

1: stall enabled

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL}as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

42.16.24 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable

0: I3C target (no controller capable)

1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

42.16.25 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device
Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

42.16.26 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

42.16.27 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CAP GRP	Res.	Res.	Res.	Res.	Res.	CAP DHOFF	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

Bits 2:0 Reserved, must be kept at reset value.

42.16.28 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]							
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS[1:0]	
						rw	rw							rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

42.16.29 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIPIMID[14:0]															IDTSEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID
This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).
This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector
This field is set as 0 (vendor fixed value).
This field represents bit[32] of the 48-bit provisioned ID.
Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID
This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.
This field represents bits[15:12] of the 48-bit provisioned ID.
Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

42.16.30 I3C register map

Table 437. I3C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0] / CCC[7:0]							RNW	DCNT[15:0]																
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	I3C_CFGR	Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0								0	0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[15:0]																
	Reset value	0	0	0	0	0	0	0	0						0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							s_e_TP	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0

Table 437. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0									
0x05C	I3C_MISR	GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS	IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0				0	0	0	0		
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]								SCLL_OD[7:0]								SCLH_I3C[7:0]								SCLL_PP[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	I3C_TIMINGR1	Res.	Res.	SDA_HD		Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value			0	0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
	Reset value																	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0

Table 437. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR	Res.	Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS [1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

43 Universal synchronous/asynchronous receiver transmitter (USART/UART)

43.1 USART introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

43.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  – Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

43.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

43.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 438. Instance implementation on STM32C5

Instance	STM32C53x/C54x	STM32C55x/C56x	STM32C59x/C5Ax
USART1	Full	Full	Full
USART2	Full	Full	Full
USART3	-	Full	Full
USART6	-	-	Full
UART4	Basic	Basic	Basic
UART5	Basic	Basic	Basic
UART7	-	-	Basic
LPUART1	Low-power	Low-power	Low-power

Table 439. USART/LPUART features

Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X

Table 439. USART/LPUART features (continued)

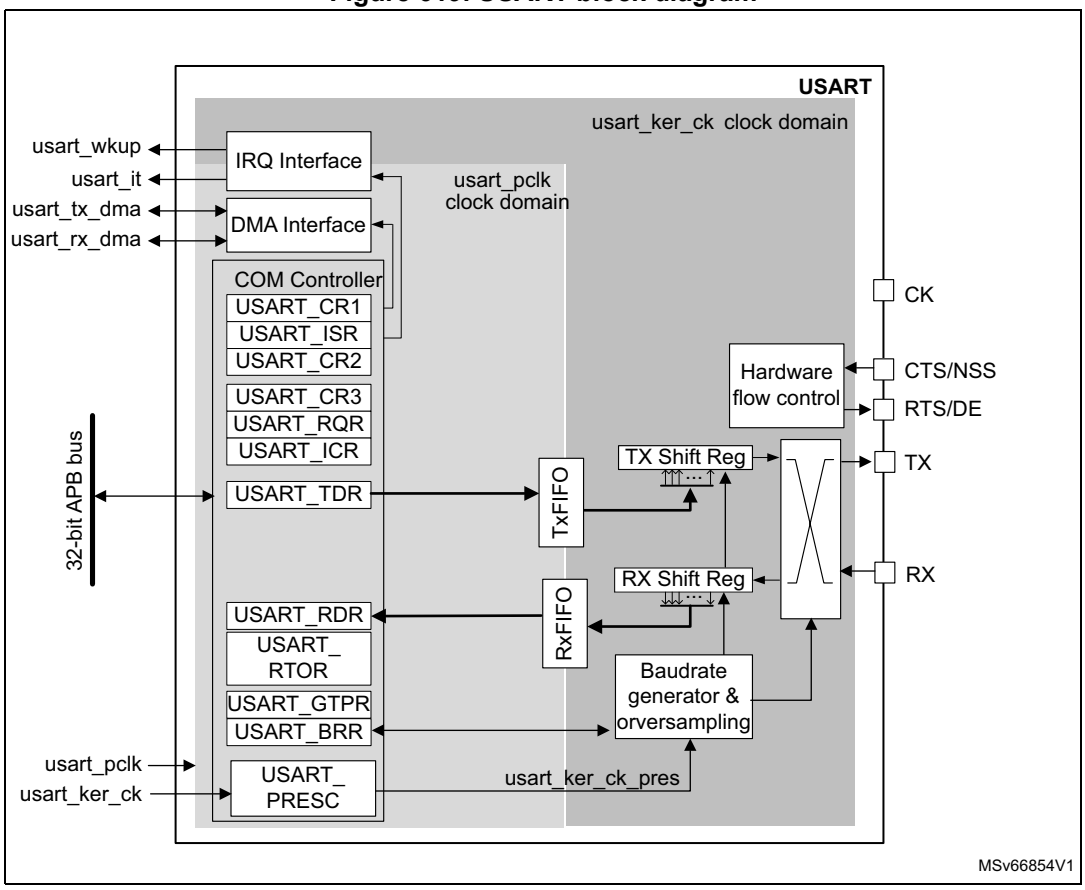
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop mode.

43.5 USART functional description

43.5.1 USART block diagram

Figure 619. USART block diagram



43.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 440* and *Table 441* for the list of USART input/output pins and internal signals.

Table 440. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 441. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request

43.5.3 USART clocks

The simplified block diagram given in *Section 43.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

43.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 620*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

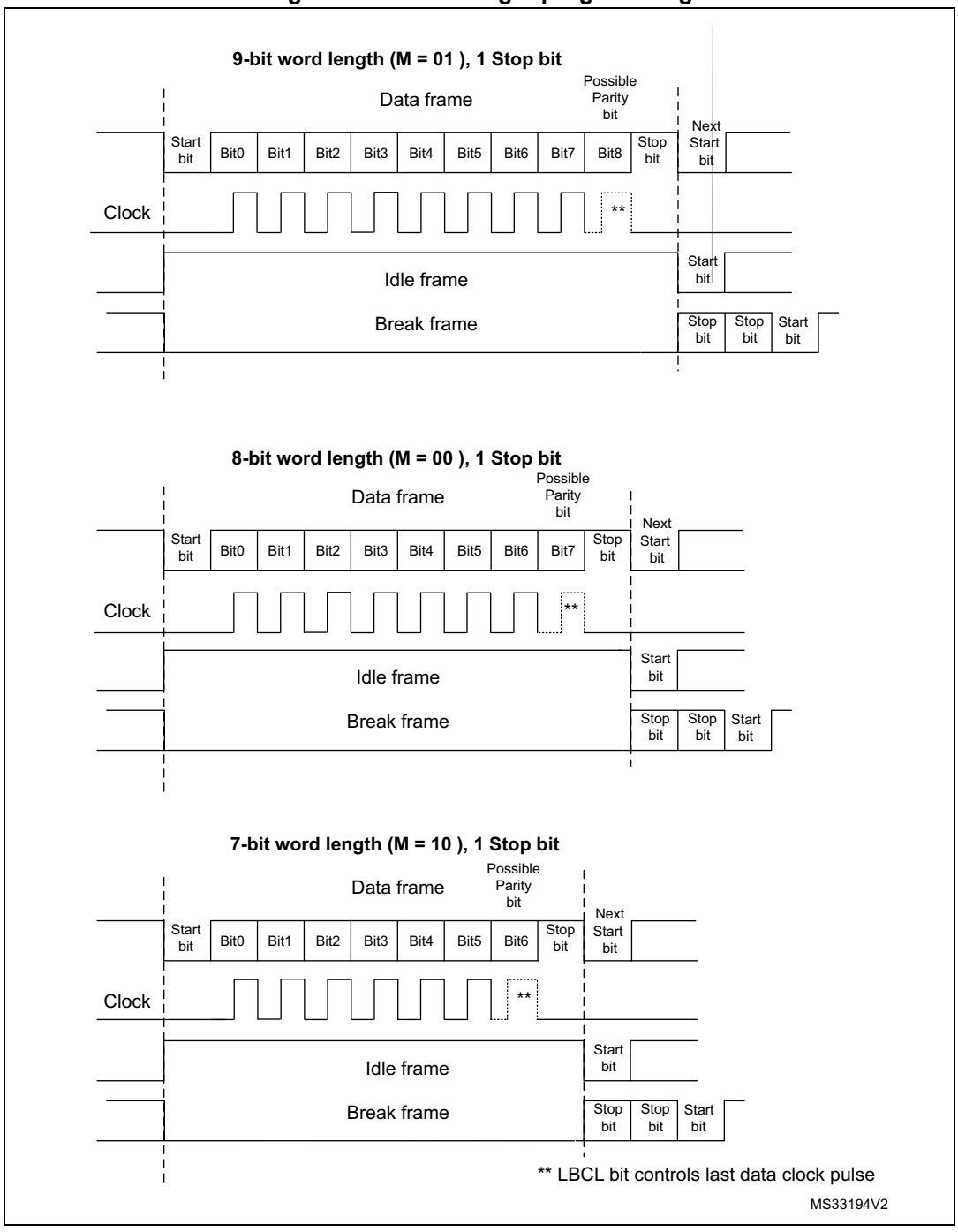
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 620. Word length programming



43.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

43.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

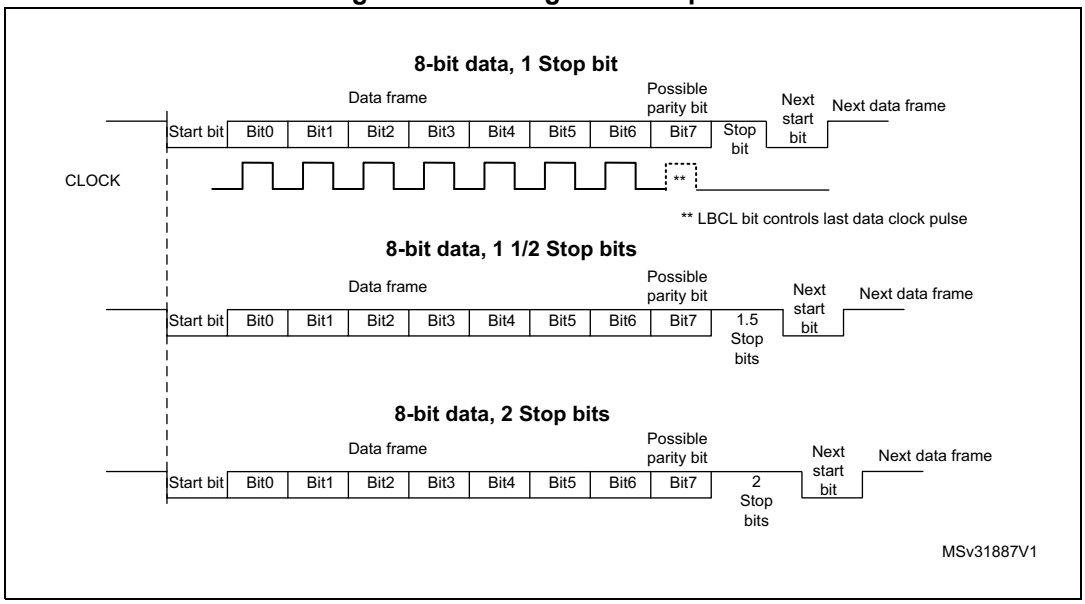
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 43.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 621. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 43.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

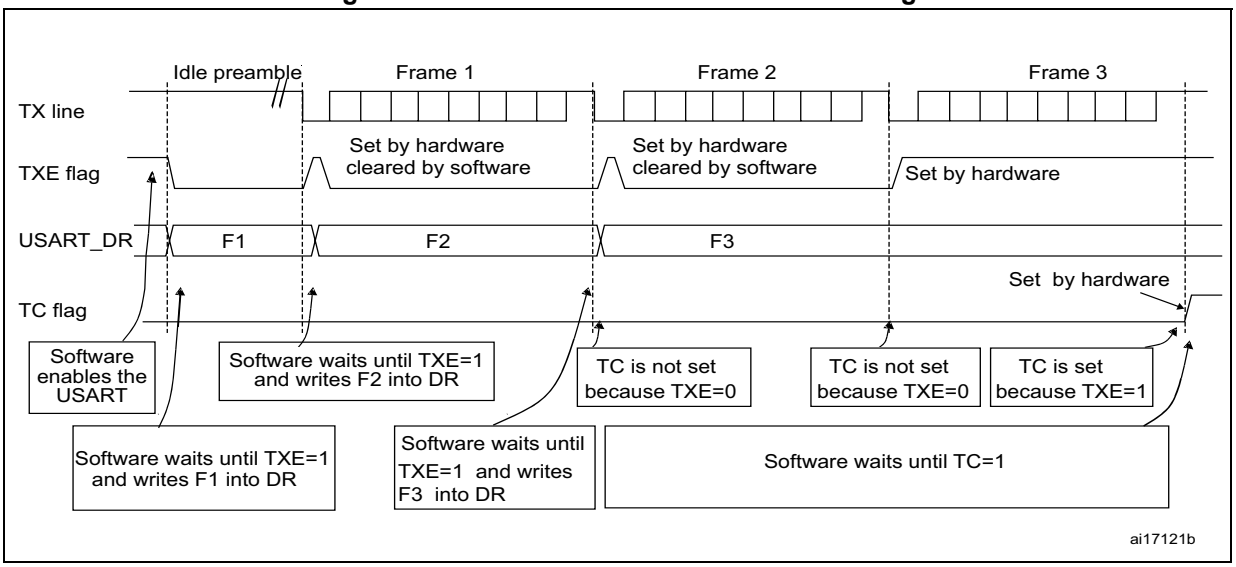
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 622*).

Figure 622. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 620*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

43.5.7 USART receiver

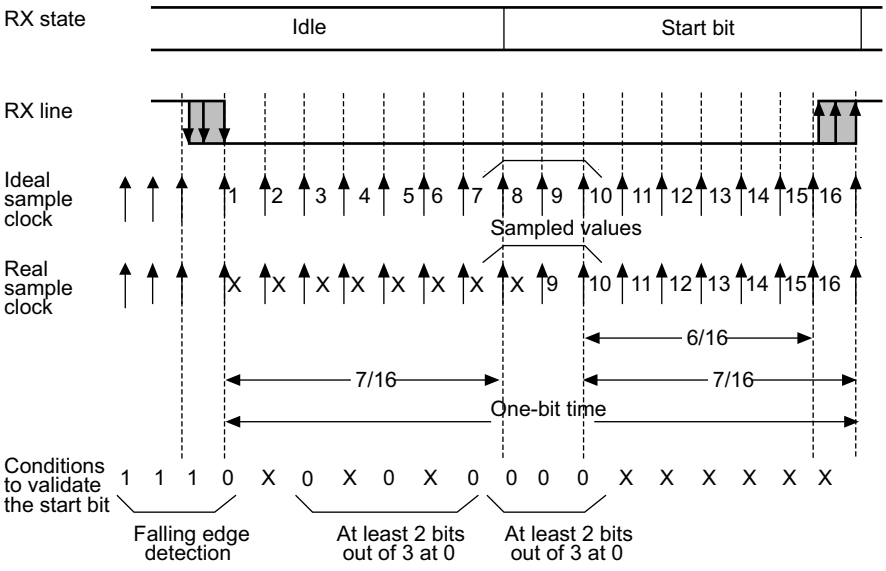
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 623. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 43.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

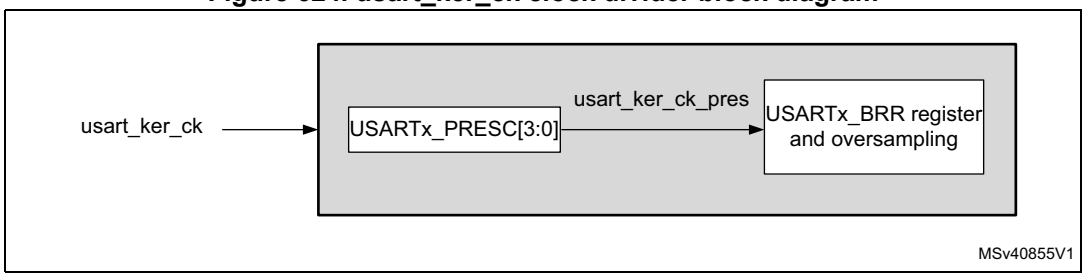
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 624. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 625* and *Figure 626*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 43.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 442*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 43.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 625. Data sampling when oversampling by 16

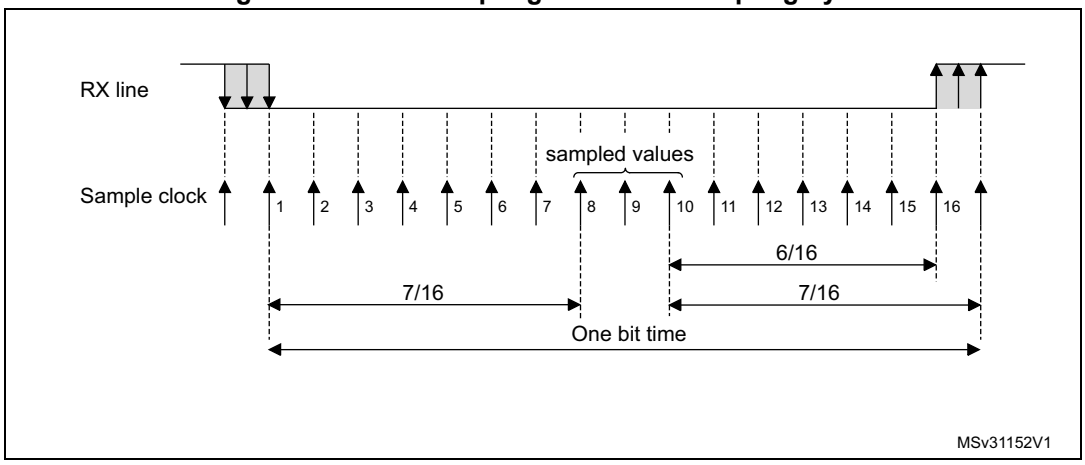


Figure 626. Data sampling when oversampling by 8

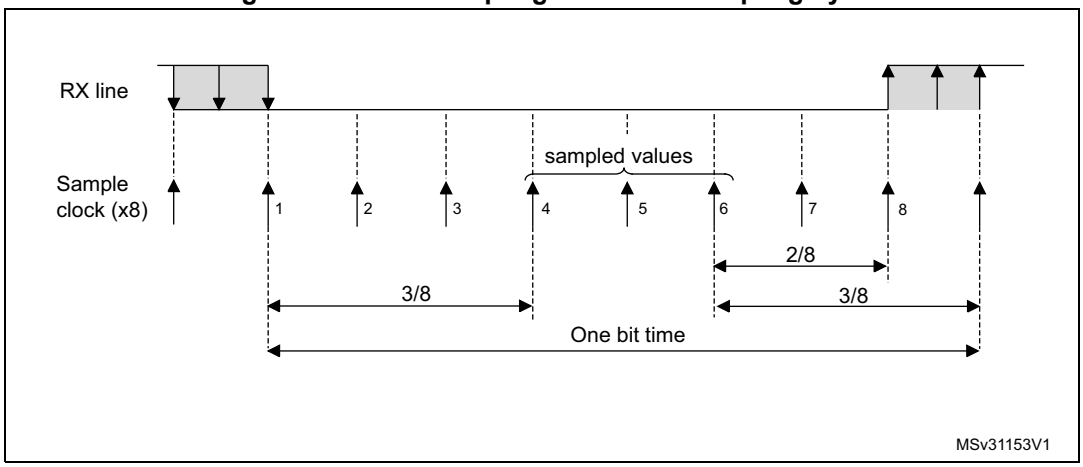


Table 442. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 43.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

43.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[3:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[3:0] = 0x681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[3:0] = USARTDIV = 52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[3:0] = 0x64

43.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 443*, *Table 444*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 443. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 444. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 443 and Table 444 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

43.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

43.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

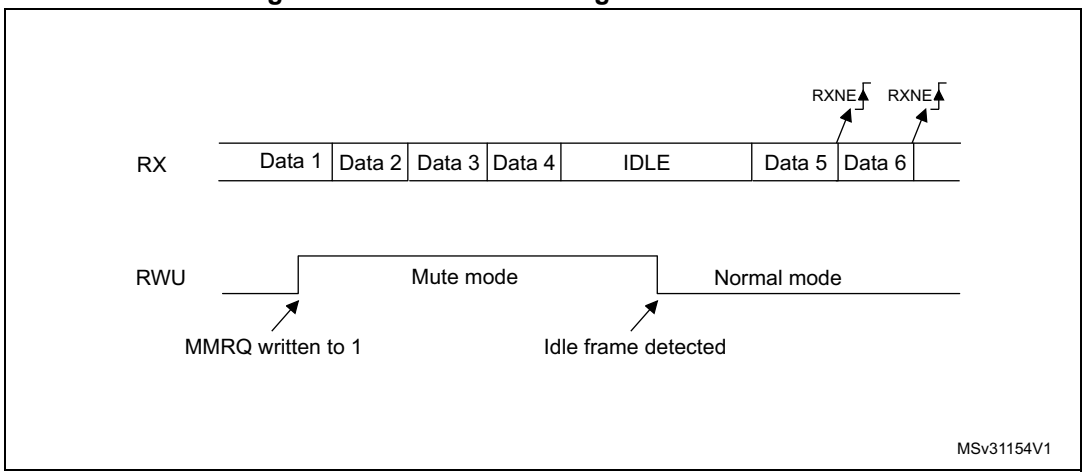
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 627*.

Figure 627. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

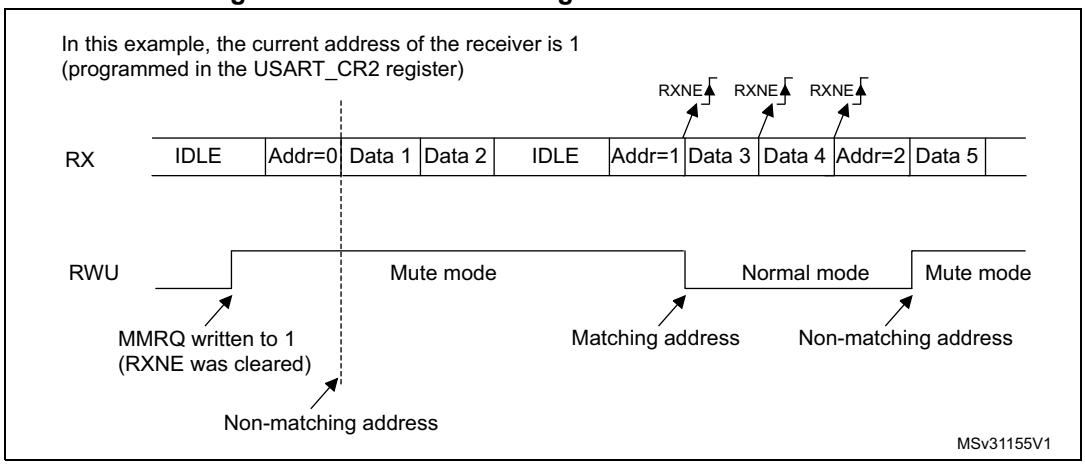
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 628*.

Figure 628. Mute mode using address mark detection



43.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

43.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 445*.

Table 445. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

43.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 43.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 43.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 629*.

Examples of break frames are given in *Figure 630*.

Figure 629. Break detection in LIN mode (11-bit break length - LBDL bit is set)

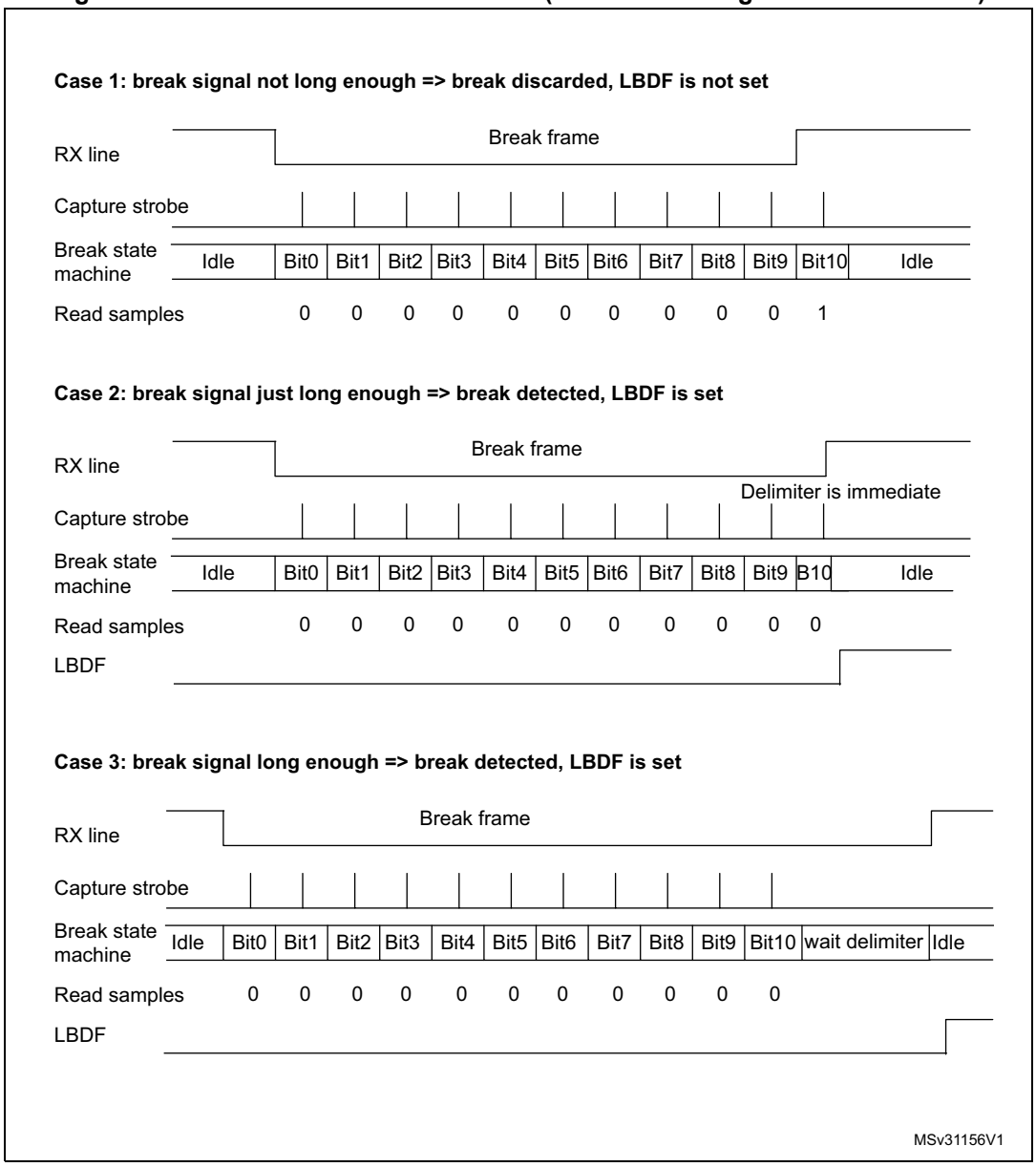
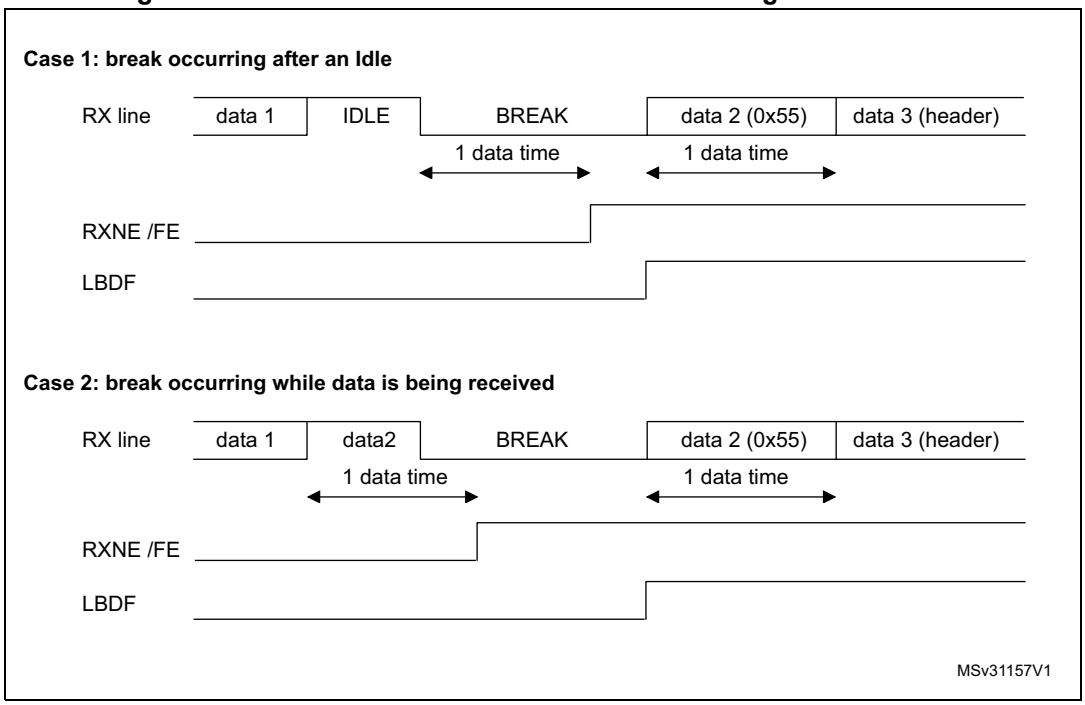


Figure 630. Break detection in LIN mode vs. Framing error detection



43.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 631*, *Figure 632* and *Figure 633*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 631. USART example of synchronous master transmission

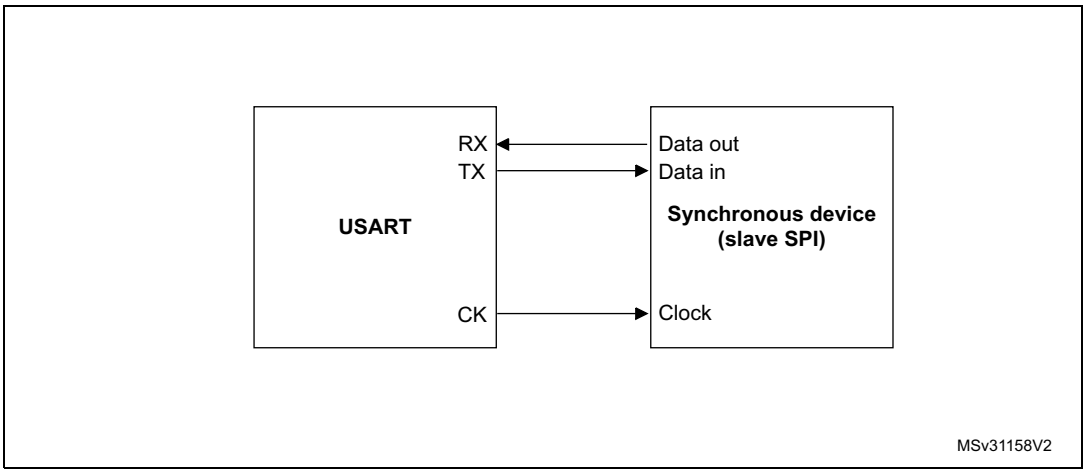


Figure 632. USART data clock timing diagram in synchronous master mode (M bits = 00)

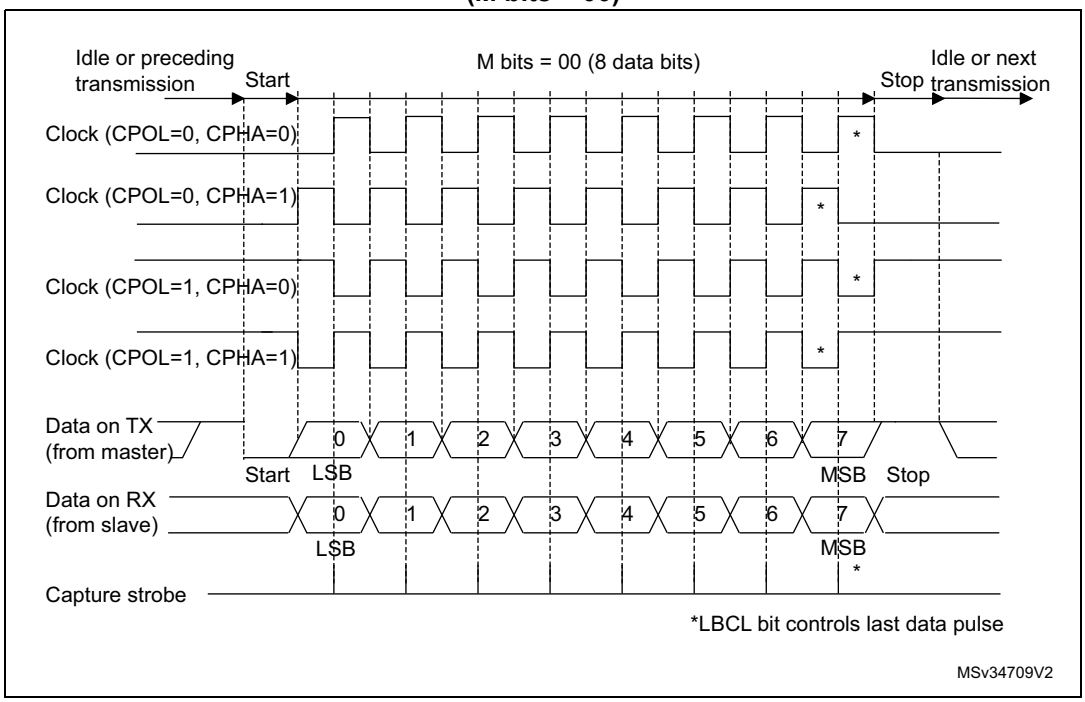
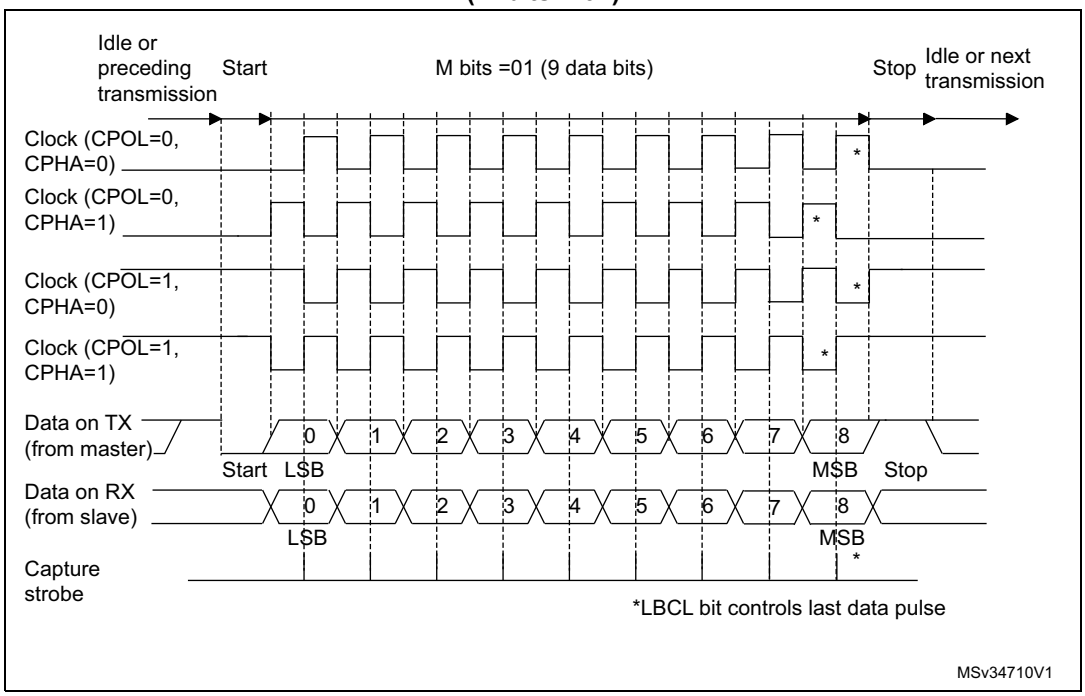


Figure 633. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

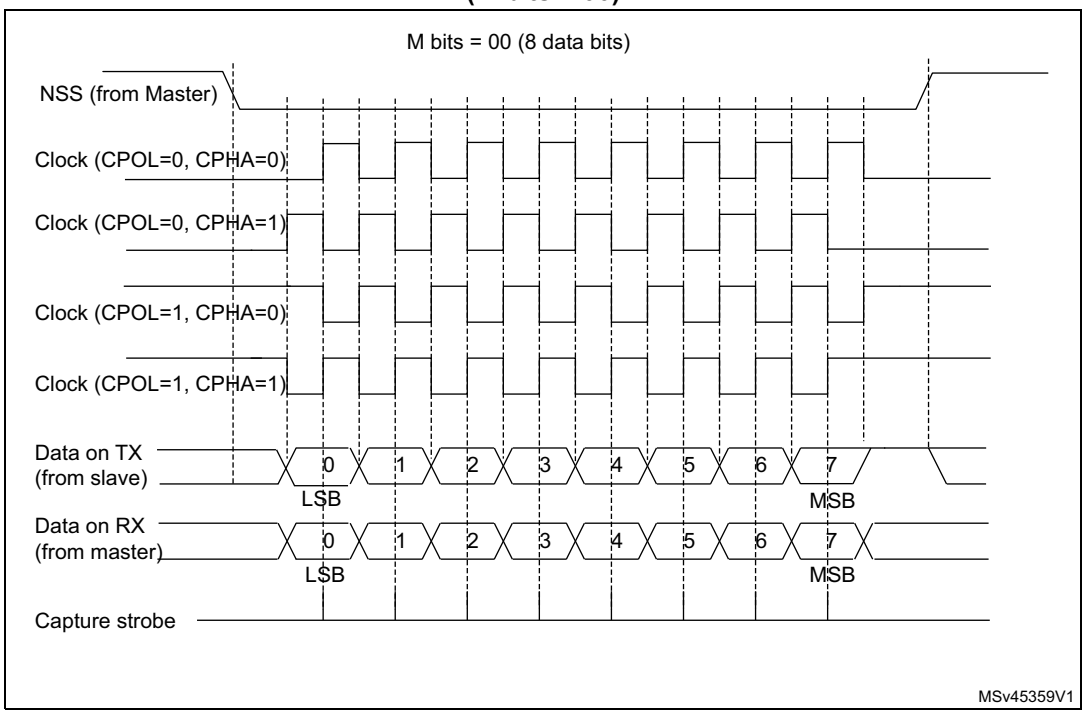
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 634*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 634. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

43.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

43.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

43.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 43.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

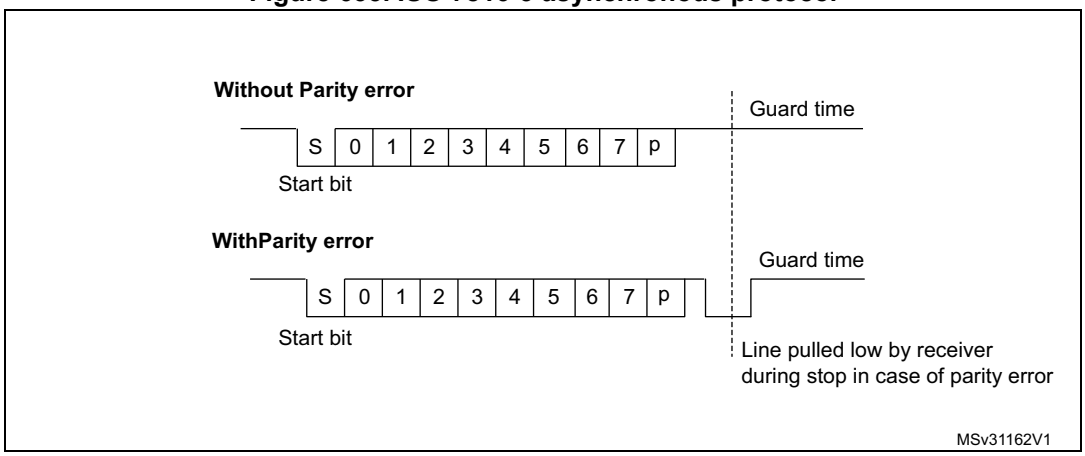
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 635 shows examples of what can be seen on the data line with and without parity error.

Figure 635. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

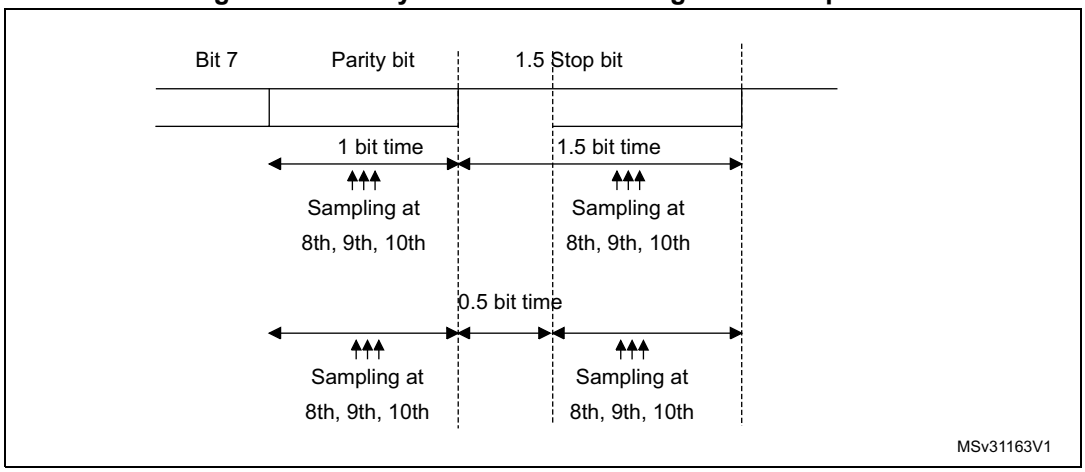
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 636 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 636. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

43.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 43.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 637*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 638*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 637. IrDA SIR ENDEC block diagram

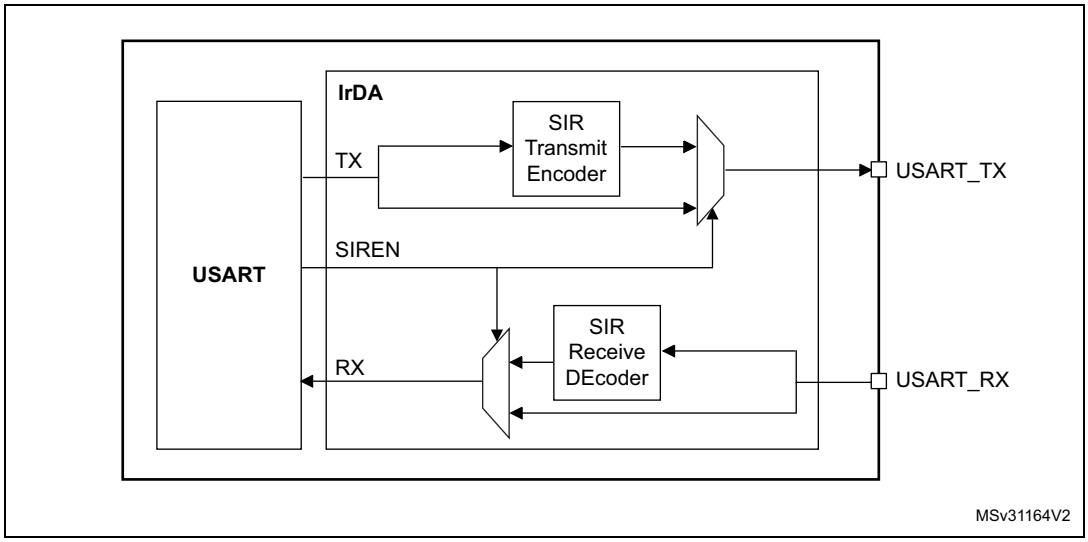
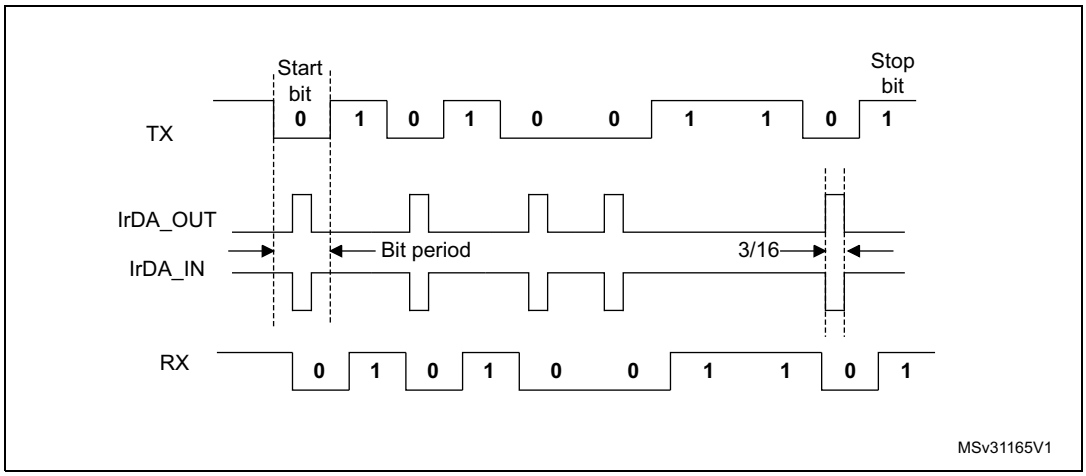


Figure 638. IrDA data modulation (3/16) - normal mode



43.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 43.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 43.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

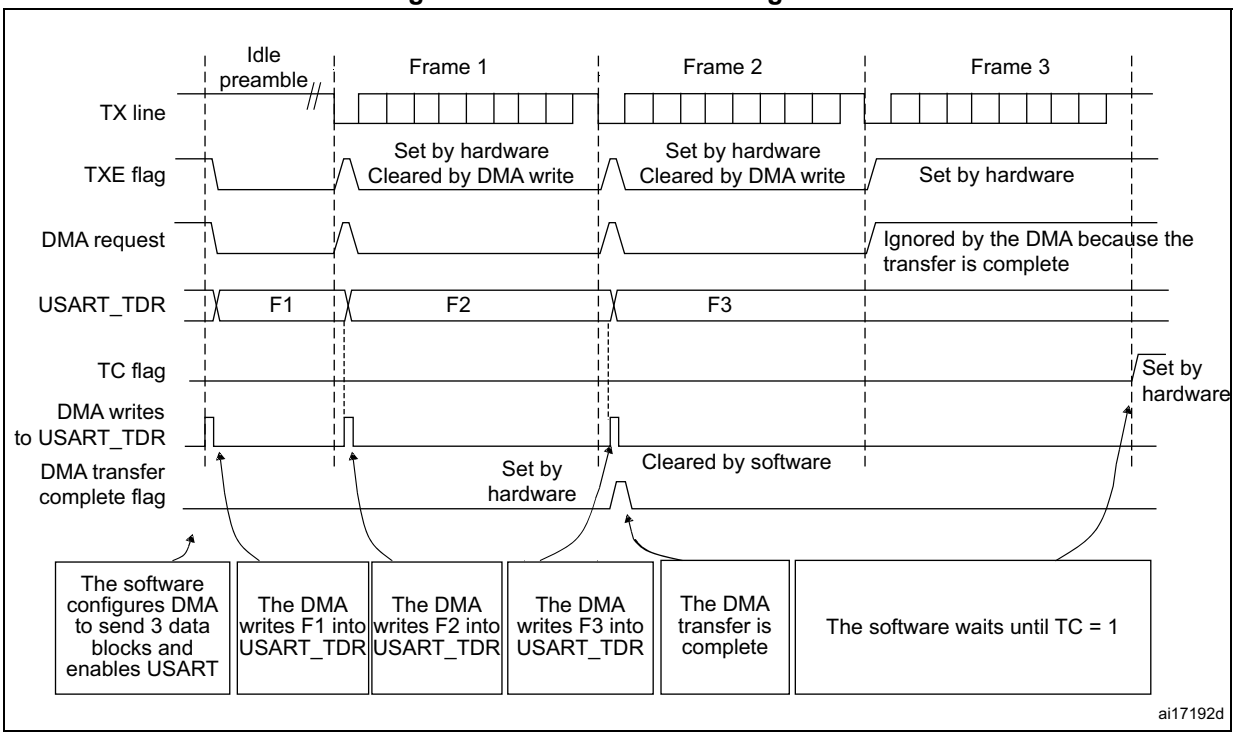
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 639. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

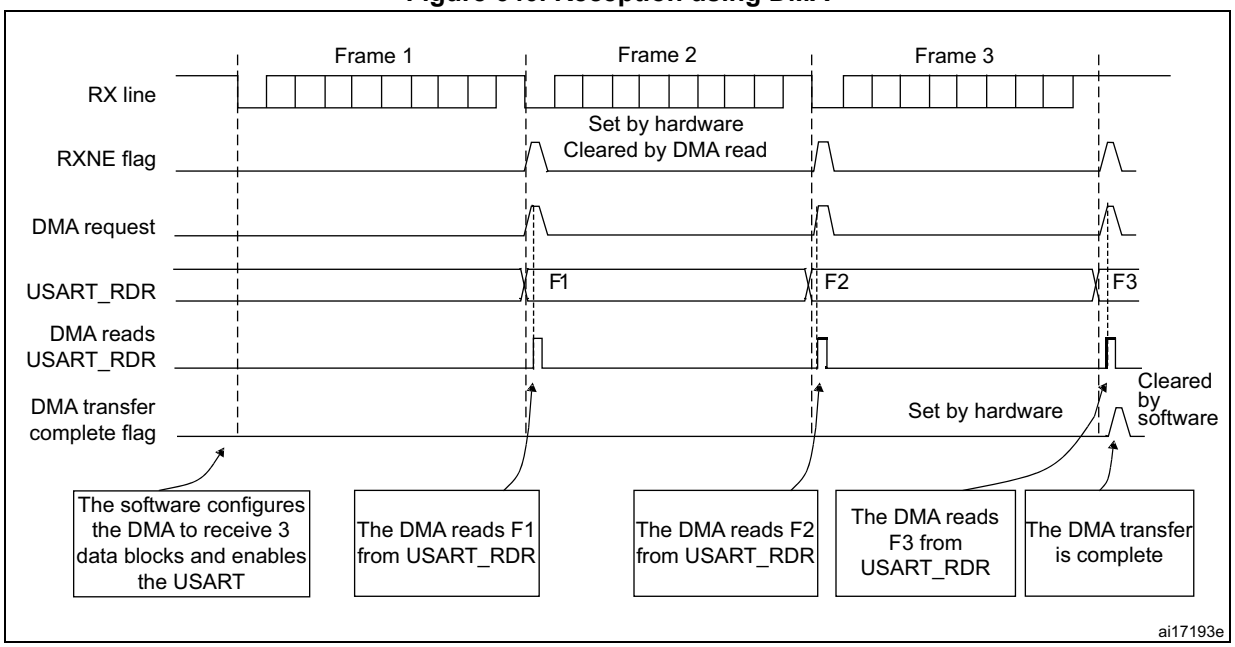
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 640. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

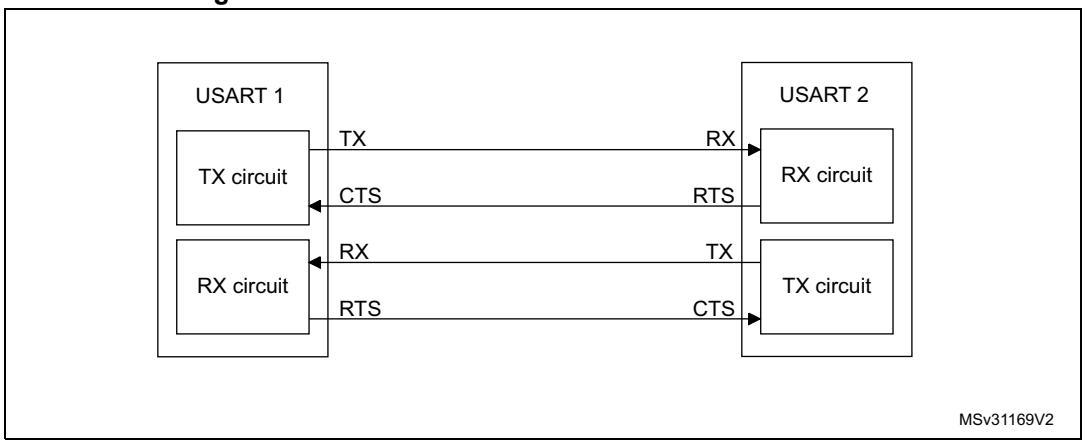
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

43.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 641* shows how to connect two devices in this mode:

Figure 641. Hardware flow control between two USARTs

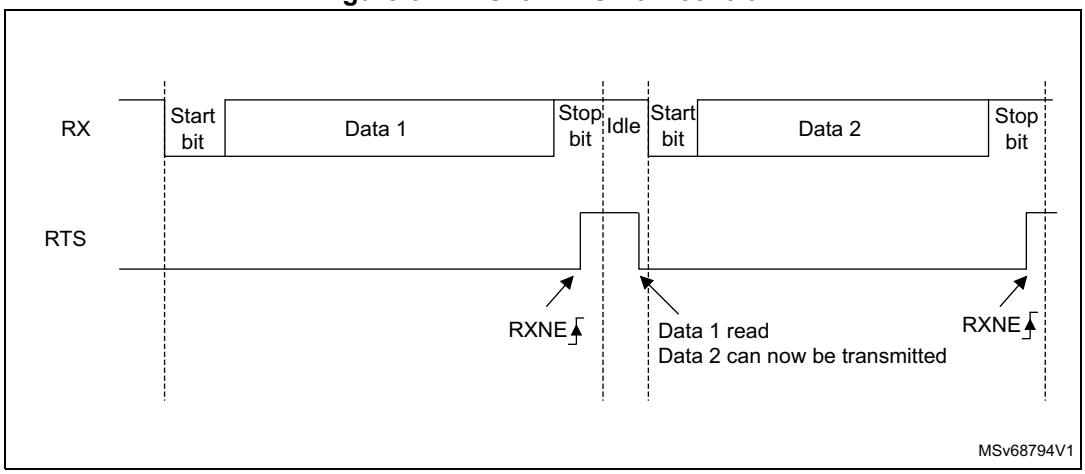


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 642* shows an example of communication with RTS flow control enabled.

Figure 642. RS232 RTS flow control



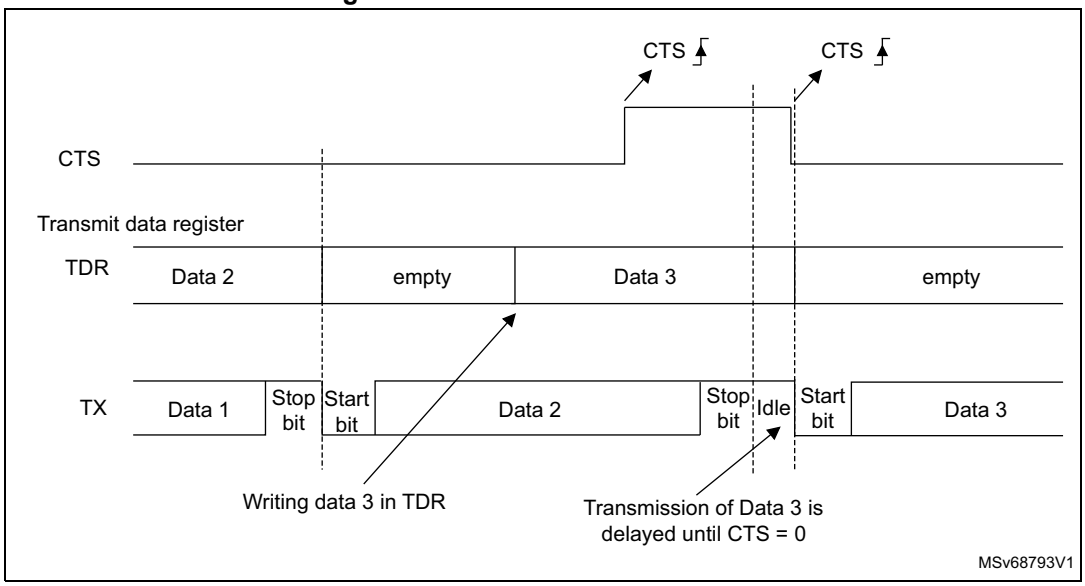
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 643* shows an example of communication with CTS flow control enabled.

Figure 643. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

43.5.22 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 644 shows the USART behavior when the wake-up event is verified.

Figure 644. Wake-up event verified (wake-up event = address match, FIFO disabled)

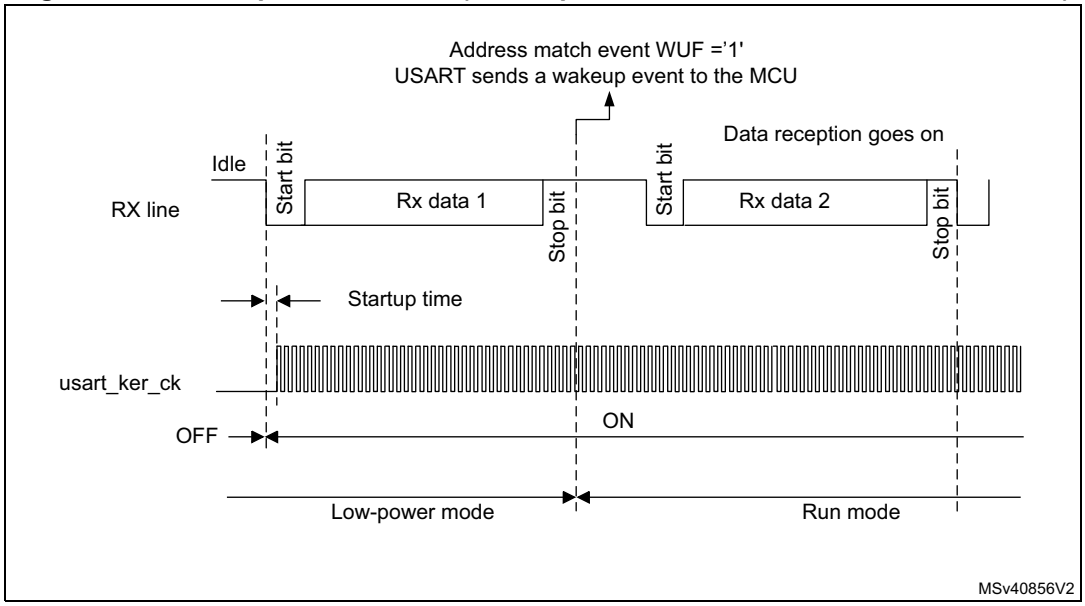
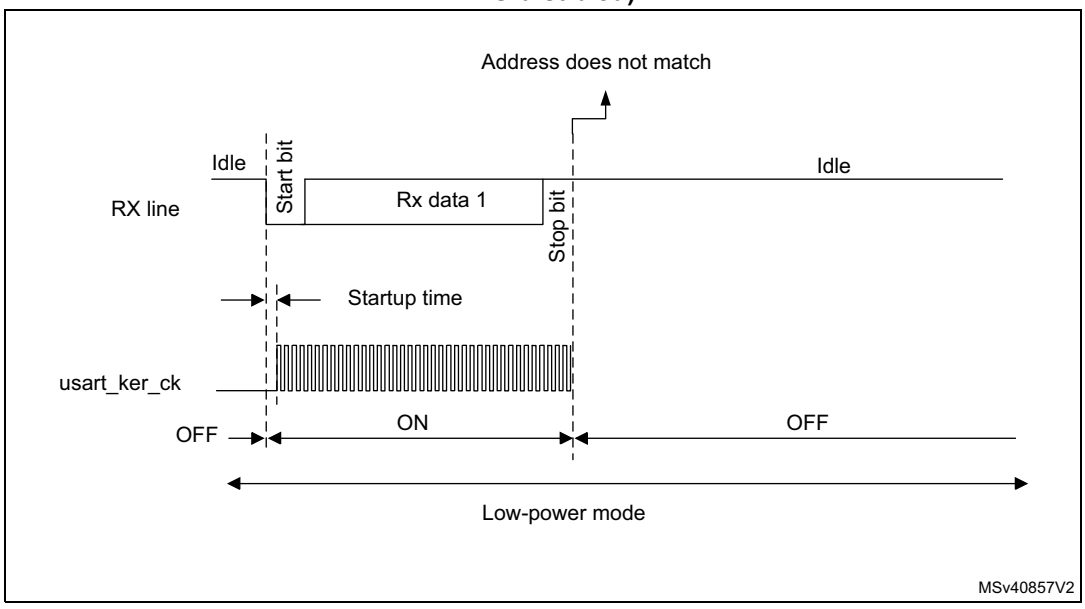


Figure 645 shows the USART behavior when the wake-up event is not verified.

Figure 645. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as a wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 43.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 443: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

43.6 USART in low-power modes

Table 446. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 43.4: USART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

43.7 USART interrupts

Refer to *Table 447* for a detailed description of all USART interrupt requests.

Table 447. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 447. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		No	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 43.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. When OVRDIS = 0.

43.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

43.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 43.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

43.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register
Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 43.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: USART interrupt is generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled
Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.
In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode
When this bit is cleared, the USART cannot wake up the MCU from low-power mode.
When this bit is set, the USART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.
Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

43.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 625* and *Figure 626*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

43.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the USART is disabled (UE = 0).
Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 43.4: USART implementation on page 1799.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

43.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 43.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

43.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

43.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 43.4: USART implementation.

43.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 43.4: USART implementation.

43.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

43.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 43.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed, and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by a write to the USART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 43.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

43.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 43.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 43.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 43.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

43.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

43.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 43.5.1: USART block diagram*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

43.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 43.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

43.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 43.4: USART implementation.

43.8.16 USART register map

Table 448. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 448. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_ RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

44 Low-power universal asynchronous receiver transmitter (LPUART)

44.1 LPUART introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

44.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

44.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 449. Instance implementation on STM32C5

Instance	STM32C53x/C54x	STM32C55x/C56x	STM32C59x/C5Ax
USART1	Full	Full	Full
USART2	Full	Full	Full
USART3	-	Full	Full
USART6	-	-	Full
UART4	Basic	Basic	Basic
UART5	Basic	Basic	Basic
UART7	-	-	Basic
LPUART1	Low-power	Low-power	Low-power

Table 450. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-

Table 450. USART/LPUART features (continued)

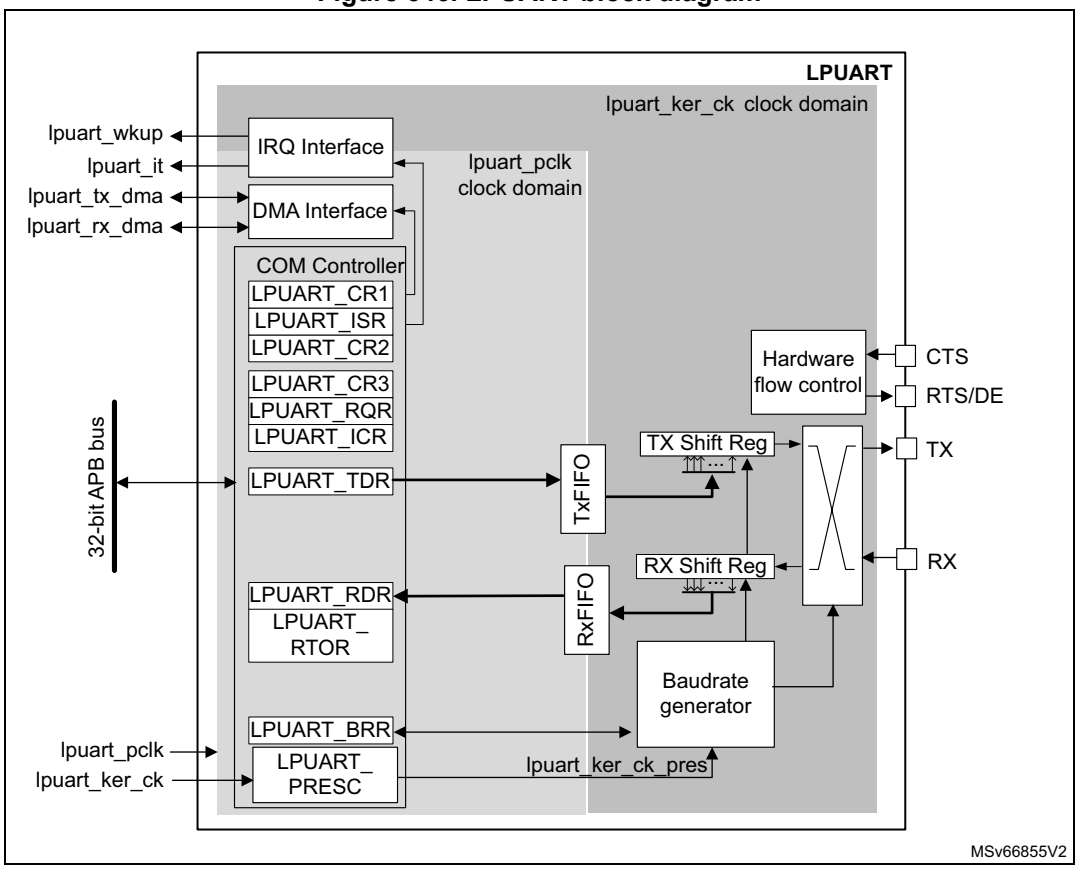
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. X = supported.
2. Wake-up supported from Stop mode.

44.4 LPUART functional description

44.4.1 LPUART block diagram

Figure 646. LPUART block diagram



44.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

    RX is the serial data input.

  – **TX (**Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pins:

  – **CTS (**Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request to send)

    When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 451* and *Table 452* for the list of LPUART input/output pins and internal signals.

Table 451. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 452. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 452. LPUART internal input/output signals (continued)

Signal name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request

44.4.3 LPUART clocks

The simplified block diagram given in *Section 44.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

44.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 620*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

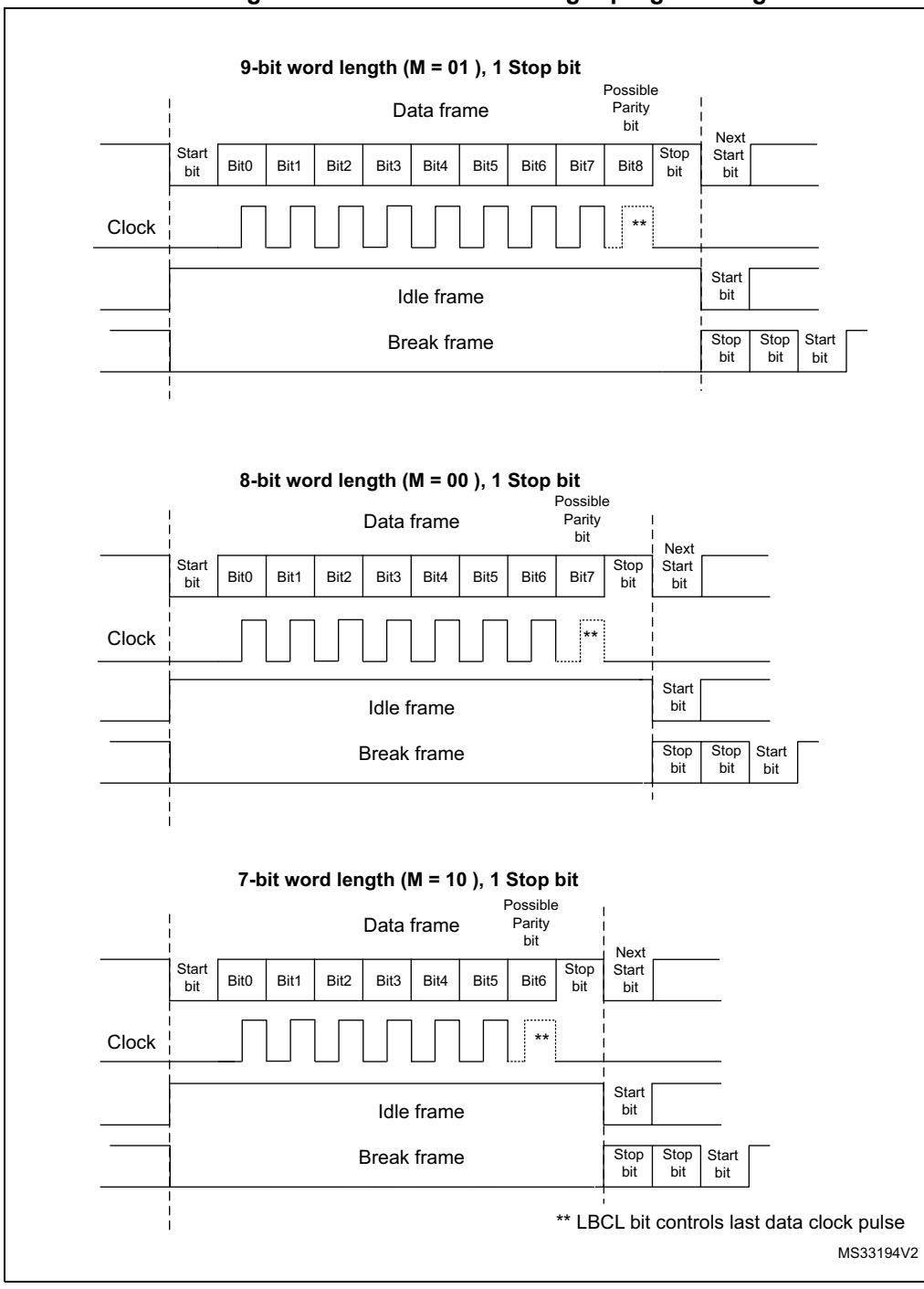
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 647. LPUART word length programming



44.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

44.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 44.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

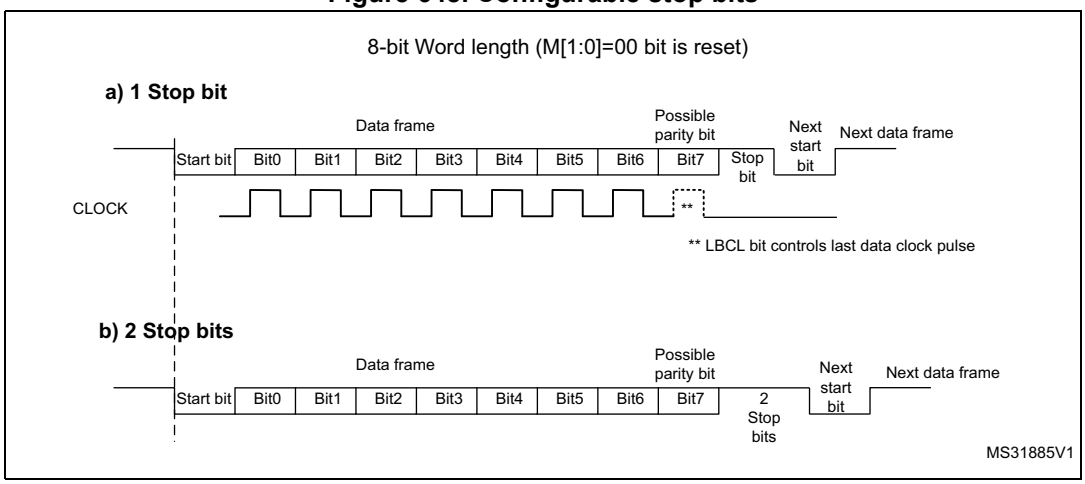
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 648. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 44.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

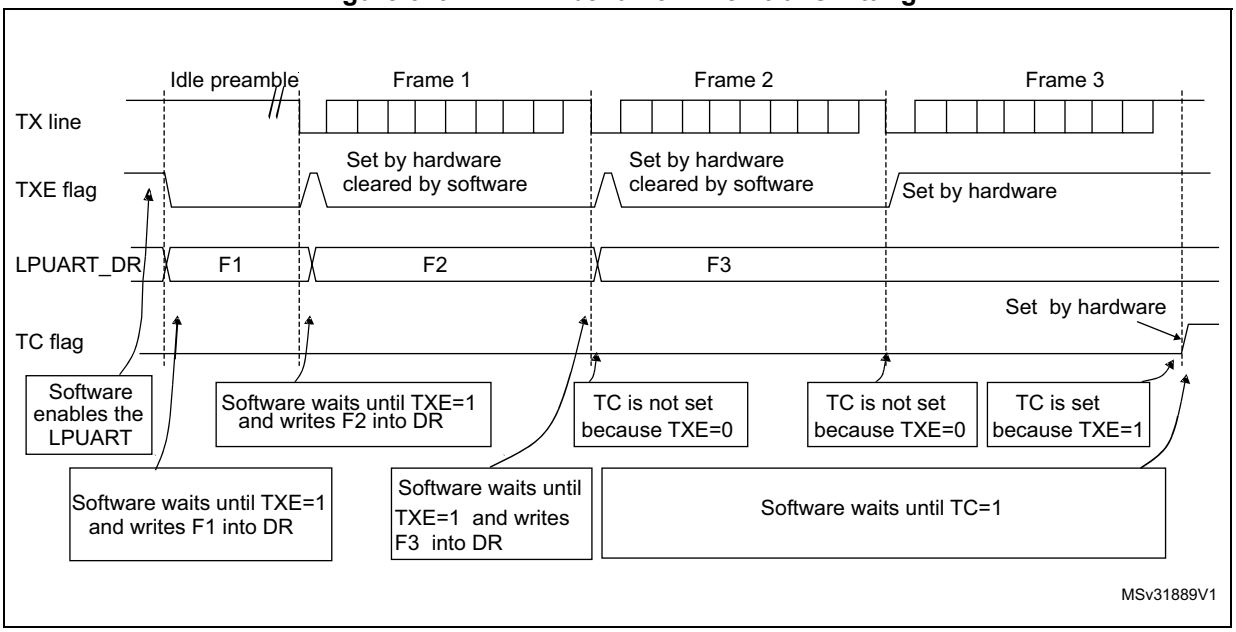
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 649: TC/TXE behavior when transmitting*).

Figure 649. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 647*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

44.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 44.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

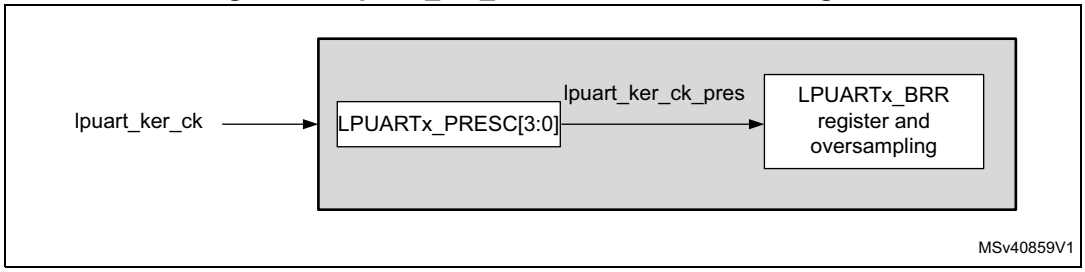
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 650. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

44.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 453. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	**Desired**	**Actual**	**Value programmed in the baud rate register**	**% Error = (Calculated – Desired) B.rate / Desired B.rate**
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

44.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 454*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 454. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 454 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

44.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

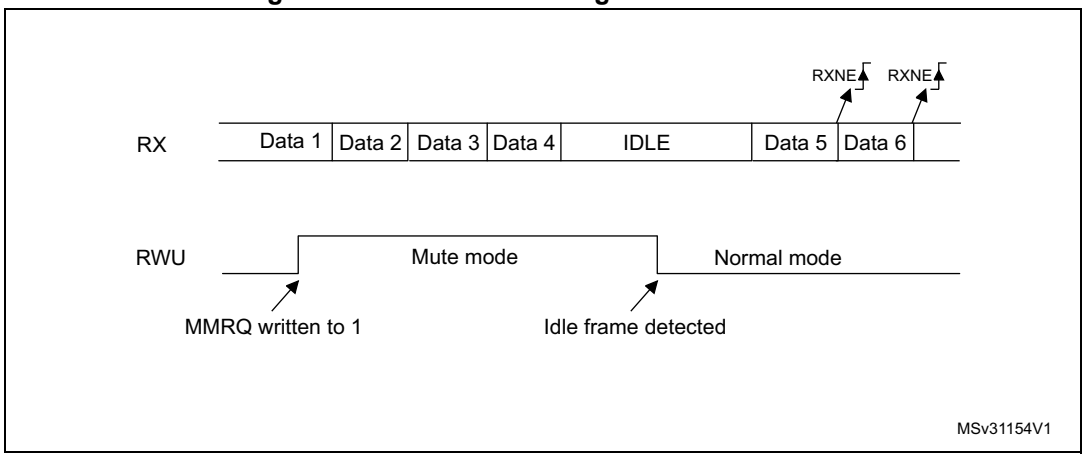
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 651*.

Figure 651. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

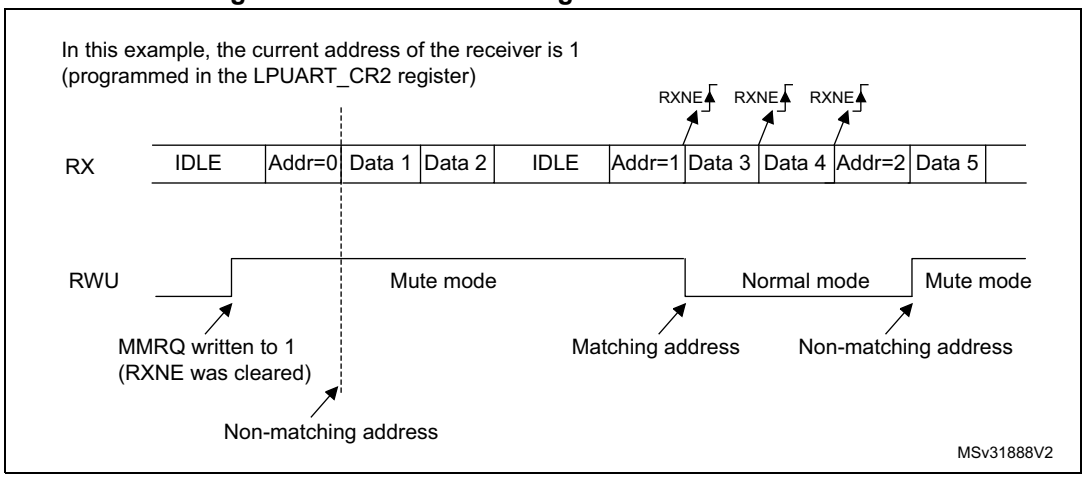
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 652*.

Figure 652. Mute mode using address mark detection



44.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 455*.

Table 455: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

44.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

44.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 44.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 44.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

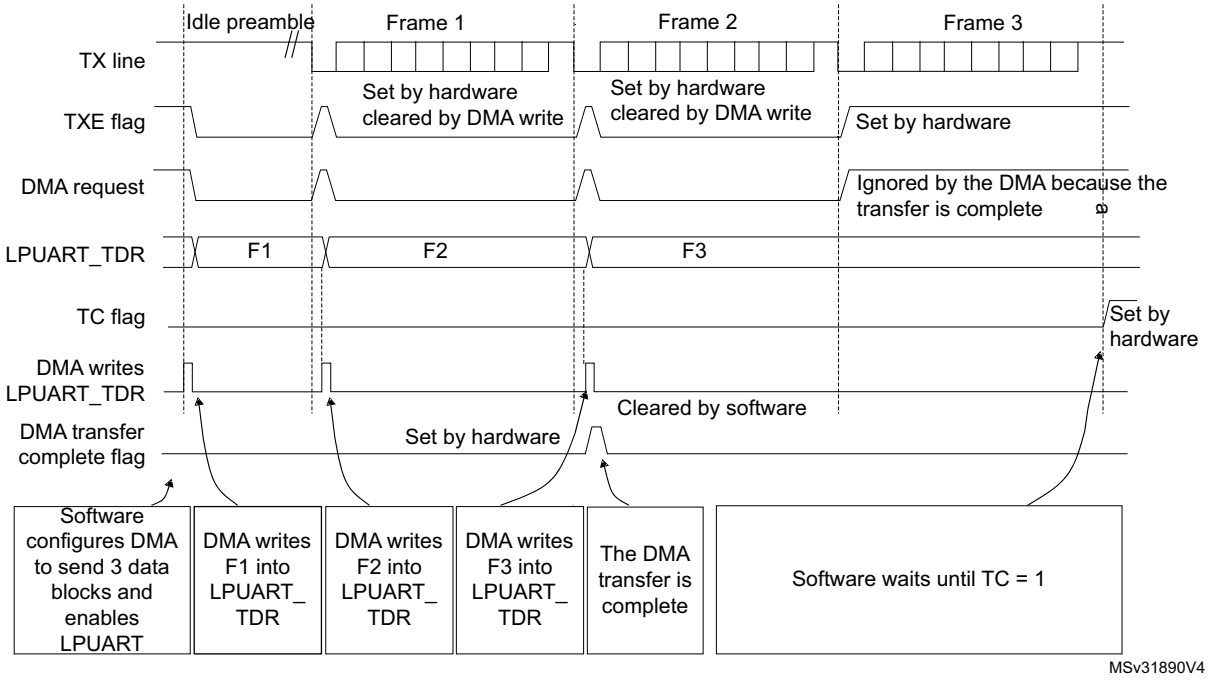
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 653. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

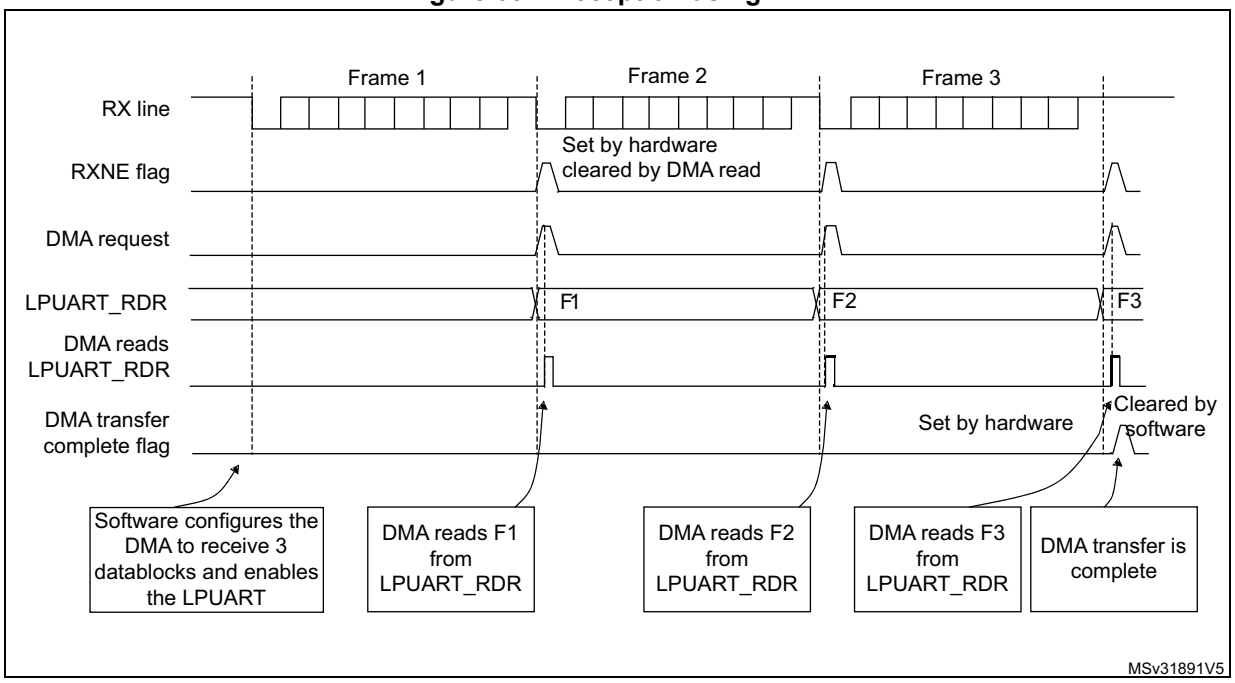
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 654. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

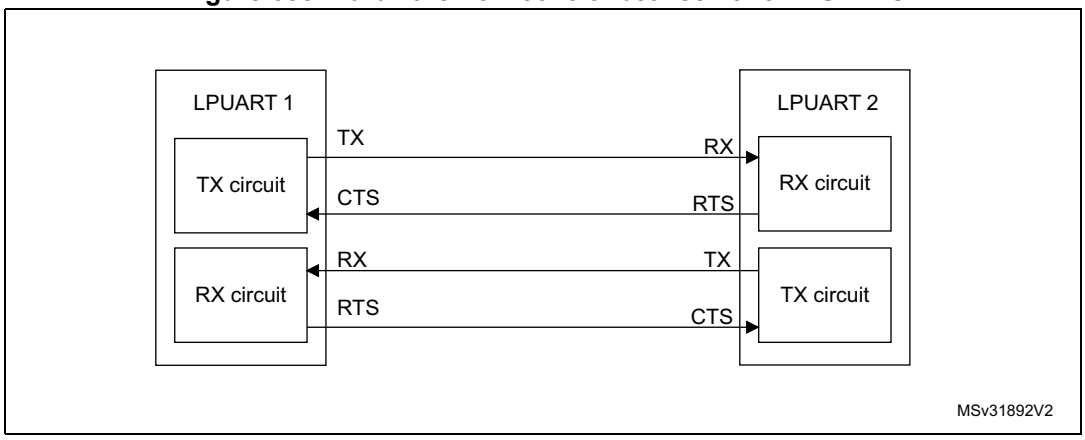
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

44.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 655* shows how to connect two devices in this mode.

Figure 655. Hardware flow control between two LPUARTs

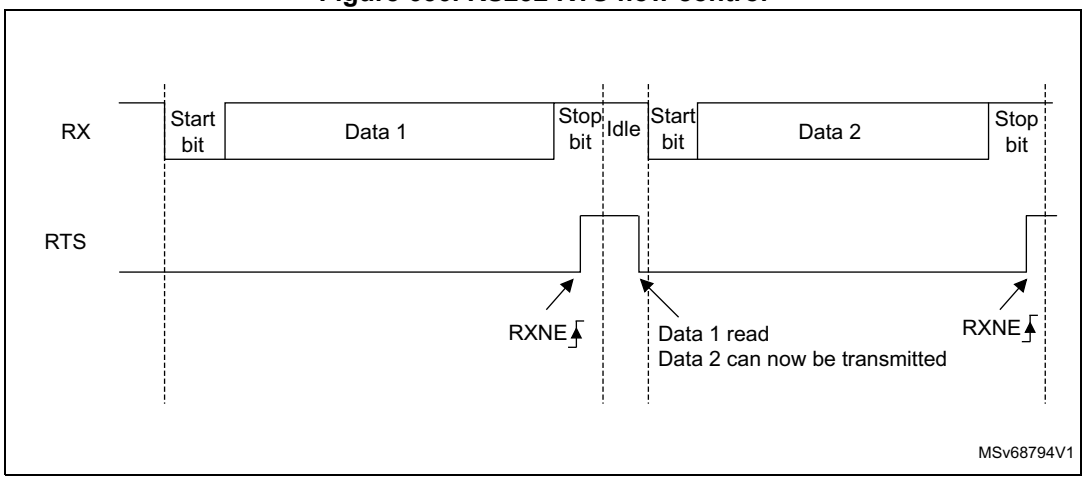


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 656* shows an example of communication with RTS flow control enabled.

Figure 656. RS232 RTS flow control



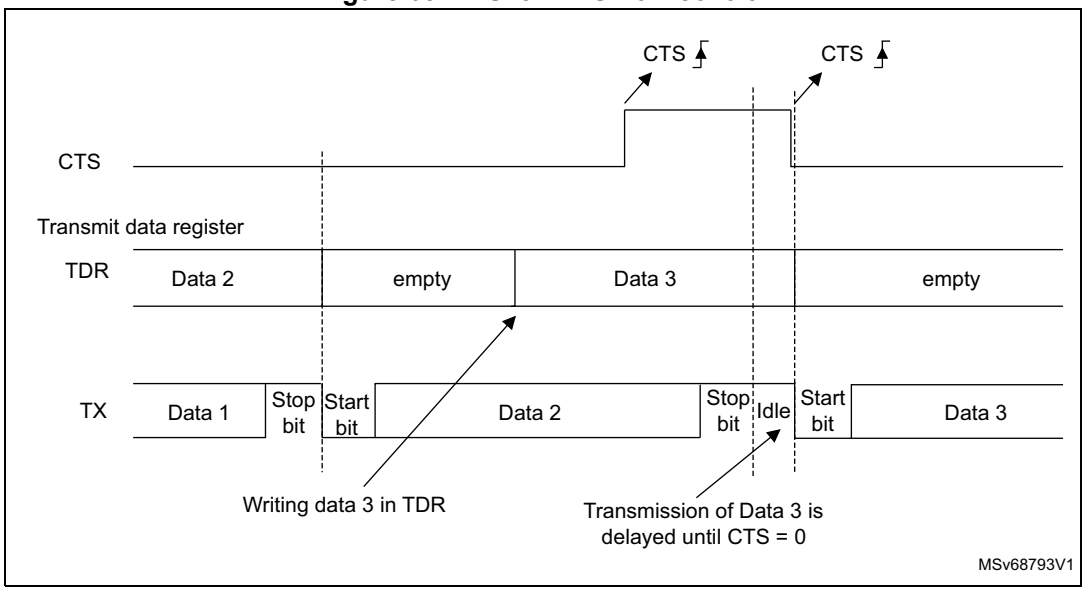
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 657* shows an example of communication with CTS flow control enabled.

Figure 657. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ($f_{lpuart_ker_ck_pres}$) cycles:

- The driver enable assertion time equals
  - (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  - (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

44.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the lpuart_pclk is gated, the LPUART provides a wake-up interrupt (**lpuart_wkup**) if a specific action requiring the activation of the **lpuart_pclk** clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many

LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **lpuart_wkup** interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and the **lpuart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure that the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not woken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 658 shows the behavior when the wake-up event is verified.

Figure 658. Wake-up event verified (wake-up event = address match, FIFO disabled)

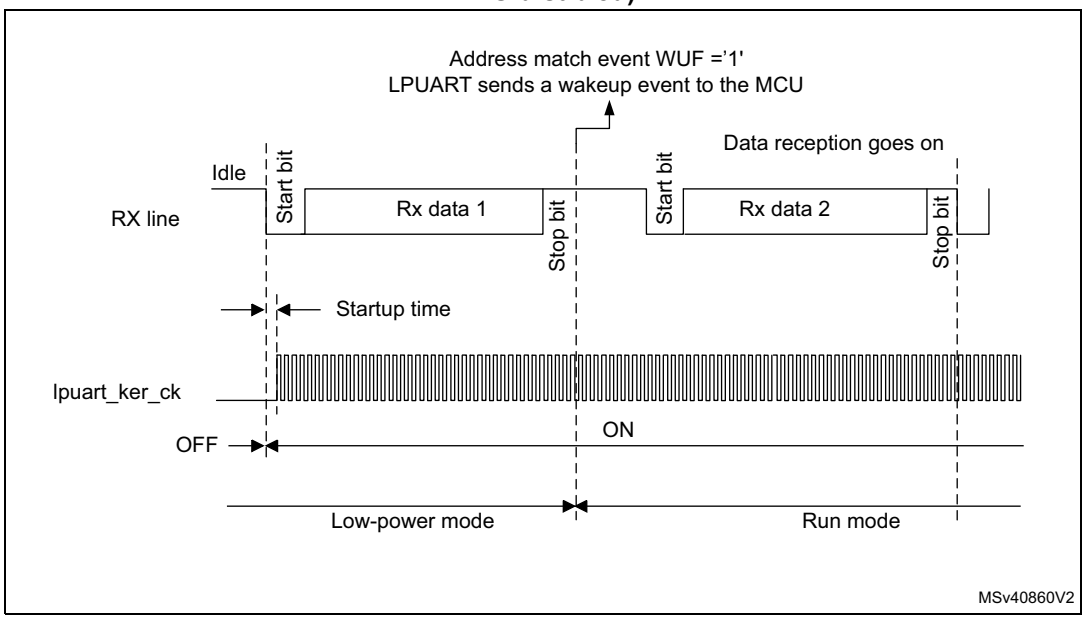
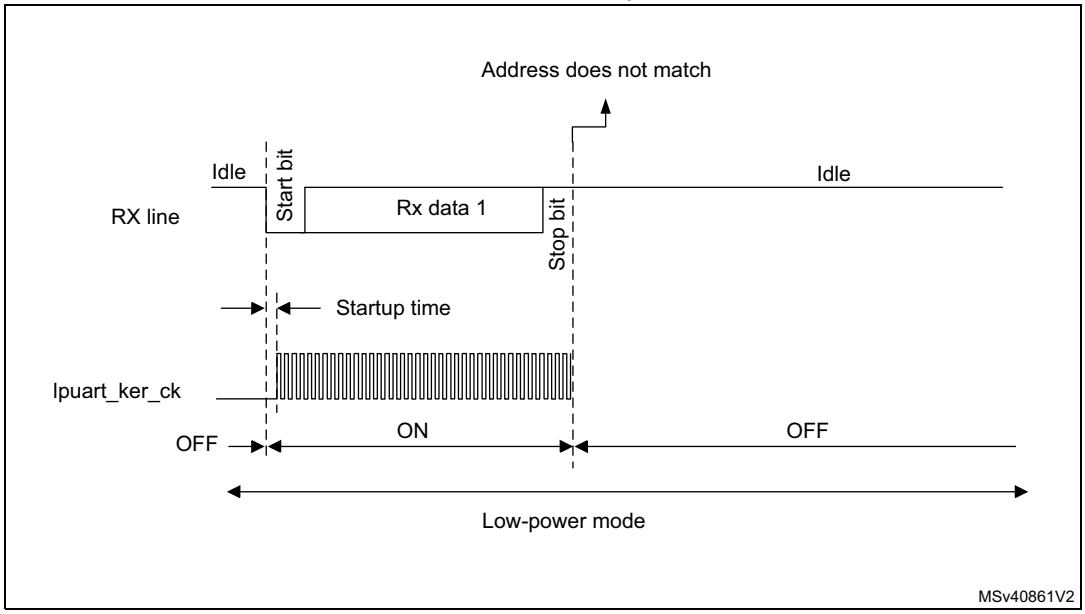


Figure 659 shows the behavior when the wake-up event is not verified.

Figure 659. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 44.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 454: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

44.5 LPUART in low-power modes

Table 456. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 44.3: LPUART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

44.6 LPUART interrupts

Refer to *Table 457* for a detailed description of all LPUART interrupt requests.

Table 457. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 44.3: LPUART implementation* for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

44.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

44.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 43.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 44.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

44.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 43.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 44.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

44.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

44.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 44.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 44.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

44.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
									rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 44.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 44.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

44.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

44.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

44.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.
An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 44.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in active mode
1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE = 1in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by a write to the LPUART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

44.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 44.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

44.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value.* Refer to *Section 44.3: LPUART implementation.*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

44.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 44.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

44.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 44.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

44.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 44.3: LPUART implementation.

44.7.14 LPUART register map

Table 458. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	LPUART_CR1FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0	0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0				0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 458. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

45 Serial peripheral interface (SPI)

45.1 SPI introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

45.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase with optional sampling delay when reading master data
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode

- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

45.3 SPI implementation

The table below describes the SPI implementation.

Table 459. SPI features

SPI feature	SPI1, SPI2, SPI3(1) (full-featured instances)
Data and CRC size	Configurable from 4 to 32 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits
FIFO size	16x8 bits
FIFO threshold	1 - 16 data
Number of data control (TSIZE)	Up to 65535
I2S feature	Yes
Autonomous in Stop modes with wake-up capability	No
Autonomous in LP-Stop and Standby modes with wake-up capability	No

1. SPI3 is available only on STM32C55x, STM32C562, STM32C59x, and STM323C5A3 devices.

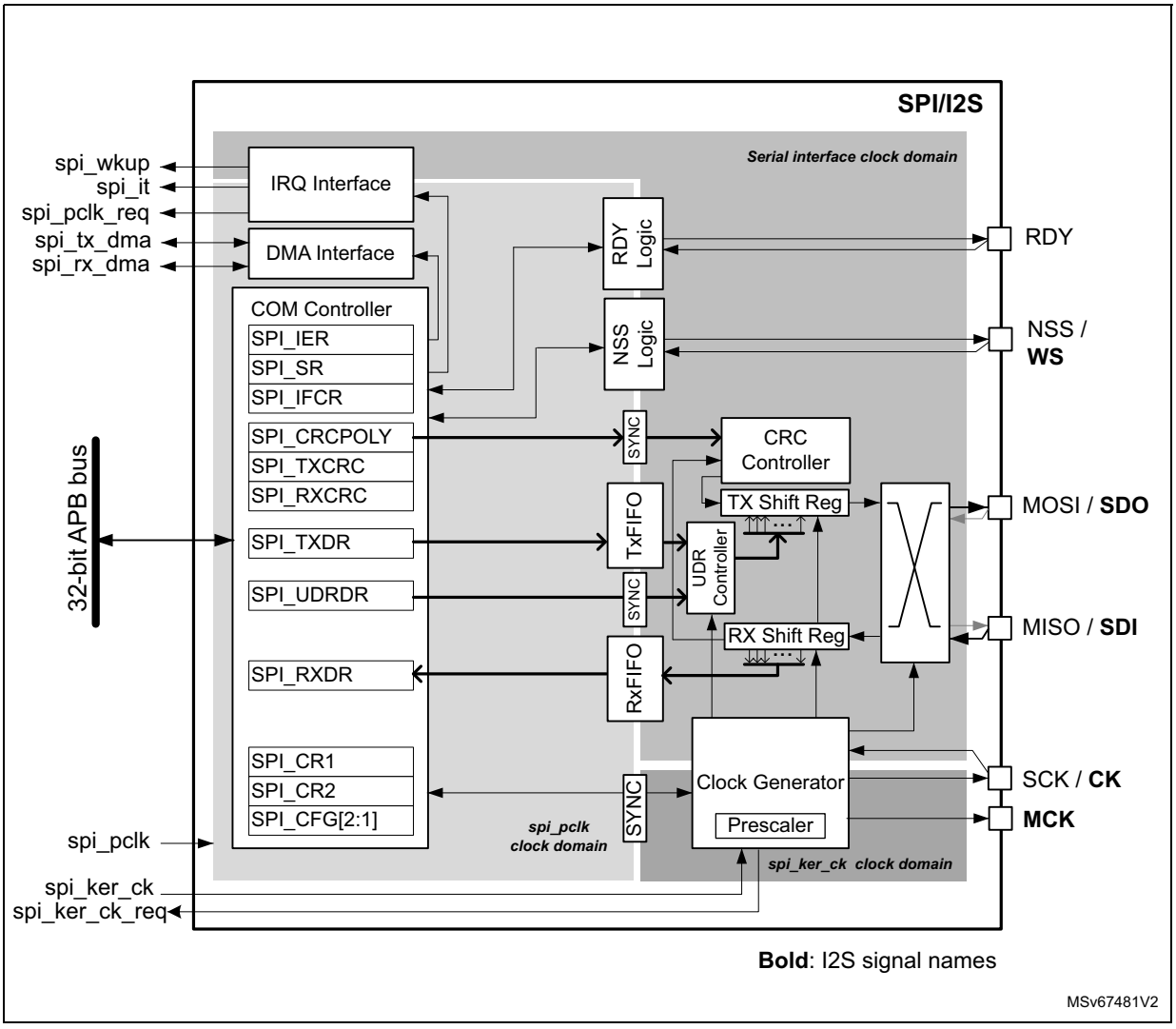
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 463: SPI wake-up and interrupt requests.

45.4 SPI functional description

45.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 660*.

Figure 660. SPI/I2S block diagram



The simplified scheme of *Figure 660* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user

has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 45.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 45.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

45.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 45.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features an additional I2S specific MCK signal. For more details about I2S signals, see *Section 45.9.2*.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 460* and *Table 461* for the list of SPI input / output pins and internal signals.

Table 460. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
MISO/SDI(2)	Input/output	Master data input / slave data output
MOSI/SDO(2)	Input/output	Master data output / slave data input
SCK/CK	Input/output	Master clock output / slave clock input
NSS/WS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output
MCK	Output	I2S master frequency output

1. Refer to the section *Section 45.9.2: Pin sharing with SPI function* for details.
2. Functionality of MOSI/SDO and MISO/SDI pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 461. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request

45.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

45.4.4 Communications between one master and one slave

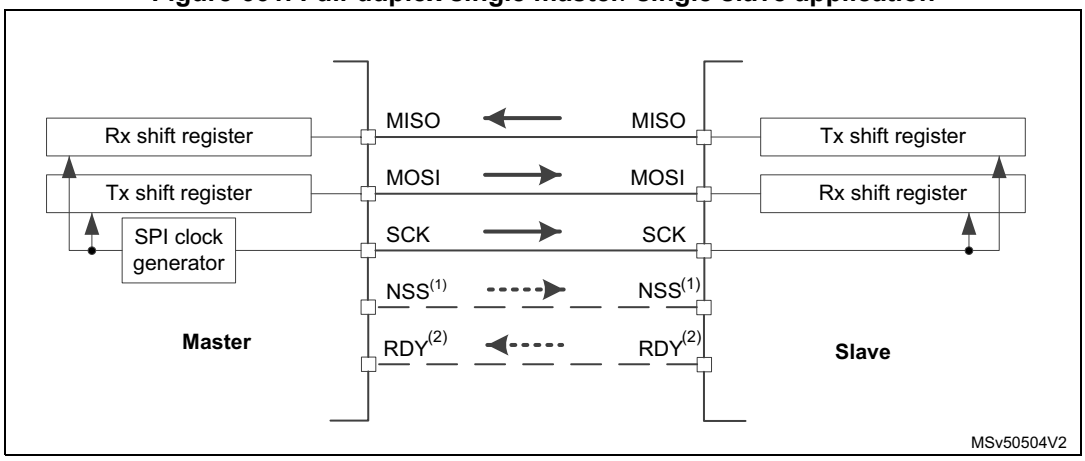
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 661. Full-duplex single master/ single slave application

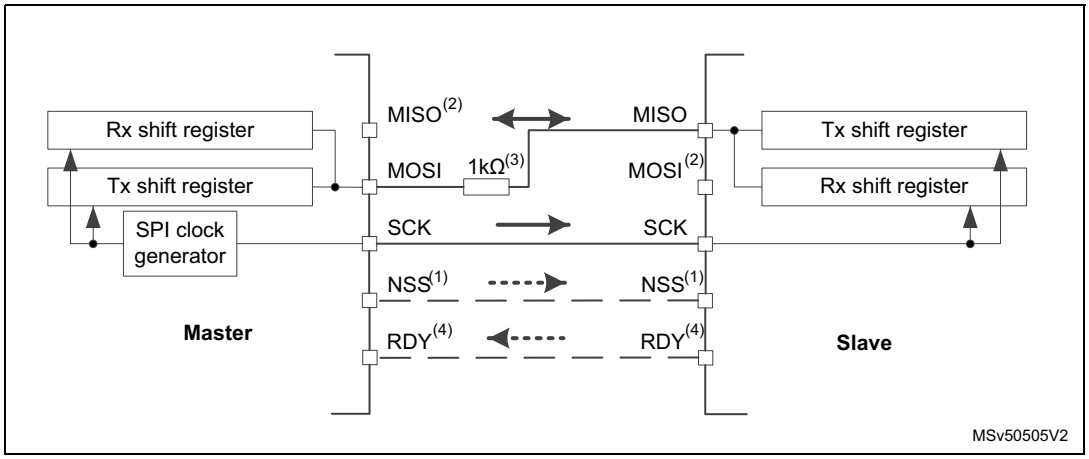


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 45.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 662. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 45.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.
4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 45.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

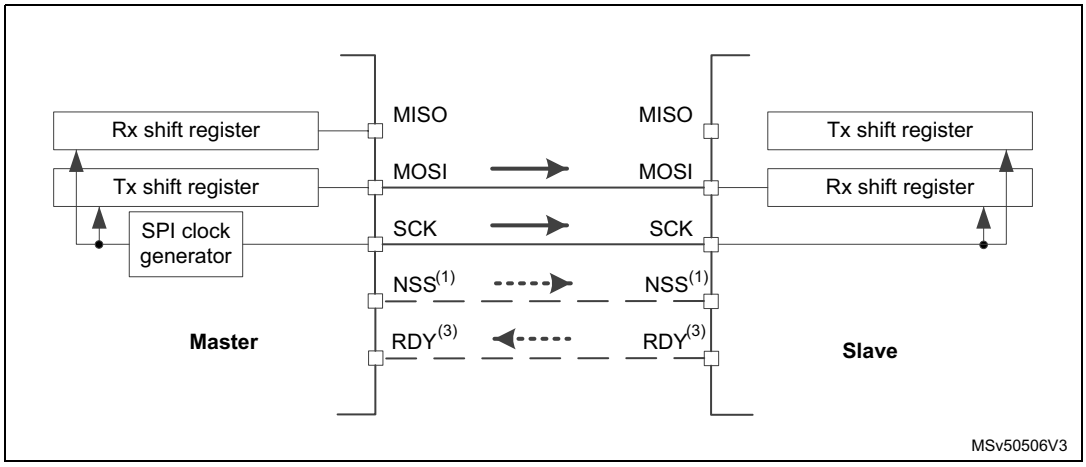
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 45.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the

HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 663. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 45.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

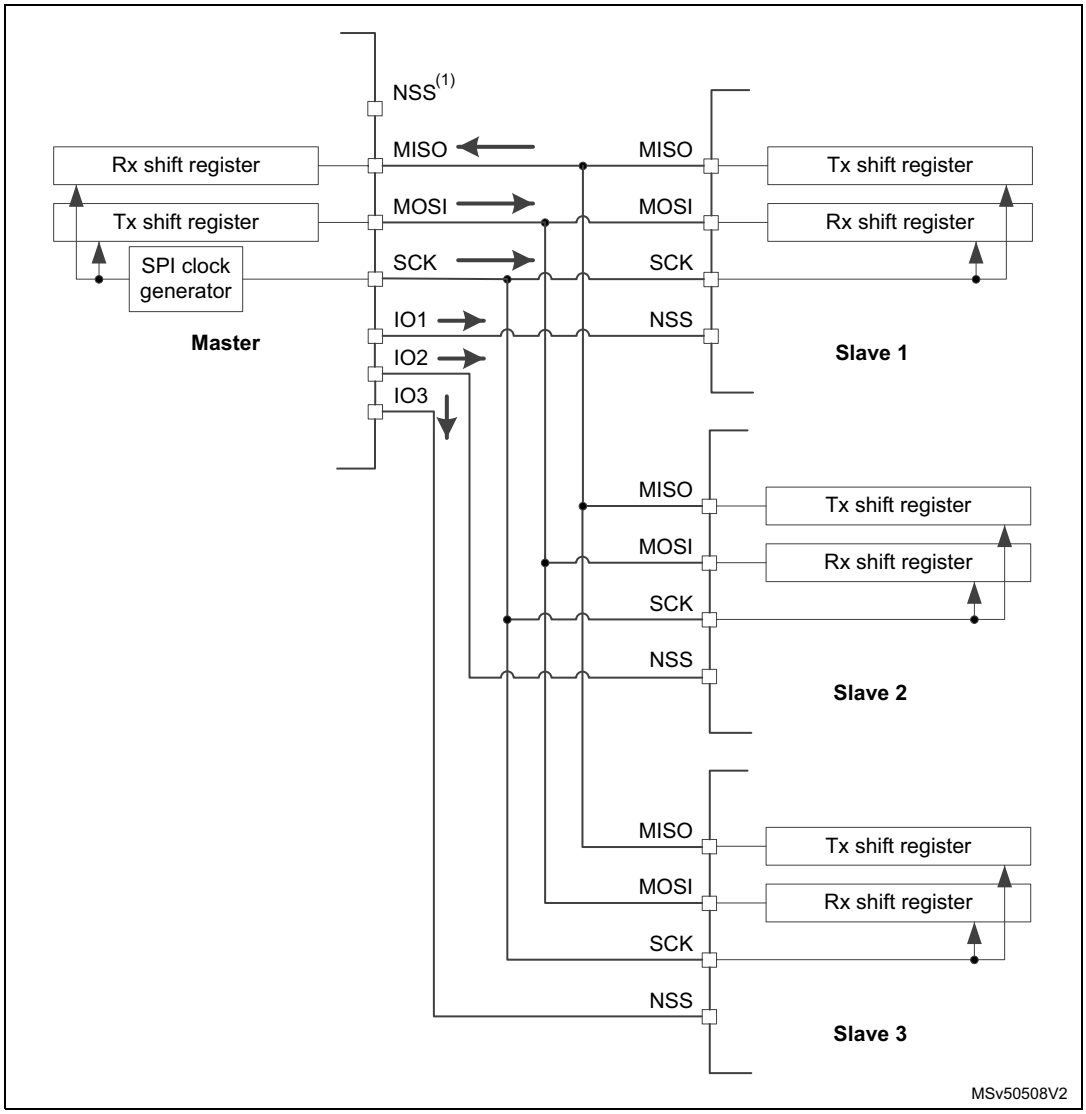
45.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 664.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 664. Master and three independent slaves connected in star topology



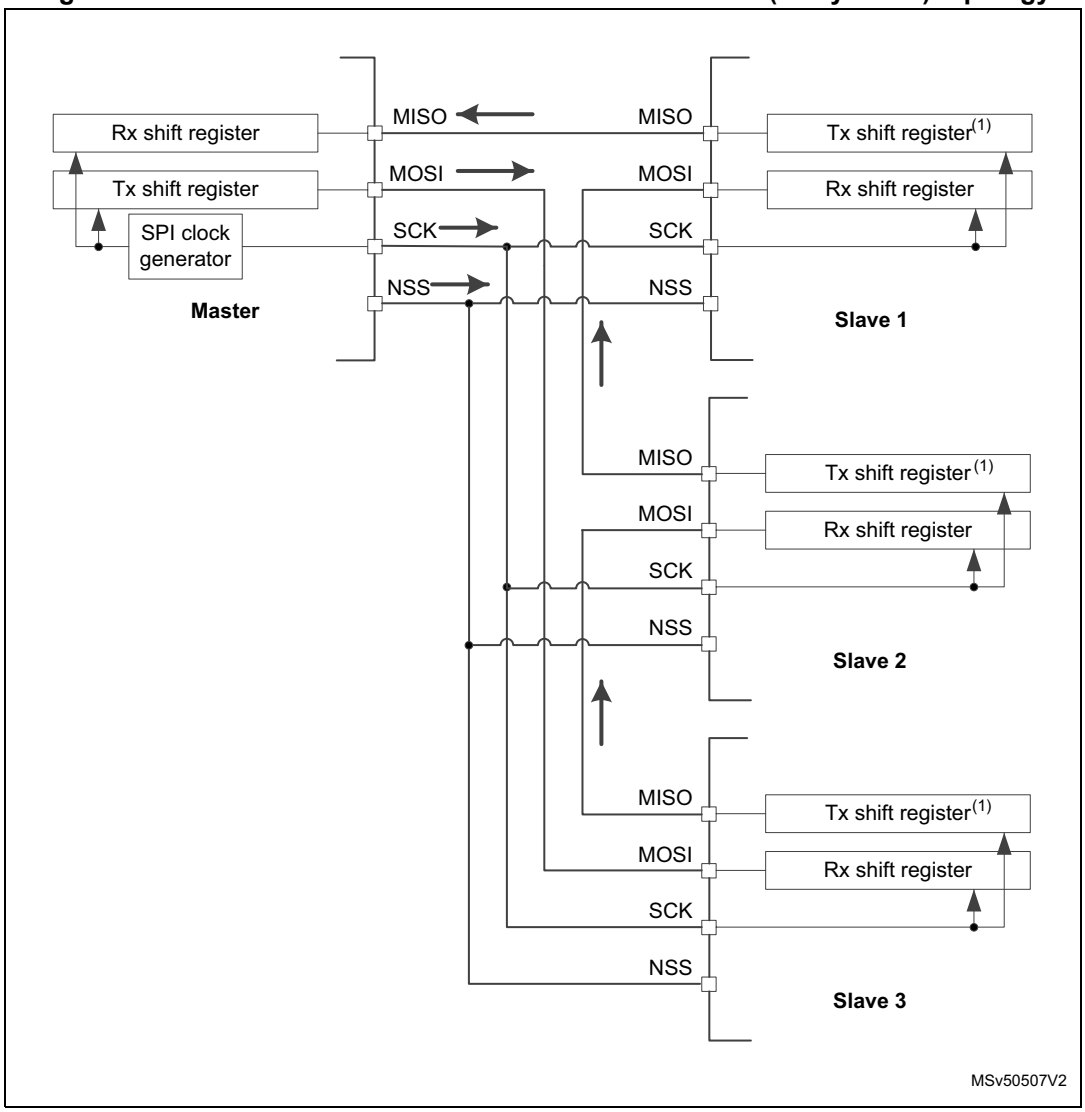
1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 45.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves at the same time when a circular topology is applied (see *Figure 665*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.

Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 665. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

45.4.6 Multimaster communication

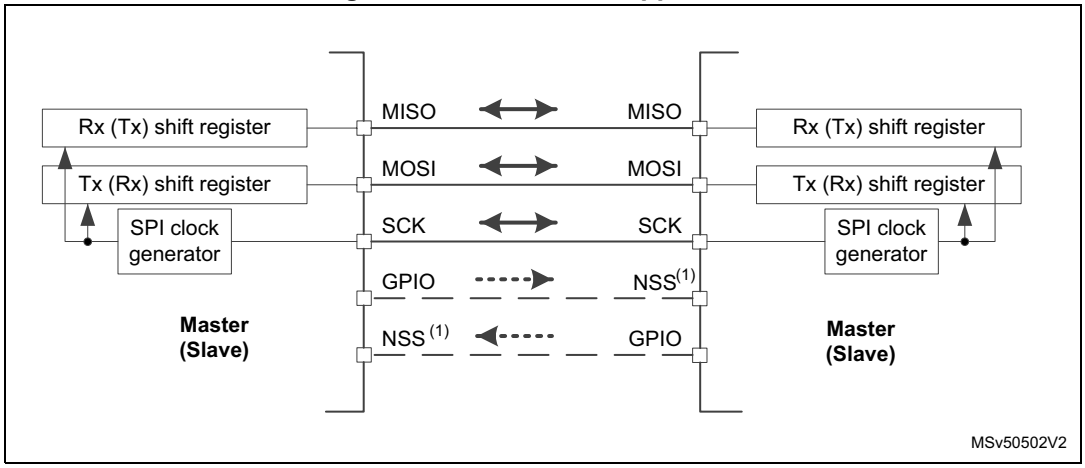
Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is

complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 666. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control since the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

45.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

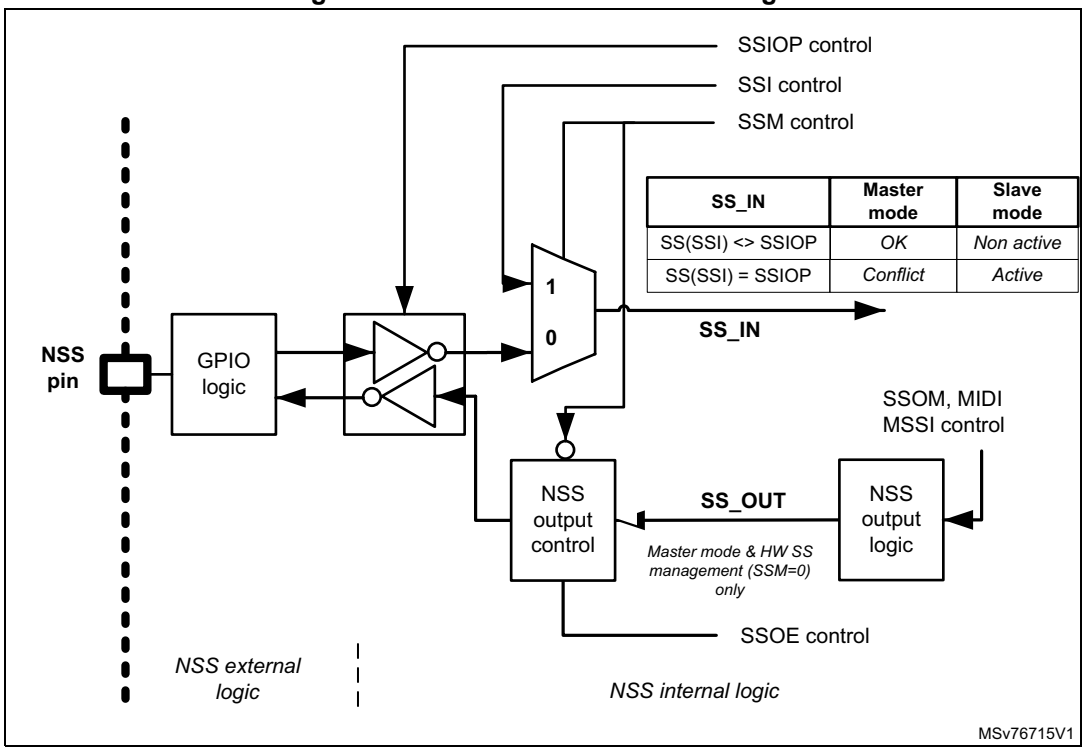
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

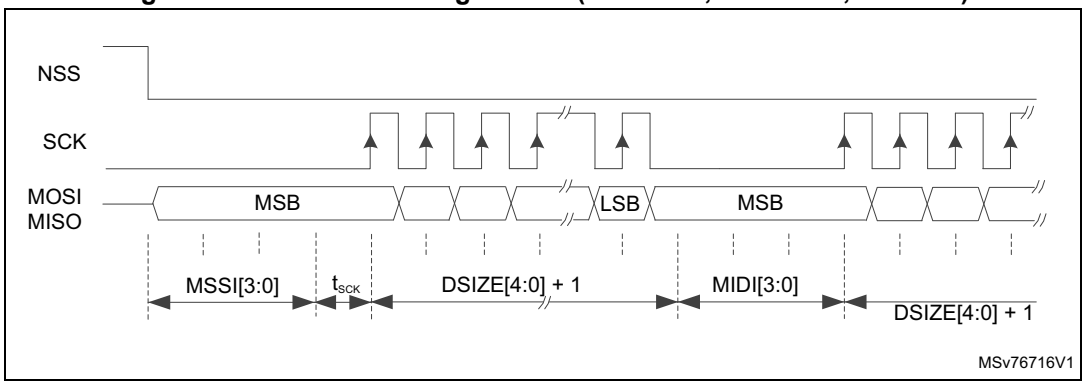
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 667. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 668*).

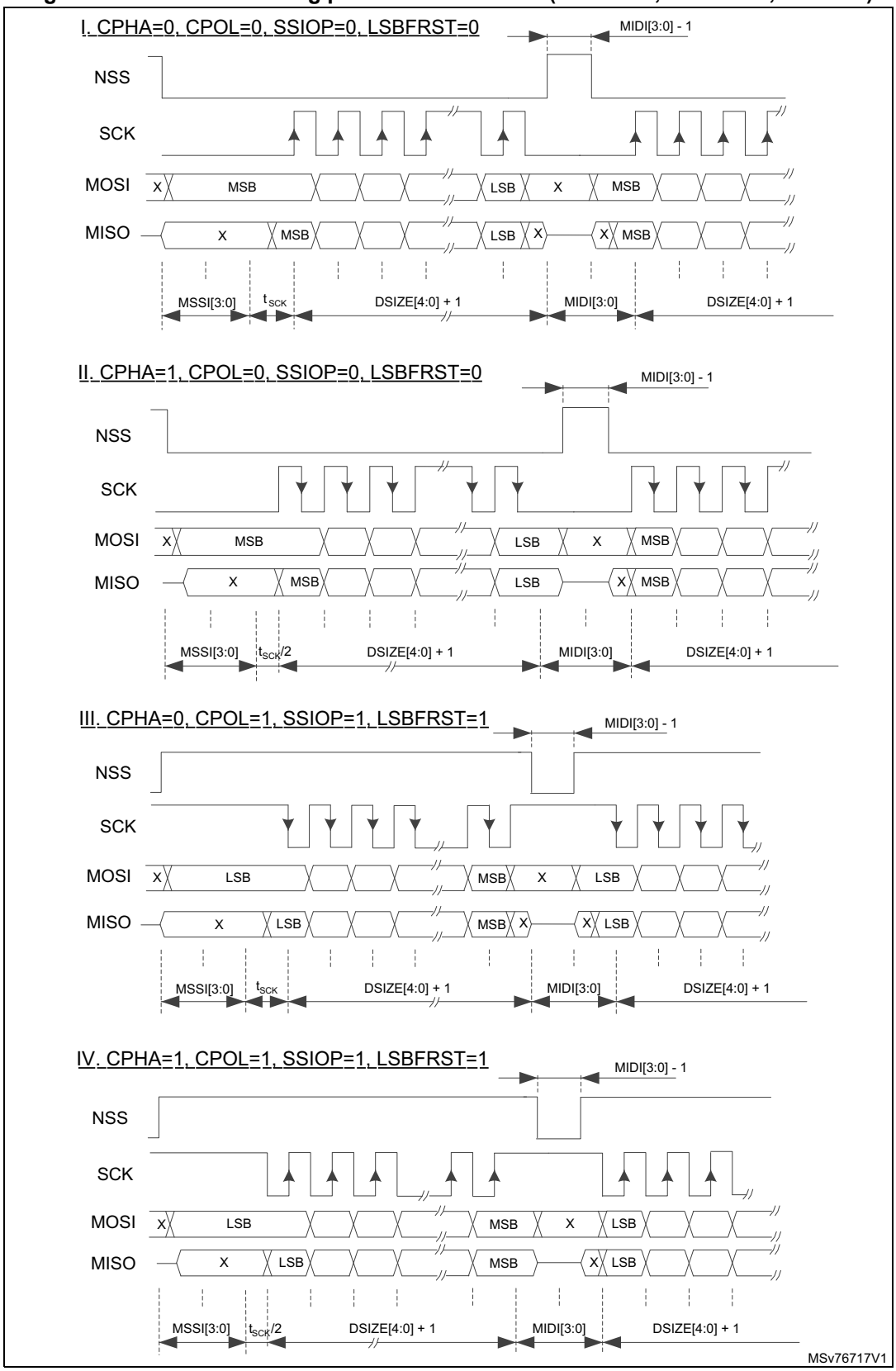
Figure 668. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 669*.

Figure 669. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

45.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

45.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

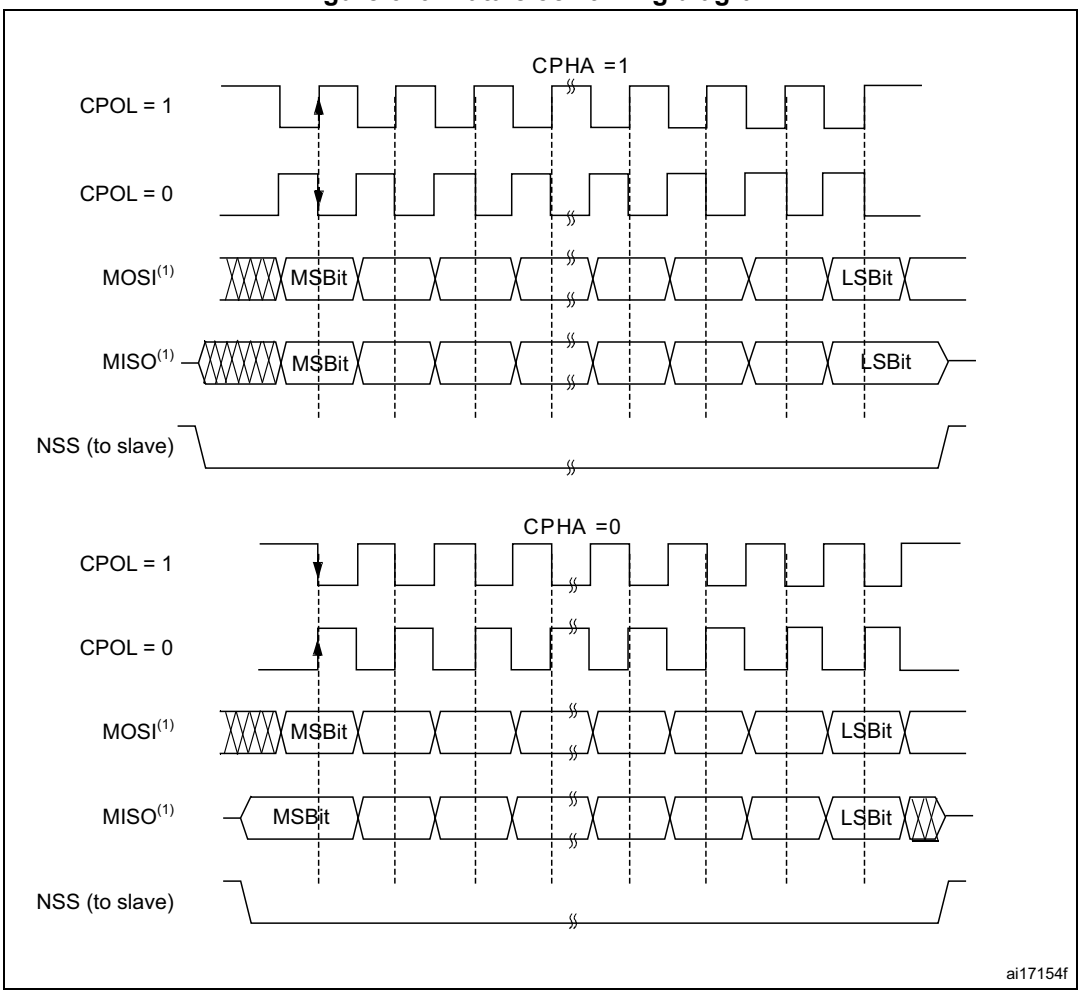
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 670*).

Figure 670 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 670. Data clock timing diagram



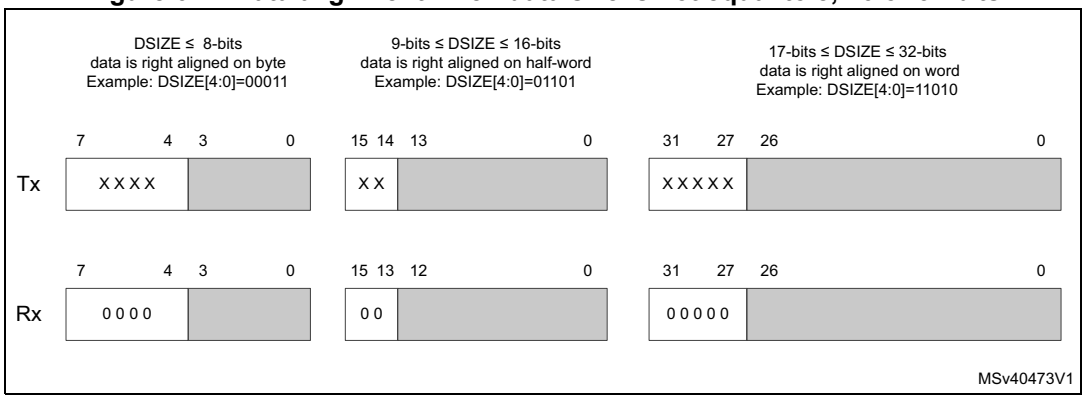
1. The order of data bits depends on the LSBFRST bit setting.
2. The input data sampling can be postponed by a half SCK cycle on the master MISO input (refer to the description of the DRDS bit of the SPI_CFG1 register).

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 45.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 671*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 45.4.12: SPI data transmission and reception procedures*.

Figure 671. Data alignment when data size is not equal to 8, 16 or 32 bits



45.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 45.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

45.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 45.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 45.4.14*.

45.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the

content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO

content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

45.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed.

TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.

2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

45.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

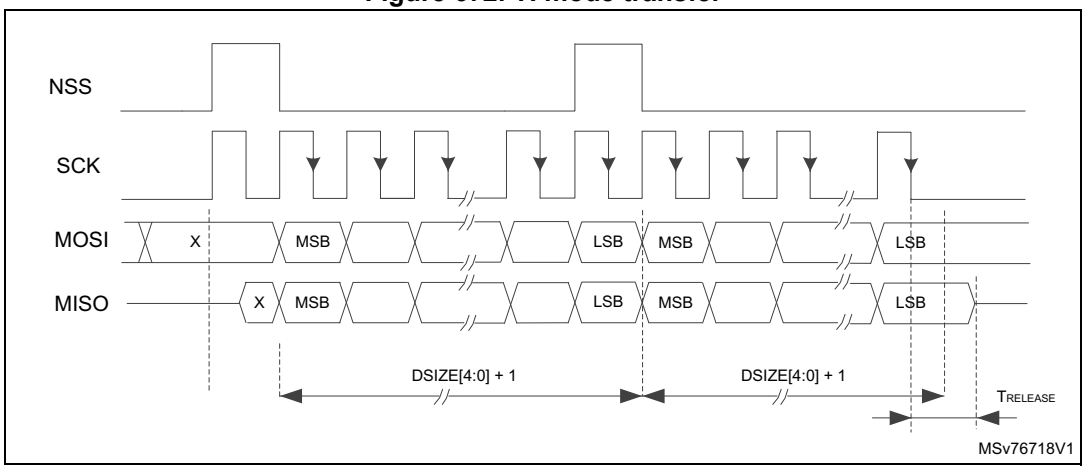
Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

45.5 SPI specific modes and control

45.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 672*.

Figure 672. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

45.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all

data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

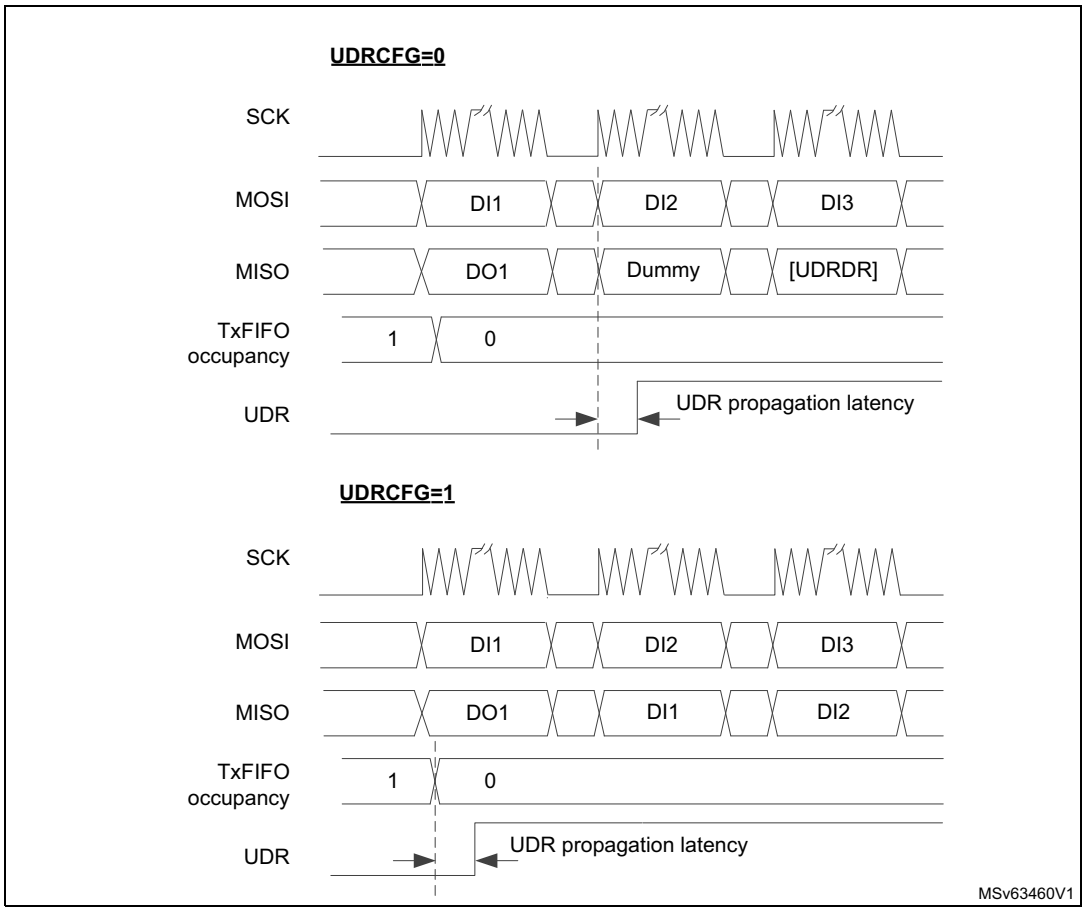
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 673*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 665*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the last received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 673. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

45.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 45.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

45.6 SPI in low-power modes

Table 462. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the SPI registers is kept.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 45.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

45.7 SPI interrupts

Table 463 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 463. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability[1][2]
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC[3]		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 45.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

45.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length can be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, overrun, and frame errors
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

45.9 I2S functional description

45.9.1 I2S general description

The block diagram shown on *Figure 660* also applies for I2S mode.

The SPI/I2S block can work in I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA, and parts of interrupt signaling are shared with the SPI function. The low-power mode function is also available in I2S mode (refer to *Section 45.6: SPI in low-power modes* and *Section 45.10: I2S interrupts*).

a. Not always available, refer to *Section 45.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

45.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: serial data output (mapped on the MOSI pin) to transmit the audio samples in master, and receive the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 1988*.
- SDI: serial data input (mapped on the MISO pin) to receive the audio samples in master, and transmit the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 1988*.
- WS: word select (mapped on the NSS pin). This is the frame synchronization. It is configured as output in master mode, and as input in slave mode.
- CK: serial clock (mapped on the SCK pin). This is the serial bit clock. It is configured as output in master mode, and as input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: master clock (mapped separately) is used when the I2S is configured in master mode.

45.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value to guarantee the behavior of the I2S/PCM function. *Table 464* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 464. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI/I2S control register 2 (SPI_CR2)	-	Set to reset value
SPI/I2S configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI/I2S configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S receive data register (SPI_RXDR)	The complete register	-

Table 464. Bitfields usable in PCM/I2S mode (continued)

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI/I2S transmitter CRC register (SPI_TXCRC)	-	
SPI/I2S receiver CRC register (SPI_RXCRC)	-	
SPI/I2S underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

45.9.4 Slave and master modes

The SPI/I2S block supports master and slave modes for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

To improve the robustness of the SPI/I2S block in slave mode, the peripheral resynchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB-justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected

Note: This resynchronization mechanism is not available for the I2S LSB-justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

45.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits of the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frame formats are supported:

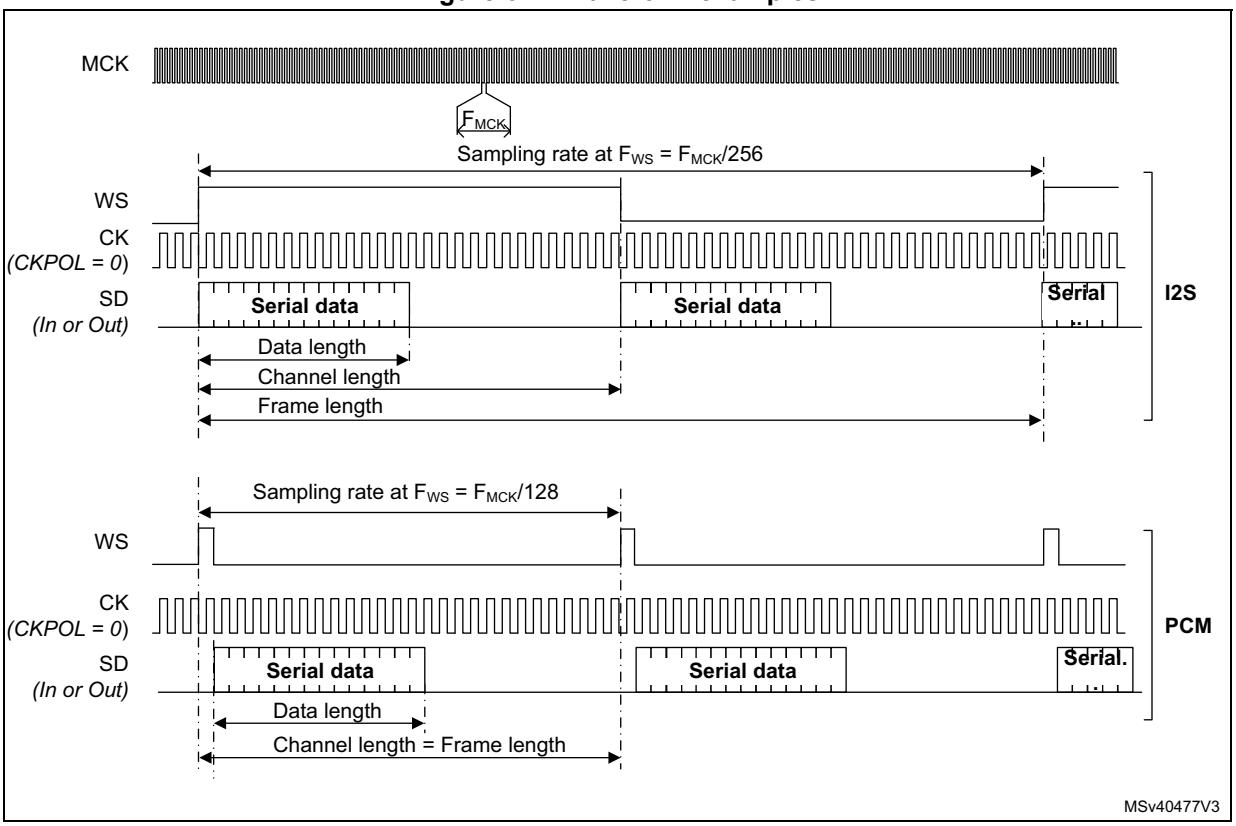
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel(a)
- 32-bit data packed in a 32-bit channel(a)

In PCM master mode, three frame formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 674. Waveform examples



For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to *Section : Serial data line swapping* for details about the direction control of serial data lines.

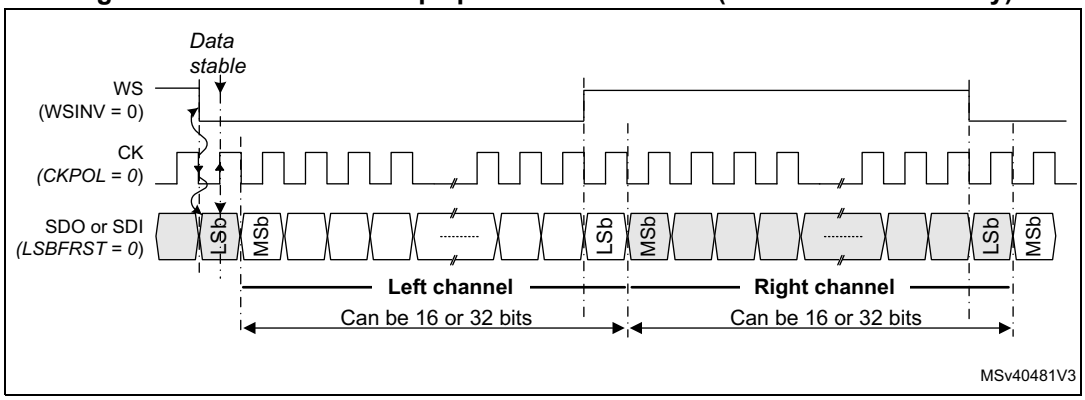
I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

a. Not always available, refer to *Section 45.3: SPI implementation* to check if 24 and 32-bit data widths are supported.

Figure 675. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



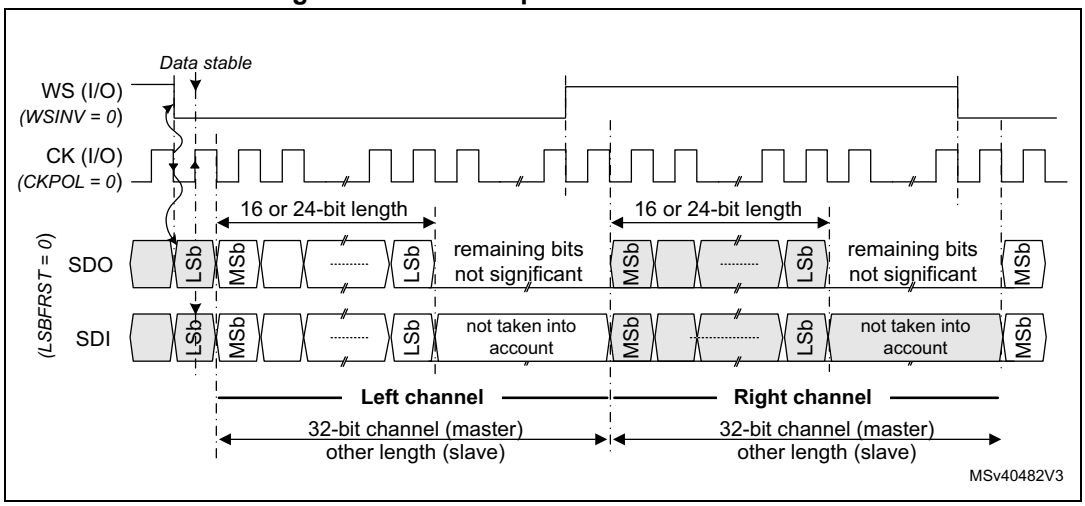
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* for the supported data sizes.

CKPOL is cleared to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

Figure 675 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

Figure 676. I2S Philips standard waveforms



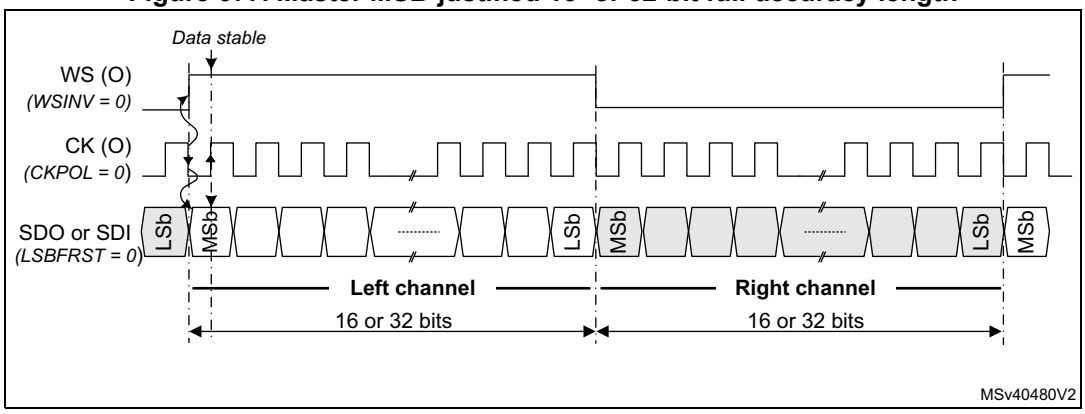
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB-justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high and the right channel if WS is low.

Figure 677. Master MSB-justified 16- or 32-bit full-accuracy length

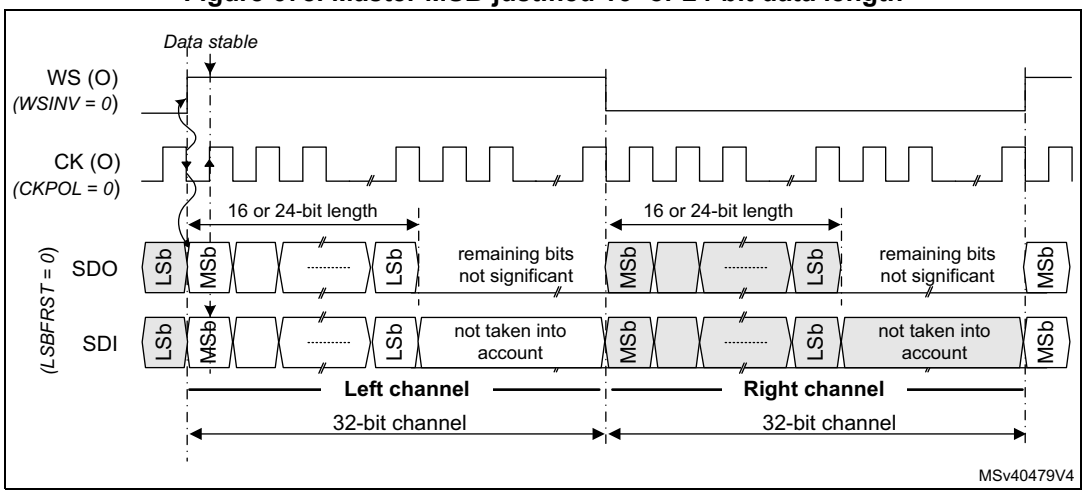


1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* to check the supported data size.

CKPOL is cleared to match the I2S MSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

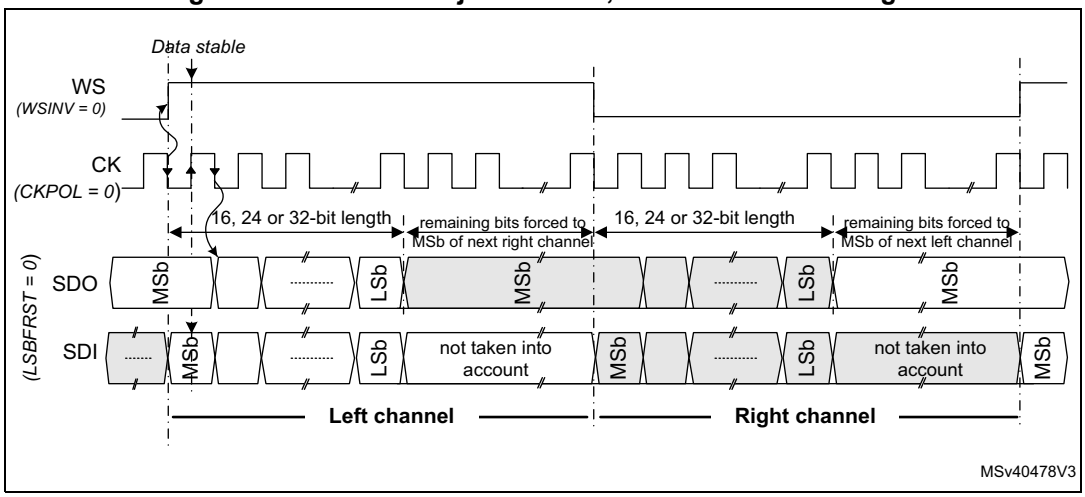
Figure 678. Master MSB-justified 16- or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit mode, the remaining bits are forced to the value of the first bit of the next data to be generated to avoid timing issues (see *Figure 679*).

Figure 679. Slave MSB-justified 16-, 24- or 32-bit data length



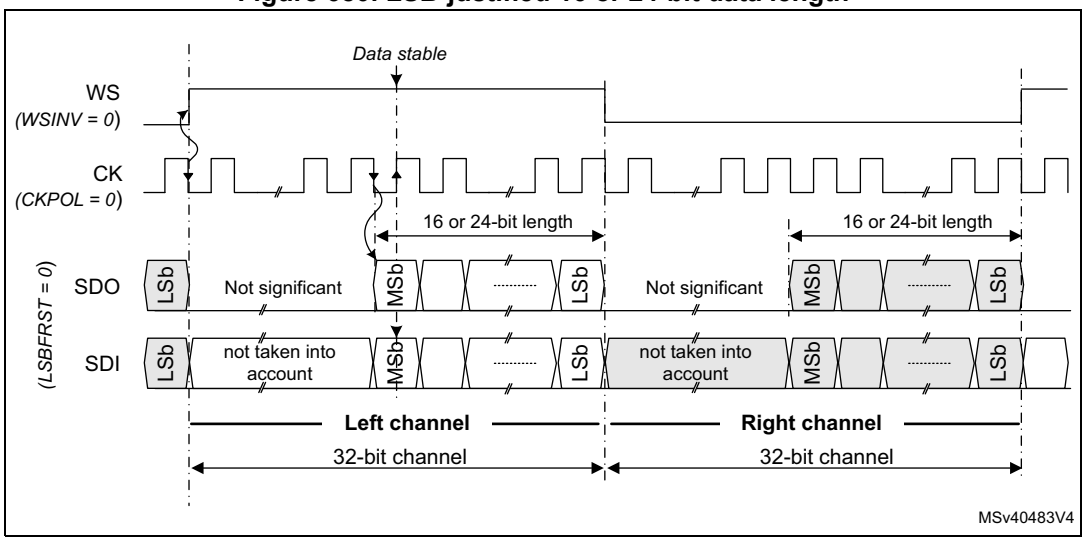
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* to check the supported data size.

LSB-justified standard

This standard is similar to the MSB-justified standard in master mode (no difference for the 16- and 32-bit full-accuracy frame formats). The LSB-justified 16- or 32-bit full-accuracy format gives similar waveforms to MSB-justified mode (see *Figure 677*) because the channel and data have the same length.

Note: In the LSB-justified format, only 16- and 32-bit channel lengths are supported in master and slave mode. This is because it is not possible to transfer properly the data if the channel length is not known by the transmitter and receiver side.

Figure 680. LSB-justified 16 or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* to check the supported data size.

CKPOL is cleared in order to match the I2S LSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit of SPI_I2SCFGR register.

In PCM long frame:

- The assertion time of the WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

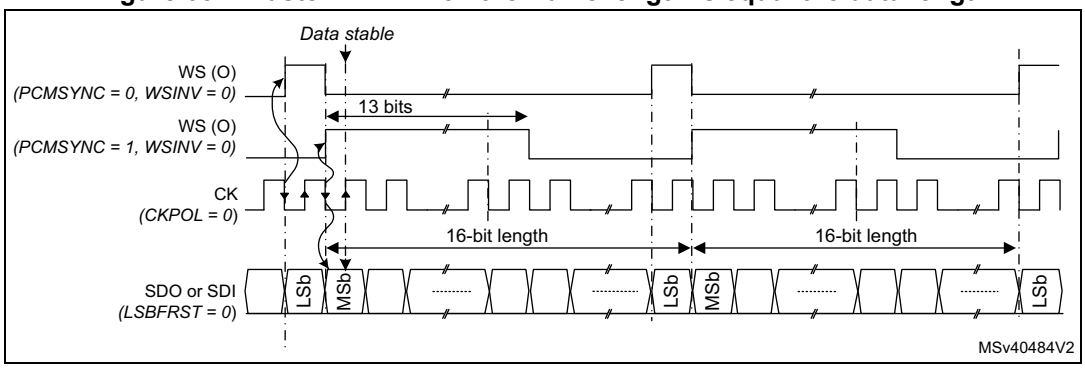
In PCM short frame:

- The assertion time of the WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:

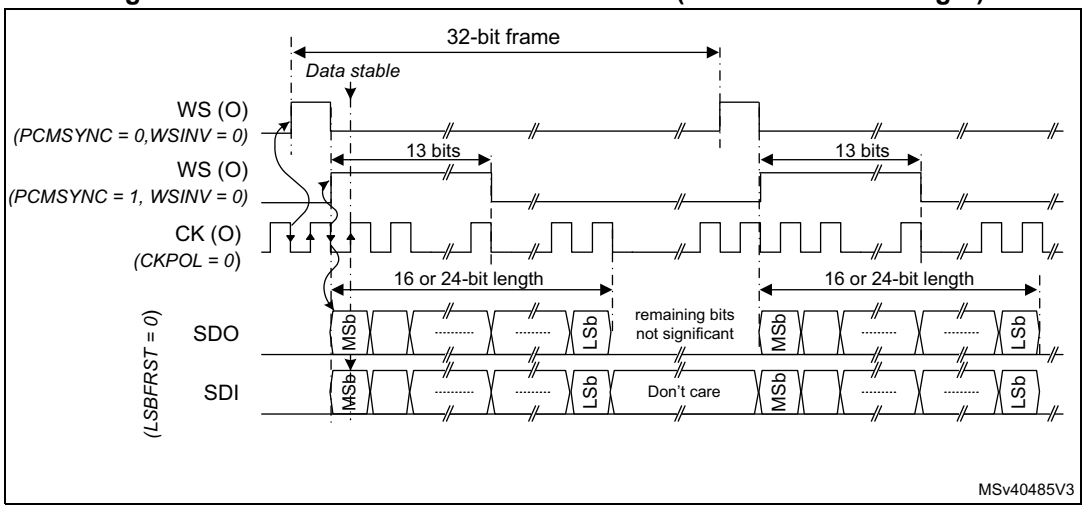
- The first data bit is MSb or LSb depending on the LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to the WSINV bit. See *Control of the WS inversion* for information concerning the handling of WS signal.

Figure 681. Master PCM when the frame length is equal the data length



A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

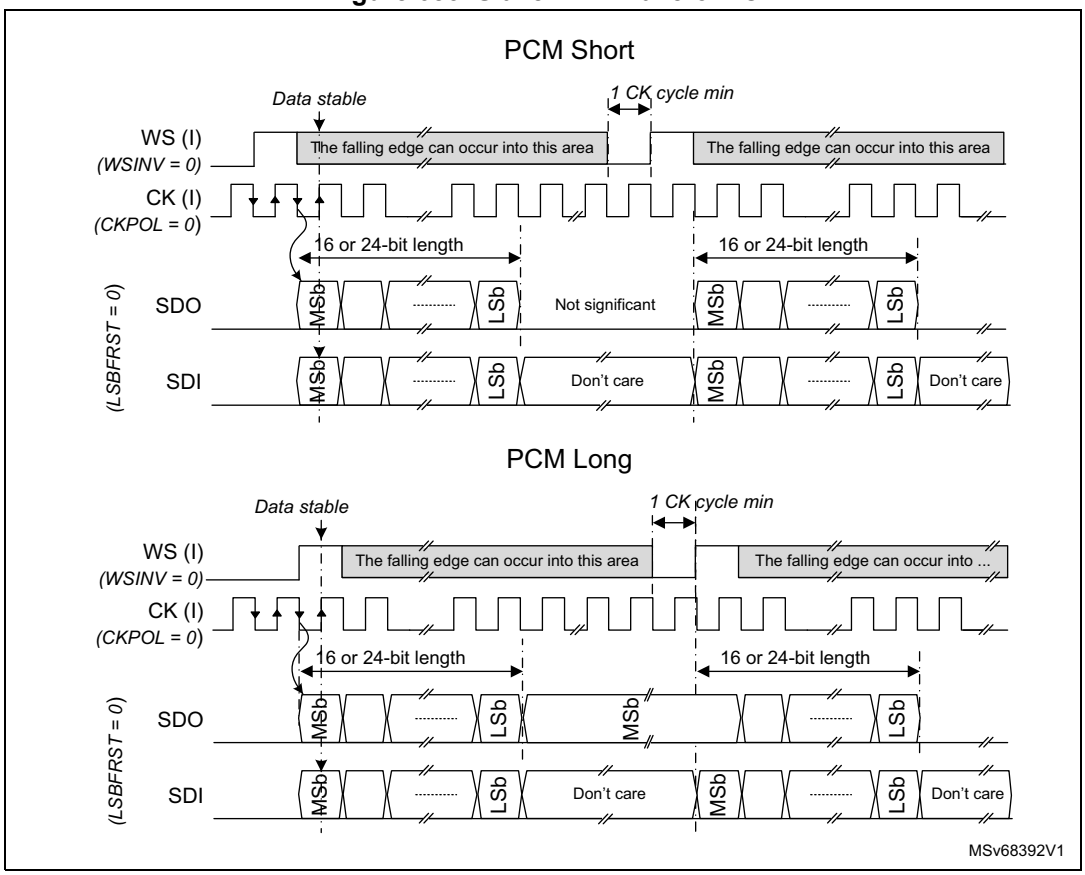
Figure 682. Master PCM standard waveforms (16 or 24-bit data length)



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* to check the supported data size.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 683*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 683. Slave PCM waveforms



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* to check the supported data size.

Note: In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits is forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

45.9.6 Additional serial interface flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for the I2S LSB-justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI/I2S configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB-justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB-justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located in the *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI/I2S configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 465 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS also depends on the protocol selected.

Table 465. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR must not be set when the SPI is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of *SPI/I2S configuration register 2 (SPI_CFG2)*, and on the slave/master mode. *Table 466* gives details on this feature.

Table 466. Serial data line swapping

Direction	IOSWP	Master mode		Slave mode	
		Input line	Output line	Input line	Output line
RX	0	SDI	-	SDO	-
	1	SDO	-	SDI	-
TX	0	-	SDO	-	SDI
	1	-	SDI	-	SDO
Full-duplex	0	SDI	SDO	SDO	SDI
	1	SDO	SDI	SDI	SDO

45.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write from/to the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

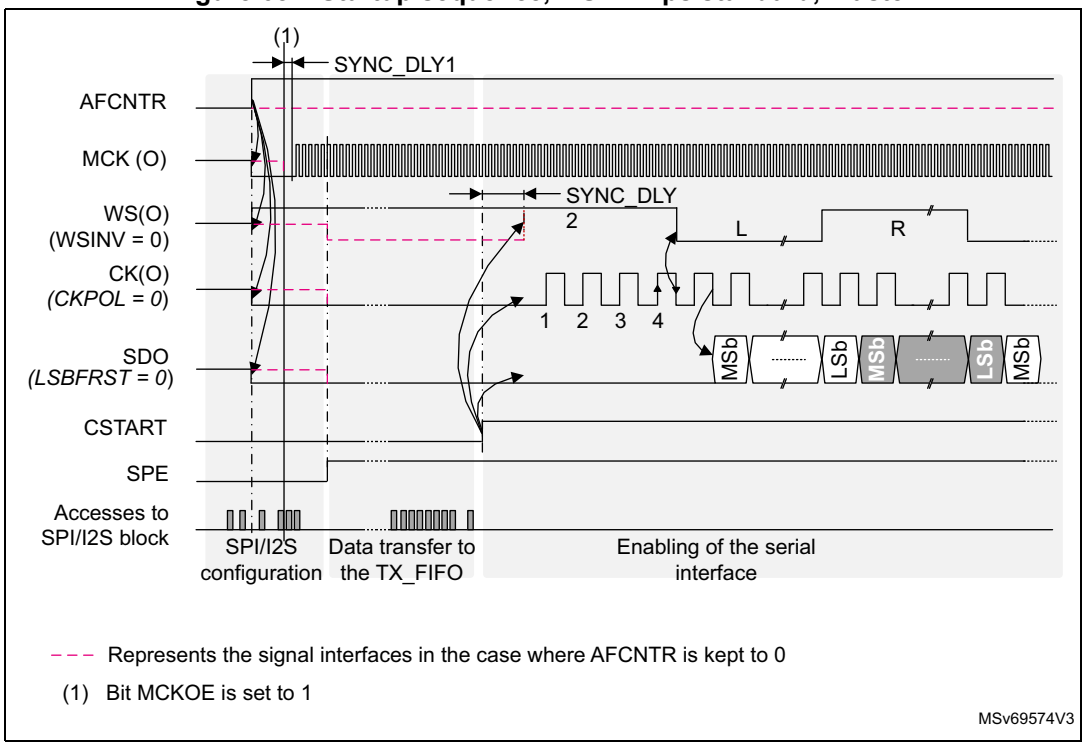
When the application wants to use the SPI/I2S block the user has to proceed as follows:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 45.9.16* for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC, and UDRC bits of the SPI_IFCR register. Note that if the flag SUSP is not cleared (via the SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO.
5. Set bit CSTART to activate the serial interface.

As shown in *Figure 684*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1 and the TxFIFO is not empty. Note that the CK bit clock is activated four rising edges before

the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 684. Startup sequence, I2S Philips standard, master



2. When MCKOE is set, the MCK is generated after a synchronization delay (SYNC_DLY1). For more information, see MCK generation in *Section 45.9.9: Clock generator*.
3. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

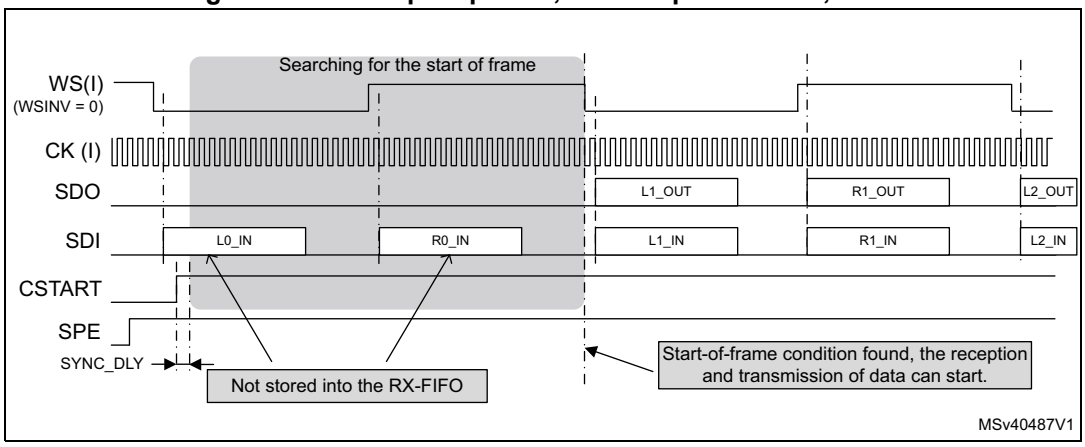
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about three periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For the I2S Philips standard, the start-of-frame condition is a falling edge of the WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of the WS signal. The transmission/reception starts at the rising edge of WS for MSB-aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 685 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 685. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after two periods of CK clock (SYNC_DLY).

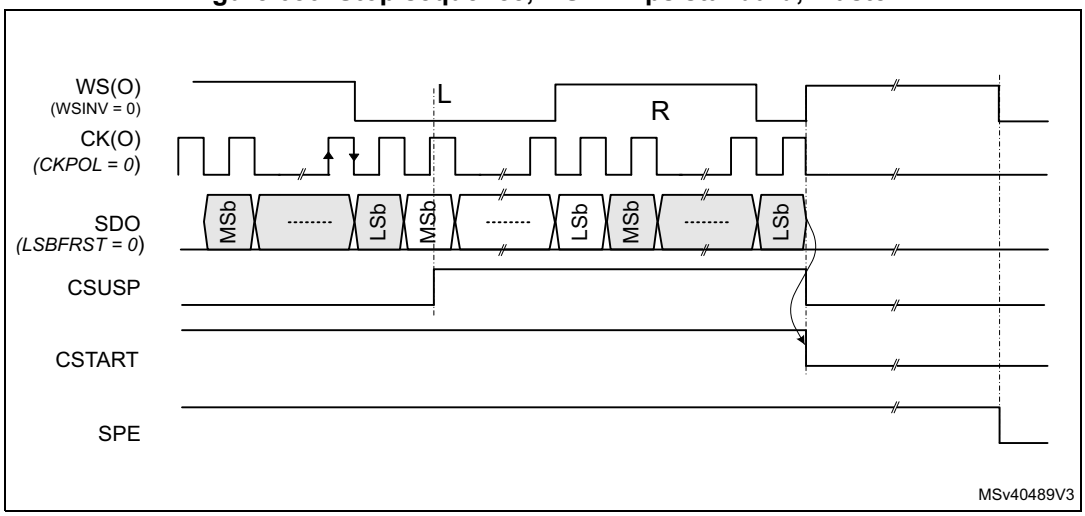
45.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

Figure 686 shows an example of stop sequence in the case of master mode. The CSUSP bit is set during the transmission of the left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear the SPE bit.

Figure 686. Stop sequence, I2S Philips standard, master

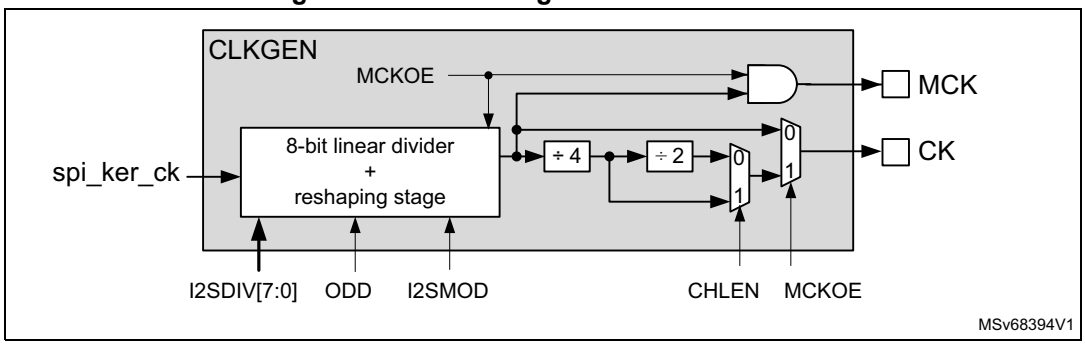


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

45.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the frame synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 687. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN, and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there are no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: MCK duty cycle is 50 %, even for odd division ratios, when the kernel clock duty cycle is 50 %. When the division ratio is set (I2SDIV = 0), the MCK duty cycle is the same as the duty cycle of the kernel clock.

Table 467 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK is generated when the following conditions are met:

- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,

Table 467. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192

Table 467. CLKGEN programming examples for usual I2S frequencies (continued)

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

45.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

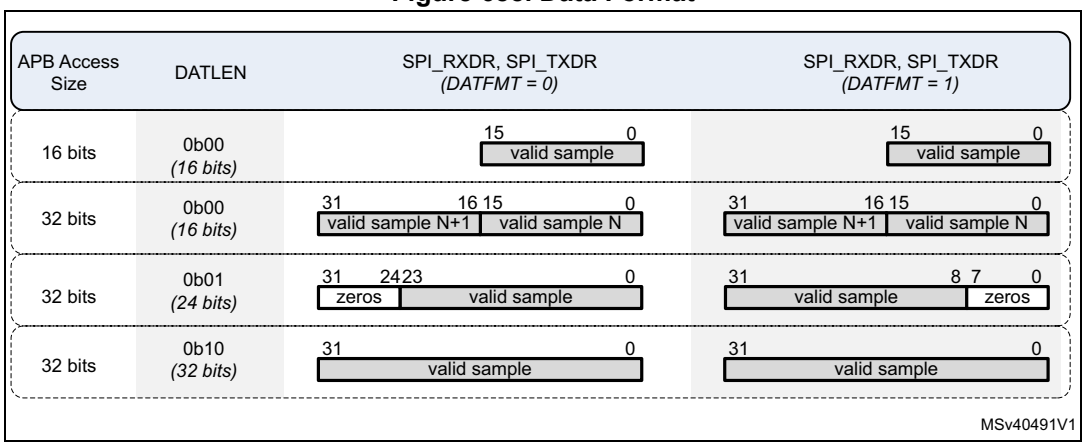
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 688 shows the allowed settings between APB access sizes, DATFMT, and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 688. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 45.3: SPI implementation* to check the supported data size.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to *Section 45.9.11* and *Section 45.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes three basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[a],
- 4 samples, if the DATLEN = 2 (32 bits)[a].

45.9.11 FIFO status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus, FTHLV new data to be transmitted can be written to SPI_TXDR. The TXP flag is reset when the number of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 45.10* for additional information on the interrupt function in I2S mode.

45.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

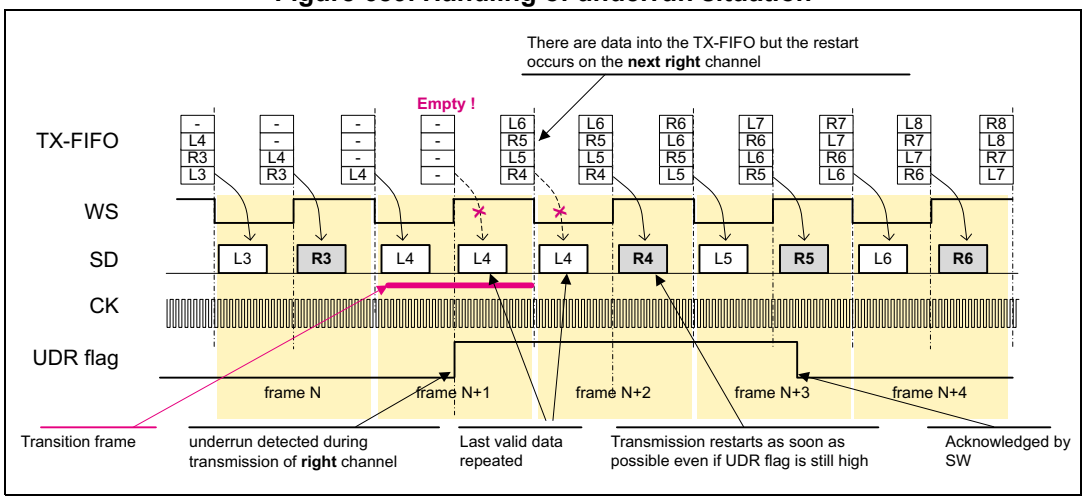
a. Not always available, refer to *Section 45.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral replays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the restart condition is met:
  - If the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - If the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

Figure 689. Handling of underrun situation



When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO.

The UDR flag can trigger an interrupt if the UDRIE bit of the SPI_IER register is set. The UDR bit is cleared by setting the UDRC bit of the SPI_IFCR register.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK, and MCK signals are not gated.

Due to resynchronization, any change of the UDR flag is taken into account by the hardware after at least two periods of CK clock.

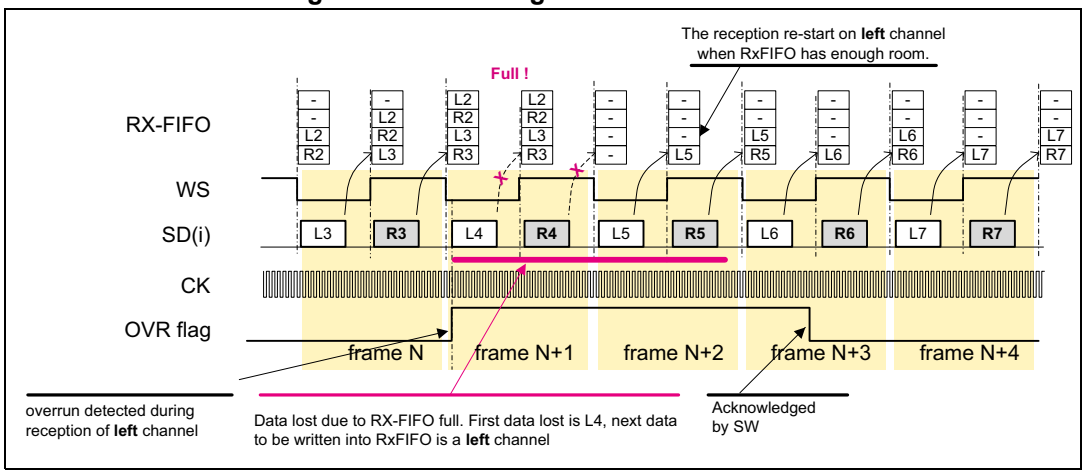
45.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.

In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 690. Handling of overrun situation



When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by setting the OVRC bit of the SPI_IFCR register.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK, and MCK signals are not gated.

45.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

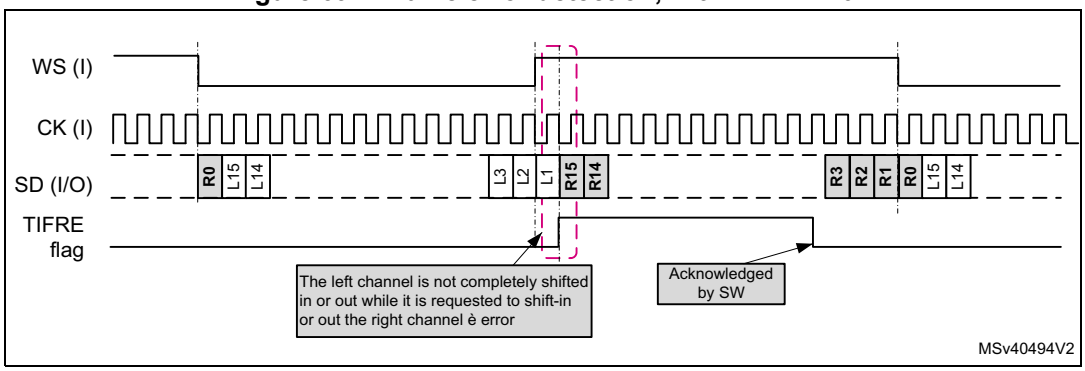
- A frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurs at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.

Figure 691 shows an example of frame error detection. The SPI/I2S block is in slave mode and the number of bit clock periods for the left channel are not enough to shift-in or shift-out the data. The figure shows that the ongoing transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 691. Frame error detection, with FIXCH = 0

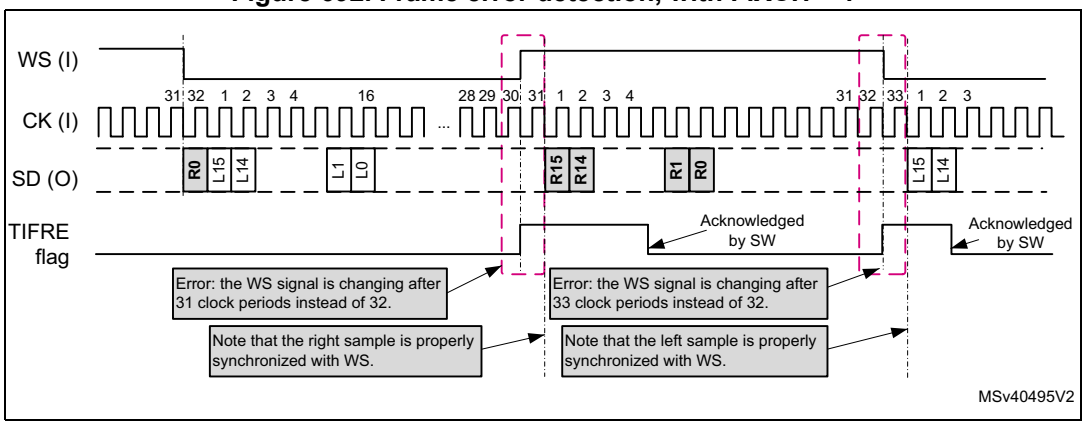


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 692*, in this mode, the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 691* and *Figure 692* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 692. Frame error detection, with FIXCH = 1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

45.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of the SPI_CFG1 register.

In receive mode, the DMA interface is working as follows:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follows:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

45.9.16 Programing examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set I2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example, if a threshold of two audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. If needed, the application can then configure the external audio codec via an I2C interface.
8. Clear all interrupt enable fields located in the SPI_IER register.
9. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of the SPI_IFCR register.
10. When the application wants to start the data transfer, set the SPE bit. Then:
    - If the data transfer uses DMA:
    a) Program the DMA peripheral: two channels, one for RX and one for TX.
    b) Initialize the memory buffer with valid audio samples for TX path.
    c) Enable the DMA channels.
    d) Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
    e) Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled up.
    - If the data transfer is done via interrupt:

      Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Stop procedure in master mode

1. Set the bit CSUSP, in order to stop ongoing transfers
2. Check the value of the CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the

SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So, we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example if a threshold of two audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   – Program the DMA peripheral: one RX channel
   – Enable the DMA channel,
   – In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally, the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop procedure in slave mode

1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

45.10 I2S interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wake-up event signal (**spi_wkup**) can be generated according to the events described in the *Table 468*.

Interrupt events can be enabled and disabled separately.

Table 468. I2S interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/Interrupt clearing method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
SPI	TxFIFO threshold reached	TXP	When the TxFIFO contains less than FTHLV empty locations	Yes	Yes	No
	RxFIFO threshold reached	RXP	When the RxFIFO contains less than FTHLV samples			
	Overrun error	OVR	Write OVRC to 1			
	Underrun error	UDR	Write UDRC to 1			
	Frame error flag	TIFRE	Write TIFREC to 1		No	

45.11 SPI/I2S registers

45.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (33-bit or 17-bit) CRC polynomial configuration

0: Full size (33-bit or 17-bit) CRC polynomial is not used

1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode

In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 45.9.8: Stop sequence*.

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero. SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

45.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

45.11.3 SPI/I2S configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	DRDS	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw				rw		rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator
0: bypass is disabled
1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting
000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256
Note: MBR setting is considered at slave working at TI mode, too (see Section 45.5.1: TI mode).

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DRDS:** data sampling delay on master input (MISO)
This delay is used to compensate the delay on the MISO line (for example due to the signal optical isolation).
0: Sampling is performed on the configured SCK edge (no delay is applied)
1: Sampling is postponed by a fixed delay equal to a half SCK cycle (the sampling is performed just at the end of the bit validity interval at MISO line).
Note: The DRDS setting has no impact on the other SCK management nor on the data frame communication format. When the delay is applied together with CRC enabled, the CRC computation and evaluation is also delayed.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared

Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.

The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.

The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

Bit 15 **TXDMAEN:** Tx DMA stream enable

0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable

0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

For more details see *Figure 673: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in the SPI_UDRDR register

1: Slave repeats the last received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8 bits, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

45.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Note: This bit can be also used in PCM and I2S modes.

The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped
Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness

Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.

0000: no delay

0001: 1 clock cycle period delay

...

1111: 15 clock cycle periods delay

Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

45.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MODFIE**: mode Fault interrupt enable
0: MODF interrupt disabled
1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable
0: TIFRE interrupt disabled
1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable
0: CRC interrupt disabled
1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable
0: OVR interrupt disabled
1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

45.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus .
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is re-enabled or by writing 1 to the CRCEC bit of the SPI_IFCR optionally.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is re-enabled or by writing 1 to the OVRC bit of the SPI_IFCR optionally.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is re-enabled or by writing 1 to the UDRC bit of the SPI_IFCR optionally.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only. In I2S/PCM mode, (when available) this flag applies to master and slave mode.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

45.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear

Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

45.11.8 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

45.11.9 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

45.11.10 SPI/I2S polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCPOLY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

45.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

45.11.12 SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

45.11.13 SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDRDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

45.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x050

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit, which needs to be kept at 0.

This register is reserved for instances not supporting I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

Refer to *Section 45.9.9: Clock generator* for details

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
Refer to *Section 45.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: The data inside the SPI_RXDR or SPI_TXDR register are right aligned
1: The data inside the SPI_RXDR or SPI_TXDR register are left aligned.

Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB- or LSB-justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB- or LSB-justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 45.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I²S standard selection
For more details on I²S standards, refer to *Section 45.9.5: Supported audio protocols*
00: I²S Philips standard.
01: MSB-justified standard (left justified)
10: LSB-justified standard (right justified)
11: PCM standard

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - Full Duplex
101: master - Full Duplex
others: not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

45.11.15 SPI/I2S register map

Table 469. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	DRDS	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0				0		0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	Reserved	Res.																															
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 469. SPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SPI_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	Reset value							0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 45.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

46 FD controller area network (FDCAN)

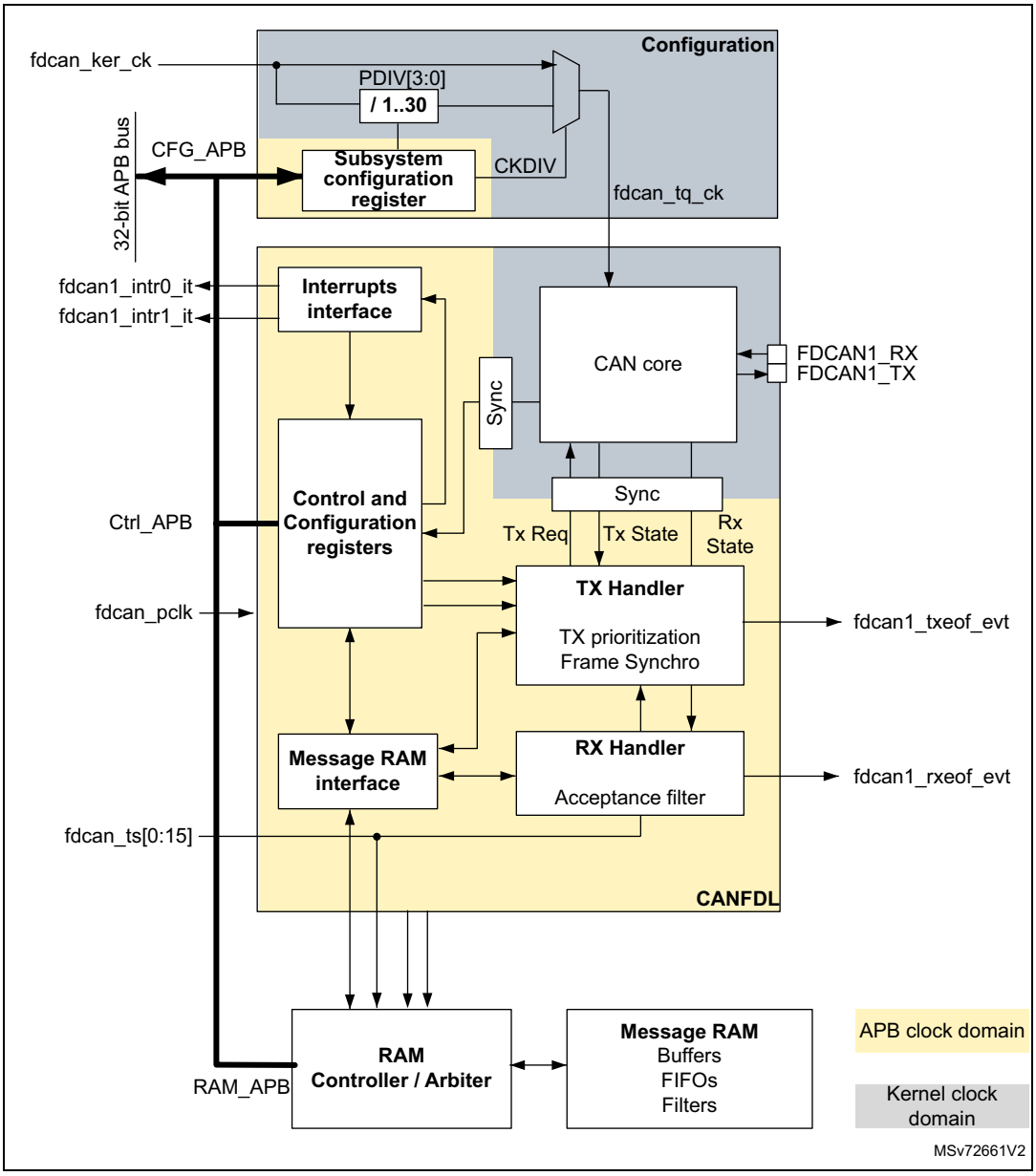
46.1 FDCAN introduction

The controller area network (CAN) subsystem (see *Figure 693*) consists of one CAN module, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance is used for filtering, transmitting event FIFOs, and receiving and transmitting FIFOs.

Figure 693. CAN subsystem.



46.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

46.3 FDCAN implementation

The table below describes the FDCAN implementation.

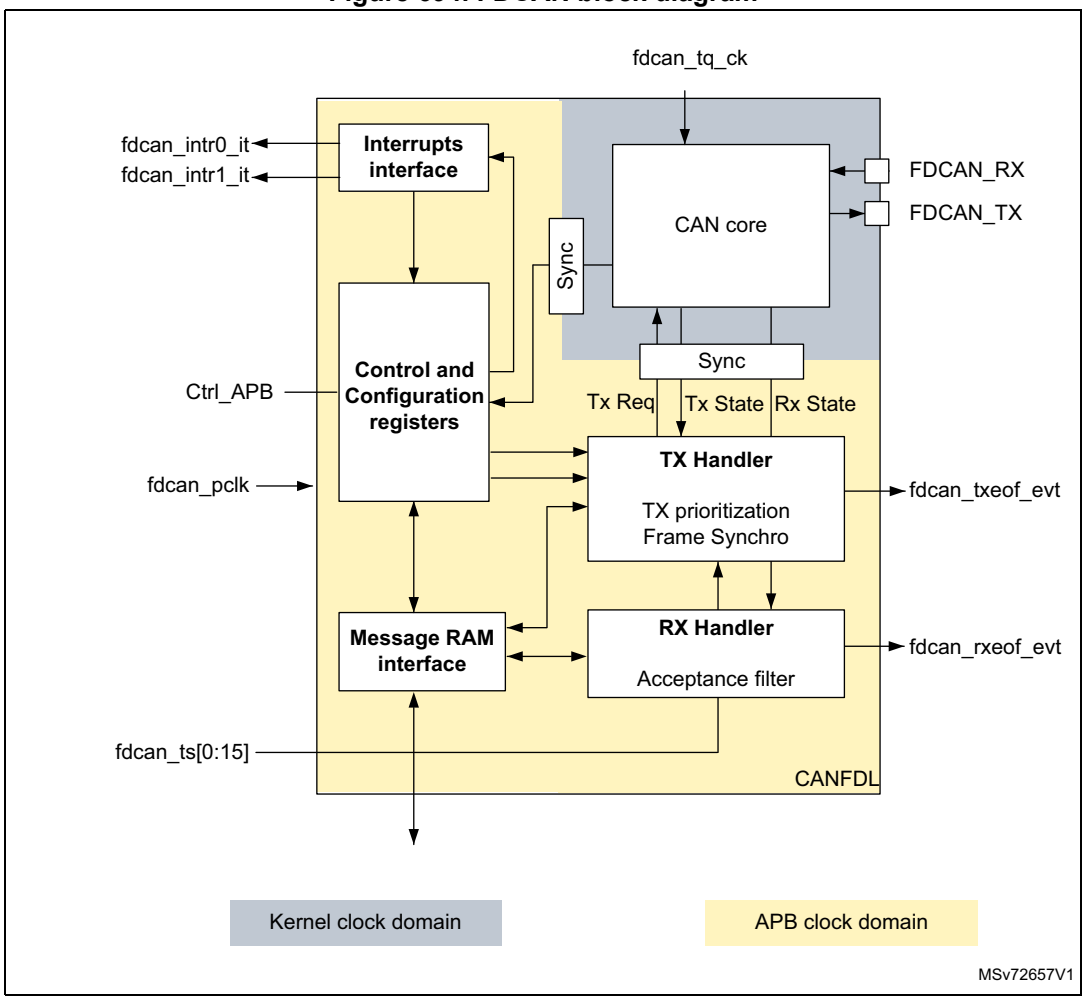
Table 470. FDCAN features

Feature	STM32C53x/C54x	STM32C55x/C56x	STM32C59x/C5Ax
FDCAN1	X	X	X
FDCAN2	X	-	X

46.4 FDCAN functional description

46.4.1 FDCAN block diagram

Figure 694. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.

By programming the EINT0 and EINT1 bits of the FDCAN_ILE register, the interrupt lines can be independently enabled or disabled.

CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

The Tx handler also generates an event on the fdcan_txeof_evt output when the end-of-frame transmission is issued. This event is internally connected to the TIM2 and TIM3 input capture.

Rx handler

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

The Rx handler also generates an event on the fdcan_rxeof_evt output when the end-of-frame reception is received. This event is internally connected to the TIM2 and TIM3 input capture.

APB interface

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

Message RAM interface

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbiter.

46.4.2 FDCAN pins and internal signals

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 471*, *Table 472*, and *Figure 693*.

Table 471. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan_intr0_it	Digital output	FDCAN interrupt0
fdcan_intr1_it		FDCAN interrupt1
fdcan_ts[0:15]	-	External timestamp vector
fdcan_txeof_evt	Digital output	End-of-frame transmission event
fdcan_rxeof_evt	Digital output	End-of-frame reception event
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

Table 472. CAN subsystem I/O pins

Name	Type	Description
FDCAN_RX	Digital input	FDCAN receive pin
FDCAN_TX	Digital output	FDCAN transmit pin

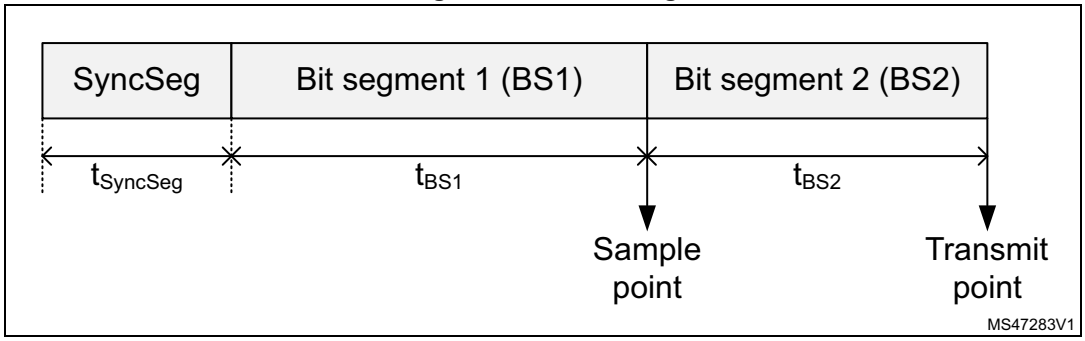
46.4.3 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 695*, this operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum ($1 \times t_q$).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable from 1 to 32 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but can also be automatically shortened to compensate for negative phase drifts.

Figure 695. Bit timing



The baud rate is the inverse of the bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components (see *Figure 695*):

bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$

Where:

– For the nominal bit time

$t_q = (NBRP[8:0] + 1) \times t_{fdcan_tq_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (NTSEG1[7:0] + 1)$

$t_{BS2} = t_q \times (NTSEG2[6:0] + 1)$

Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN_NBTP register.

– For the data bit time
$t_q = (DBRP[4:0] + 1) \times t_{fdcan_tq_clk}$
$t_{SyncSeg} = 1 \times t_q$
$t_{BS1} = t_q \times (DTSEG1[4:0] + 1)$
$t_{BS2} = t_q \times (DTSEG2[3:0] + 1)$
Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN_DBTP and FDCAN_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA_CCCR register are set.

The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock ($f_{dcan_tq_ck} \leq f_{dcan_pclk}$).

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

46.4.4 Operating modes

Configuration

Access to peripheral version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

The reachable FDCAN bitrates are:

- Nominal bitrate: up to 1 Mbit/s
- Data bitrate: between nominal bitrate and 8 Mbit/s

Software initialization

Software initialization is started by setting the INIT bit of the FDCAN_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN_CCCR register are both set.

The CCE bit of the FDCAN_CCCR register can only be set/cleared while the INIT bit of FDCAN_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.

The following registers are reset when the CCE bit of the FDCAN_CCCR register is set:

- FDCAN_HPMS: High priority message status
- FDCAN_RXF0S: Rx FIFO 0 status
- FDCAN_RXF1S: Rx FIFO 1 status
- FDCAN_TXFQS: Tx FIFO/queue status
- FDCAN_TXBRP: Tx buffer request pending
- FDCAN_TXBTO: Tx buffer transmission occurred
- FDCAN_TXBCF: Tx buffer cancellation finished
- FDCAN_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN_TOCC register when the CCE bit of the FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN_TXBAR: Tx buffer add request
- FDCAN_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN_CCCR can only be set/cleared while the INIT and CCE bits are both set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

- Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes.
- Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame.

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN_PSR) to integrating (ACT[1:0] = 00 in FDCAN_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRS bits set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 473*.

Table 473. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN_DBTP register. In the following FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (t_q), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mt_q (minimum time quantum, one period of fdcan_tq_ck clock).

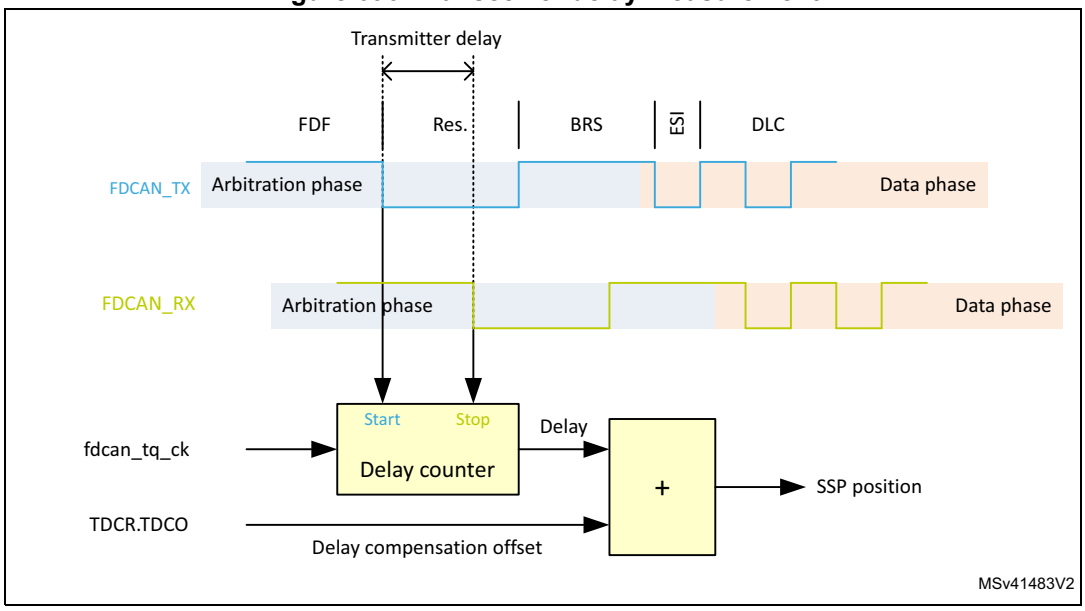
The TDCV[6:0] bitfield of the FDCAN_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN_CCCR. It is updated at each transmission of an FD frame while the TDC bit of the FDCAN_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to 127 × mt_q. If the sum exceeds this value, the maximum value (127 × mt_q) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mt_q.

Figure 696. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

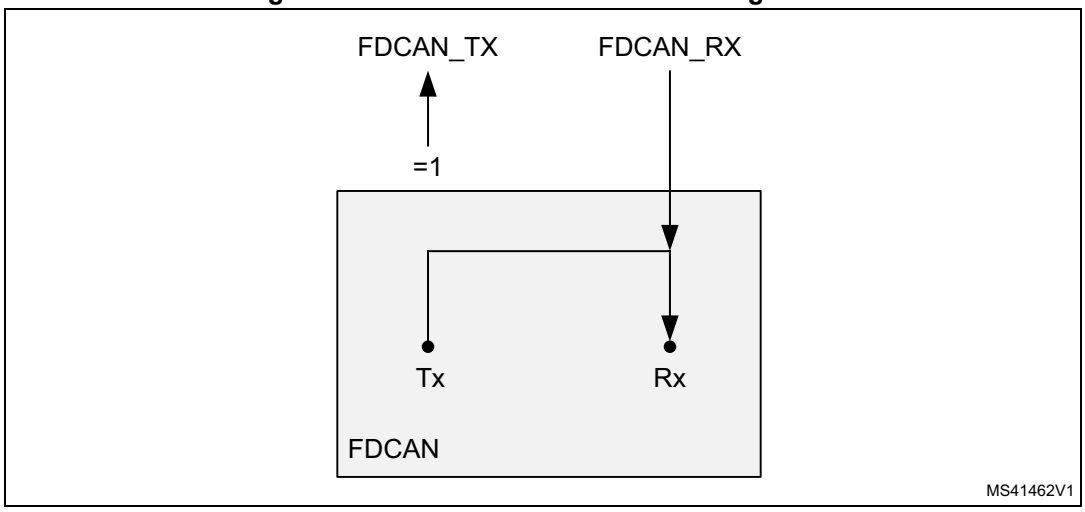
Note: The restricted operation mode must not be combined with the loop-back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 697* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 697. Pin control in bus monitoring mode



Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN_CCCR register.

Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN_TXBCF register.
- Successful transmission in spite of cancellation
  - The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.
- Arbitration loss or frame transmission disturbed
  - The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power-down (Sleep mode)

Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

Test modes

To enable write access to *FDCAN test register (FDCAN_TEST)*, the TEST bit of the FDCAN_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming the TX[1:0] bitfield of the FDCAN_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from the RX bit of FDCAN_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via RX.

Note: Test modes must be used for production tests or self-test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop-back mode

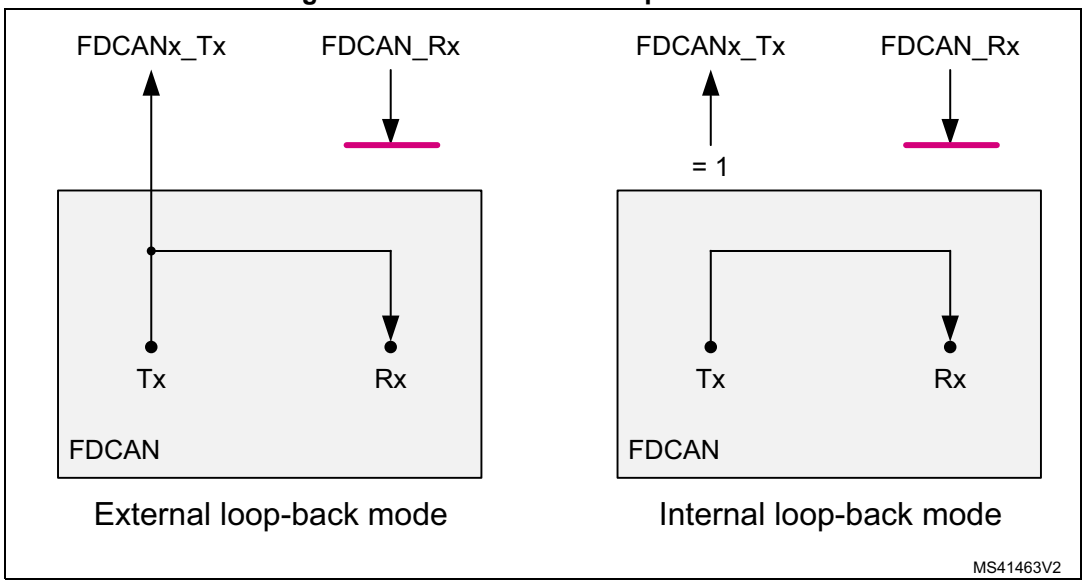
The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 698* shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop-back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN_TEST and the MON bit of FDCAN_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 698* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop-back mode.

Figure 698. Pin control in loop-back mode



Timestamp generation

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN_TSCV register. A write access to TSCV15:0] resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming TSS[1:0] of FDCAN_TSCC, a 16-bit timestamp can be used.

Debug mode behavior

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN_TOCV. The timeout counter can only be started while the INIT bit of FDCAN_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the TOO interrupt flag is set in the FDCAN_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

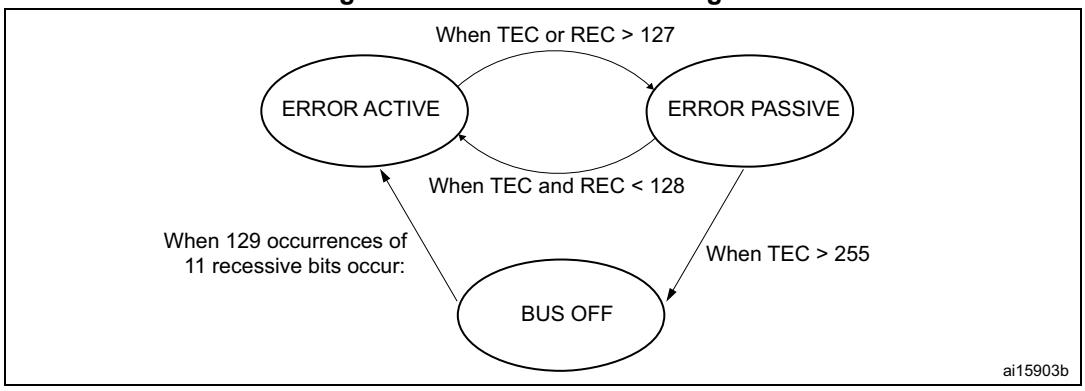
Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.

46.4.5 Error management

As described in the CAN protocol, the error management is handled entirely by hardware using the transmit error counter (the TEC[7:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*) and the receive error counter (the REC[6:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*). These values are incremented or decremented according to the error condition. For detailed information on TEC[7:0] and REC[6:0] management, refer to the CAN standard. Both values can be read by software to determine the stability of the network.

The bus-off state is reached when TEC[7:0] is greater than 255. This state is also indicated by the BO flag of the *FDCAN protocol status register (FDCAN_PSR)*. In bus-off state, the CAN is no longer able to transmit and receive messages. It has to wait at least for the duration of the recovery sequence specified in the CAN standard (129 occurrences of 11 consecutive recessive bits monitored on FDCAN_RX input).

Figure 699. CAN error state diagram

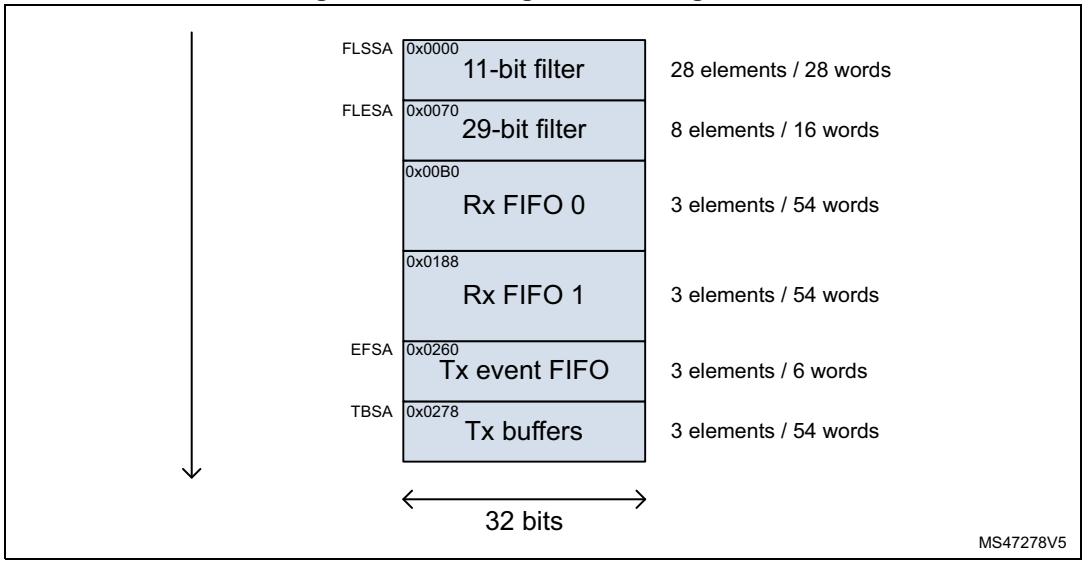


Note: In initialization mode, the CAN does not monitor the FDCAN_RX *signal, and therefore cannot complete the recovery sequence. To recover from an error state, the CAN must operate in normal mode.*

46.4.6 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in *Figure 700*.

Figure 700. Message RAM configuration



When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in *Figure 700*)
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN_IR
- Set the high priority message interrupt flag HPM in FDCAN_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**

  The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT[1:0] = 00: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- EFT[1:0] = 11: the extended ID AND mask (XIDAM) is not used for range filtering.

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

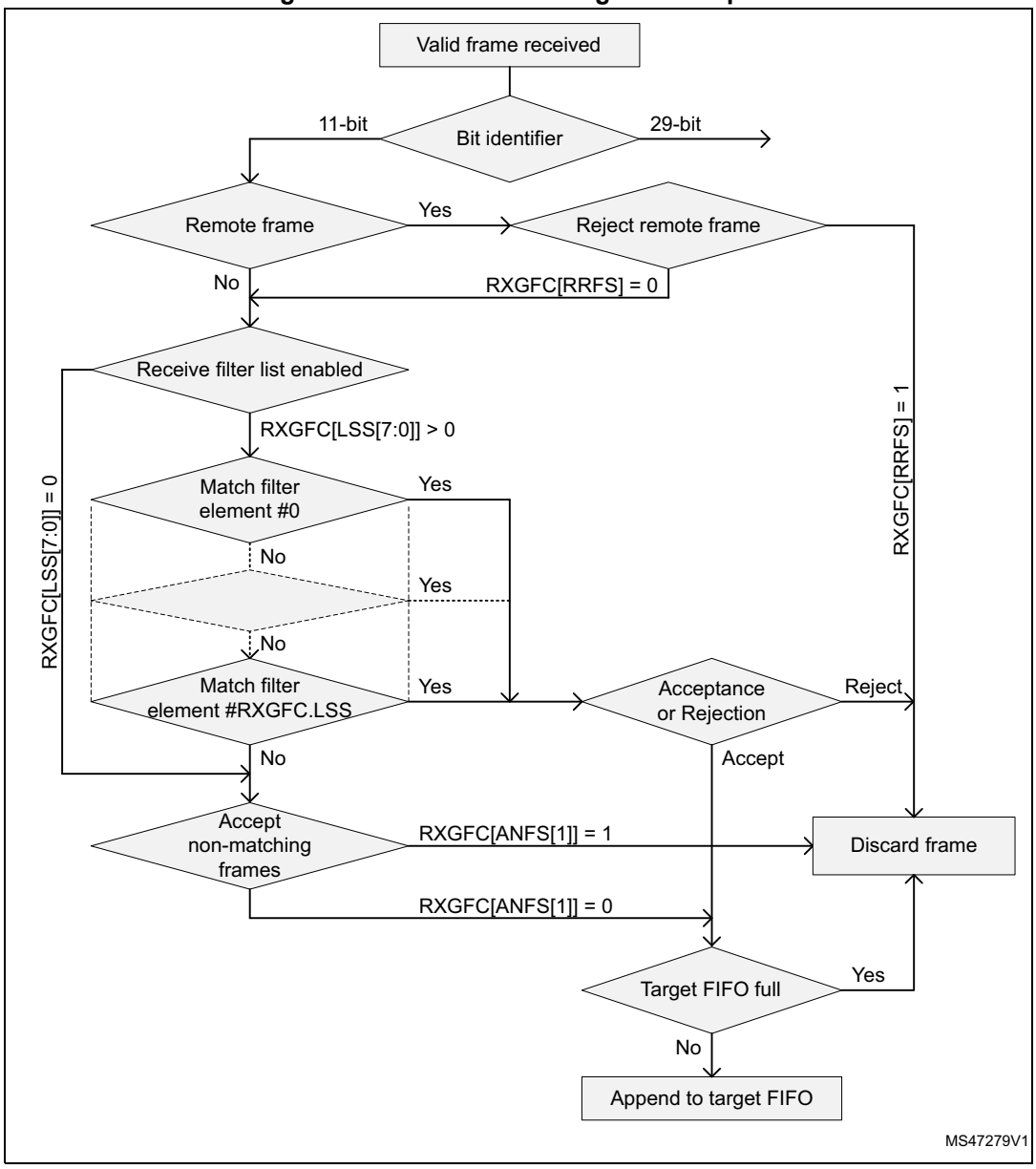
In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 701 shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in *Section 46.4.11*.

The standard message filtering is controlled by the FDCAN_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 701. Standard message ID filter path

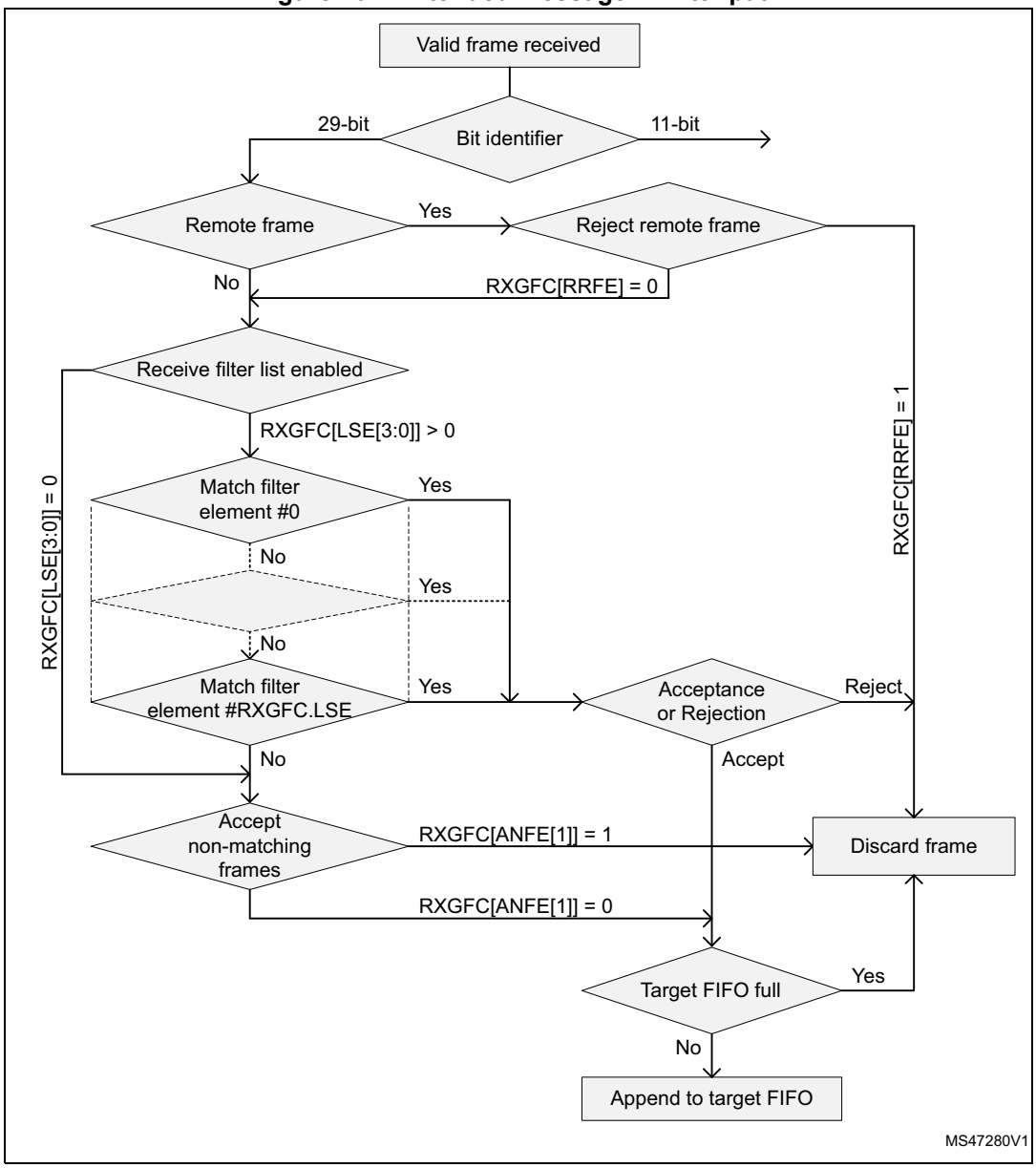


Extended message ID filtering

Figure 702 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in *Section 46.4.12*.

The extended message filtering is controlled by the FDCAN_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 702. Extended message ID filter path



The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 46.4.8*.

When an Rx FIFO full condition is signaled by RFnF in FDCAN_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGI of FDCAN_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN_IR.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGI of FDCAN_RXFnS) is signaled by FnF = 1 in FDCAN_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN_RXFnS).

Tx handling

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. the Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 474. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN_TXBRP) is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN_CCCR).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCA_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. The Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding 18 times the put index TFQPI[1:0] (0 … 2) to the Tx buffer start address TBSA.

Tx queue

Tx queue operation is configured by setting the TFQM of the FDCAN_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application can use the FDCAN_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding 18 times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN _TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in *Tx FIFO*.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.

46.4.7 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see *Section 46.5.23* and *Section 46.5.25*). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

46.4.8 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in *Table 475*. The description is provided in *Table 476*.

Table 475. Rx FIFO element

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R0	ESI	XTD	RTR	ID[28:0]																												
R1	ANMF	FIDX[6:0]							Res.				FDF		BRS		DLC[3:0]				RXTS[15:0]											
R2	DB3[7:0]								DB2[7:0]								DB1[7:0]								D[7:0]							
R3	DB7[7:0]								DB6[7:0]								DB5[7:0]								DB4[7:0]							
⋮	⋮								⋮								⋮															
Rn	DBm[7:0]								DBm-1[7:0]								DBm-2[7:0]								DBm-3[7:0]							

The element size configured for storage of CAN FD messages is set to 64-byte data field.

Table 476. Rx FIFO element description

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 476. Rx FIFO element description (continued)

Field	Description
R0 Bit 29 RTR	Remote transmission request Signals to the host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.
R2 Bits 31:24 DB3[7:0]	Data byte 3
R2 Bits 23:16 DB2[7:0]	Data byte 2
R2 Bits 15:8 DB1[7:0]	Data byte 1
R2 Bits 7:0 D[7:0]	Data byte 0
R3 Bits 31:24 DB7[7:0]	Data byte 7
R3 Bits 23:16 DB6[7:0]	Data byte 6

Table 476. Rx FIFO element description (continued)

Field	Description
R3 Bits 15:8 DB5[7:0]	Data byte 5
R3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data byte m
Rn Bits 23:16 DBm-1[7:0]	Data byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data byte m-3

46.4.9 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration TFQM bit of the FDCAN_TXBC register. The element size is configured for storage of CAN FD messages with up to 64-byte data.

Table 477. Tx buffer and FIFO element

Bit	31			24	23					16	15	8	7	0
T0	ESI	XTD	RTR	ID[28:0]										
T1	MM[7:0]				EFC	Res.	FDF	BRS	DLC[3:0]		Res.			
T2	DB3[7:0]				DB2[7:0]						DB1[7:0]		D[7:0]	
T3	DB7[7:0]				DB6[7:0]						DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮						⋮			
Tn	DBm[7:0]				DBm-1[7:0]						DBm-2[7:0]		DBm-3[7:0]	

Table 478. Tx buffer element description

Field	Description
T0 Bit 31 ESI(1)	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 478. Tx buffer element description (continued)

Field	Description
T0 Bit 29 RTR(2)	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS(3)	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data byte 3
T2 Bits 23:16 DB2[7:0]	Data byte 2
T2 Bits 15:8 DB1[7:0]	Data byte 1
T2 Bits 7:0 D[7:0]	Data byte 0
T3 Bits 31:24 DB7[7:0]	Data byte 7
T3 Bits 23:16 DB6[7:0]	Data byte 6
T3 Bits 15:8 DB5[7:0]	Data byte 5
T3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮

Table 478. Tx buffer element description (continued)

Field	Description
Tn Bits 31:24 DBm[7:0]	Data byte m
Tn Bits 23:16 DBm-1[7:0]	Data byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node can optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if the transmission in CAN FD format is enabled by the FDOE bit of the FDCAN_CCCR.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled by setting the FDOE bit of the FDCAN_CCCR. Bit BRS is only evaluated when in addition BRSE bit is set in FDCAN_CCCR.

46.4.10 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN_TXEFS register.

Table 479. Tx event FIFO element

Bit	31			24	23						16	15		8	7		0
E0	ESI	XTD	RTR	ID[28:0]													
E1	MM[7:0]				ET[1:0]		EDL	BRS	DLC[3:0]			TXTS[15:0]					

Table 480. Tx event FIFO element description

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 480. Tx event FIFO element description (continued)

Field	Description
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.

46.4.11 FDCAN standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 … 27).

Table 481. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 482. Standard message ID filter element field description

Field	Description
Bit 31:30 SFT[1:0](1)	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT[1:0] = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC[2:0] = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

46.4.12 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 … 7).

Table 483. Extended message ID filter element

Bit	31		24	23 16	15 8	7 0
F0	EFEC[2:0]		EFID1[28:0]			
F1	EFT[1:0]	Res.	EFID2[28:0]			

Table 484. Extended message ID filter element field description

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

46.5 FDCAN registers

46.5.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

46.5.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endianness issue.

46.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods: t_q = (DBRP[4:0] + 1) FDCAN clock periods.

DTSEG1[4:0] is the sum of PROP_SEG and PHASE_SEG1. DTSEG2[3:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [DTSEG1[4:0] + DTSEG2[3:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values for the baud rate prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is t_{BS1} = (DTSEG1[4:0] + 1) × t_q.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2[3:0] + 1) × t_q.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: t_{SJW} = (DSJW[3:0] + 1) × t_q.

Note: With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s.

The data phase bit rate must be higher than or equal to the nominal bit rate.

46.5.4 FDCAN test register (FDCAN_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin

This bit is used to monitor the actual value of FDCANx_RX. It is synchronized with the FDCANx_RX pin, so it is set after reset if the FDCAN is connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop-back mode

0: Reset value, loop-back mode is disabled

1: Loop-back mode is enabled (see *Power-down (Sleep mode)*)

Bits 3:0 Reserved, must be kept at reset value.

46.5.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN_IR register. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.
This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

46.5.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non-ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP:** Transmit pause enable
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant t_q required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching
0: Bit rate switching for transmissions disabled
1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, FDCAN_TEST holds reset values
1: Test mode, write access to FDCAN_TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN can be set in power-down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or over.load frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN_CCCR).

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization started

Note: Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

46.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR register are both set. The CAN bit time can be programed in the range of 4 to 81 × t_q. The CAN time quantum can be programmed in the range of 1 to 1024 FDCAN kernel clock periods:
t_q = (BRP + 1) × FDCAN clock period fdcan_ker_ck.

NTSEG1[7:0] is the sum of PROP_SEG and PHASE_SEG1. NTSEG2[6:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [NTSEG1[7:0] + NTSEG2[6:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

This bitfield is write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN kernel clock of 8 MHz, the reset value 0x0600 0A03 configures the FDCAN for a bit rate of 500 kBit/s.

46.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM2 value (tim2_cnt[0:15]) or from TIM3 value (tim3_cnt[0:15])

11: Same as 00.

These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

46.5.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSS[1:0] = 01 in FDCAN_TSCC, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TCP[3:0] in FDCAN_TSCC. A wrap around sets the TSW interrupt flag in FDCAN_IR. Write access resets the counter to 0.
When TSS[1:0] = 10, TSC[15:0] reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN_TSCV.

46.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.
00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable

0: Timeout counter disabled
1: Timeout counter enabled
This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

For more details, see *Timeout counter*.

46.5.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN_TOCC.

46.5.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN_IR.

Access type is rc_r: cleared on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.

1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When the ASM bit of the FDCAN_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented.

46.5.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCO[6:0] in FDCAN_TDCR. The SSP position is, in the data phase, the number of minimum time quanta (mt_q) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 × mt_q.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was cleared by the CPU, no FDCAN message has been received.

1: Message in FDCAN format with EDL flag set has been received.

Access type is rc_r: cleared on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set.

1: Last received FDCAN message had its BRS flag set.

Access type is rc_r: cleared on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its ESI flag set.

1: Last received FDCAN message had its ESI flag set.

Access type is rc_r: cleared on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is rs: set on read.

Bit 7 **BO**: Bus-off status

0: The FDCAN is not in bus-off state.

1: The FDCAN is in bus-off state.

Bit 6 **EW**: Warning status

0: Both error counters are below the error-warning limit of 96.

1: At least one of error counter has reached the error-warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the error-passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]:** Last error code

LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.

000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.

001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error. A fixed format part of a received frame has the wrong format.

011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is rs: set on read.

Note: When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.

The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN_PSR, enabling the CPU to check up whether the CAN bus is

stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The REC[6:0] bitfield of the FDCAN_ECR register is used to count these sequences.

46.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 × mt_q.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

46.5.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN_IE controls whether an interrupt is generated. The configuration of FDCAN_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus-off status
0: Bus-off status unchanged
1: Bus-off status changed

Bit 18 **EW**: Warning status
0: Error-warning status unchanged
1: Error-warning status changed

Bit 17 **EP**: Error passive
0: Error-passive status unchanged
1: Error-passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN_CCCR register.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

46.5.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **BOE**: Bus-off status
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **HPME**: High-priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

46.5.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line must be enabled via the EINT0 and EINT1 bit of the FDCAN_ILE register. 0 indicates that the interrupt is assigned to interrupt line fdcan1_intr0_it, 1 indicates that the interrupt is assigned to interrupt line fdcan1_intr1_it.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERR:** Protocol error grouping the following interruption
- ARAL: Access to reserved address line
- PEDL: Protocol error in data phase line
- PEAL: Protocol error in arbitration phase line
- WDIL: Watchdog interrupt line
- BOL: Bus-off status
- EWL: Warning status interrupt line

Bit 5 **BERR:** Bit and line error grouping the following interruption
- EPL Error passive interrupt line
- ELOL: Error logging overflow interrupt line

Bit 4 **MISC:** Interrupt regrouping the following interruption
- TOOL: Timeout occurred interrupt line
- MRAFL: Message RAM access failure interrupt line
- TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR:** Tx FIFO ERROR grouping the following interruption
- TEFLL: Tx event FIFO element lost interrupt line
- TEFFL: Tx event FIFO full interrupt line
- TEFNL: Tx event FIFO new entry interrupt line
- TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG:** Status message bit grouping the following interruption
- TCFL: Transmission cancellation finished interrupt line
- TCL: Transmission completed interrupt line
- HPML: High-priority message interrupt line

Bit 1 **RXFIFO1:** RX FIFO bit grouping the following interruption
- RF1LL: Rx FIFO 1 message lost interrupt line
- RF1FL: Rx FIFO 1 full interrupt line
- RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RXFIFO0:** RX FIFO bit grouping the following interruption
RF0LL: Rx FIFO 0 message lost interrupt line
RF0FL: Rx FIFO 0 full interrupt line
RF0NL: Rx FIFO 0 new message interrupt line

46.5.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

46.5.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in *Figure 701* and *Figure 702*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list
0: No extended message ID filter
1 to 8: Number of extended message ID filter elements
> 8: Values greater than 8 are interpreted as 8.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list
0: No standard message ID filter
1 to 28: Number of standard message ID filter elements
> 28: Values greater than 28 are interpreted as 28.
This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard
Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard IDs
1: Reject all remote frames with 11-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard IDs
1: Reject all remote frames with 29-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

46.5.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask
For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

46.5.21 FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
r			r	r	r	r	r	r	r				r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element:
0: Standard filter list
1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index
Index of matching filter element.
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

46.5.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x0090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of the RF0L interrupt flag of the FDCAN_IR register. When RF0L is cleared, this bit is also cleared.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer.
Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0.
Range: 0 to 3.

46.5.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index

After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN_RXF0S).

46.5.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of the RF1L interrupt flag of the FDCAN_IR register. When RF1L is cleared, this bit is also cleared.

0: No Rx FIFO 1 message lost

1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full

0: Rx FIFO 1 not full

1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer.

Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1.
Range: 0 to 3.

46.5.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN_RXF1S).

46.5.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:0 Reserved, must be kept at reset value.

46.5.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										r				r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
						r	r						r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full
0: Tx FIFO/queue not full
1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 3

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index
Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC)

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC).

46.5.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN_TXBCR register.

After the FDCAN_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN_TXBCF in the following cases:

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN_TXBRP bit is cleared.

46.5.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

46.5.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. The bits remain set until the corresponding FDCAN_TXBRP bit is cleared.

0: No cancellation pending

1: Cancellation pending

46.5.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

46.5.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN_TXBAR register.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

46.5.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

46.5.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.
Each Tx buffer has its own CFIE bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

46.5.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EFGI[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
						r	r						r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost
This bit is a copy of the TEFL interrupt flag of the FDCAN_IR. When TEFL is cleared, this bit is also cleared.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index
Tx event FIFO write index pointer.
Range: 0 to 3.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **EFGI[1:0]**: Event FIFO get index
Tx event FIFO read index pointer.
Range: 0 to 3.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO.
Range: 0 to 3.

46.5.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index
After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) to EFAI[1:0] + 1 and updates the FIFO 0 fill level (EFFL[2:0] of FDCAN_TXEFS).

46.5.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

Address offset: 0x0100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE bit of the FDCAN_CCCR register is set.

Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN_CKDIV register, which changes clock divider for all instances.

46.5.38 FDCAN register map

Table 485. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003C	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
	Reset value	0																									0	0	0	0	0	0	0

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060-0x007C	Reserved	Reserved																															
0x0080	FDCAN_RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
	Reset value																	0			0	0	0	0	0	0	0				0	0	0
0x0090	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0
0x0094	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
	Reset value																													0	0	0	
0x0098	FDCAN_RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0
0x009C	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
	Reset value																													0	0	0	
0x00A0-0x00BC	Reserved	Reserved																															
0x00C0	FDCAN_TXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0																							
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
	Reset value											0				0	0							0	0						0	1	1
0x00C8	FDCAN_TXBRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP2	TRP1	TRP0
	Reset value																														0	0	0

Table 485. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00CC	FDCAN_TXBAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR2	AR1	AR0
	Reset value																														0	0	0
0x00D0	FDCAN_TXBCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR2	CR1	CR0
	Reset value																														0	0	0
0x00D4	FDCAN_TXBTO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO2	TO1	TO0
	Reset value																														0	0	0
0x00D8	FDCAN_TXBCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF2	CF1	CF0
	Reset value																														0	0	0
0x00DC	FDCAN_TXBTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE2	TIE1	TIE0
	Reset value																														0	0	0
0x00E0	FDCAN_TXBCIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE2	CFIE1	CFIE0
	Reset value																														0	0	0
0x00E4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EFG[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
	Reset value							0	0							0	0							0	0						0	0	0
0x00E8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
	Reset value																															0	0
0x0100	FDCAN_CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																												0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

47 Universal serial bus full-speed host/device interface (USB)

47.1 USB introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB2 bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

47.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports both Host and Device modes
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

47.3 USB implementation

Table 486 describes the USB implementation in the devices.

Table 486. STM32C5 USB implementation

USB features(1)	USB
Host mode	X
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	2048 bytes
Dedicated packet buffer memory SRAM access scheme	32 bits
USB 2.0 Link Power Management (LPM) support in device	X
Battery Charging Detection (BCD) support for device	X
Embedded pull-up resistor on USB_DP line	X

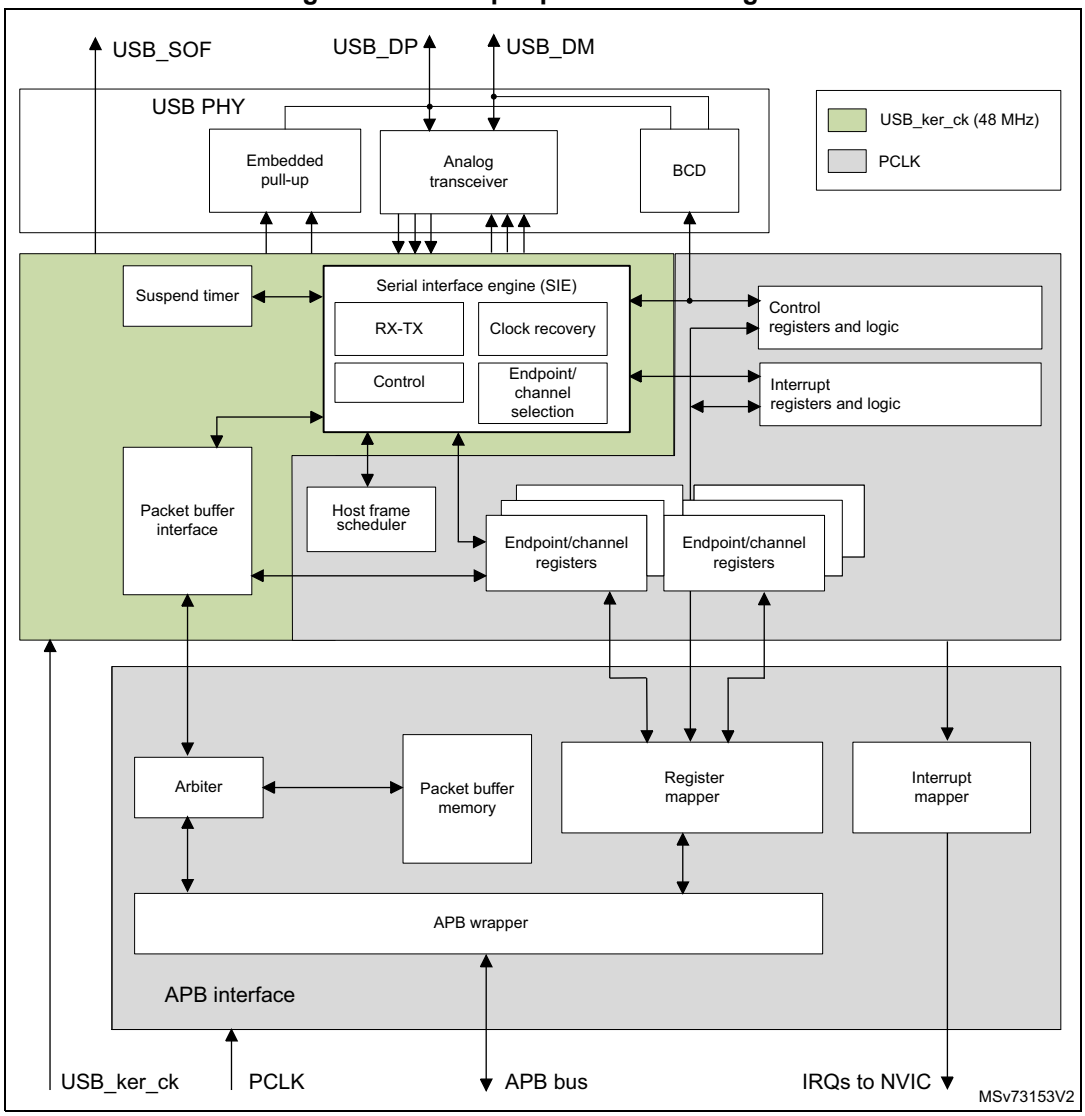
1. X= supported

47.4 USB functional description

47.4.1 USB block diagram

Figure 703 shows the block diagram of the USB peripheral.

Figure 703. USB peripheral block diagram



47.4.2 USB pins and internal signals

Table 487. USB input/output pins

Pin name	Pin type	Description
USB_DP	Digital input/output	D+ line
USB_DM	Digital input/output	D- line
USB_SOF	Digital output	SOF (start of frame indicator)

47.4.3 USB reset and clocks

A single reset is present on USB. The RCC allows a reset to be forced by software.

There are two clocks:

- PCLK for the APB bus interface and registers.
- USB_ker_ck (48 MHz) for the main protocol logic including notably the serial interface engine (SIE), see USB_ker_ck clock domain in block diagram.

47.4.4 General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

Host mode and specific functionality

A single bit, HOST, in register USB_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology “channel” is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

Note: Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).

For LPM (link power management) this feature is not supported in Host mode.

47.4.5 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for battery charging detection (BCD), multiplexed on same USB_DP and USB_DM lines.
- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB_Reset, data errors etc. and to endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.
- Timer: this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet buffer interface: this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.

- Endpoint/channel-related registers: each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control registers: these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt registers: these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note: * Endpoint/channel 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB2 bus through an APB2 interface, containing the following blocks:

- Packet memory: this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.
- Arbiter: this block accepts memory requests coming from the APB2 bus and from the USB interface. It resolves the conflicts by giving priority to APB2 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB2 transfers of any length are also allowed by this scheme.
- Register mapper: this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB2.
- APB2 wrapper: this provides an interface to the APB2 for the memory and register. It also maps the whole USB peripheral in the APB2 address space.
- Interrupt mapper: this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

47.4.6 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.

Host channels are divided in two categories:

- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwith access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows

- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame, the host scheduler:

1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non-periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.

47.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

47.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

47.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing

the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB bus reset (RST_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB_DADDR register and initializing the CHEP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RST_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

USB bus reset in Host mode

In Host mode a bus reset is activated by setting the USBRST bit of the USB_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

Structure and usage of packet buffers

Each bidirectional endpoint can receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB2 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB2 bus. Different clock configurations are possible where the APB2 clock frequency can be higher or lower than the USB peripheral one.

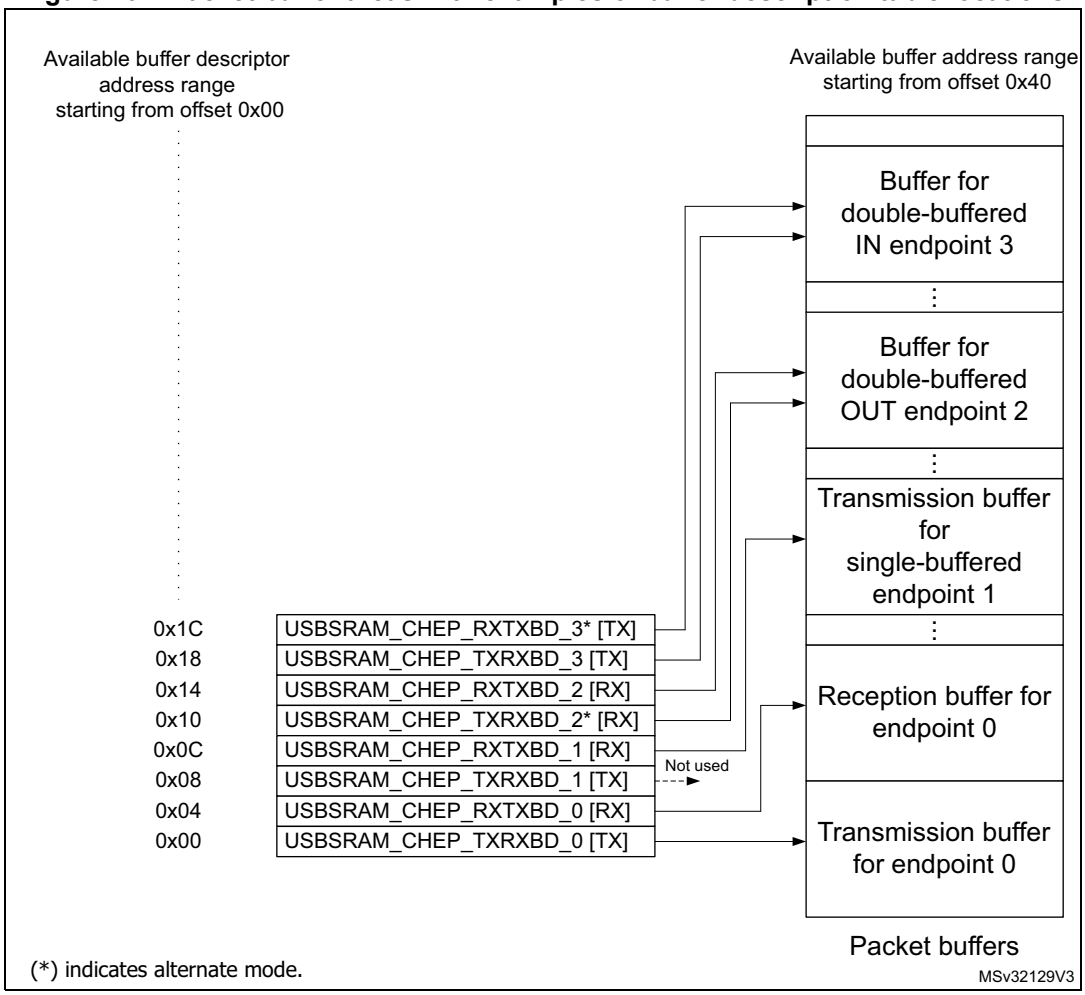
Note: Due to USB data rate and packet memory interface requirements, the APB2 clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also

located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-byte boundary. Buffer descriptor table entries are described in *Section 47.7: USBSRAM registers*. If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to *Section 47.5.5: Isochronous transfers in Device mode* and *Section 47.5.3: Double-buffered endpoints and usage in Device mode* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 704*.

For Host mode different sections explain the buffer usage model, notably *Section 47.5.6: Isochronous transfers in Host mode* and *Section 47.5.4: Double buffered channels: usage in Host mode*.

Figure 704. Packet buffer areas with examples of buffer description table locations



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX fields in the CHEP_TXBD_n and CHEP_RXBD_n registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB_CHEPnR register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB_CHEPnR register and COUNTn_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BLSIZE and NUM_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_CHEPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), must not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP_TXBD_n (fields ADDRn_TX and COUNTn_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to *Structure and usage of packet buffers on page 2093*) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB_CHEPnR register.

The ADDRn_TX field in the internal register CHEP_TXBD_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR_TX if one of the following conditions occurs:

- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

Data reception in Device mode (OUT and SETUP packets)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM_BLOCK bit fields, which are read within USBSRAM_CHEP_RXBD_n content, are used to initialize BUF_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits

STATRX in the USB_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BLSIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet

an error condition is signaled via the bits VTRX and ERR_RX if one of the following conditions occurs:

- False EOP
- Bit stuffing error
- Wrong CRC

Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_CHEPnR register at each VTRX event to distinguish normal OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EPKIND in the USB_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a

control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR_TX set (which can generate an interrupt).

47.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB_CHEPnR register) for ‘reception’ double-buffered bulk endpoints or DTOGTX (bit 6 of USB_CHEPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in

use by the application software, so to be aware of any conflict. Since in the USB_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_CHEPnR register bits and DTOG/SW_BUF definition is explained, for the cases of 'transmission' and 'reception' double-buffered bulk endpoints.

Table 488. Double-buffering buffer flag definition

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOGTX (USB_CHEPnR bit 6)	DTOGRX (USB_CHEPnR bit 14)
SW_BUF	USB_CHEPnR bit 14	USB_CHEPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 489. Bulk double-buffering memory buffers usage (Device mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (IN)	0	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by performing the two following actions:

- Writing UTYPE bit field at 00 in its USB_CHEPnR register, to define the endpoint as a bulk
- Setting EPKIND bit at 1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see *Table 489*). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

47.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below *Table 490: Bulk double-buffering memory buffers usage (Host mode)* shows the programming settings for OUT and IN tokens.

Table 490. Bulk double-buffering memory buffers usage (Host mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (OUT)	0	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (IN)	0	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

47.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering

to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOGRX for ‘reception’ isochronous endpoints, DTOGTX for ‘transmission’ isochronous endpoints, both in the related USB_CHEPnR register) according to *Table 491*.

Table 491. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (IN)	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overrun conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB_ISTR register anyway, in order to notify the software of the possible data corruption.

47.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.

The mechanism is based on a table very similar to that for Device mode. See *Table 492* to understand the relationship between the DTOG bit buffers and the buffer usage.

Table 492. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (OUT)	0	USBSRAM_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations
	1	USBSRAM_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.
Receive (IN)	0	USBSRAM_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.
	1	USBSRAM_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID),

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

47.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set SUSPRDY bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to ensure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear SUSPEN bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to *Table 493*, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST_DCON bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.

Table 493. Resume event detection

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB_CNTR register to 1.

Suspend and resume in Host mode

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB_CNTR to 1.

Below in *Table 494*, the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.

Table 494. Resume event detection for host

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Not allowed (noise on bus)	Go back in Suspend mode
“10”	Full speed capable device: Not allowed (noise on bus) Low speed device: Device remote wake-up resume	None
“01”	Full speed capable device: Device remote wake-up resume Low speed device: Not allowed (noise on bus)	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

47.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common registers: interrupt and control registers. These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.
  - USB_CNTR
  - USB_ISTR
  - USB_FNR
  - USB_DADDR
  - USB_LPMCSR
  - USB_BCDR
- Endpoint/channel registers: endpoint/channel configuration and status
  - USB_CHEPnR

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

47.6.1 USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M
rw														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	r	rw	rw

Bit 31 **HOST:** HOST mode

HOST bit selects betweens host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.

0: USB Device function

1: USB host function

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **DDISCM**: Device disconnection mask

– Host mode

0: Device disconnection interrupt disabled

1: Device disconnection interrupt enabled

Bit 16 **THR512M**: 512 byte threshold interrupt mask

0: 512 byte threshold interrupt disabled

1: 512 byte threshold interrupt enabled

Bit 15 **CTRM:** Correct transfer interrupt mask

0: Correct transfer (CTR) interrupt disabled.

1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask

0: PMAOVR interrupt disabled.

1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM:** Error interrupt mask

0: ERR interrupt disabled.

1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM:** Wake-up interrupt mask

0: WKUP interrupt disabled.

1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 **SUSPM:** Suspend mode interrupt mask

0: Suspend mode request (SUSP) interrupt disabled.

1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RST_DCONM:** USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask
0: RESET interrupt disabled.
1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM:** Start of frame interrupt mask
0: SOF interrupt disabled.
1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM:** Expected start of frame interrupt mask
0: Expected start of frame (ESOF) interrupt disabled.
1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 **L1REQM:** LPM L1 state request interrupt mask
0: LPM L1 state request (L1REQ) interrupt disabled.
1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **L1RES:** L1 remote wake-up / resume driver
– Device mode
Software sets this bit to send a LPM L1 50 µs remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.
0: No effect
1: Send 50 µs remote-wake-up signaling to host

Bit 4 **L2RES:** L2 remote wake-up / resume driver
– Device mode
The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.
– Host mode
Software sets this bit to send resume signaling to the device.
Software clears this bit to send end of resume to device and restart SOF generation.
In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.
0: No effect
1: Send L2 resume signaling to device

Bit 3 **SUSPEN:** Suspend state enable

– Condition: Device mode

Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.

As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

– Condition: Host mode

Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.

As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

Bit 2 **SUSPRDY:** Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

0: Normal operation

1: Suspend state

Bit 1 **PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.
0: Exit power down
1: Enter power down mode

Bit 0 **USBRST:** USB Reset

– Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB.The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.
0: No effect
1: USB core is under reset

– Condition: Host mode

Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.
0: No effect
1: USB reset driven

47.6.2 USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_CHEPnR register (the CTR bit is actually a read only bit). For endpoint-/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LS_ DCON	DCON_ STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR 512
	r	r												rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RST_ DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LS_DCON**: Low speed device connected

– Host mode:

This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state.

Bit 29 **DCON_STAT**: Device connection status

– Host mode:

This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected.

0: No device connected

1: FS or LS device connected to the host

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **DDISC**: Device connection

– Host mode

This bit is set when a device connection is detected. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 16 **THR512**: 512 byte threshold interrupt

This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel.

Bit 15 **CTR**: Completed transfer in host mode

This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR:** Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOVR interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 13 **ERR:** Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 12 **WKUP:** Wake-up

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 11 **SUSP:** Suspend mode request

– Device mode

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 10 **RST_DCON:** USB reset request (Device mode) or device connect/disconnect (Host mode)

– Device mode

This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.

– Host mode

This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.

Bit 9 **SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine can monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 8 **ESOF:** Expected start of frame

– Device mode

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 **L1REQ:** LPM L1 state request

– Device mode

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.
If DIR bit = 0, VTTX bit is set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).
If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]:** Device Endpoint / host channel identification number

These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.

47.6.3 USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0000 0XXX (where X is undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 13 **LCK:** Locked

– Device mode

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

– Device mode

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

47.6.4 USB Device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EF:** Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB_CHEPnR registers.

Bits 6:0 **ADD[6:0]:** Device address

– Device mode

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

– Host mode

These bits contain the address transmitted with the LPM transaction

47.6.5 USB LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN
								r	r	r	r	r		rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **BESL[3:0]:** BESL value

– Device mode

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REMWAKE:** bRemoteWake value

– Device mode

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LPMACK:** LPM token acknowledge enable

– Device mode:

0: the valid LPM token is NYET.
1: the valid LPM token is ACK.
The NYET/ACK is returned only on a successful LPM transaction:
No errors in both the EXT token and the LPM token (else ERROR)
A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN:** LPM support enable

– Device mode

This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.

47.6.6 USB battery charging detector (USB_BCDR)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
rw								r	r	r	r	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DPPU_DPD:** DP pull-up / DPDM pull-down

– Device mode

This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.

– Host mode

This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET:** DM pull-up detection status

– Device mode

This bit is active only during PD and gives the result of comparison between DM voltage level and V_{LGC} threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

Bit 6 **SDET:** Secondary detection (SD) status

– Device mode

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

Bit 5 **PDET:** Primary detection (PD) status

– Device mode

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 **DCDET:** Data contact detection (DCD) status

– Device mode

This bit gives the result of DCD.

0: data lines contact not detected.

1: data lines contact detected.

Bit 3 **SDEN:** Secondary detection (SD) mode enable

– Device mode

This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 2 **PDEN:** Primary detection (PD) mode enable

– Device mode

This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 1 **DCDEN:** Data contact detection (DCD) mode enable

– Device mode

This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 0 **BCDEN:** Battery charging detector (BCD) enable

– Device mode

This bit is set by the software to enable the BCD support within the USB Device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to 0 in order to allow the normal USB operation.

47.6.7 USB endpoint/channel n register (USB_CHEPnR)

Address offset: 0x00 + 0x4 * n, (n = 0 to 7)

Reset value: 0x0000 0000

The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an USB_CHEPnR register is available to store the endpoint/channel specific information.

They are also reset when an USB reset is received from the USB bus or forced through bit USBRST in the CTLR register, except the VTRX and VTTX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its USB_CHEPnR register where *n* is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	THREE_ERR_RX[1:0]		THREE_ERR_TX[1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	rc_w0	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VTRX	DTOG RX	STATRX[1:0]		SETUP	UTYPE[1:0]		EP KIND	VTTX	DTOG TX	STATTX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **THREE_ERR_RX[1:0]:** Three errors for an IN transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. THREE_ERR_RX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bits 28:27 **THREE_ERR_TX[1:0]:** Three errors for an OUT or SETUP transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. THREE_ERR_TX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bit 26 **ERR_RX:** Received error for an IN transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 25 **ERR_TX:** Received error for an OUT/SETUP transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 24 **LS_EP:** Low speed endpoint – host with HUB only

– Host mode

This bit is set by the software to send an LS transaction to the corresponding endpoint.
0: Full speed endpoint
1: Low speed endpoint

Bit 23 **NAK:**

– Host mode

This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.

Bits 22:16 **DEVADDR[6:0]:**

– Host mode

Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX:** USB valid transaction received

– Device mode

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written, writing 1 has no effect.

– Host mode

This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit
If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.
If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.
Errors can be seen via the bits ERR_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.

Bit 14 **DTOGRX:** Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).
If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to *Section 47.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to *Section 47.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STATRX[1:0]:** Status bits, for reception transfers

– Device mode

These bits contain information about the endpoint status, which are listed in *Table 495: Reception status encoding on page 2125*. These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred (VTRX = 1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.
Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 47.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

These bits are the host application controls to start, retry, or abort host transactions driven by the channel.
These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

- DISABLE
  DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.
- VALID
  A host channel is actively trying to submit USB transaction to device only when in VALID state.VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.
  VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.
- NAK
  NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE
- STALL
  STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.

Bit 11 **SETUP:** Setup transaction completed

– Device mode

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.

– Host mode

This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]:** USB type of transaction

These bits configure the behavior of this endpoint/channel as described in *Table 496: Endpoint/channel type encoding*. Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there can be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.
The usage of isochronous channels/endpoints is explained in *Section 47.5.5: Isochronous transfers in Device mode*

Bit 8 **EPKIND:** endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. *Table 497* summarizes the different meanings.
DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 47.5.3: Double-buffered endpoints and usage in Device mode*.
STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit can be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX:** Valid USB transaction transmitted

– Device mode

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written.

– Host mode

Same as VTRX behavior but for USB OUT and SETUP transactions.

Bit 6 **DTOGTX:** Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).
If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to *Section 47.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to *Section 47.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STATTX[1:0]:** Status bits, for transmission transfers

– Device mode

These bits contain the information about the endpoint status, listed in *Table 498*. These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred (VTTX = 1) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.
Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to *Section 47.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint is defined as isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to ‘STALL’ or ‘NAK’ for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

The STATTX bits contain the information about the channel status. Refer to *Table 498* for the full descriptions (“Host mode” descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

Bits 3:0 **EA[3:0]:** endpoint/channel address

– Device mode

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

– Host mode

Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.

Table 495. Reception status encoding

STATRX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.
10	**NAK**: Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request.
11	**VALID**: this endpoint/channel is enabled for reception.

Table 496. Endpoint/channel type encoding

UTYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 497. Endpoint/channel kind meaning

UTYPE[1:0]		EPKIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint
11	INTERRUPT	Not used

Table 498. Transmission status encoding

STATTX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.

Table 498. Transmission status encoding (continued)

STATTX[1:0]	Meaning
10	**NAK**: Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request.
11	**VALID**: this endpoint/channel is enabled for transmission.

47.6.8 USB register map

The table below provides the USB register map and reset values.

Table 499. USB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_CHEP0R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_CHEP1R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USB_CHEP2R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_CHEP3R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USB_CHEP4R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_CHEP5R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 499. USB register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	USB_CHEP6R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	USB_CHEP7R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	USB_CNTR	HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
	Reset value	0														0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1
0x44	USB_ISTR	Res.	LS_DCON	DCON_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR512	CTR	PMAOVR	ERR	WKUP	SUSP	RST_DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
	Reset value		0	0												0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x48	USB_FNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	USB_DADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x54	USB_LPMCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REMWAKE	Res.	LPMACK	LPMEN
	Reset value																									0	0	0	0	0		0	0
0x58	USB_BCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
	Reset value																	0								0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

47.7 USBSRAM registers

Note: The buffer descriptor table is located inside the packet buffer memory in the separate "USBSRAM" address space.

Although the buffer descriptor table is located inside the packet buffer memory ("USBSRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USBSRAM base address. The buffer descriptor table entry associated with the USB_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 2093*.

47.7.1 Channel/endpoint transmit buffer descriptor n (USBSRAM_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

47.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USBSRAM_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during

the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 500*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

Table 500. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
0 (00000)	Not allowed	32 bytes
1 (00001)	2 bytes	64 bytes
2 (00010)	4 bytes	96 bytes
3 (00011)	6 bytes	128 bytes
...	...	...
14 (01110)	28 bytes	480 bytes
15 (01111)	30 bytes	512
16 (10000)	32 bytes	544
...	...	...

Table 500. Definition of allocated buffer memory (continued)

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
29 (11101)	58 bytes	...
30 (11110)	60 bytes	992 bytes
31 (11111)	62 bytes	1023 bytes

47.7.3 Channel/endpoint receive buffer descriptor n (USBSRAM_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 500*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

47.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USBSRAM_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

47.7.5 USBSRAM register map

The table below provides the USB register map and reset values.

Table 501. USBSRAM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USBSRAM_CHEP_TXRXBD_0	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x00	USBSRAM_CHEP_TXRXBD_0 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	USBSRAM_CHEP_RXTXBD_0	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	USBSRAM_CHEP_RXTXBD_0 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	USBSRAM_CHEP_TXRXBD_1	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	USBSRAM_CHEP_TXRXBD_1 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	USBSRAM_CHEP_RXTXBD_1	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	USBSRAM_CHEP_RXTXBD_1 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
. . . .	. . .	. . .																															
0x38	USBSRAM_CHEP_TXRXBD_7	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x38	USBSRAM_CHEP_TXRXBD_7 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	USBSRAM_CHEP_RXTXBD_7	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	USBSRAM_CHEP_RXTXBD_7 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

48 Ethernet (ETH): media access control (MAC) with DMA controller

48.1 Ethernet introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The Ethernet peripheral enables to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2015 standard.

The peripheral is configurable to meet the needs of a large variety of consumer and industrial applications.

48.2 Ethernet main features

The Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

48.2.1 Standard compliance

The Ethernet peripheral is compliant with the following standards:

- IEEE 802.3-2015 for Ethernet MAC and media independent interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master and AHB slave ports
- RMII specification version 1.2 from RMII consortium

48.2.2 MAC features

MAC Tx and Rx common features

- Separate transmission, reception, and control interfaces to the application
- 10, 100 Mbit/s data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD protocol support
  - Flow control using back pressure (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)

Note: Ensure that only one queue is active for half-duplex operation.

- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII PHYs

- 32-bit data transfer interface on the application side
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Network statistics with RMON or MIB counters (partial support of RFC2819/RFC2665)
- Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping are supported in Tx direction.
- Flexibility to control pulse-per-second (PPS) output signal (eth_ptp_pps_out and ETH_PPS_OUT)
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

MAC Tx features

- Preamble and start-of-frame data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or Jumbo Ethernet packets of up to 16 Kbytes
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in Full-duplex mode)
- Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in Full-duplex mode
- Insert, replace, or delete queue/channel-based VLAN tags

MAC Rx features

- Automatic Pad and CRC stripping options
- Option to disable automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - Four 48-bit perfect (DA) address filters with masks for each byte
  - Four 48-bit SA address comparison check with masks for each byte
  - 64 bit Hash filter for multicast and unicast (DA) addresses
- Option to pass all multicast addressed packets
- Promiscuous mode to pass all packets without any filtering for network monitoring
- Pass all incoming packets (as per filter) with a status report

- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering based either on the outer or inner VLAN tag
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- Detection of remote wake-up packets and AMD magic packets
- Optional forwarding of received Pause packets to the application (in Full-duplex mode)
- Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN tags and providing the tags in the status

48.2.3 Transaction layer (MTL) features

MTL Tx and Rx common features

- 32-bit transaction layer block (bridges the application and the MAC)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)

MTL Tx features

- 2048-byte transmit FIFO with programmable threshold capability
- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Automatic retransmission of collision packets in Half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum on frames transmitted in Store-and-forward mode
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Packet-level control for
  - VLAN tag insertion or replacement
  - Ethernet source address insertion
  - Layer 3/Layer 4 checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control

MTL Rx features

- 2048-byte receive FIFO with configurable threshold
- Programmable Rx queue threshold (default fixed at 64 bytes) in threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO

48.2.4 DMA block features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see *Section 48.11.2: Ethernet DMA registers*) to control the DMA operations. The DMA block supports the following features:

- 32-bit data transfers
- Separate DMA in transmit path and receive paths
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet transmit or receive complete interrupt control
- Round-robin or fixed-priority arbitration between the receive and transmit engines
- Start and Stop modes
- Separate ports for host control (AHB) access and host data interface
- Tx DMA channel with TCP segmentation offload (TSO) feature enabled
- Programmable control for transmit descriptor posted writes to improve the throughput

48.2.5 Bus interface features

AHB master interface

The AHB master interface features are the following:

- Interfaces with the application through AHB
- 32-bit data on the AHB master port
- Split, Retry, and Error AHB responses
- AHB 1-Kbyte boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)

The AHB master interface does not generate the following:

- Wrap burst
- Locked or protected transfers
- 32-bit address width

AHB slave interface

The AHB slave interface supports the following features:

- Interfaces with the application through AHB
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:

- Split
- Retry

48.3 Ethernet pins and internal signals

Table 502 lists the Ethernet inputs and output signals connected to package pins or balls. Active pins depend on the PHY type selected (MII or RMII) and on the device configuration.

Table 503 shows the internal Ethernet signals.

Table 502. Ethernet peripheral pins

Port name	Digital port type	Description
ETH_COL	Input	Collision detection signal, MII only.
ETH_CRS	input	Carrier sense signal, MII only
ETH_REF_CLK	Input	RMII reference clock
ETH_RX_CLK	Input	MII timing reference for Rx data transfers
ETH_RXD[3:0]	Input	Receive data. 4 pins for MII, 2 for RMII.
ETH_RX_DV	Input	Receive data valid
ETH_CRS_DV	Input	RMII: Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
ETH_RX_ER	Input	Receive error
ETH_TX_CLK	Input	MII timing reference for Tx data transfers
ETH_TXD[3:0]	Output	Transmit data. 4 pins for MII, 2 for RMII.
ETH_TX_EN	Output	Transmit data enable
ETH_TX_ER	Output	Transmit error
ETH_MDC	Output	Management data clock
ETH_MDIO	Input/output	Management data
ETH_PHY_INTN	Input	PHY interrupt

Table 502. Ethernet peripheral pins (continued)

Port name	Digital port type	Description
ETH_PPS_OUT	Output	PTP pulse-per-second output
ETH_PTP_AUX_TS(1)	Input	Trigger input for the auxiliary snapshots of the PTP system time

1. This pin is not available on all devices.

Table 503. Internal Ethernet input/output signals

Signal name	Signal type	Description
eth_hclk	Digital input	AHB clock
eth_sbd_intr_it	Digital output	Main Ethernet interrupt
lpi_intr_o	Digital output	Sideband signal generated when the transmitter or receiver enters or exits the LPI state.
pmt_intr_o	Digital output	Sideband signal generated when a valid remote wake-up packet is received
eth_tx_clk	Digital input	Tx kernel clock
eth_rx_clk	Digital input	Rx kernel clock
eth_rmii_ref_clk	Digital input	RMII reference kernel clock
eth_ptp_pps_out	Digital output	PTP pulse-per-second signal
mac_speed_o[1:0]	Digital output	MAC speed information used by the RCC
clk_ptp_ref_i	Digital input	PTP reference clock input
eth_ptp_aux_ts_trig[3:0]	Digital input	Trigger input for auxiliary snapshots of the PTP system time

Table 504. Ethernet trigger connections

Type	ID	Source	Destination
Internal	eth_ptp_aux_ts_trig[0]	tim2_trgo	-
	eth_ptp_aux_ts_trig[1]	tim3_trgo	-
	eth_ptp_aux_ts_trig[3]	-	-
	eth_ptp_pps_out (PTP Flexible pulse-per-second output 0)	-	tim2_etr14 tim3_etr14 tim2_ti1_2 tim3_ti1_2
External	Flexible pulse-per-second output 0	-	ETH_PPS_OUT
	eth_ptp_aux_ts_trig[2]	ETH_PTP_AUX_TS	-

48.4 Ethernet architecture

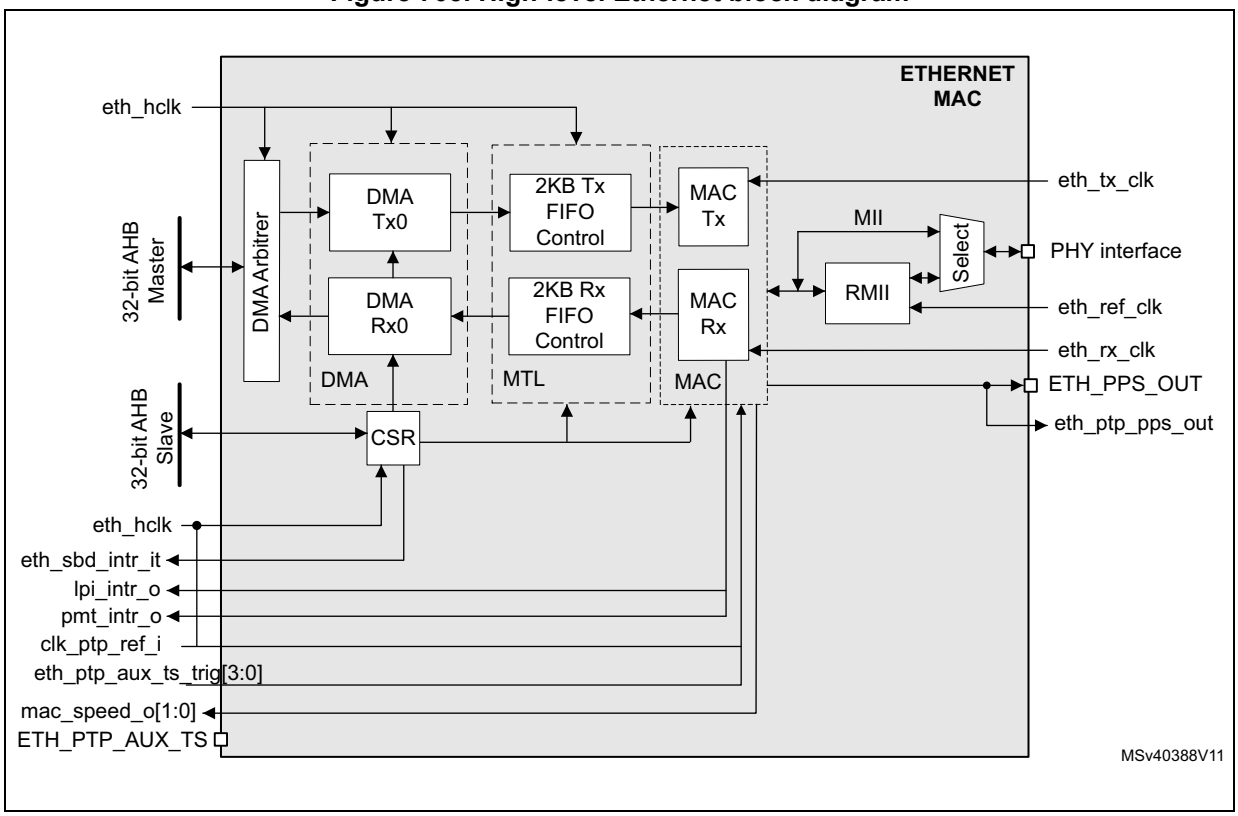
The Ethernet peripheral includes four main functional modules:

- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)

  This is the logical DMA module with one physical channel for reception and one for transmission. It controls the data transfers between MAC and system memory through the AMBA AHB 32-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The MAC transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII PHY media independent interfaces.

Figure 705. High-level Ethernet block diagram



1. For a definition of the internal signals, refer to *Table 503*.
2. Refer to RCC chapter “Clock distribution for Ethernet” for a detailed description of the Ethernet clock architecture.
3. The ETH_PTP_AUX_TS pin is not available on all devices.

48.4.1 DMA controller

The DMA has independent transmit (Tx) and receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet transmit and receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The base address of each list is written to the respective Tx and Rx registers: *Channel 0 Tx descriptor list address register (ETH_DMAC0TXDLAR)*) and *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*.

The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, *Channel 0 Tx descriptor ring length register (ETH_DMAC0TXRLR)* and *Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)*.

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

Descriptors are specified in *Section 48.10: Descriptors*.

DMA arbitration

The DMA module incorporates an arbiter that performs the arbitration between the Tx and Rx channels accesses from the AHB master interface. The following two types of arbitrations are supported and can be selected through *DMA mode register (ETH_DMAMR)*:

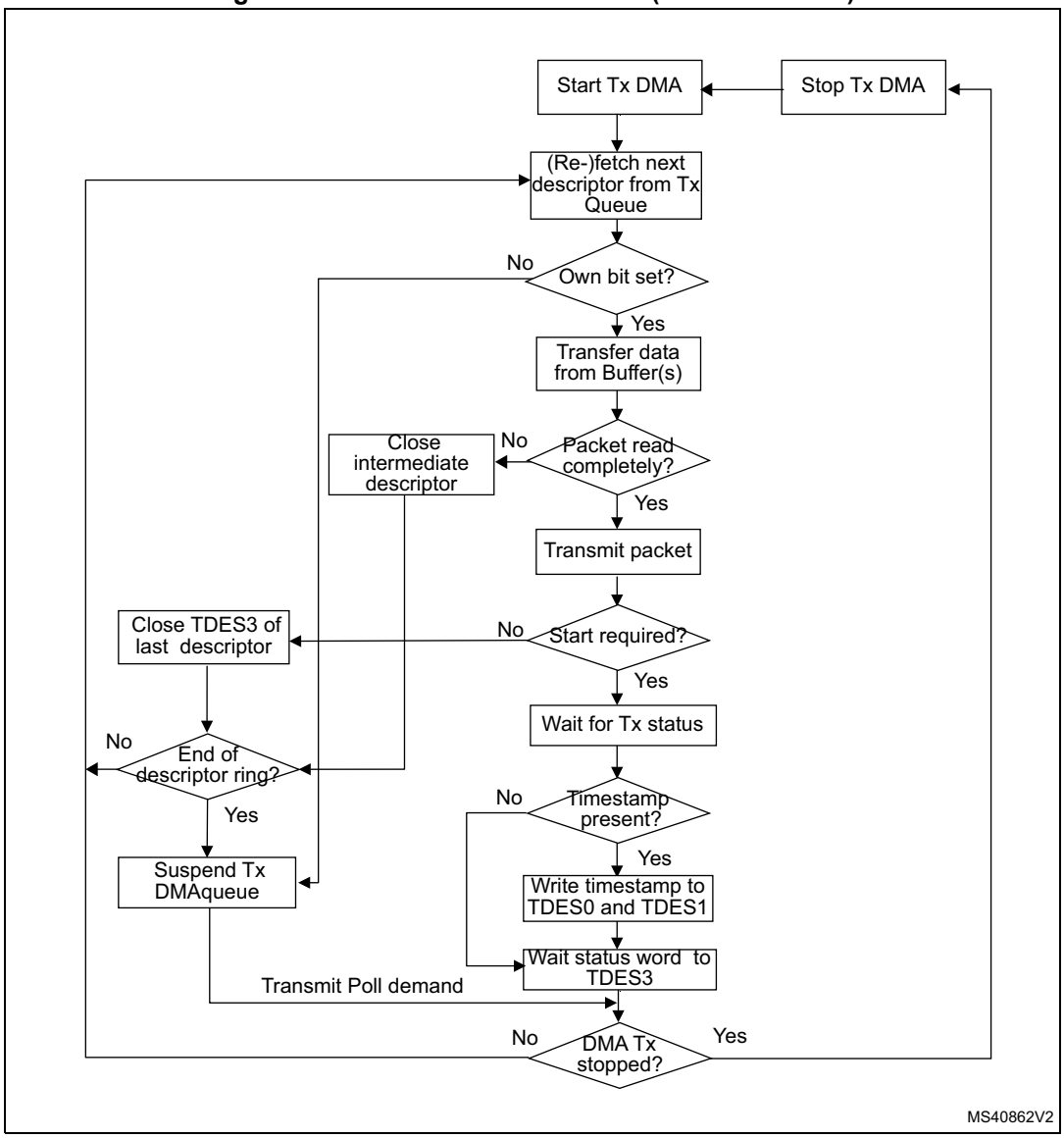
- Round-robin arbitration**:** the arbiter allocates the data bus between Rx and Tx in ratio set by Bits [14:12] of ETH_DMAMR.
- Fixed-priority arbitration**:** by default Rx DMA always gets priority over Tx DMA for data access. Setting bit 11 of ETH_DMAMR register gives priority to the Tx DMA.

DMA transmission in default mode

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application shifts the descriptor tail pointer offset value of the transmit channel.
3. The DMA fetches the descriptor from the application memory.
4. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   – The descriptor is flagged as owned by the application (TDES3 [31] = 0).
   – The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   – An error condition occurs.
5. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
6. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
7. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.
8. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.
9. Bit 0 of *Channel 0 status register (ETH_DMAC0SR)*) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
10. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the *Channel 0 Tx descriptor tail pointer register (ETH_DMAC0TXDTPR)* when it receives a Transmit Poll demand and the underflow interrupt status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.

Figure 706. DMA transmission flow (standard mode)



DMA transmission in OSP (operate on second packet) mode

In Run state, if bit 4 is set in the *Channel 0 transmit control register (ETH_DMAC0TXCR)*), the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

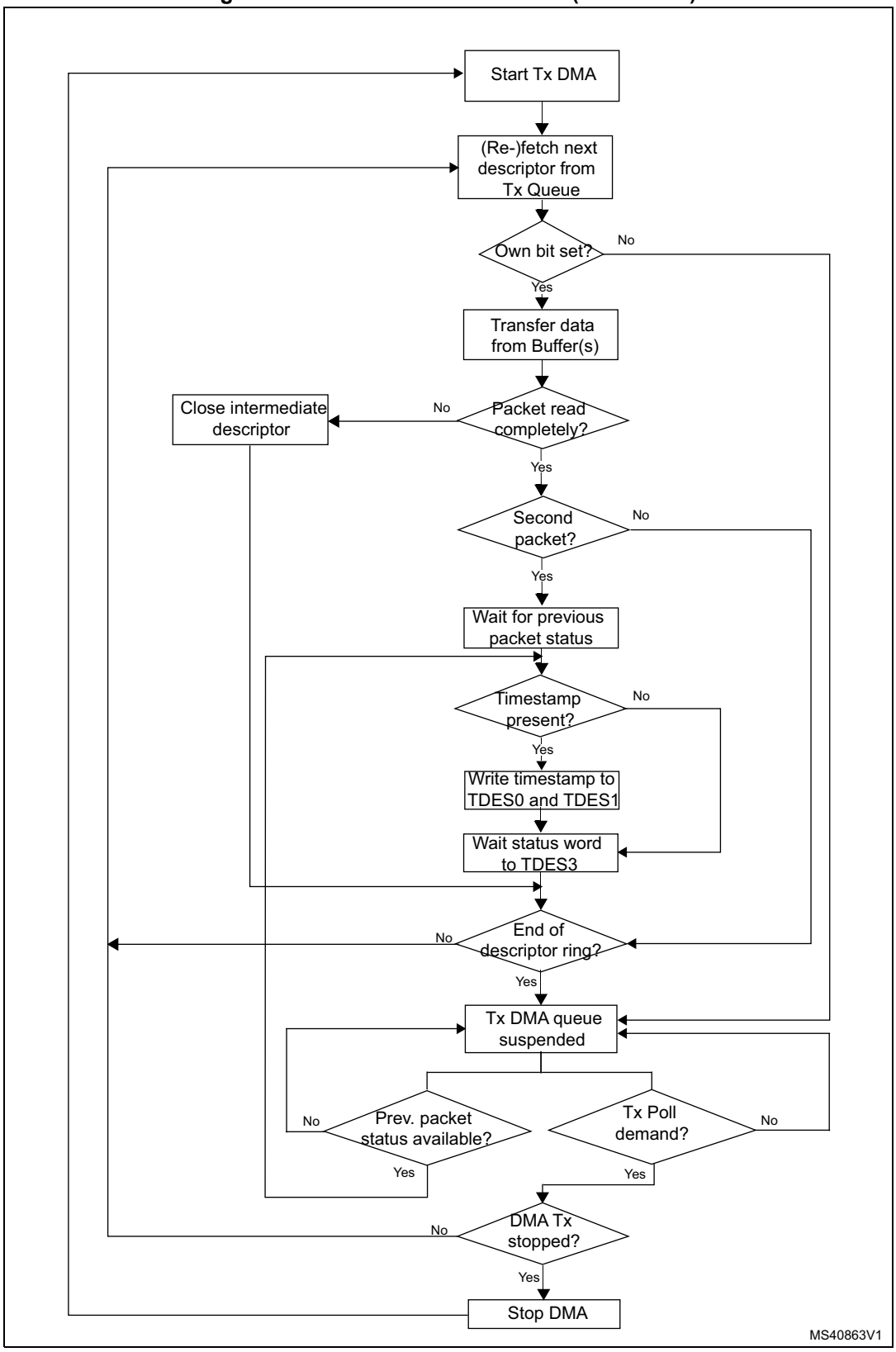
In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see *Section : DMA transmission in default mode*).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:
   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.
   a) The DMA writes the status to the corresponding TDES3.
   a) The DMA sets the relevant interrupts and returns to Suspend mode.

   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in the transmit descriptor tail pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in *Figure 708: Receive DMA flow*.

Figure 707. DMA transmission flow (OSP mode)



DMA reception

In the receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

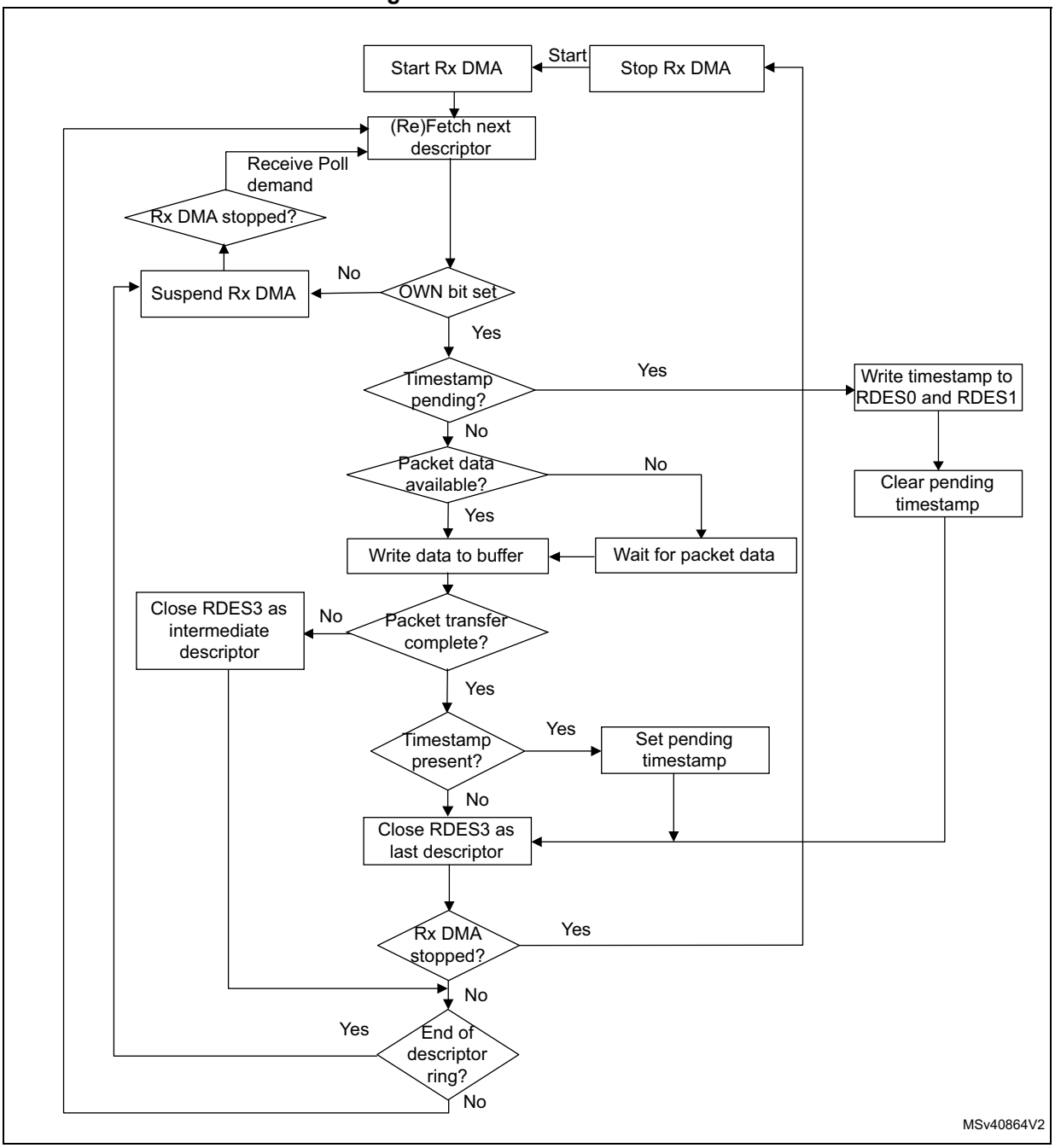
The DMA Rx descriptor ring structure is described in *Section 48.10: Descriptors*.

The reception sequence for Rx DMA engine is as follows (see also *Figure 708: Receive DMA flow*):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application should set the correct value in the receive descriptor tail pointer register of corresponding DMA channel to indicate the location of the last valid descriptor for the DMA. If the tail pointer points to descriptor N, the last valid descriptor for the DMA is descriptor N - 1.
2. When bit 0 of *Channel 0 receive control register (ETH_DMAC0RXCR)*) is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. The descriptors referenced between the current descriptor and the tail pointer registers are available for the DMA. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).
5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA retrieves the status of the receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.
8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. The DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).
11. The receive DMA exits the Suspend state when a receive Poll demand is given, and the application updates the channel receive descriptor tail pointer register to indicate the location of the last valid descriptor for DMA. Then, the engine proceeds to step 2 and fetches again the next descriptor.

Note: Refer to Section : Descriptor tail pointer handling examples for updating the correct value in receive descriptor tail pointer register.

Figure 708. Receive DMA flow



Priority scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channel 0 to access descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the *DMA mode register (ETH_DMAMR)* specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a given channel.

If the Tx DMA and Rx DMA of a given channel are enabled, the DMA which gets the bus when the channel gets control of the bus must be specified. The priority between the

corresponding Tx DMA and Rx DMA can be configured through the TXPR field of the *DMA mode register (ETH_DMAMR)*. For round-robin arbitration, the weighted priority between the Tx DMA and Rx DMA is configured through the PR field of the *DMA mode register (ETH_DMAMR)*. *Table 505* provides information about the priority scheme between Tx DMA and Rx DMA.

Table 505. Priority scheme for Tx DMA and Rx DMA

DMA mode register (ETH_DMAMR)					Priority scheme
PR[2:0]			TXPR	DA	
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.
0	0	1	0	0	Rx has priority over Tx in ratio 2:1.
0	1	0	0	0	Rx has priority over Tx in ratio 3:1.
0	1	1	0	0	Rx has priority over Tx in ratio 4:1.
1	0	0	0	0	Rx has priority over Tx in ratio 5:1.
1	0	1	0	0	Rx has priority over Tx in ratio 6:1.
1	1	0	0	0	Rx has priority over Tx in ratio 7:1.
1	1	1	0	0	Rx has priority over Tx in ratio 8:1.
x	x	x	1	1	Tx always has priority over Rx.
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.
0	0	1	1	0	Tx has priority over Rx in ratio 2:1.
0	1	0	1	0	Tx has priority over Rx in ratio 3:1.
0	1	1	1	0	Tx has priority over Rx in ratio 4:1.
1	0	0	1	0	Tx has priority over Rx in ratio 5:1.
1	0	1	1	0	Tx has priority over Rx in ratio 6:1.
1	1	0	1	0	Tx has priority over Rx in ratio 7:1.
1	1	1	1	0	Tx has priority over Rx in ratio 8:1.

48.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 32-bit wide data paths: the Transmit path and the receive path.

- Transmit path

  The application or internal DMA pushes the Ethernet packets read from the application or system memory into the Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 2048 bytes.

- Receive path

  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured receive threshold (RTC bits[1:0] defined in *Rx queue 0 operating mode register (ETH_MTLRXQ0OMR)*), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 2048 bytes.

48.4.3 MAC

The MAC is responsible of the Ethernet protocol processing.In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

MAC transmission

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   - The MAC completes the normal transmission and provides the transmission status to the MTL.
   - If a normal collision (in Half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:

     the packet was successfully transmitted;

     the maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) is observed from the MAC.

– If any one of the following event happens, the MAC aborts the packet transmission:

no carrier (Half-duplex mode)

loss of carrier (Half-duplex mode)

excessive deferral (Half-duplex mode)

late collisions (Half-duplex mode)

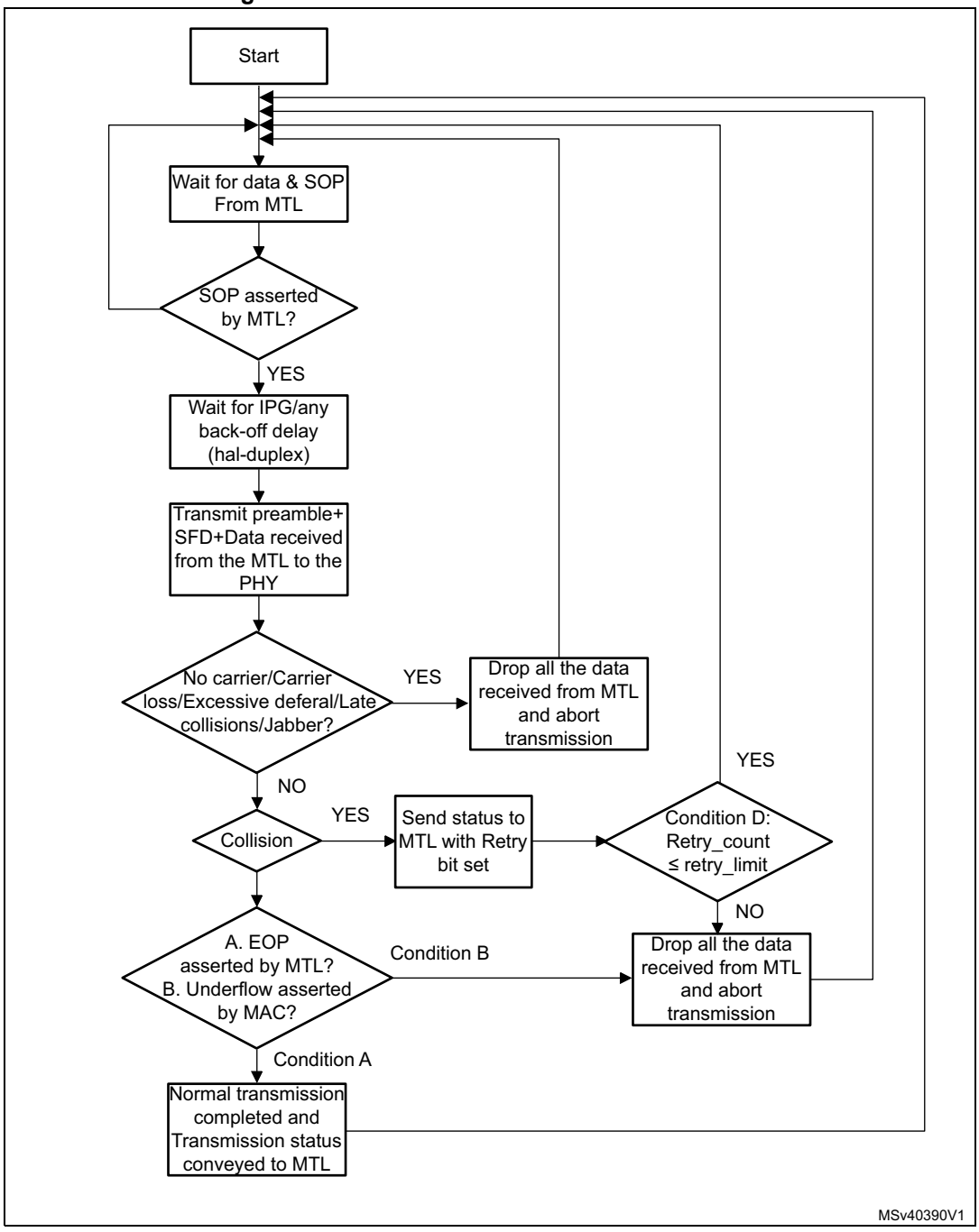
jabber

the MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.
5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

Figure 709: Overview of MAC transmission flow illustrates the MAC transmission process flow.

Figure 709. Overview of MAC transmission flow



MAC reception

A receive operation is initiated when the MAC detects an SFD on MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

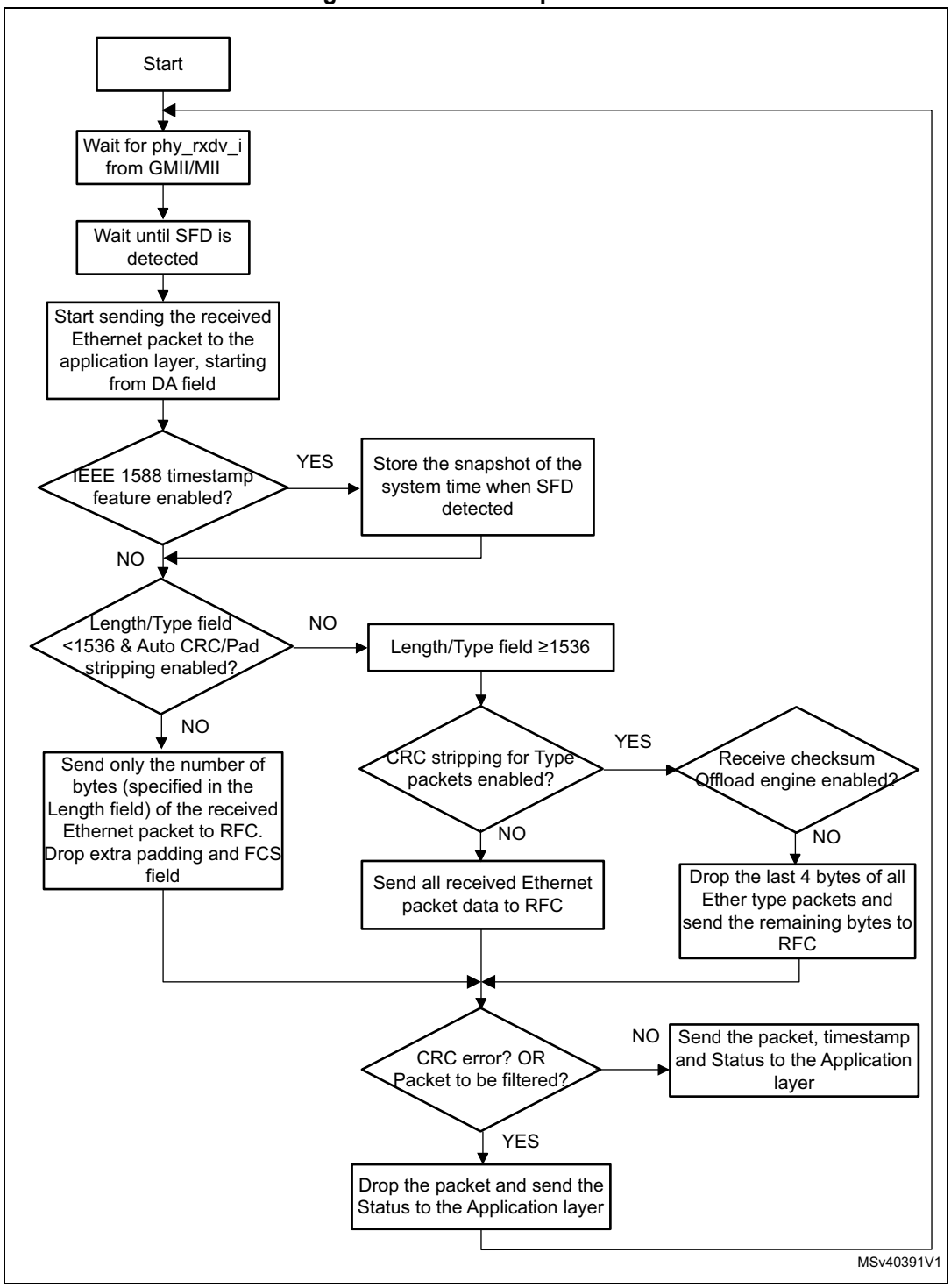
The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of MII becomes active, the receive State Machine (RSM) starts looking for the SFD field (0xD nibble).

   The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).
3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application. The MAC converts the received nibble data into bytes and forwards the valid packet data to the RFC module.
4. The receive state machine decodes the Length/Type field of the Ethernet packet being received.

   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the *Operating mode configuration register (ETH_MACCR)*), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.
5. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the *Operating mode configuration register (ETH_MACCR)*. However, if the CRC stripping has been enabled for Type packets and not enabled the receive checksum offload engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.
6. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of *Watchdog and jabber timeout register (ETH_MACWJBTR)*) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of *Operating mode configuration register (ETH_MACCR)*. However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.

Figure 710. MAC reception flow



48.5 Ethernet functional description: MAC

48.5.1 Double VLAN processing

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the receive path. Stripping and providing up to two VLAN Tags in the receive path as a part of the receive status

Transmit path

Table 506: Double VLAN processing features in Tx path describes the features supported by the MAC on the Transmit side.

Table 506. Double VLAN processing features in Tx path

Feature	Description
Support for C-VLAN and S-VLAN tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of *VLAN inclusion register (ETH_MACVIR)* and *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN tag deletion	VLAN tag deletion can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN tag insertion or replacement	VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the control word.

Receive path

Table 507: Double VLAN processing in Rx path describes the features supported by the MAC on the receive side and the corresponding bits in the *VLAN tag register (ETH_MACVTR)*.

Table 507. Double VLAN processing in Rx path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.
16-bit outer and inner VLAN tag and type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

48.5.2 Source address and VLAN insertion, replacement, or deletion

Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets

  To enable this feature for all packets, program the SARC field of the *Operating mode configuration register (ETH_MACCR)*.
- Enabling SA insertion or replacement for selective packets

  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of transmit descriptor Word 3/TDES3, refer to *Section 48.10.3: Transmit descriptor*) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or replacement by MAC

Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

VLAN insertion, replacement, or deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

  Insertion or replacement is performed based on the setting of VLTI bit in the *VLAN inclusion register (ETH_MACVIR)* as described in *Table 508: VLAN insertion or replacement based on VLTI bit*.

Table 508. VLAN insertion or replacement based on VLTI bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with VT field of Transmit context descriptor of the packet
VLTI bit is reset	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with the VLT field of *VLAN inclusion register (ETH_MACVIR)*

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of *VLAN inclusion register (ETH_MACVIR)*.
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see *Table 541: TDES2 normal descriptor (read format)*).

  In addition, the VLP (VLAN Priority control) bit must be reset in *VLAN inclusion register (ETH_MACVIR)* (for outer VLAN) and *Inner VLAN inclusion register (ETH_MACIVIR)* (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.

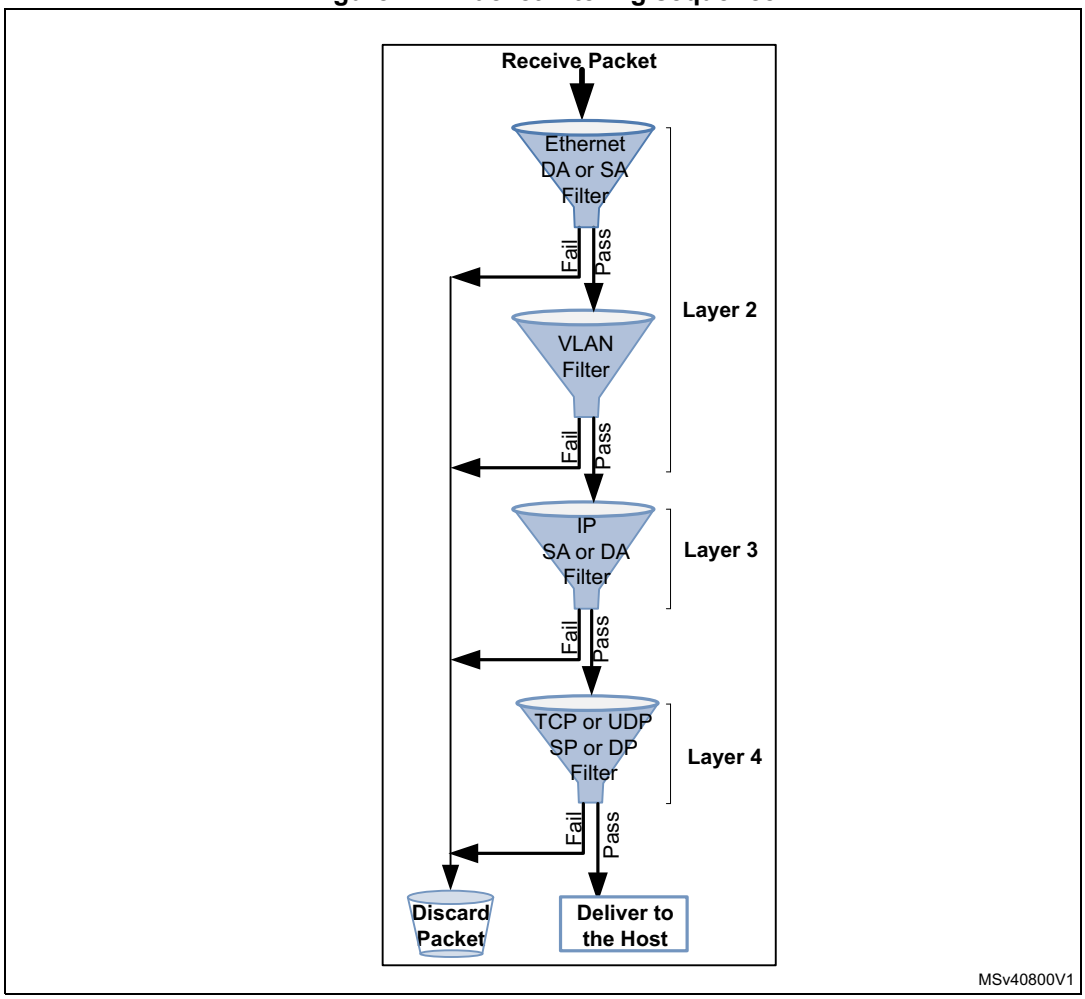
48.5.3 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- *MAC source or destination address filtering*: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- *VLAN filtering*: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- *Layer 3 and Layer 4 filtering*: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 711* shows the filtering sequence for Rx packets.

Figure 711. Packet filtering sequence



The sequence shown in *Figure 711* is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the Layer filters are not enabled, that filter is bypassed and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control.

For example, when RA bit of *Packet filtering control register (ETH_MACPFR)* is set to 1, all the discarded packets are forwarded to the host but with their packet status indicating the

specific filter failure. If RA bit is cleared to 0, VTFE and IPFE bits of *Packet filtering control register (ETH_MACPFR)* control if the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

MAC source or destination address filtering

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast destination address filtering

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of *Packet filtering control register (ETH_MACPFR)* is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in *MAC address x high register (ETH_MACAxHR)*. This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

Multicast destination address filtering

To program the MAC to pass all multicast packets, set the PM bit in *Packet filtering control register (ETH_MACPFR)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the *Packet filtering control register (ETH_MACPFR)*.

In Perfect filtering mode, the multicast address is compared with the programmed MAC destination address registers. Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

Hash or Perfect address filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Packet filtering control register (ETH_MACPFR)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

Broadcast address filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in *Packet filtering control register (ETH_MACPFR)*.

Unicast source address filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of *MAC address x high register (ETH_MACAxHR)*.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in *Packet filtering control register (ETH_MACPFR)*. Otherwise, the result of the SA filter is given as a status bit in the receive Status word (see *Table 510*). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

Inverse filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of *Packet filtering control register (ETH_MACPFR)*. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 509 and *Table 510* summarize the DA and SA filtering based on the type of packets received.

Note: When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

Table 509. Destination address filtering

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match

Table 509. Destination address filtering (continued)

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x
Multicast	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 510. Source address filtering

Packet type	PR	SAIF	SAF	SA filter operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN filtering

The MAC supports Perfect and Hash VLAN filtering. Refer to *Section 48.9.15: Programming guidelines to perform VLAN filtering on the receiver* for detailed programming steps.

VLAN tag Perfect filtering

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of *VLAN tag register (ETH_MACVTR)*. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of *VLAN tag register (ETH_MACVTR)*. The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

VLAN tag Hash filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN tag Hash filtering feature can be enabled using the VTHM (VLAN tag Hash Table match enable) bit of the *VLAN tag register (ETH_MACVTR)*. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag register (ETH_MACVTR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 511 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of *Packet filtering control register (ETH_MACPFR)* is set, all packets are received and the VLAN match status is indicated in the VF bit of *RDES2 normal descriptor (write-back format)*. When the RA bit is not set and the VTFE bit is set in *Packet filtering control register (ETH_MACPFR)*, the packet is dropped if the final VLAN match status is Fail. In *Table 511*, value *X* means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of *VLAN tag register (ETH_MACVTR)*, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in *VLAN tag register (ETH_MACVTR)*.

Table 511. VLAN match status(1)

VID	VLAN perfect filter match result	VTHM bit	VLAN hash filter match result	VTIM bit	Final VLAN match status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail

Table 511. VLAN match status(1) (continued)

VID	VLAN perfect filter match result	VTHM bit	VLAN hash filter match result	VTIM bit	Final VLAN match status
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, 'X' represents any value.

Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the *Operating mode configuration register (ETH_MACCR)* to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets

  The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of *Operating mode configuration register (ETH_MACCR)* is set and one of the following conditions is true:

  – All enabled Layer 3 and Layer 4 fields match.

  – At least one of the enabled field matches and other fields are bypassed or disabled

  When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets

  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets.The aborted or partial packets are dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programmed to a small value. Such packet transfer to the application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.
- Non-TCP or UDP IP Packets

  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

Layer 4 filtering

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 filters register set

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- *Layer4 address filter 0 register (ETH_MACL4A0R)*
- *Layer3 address 0 filter 0 register (ETH_MACL3A00R)*
- *Layer3 address 1 filter 0 register (ETH_MACL3A10R)*
- *Layer3 address 2 filter 0 register (ETH_MACL3A20R)*
- *Layer3 address 3 filter 0 register (ETH_MACL3A30R)*

The second independent set of registers includes: *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, *Layer 4 address filter 1 register (ETH_MACL4A1R)*, *Layer3 address 0 filter 1 Register (ETH_MACL3A01R)*, *Layer3 address 1 filter 1 register (ETH_MACL3A11R)*, *Layer3 address 2 filter 1 Register (ETH_MACL3A21R)* and *Layer3 address 3 filter 1 register (ETH_MACL3A31R)*.

48.5.4 IEEE 1588 timestamp support

The IEEE 1588 standard defines a precision time protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources.

This chapter contains the following sections:

- *IEEE 1588 timestamp support*
- *IEEE 1588 system time source*
- *IEEE 1588 auxiliary snapshots*
- *Flexible pulse-per-second output*
- *PTP timestamp offload function*
- *One-step timestamp*”

IEEE 1588 timestamp support

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588 2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement subsecond time in digital or binary format

Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- Ordinary clock

  The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

– Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in *Timestamp control register (ETH_MACTSCR)*.

– Maintenance of the data sets such as timestamp values.

The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 512. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in *Timestamp control register (ETH_MACTSCR)*.

- Boundary clock

  The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and signaling end in the protocol engine of the boundary clock. Such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

  The boundary clock is similar to the ordinary clock except for the following features:

  – The clock data sets are common to all ports of the boundary clock.

  – The local clock is common to all ports of the boundary clock.

- End-to-end transparent clock

  The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the ingress port to the egress port.

  The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both ingress and egress ports only for the messages mentioned in *Table 513*. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the *Timestamp control register (ETH_MACTSCR)*.

Table 513. End-to-end transparent clock: PTP messages for snapshot

PTP messages
SYNC
Delay_Req

- Peer-to-peer transparent clock

  In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

  The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

  The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in *Table 521*.

Table 514. Peer-to-peer transparent clock: PTP messages for snapshot

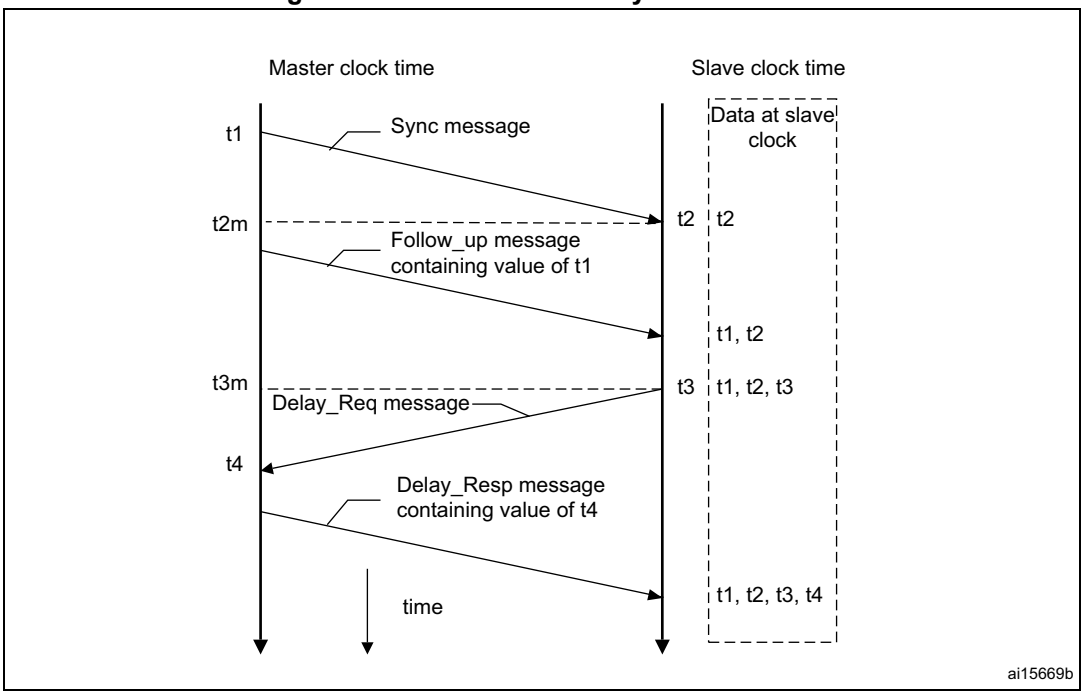
PTP messages
SYNC
Pdelay_Req
Pdelay_Resp

You can take the snapshot by setting the SNAPTYPESEL bit to 11 in *Timestamp control register (ETH_MACTSCR)*.

Delay request-response mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 712* shows the process that PTP uses for synchronizing a slave node with a master node by exchanging PTP messages.

Figure 712. Networked time synchronization



As shown in *Figure 712*, the PTP uses the following process:

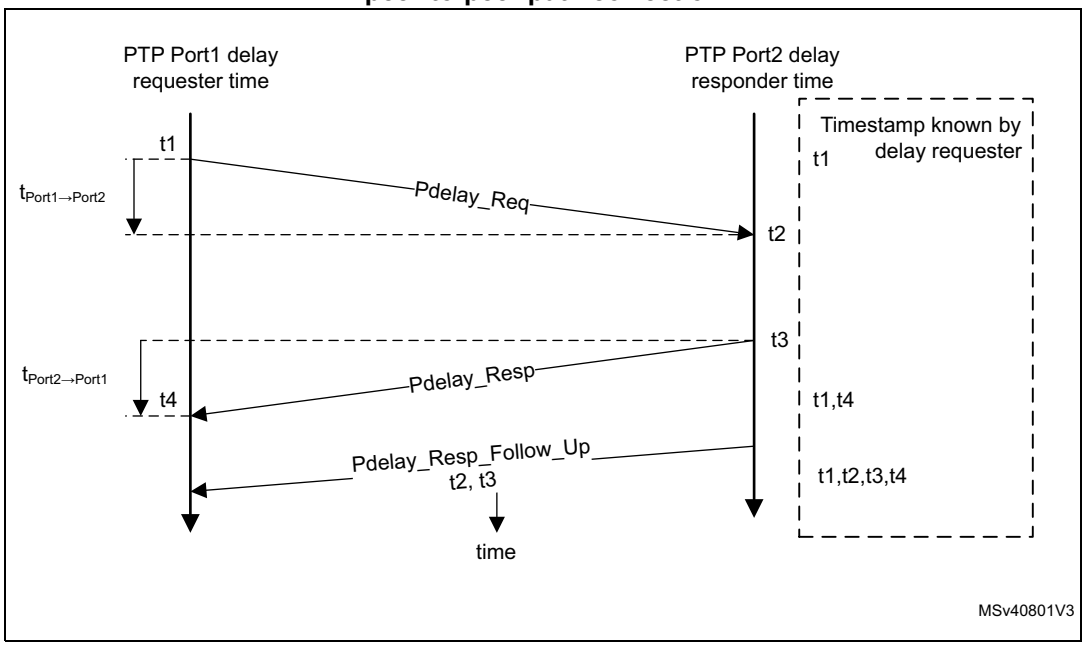
1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t_1. This time must be captured for Ethernet ports at MII.
2. The slave receives the SYNC message and also captures the exact time, t_2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, t_3, at which this packet leaves the MII interface.
5. The master receives the message, capturing the exact time t_4, at which the message enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference with the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

Peer-to-peer PTP transparent clock (P2P TC) message support

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow_up, and Delay Response messages. *Figure 713* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 713. Propagation delay calculation in clocks supporting peer-to-peer path correction



As shown in *Figure 713*, the propagation delay is calculated as follows:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t_1) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t_2) for this message.
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t_3) for this message.

   To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp message
   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp_Follow_Up message
   – Timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively
4. Port 1 generates a timestamp (t_4) on receiving the Pdelay_Resp message.
5. Port 1 uses all four timestamps to compute the mean link delay.

Timestamp correction

According to the IEEE 1588 specifications, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

As the MAC takes the timestamp at an internal point far from the actual boundary of the node and network, this captured timestamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (MII Tx, Rx clock) being different at the capture point as compared to the PTP clock (clk_ptp_ref_i) that is used to generate the time. The resultant CDC (clock domain crossing) circuits add an error the depends on the clock period of the MII and PTP clocks.

Ingress correction

In the receive side, the timestamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet SFD bit is received at the port boundary. Therefore, the captured timestamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp ingress correction nanosecond register (ETH_MACTSICNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between boundary point and the input of the core

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from the PHY datasheet or timing characteristics.
2. Internal latency from the input of the core to the internal capture point

   The internal ingress latency can be read from the *Timestamp ingress latency register (ETH_MACTSILR)*. This is a read-only register that gives the latency in scaledNanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs

based on the active PHY interface (such as MII or RMII) and the operating speed. Therefore, the software must read this register after any speed change in the MAC, to determine the current internal latency.

3. CDC synchronization

   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_ptp_ref_i).

The values determined from these three components should be added by the software and must be written into the TSIC field of the *Timestamp ingress correction nanosecond register (ETH_MACTSICNR)*.

Note: The value written to the register must be negative (two's complement), because it has to be subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is set, the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1 ns. So bit 31 of TSIC must be set to 1 (for negative value) and bits 30 to 0 must be programmed with "10^9 – total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is –5 ns, then the value is 0xBB9A C9FB.

When TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is reset, the nanoseconds field of the captured timestamp is in binary format with a granularity of ~0.466 ns. Therefore, bits[30:0] must be written with "2^{31} – total ingress_correction_value" represented in binary with bit[31] = 1.

Egress correction

In the Transmit side, the timestamp captured at the internal snapshot point is earlier (advanced in time) as compared to the time at which that packet SFD bit is output at the port boundary. Therefore, the captured timestamp must be compensated by the egress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp egress correction nanosecond register (ETH_MACTSECNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), the egress latency must be determined from the PHY datasheet or timing characteristics.

2. Internal latency from the internal capture point and the output of the core

   This internal egress latency can be read from the *Timestamp egress latency register (ETH_MACTSELR)*. This is a read-only register that gives the latency in scaledNanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface and the operating speed. Therefore, the software

must read this register after any speed change in the MAC, to determine the current internal latency.

3. CDC synchronization error

   The timestamp correction because of synchronization is compensated by adding

$$\text{EGRESS_SYNC_CORR} = (1 \times \text{PTP_CLK_PER} + 4 \times \text{TX_CLK_PER})$$

Table 515 lists the egress and ingress latency values for various PHY interfaces:

Table 515. Egress and ingress latency for PHY interfaces

PHY interface		Egress latency	Ingress latency
RMII	100 Mbit/s	40	120
RMII	10 Mbit/s	400	800

Frequency range of reference timing clock

The timestamp information are transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of MII and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

The PTP clock frequency limitations are the following:

- Maximum PTP clock frequency

  The maximum PTP clock frequency is limited by the maximum resolution of the reference time (10 ns at 100 MHz). In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

- Minimum PTP clock frequency

  The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time with the MII clock domain. This relationship is given by the following equation:

  3 × PTP clock period + 4 × MII clock period ≤ Minimum gap between two SFDs

  The MII clock frequency is fixed by IEEE specifications. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in *Table 516*.

Table 516. Minimum PTP clock frequency example

Mode	Minimum gap between two SFDs	Minimum PTP frequency with internal timestamp
10 Mbit/s full-duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
10 Mbit/s half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz

Table 516. Minimum PTP clock frequency example (continued)

Mode	Minimum gap between two SFDs	Minimum PTP frequency with internal timestamp
100 Mbit/s full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
100 Mbit/s half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz

PTP processing and control

Table 517 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008 specifications.

Table 517. Message format defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField(1)								1	32
logMessageInterval								1	33

1. The controlField is used in version 1. In version 2, the messageType field is used for detecting different message types.

Some fields of the Ethernet payload can be used to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP packets over IPv4
- PTP frames over IPv6
- PTP packets over Ethernet

PTP packets over IPv4

Table 518 provides information about the fields that are matched to control the snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 517*.

Table 518. Message format defined in IEEE 1588-2008

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x0800	IPv4 datagram
IP version and header length	14	0x45	IP version is IPv4
Layer 4 protocol	23	0x11	UDP
IP multicast address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP multicast address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP Primary multicast address: 224.0.1.129 PTP Pdelay multicast address: 224.0.0.107
UDP destination port	36, 37	0x013F, 0x0140	0x013F: PTP event messages(1) 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

PTP frames over IPv6

Table 519 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 517*.

Table 519. IPv6-UDP PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x86DD	IP datagram
IP version	14 (bits [7:4])	0x6	IP version is IPv6
Layer 4 protocol	20(1)	0x11	UDP
PTP multicast address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP destination port	56, 57a	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	94a	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP message type field (IEEE 1588 version 2)	62a (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The Extension header is not defined for PTP packets.

PTP packets over Ethernet

Table 520 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

Table 520. Ethernet PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC destination multicast address(1)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses(2): 01-1B-19-00-00-00 01-80-C2-00-00-0E(3)
MAC packet type	12, 13	0x88F7	PTP Ethernet packet
PTP control field (IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the TSENMACADDR bit of *Timestamp control register (ETH_MACTSCR)* is set.
2. IEEE 1588-2008, Annex F
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

Transmit path functions

The MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the MII interface. The packets, for which a timestamp has to be captured, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The packets for which a timestamp has to be captured must be specified. The packets can be defined by using the control bits in the transmit descriptor (see *Section 48.10.3: Transmit descriptor*). The MAC returns the timestamp to the software inside the corresponding

Transmit descriptor, thus automatically connecting the timestamp to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. The snapshot of the time to be sent to the application can be controlled by using the following options of the *Timestamp control register (ETH_MACTSCR)*:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type

  This feature controls the type of messages for which snapshots are taken.

Note: The peripheral also supports the PTP messages over VLAN packets.

Table 521 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPSEL field in *Timestamp control register (ETH_MACTSCR)*.

Table 521. Timestamp snapshot dependency on ETH_MACTSCR bits

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP messages
00	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	X	X	SYNC, Delay_Req
11	X	X	Pdelay_Req, Pdelay_Resp

The DMA returns the timestamp to the software application inside the corresponding receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Programming guidelines for IEEE 1588 timestamping (system time correction)

See *Section : System time correction* in *Section 48.9.9: Programming guidelines for IEEE 1588 timestamping on page 2239*.

IEEE 1588 system time source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

Description of IEEE 1588 system time source

The peripheral uses the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:

- UInteger48 seconds field

  The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bit wide. For example, 2.000000001 seconds are represented as seconds Field = 0x0000 0000 0002.
- UInteger32 nanosecondsField

  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 seconds are represented as nanoSeconds = 0x0000 0001.

  The nanoseconds field supports the following two modes:
  - **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A C9FF, that is, (10e9-1) nanoseconds.
  - **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF FFFF. Accuracy is ~0.466 ns per bit.

    These modes can be set through TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)*.

System time register module

The 64-bit PTP time is updated using the PTP input reference clock, clk_ptp_ref_i. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being transmitted or received at the MII.

The system time counter can be initialized or corrected using either the coarse or the fine correction method.

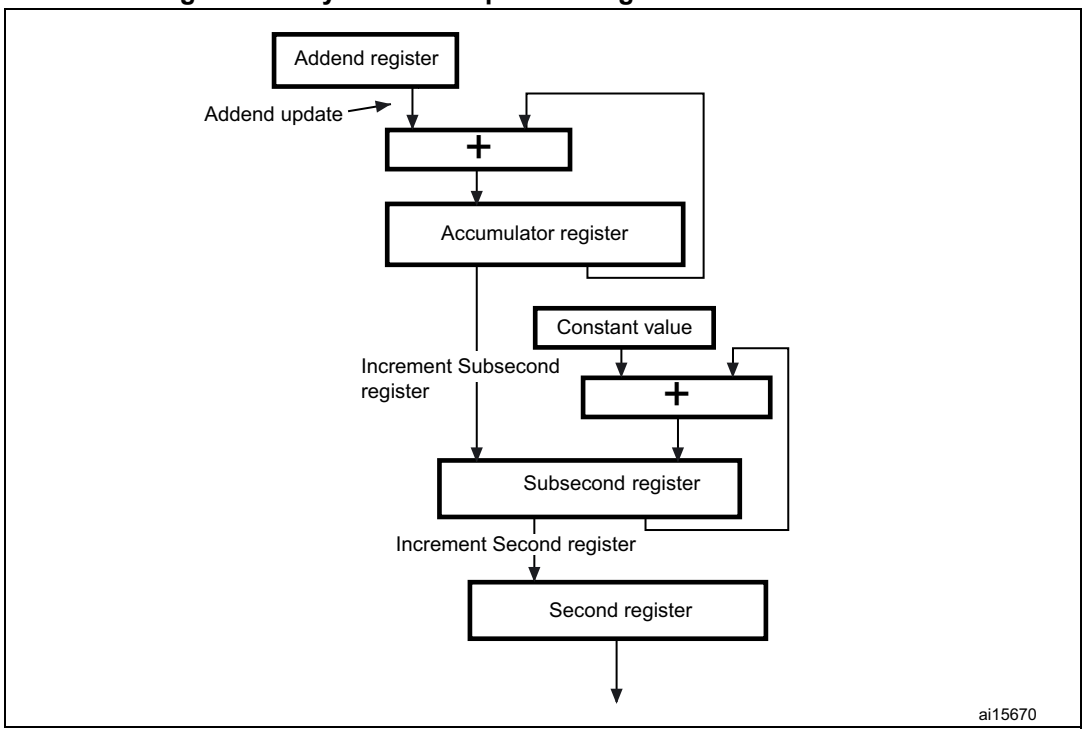
In the coarse correction method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is programmed with the value in the timestamp update registers, whereas for system time correction the offset value (timestamp update register) is added to or subtracted from the system time.

In the fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588-2002 specifications) is corrected over a period of time, unlike in the coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in *Figure 714*. The arithmetic carry that the accumulator generates is used as a pulse to

increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in *Figure 714*.

Figure 714. System time update using fine correction method



The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC 4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B 7672. When there is not clock drift, the default addend value of 0xC1F0 7C1F (2^{32} / 1.32) must be programmed.

In *Figure 714*, the constant value used to accumulate the subsecond register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

1. At time $MasterSyncTime_n$ the master sends the slave clock a SYNC message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as follows:

$$MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$$

2. The master clock counts for current Sync cycle, $MasterClockCount_n$ is

$$MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$$

(assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

3. The slave clock count for current Sync cycle, $SlaveClockCount_n$ is

$$SlaveClockCount_n = SlaveClockTime_n - SlaveClockTime_{n-1}$$

4. The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is

$$ClockDiffCount_n = MasterClockTime_n - SlaveClockTime_n$$

5. The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is

$$FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$$

6. The frequency compensation value for Addend register, $FreqCompensationValue_n$ is

$$FreqCompensationValue_n = FreqScaleFactor_n \times FreqCompensationValue_{n-1}$$

In theory, this algorithm achieves the lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value by the master, the algorithm corrects it at the cost of additional Sync cycles.

Refer to *Section 48.9.9: Programming guidelines for IEEE 1588 timestamping* for detailed programming steps.

IEEE 1588 auxiliary snapshots

The auxiliary snapshot feature enables to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the eth_ptp_trgx (where x = 0 to 3) sideband signal.

Up to four auxiliary snapshot inputs can be configured and up to four snapshots can be stored. A FIFO is accessible through registers: *Auxiliary timestamp seconds register (ETH_MACATSSR)* and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*.

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the *Timestamp status register (ETH_MACTSSR)* to know the timestamp of which input is available for reading at the top of this FIFO.

When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the *Timestamp status register (ETH_MACTSSR)*. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in *Auxiliary control register (ETH_MACACR)*. When multiple snapshots are present in the FIFO, the count is indicated in bits[27:25] of *Timestamp status register (ETH_MACTSSR)*.

Flexible pulse-per-second output

The MAC supports either a fixed pulse-per-second output mode (also called fixed mode) or a flexible pulse-per-second output mode for the ETH_PPS_OUT and eth_ptp_pps_out outputs:

- Fixed pulse-per-second output

  In this mode, only the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the *PPS control register (ETH_MACPPSCR)*.
- Flexible pulse-per-second output

  In this mode, the software has the flexibility to program the start or stop time, width, and interval of the pulse generated on the eth_ptp_pps_out output:

  The start and stop times are programmed through *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*.

  The PPS width and interval are programmed in terms of granularity of system time (number of the units of subsecond increment value) through *PPS width register (ETH_MACPPSWR)* and *PPS interval register (ETH_MACPPSIR)*, respectively.

Note: By default, the peripheral is in Fixed mode and indicates one second interval. When Fixed mode is selected by clearing PPSEN0 to 0 in the PPS control register (ETH_MACPPSCR):

- *the output on all PPS outputs is controlled by the value programmed in the PPSCTRL_PPSCMD field. Independent control of individual PPS output is not supported in Fixed mode.*
- *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR) are used only for generating target time reached interrupt; they are not used for PPS output generation.*
- *TRGTMODSEL0/1/2/3 must be programmed to 0.*
- *the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).*

Description of flexible pulse-per-second (PPS) output

The peripheral supports the following features with the flexible PPS outputs:

- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, the stop time can be programmed before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of subsecond increment value programmed in the *Subsecond increment register (ETH_MACSSIR)*. The pulse width can be programmed from 1 to 232–1 units of subsecond increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment value. You can program the interval between pulses from 1 to 232–1 units of subsecond increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

Note: The PTP reference clock mentioned in the following sections is the clock at which the system time is updated. When the TSCFUPDT bit of Timestamp control register (ETH_MACTSCR) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In Fine correction mode, this is the clock tick at which the system time is updated (using Subsecond increment register (ETH_MACSSIR) (as shown in Figure 714).

Refer to *Section 48.9.12: Programming guidelines for flexible pulse-per-second (PPS) output* for further details on how configuring flexible pulse output.

PPS start and stop times

The initial start time can be programmed in the Target Time registers.

If required, the start or stop time can be programmed again. However, this can be done only after the earlier programmed value is synchronized with the PTP clock domain. Bit 31 of *PPS target time nanoseconds register (ETH_MACPPSTTNR)* indicates that the synchronization is complete. This enables to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS width and interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to obtain a PPS pulse width of 40 ns and an interval of 100 ns with a PTP reference clock of 50 MHz, program the width and interval to values 2 and 5, respectively. Smaller granularity can be achieved by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, program or update the interval and width of the PPS signal output.

PTP timestamp offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network.

These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

The PTP offload feature is selected through *PTP offload control register (ETH_MACPOCR)*. 80-bit PTP node identity is configured through the following three registers: *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.

Description of PTP offload function

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. *Table 522* indicates the PTP message generation criteria.

Table 522. PTP message generation criteria

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
00	0	1	Ordinary or Boundary Slave	SYNC message reception	Delay_Req
00	1	1	Ordinary or Boundary Master	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
01	0	1	Transparent Slave	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req

Table 522. PTP message generation criteria (continued)

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
01	1	1	Transparent Master	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
11	X	X	Peer-to-Peer Transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
All other programming combinations are invalid for the PTP offload feature.					

Note: Clocks supporting peer delay mechanism must not generate delay request/delay response messages, according to IEEE 1588-2008 specifications. However, the peripheral controller supports this for flexibility, with a programmable control bit (DRRDIS) in the PTP offload control register (ETH_MACPOCR).

The DRRDIS bit can be used to control the response generation for delay request/delay response message. For example, in transparent slave mode, delay request is generated in response to received SYNC *only when the bit is reset.*

When the MAC is set as an Ordinary or Boundary Slave clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission of the corresponding Delay_Req message. Similarly, various other modes of operation are explained in *Table 522*.

The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the Destination Address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B 1900 0000 for all except peer delay mechanism messages and 0x0180 C200 000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with unicast destination address, it takes the SA field of the received packets and makes

them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all the received PTP messages are forwarded to the application along with Rx status, indicating whether the response was generated by the MAC, if it satisfies the packet filtering logic of the MAC receiver

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress timestamp of these two PTP messages are provided in the Tx TS status (Tx Timestamp Status register and interrupt generated).

In addition to messageType and versionPTP fields match for basic PTP over Ethernet message detection, the following additional fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one-step indication (0b0).
3. The transportSpecific field is checked for Default PTP over Ethernet (0b0000) or 802.1AS mode (0b1111) when enabled.

PTP packet generation

This section explains the format and content of the automatically generated PTP packets by the MAC when this mode is enabled. It provides the template of the common PTP message header, as well as the detailed description of the fields of the specific PTP packets generated.

Table 523. Common PTP message header fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

PTP message header fields

- **messageType**

  The following encoded values are used for PTP message types:
  - SYNC: 0000
  - Delay_Req: 0001
  - Pdelay_Req: 0010
  - Pdelay_Resp: 0011
  - Delay_Resp: 1001
- **transportSpecific**

  The following transport protocol encoding is used:
  - Default PTP over Ethernet: 0000
  - 802.1AS mode: 0001
- **versionPTP**

  It is always set to 2 because PTP version 2 is supported.
- **domainNumber**

  This field contains the value from the *PTP offload control register (ETH_MACPOCR)*.
- **flagField**

  The following values are used:
  - alternateMasterFlag (Octet 0 bit 0): 0 for SYNC and Delay_Resp
  - twoStepFlag (Octet 0 bit 1): 0 for SYNC and Pdelay_Resp
  - unicastFlag (Octet 0 bit 2): 0 for Multicast Address, 1 for Unicast Address
- **correctionField**

  For more information, see *Table 524*.
- **sourcePortIdentity**

  This field takes the value programmed in the *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.
- **sequenceId**

  Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.
- **controlField**

  The following encoded values are used for controlField:
  - SYNC: 0000 0000
  - Delay_Req: 0000 0001
  - Pdelay_Req: 0000 0010
  - Pdelay_Resp: 0000 0101
  - Delay_Resp: 0000 0011
- **logMessageInterval**
  - SYNC:

    This field contains logSyncInterval from the corresponding MAC_Log_Message_Interval register.

– Delay_Resp:

This field contains the sum of DRSYNCR and logSyncInterval value taken from the *Log message interval register (ETH_MACLMIR)* for a multicast PTP message and 0111 1111 for unicast PTP message.

– Delay_Req, Pdelay_Req and Pdelay_Resp: 0111 1111

where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of –15 to 15 for logSyncInterval fields, which translates to a range from 32.768 micro second (2–15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- $2^{(30+N)}$ ns, when N is negative (–1 to –15)
- 2^{N} seconds, when N is positive (0 to 15)

For example:

- When logSyncInterval is programmed to 1, the interval is 2^{1}; therefore, the SYNC message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is 2^{-1} = 0.536 seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because 2^{-30} = 1 ns.
- When logSyncInterval is programmed to –-5, the interval is 2^{-5} = 33.55 ms; therefore, the SYNC message is sent once every 33.55 ms.

Note: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation $2^{(30+N)}$, because of the non-binary nature of the nanoseconds field of the system time.

PTP message-specific fields

The message-specific fields are the following:

- **messageLength**

  There is no suffix supported, so this field contains the length of the PTP message that includes 34-byte PTP common header and the body specific to the message type.

  For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.

- **originTimestamp**

  This field is the captured egress timestamp for SYNC, Delay_Req, and Pdelay_Req PTP messages.

- **receiveTimestamp**

  For Delay_Resp PTP message, this is the ingress timestamp of the corresponding received Delay_Req PTP message.

- **requestingPortIdentity**

  For Delay_Resp and Pdelay_Resp PTP messages, this is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.

- **requestReceiptTimestamp**

  For the Pdelay_Resp PTP message, this field is set to 0.

One-step timestamp

The MAC supports the one-step timestamp feature that enables to identify the offset in the packet and inserts the timestamp received from the application at that offset.

MAC Transmit PTP mode for one-step timestamp

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 524 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the *Timestamp control register (ETH_MACTSCR)* and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 524. MAC transmit PTP mode and one-step timestamping operation

Programming			Mode	Per packet control(1)			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE (2)	OSTC (3)	TTS(4)	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
							Delay_Req (correction field for residence time and egress asymmetric correction)
00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
01	X	0	End-to-end Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Pdelay_Req (correction field for residence time and egress asymmetric correction)
						Ingress TS	Pdelay_Resp (correction field for residence time and ingress asymmetric correction)

Table 524. MAC transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[1]			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
01	0	1	Ordinary or Boundary Slave with support for peer delay mechanism or Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction) (applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
10	X	X	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Delay_Req (correction field for residence time and egress asymmetric correction)

Table 524. MAC transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control(1)			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE (2)	OSTC (3)	TTS(4)	
11	X	X	Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)

1. The per-packet control values provided here are the recommended settings used by devices in typical PTP operation and for the programmed mode.
2. TTSE represents TTSE bit of TDES2 transmit normal descriptor. The TTSE function is independent from the OST function and the programmed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of TDES3 transmit context descriptor
4. TTS represents the timestamp value provided in the TTSH, TTSL fields of TDES0 and TDES1 transmit normal descriptor (write-back format).

Note: Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp. Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

Enabling one-step timestamp

The one-step timestamp feature can be enabled for a given packet by setting bit 20 (OSTC) in TDES3 context descriptor. To update the correction field in certain PTP packets, the ingress timestamp must be given in the TSSL and TSSH fields.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

48.5.5 Checksum offload engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the receive path.

Transmit checksum offload engine

In the transmit path, the MAC calculates the checksum and inserts it in the Tx packet. This feature helps reducing the load on the software and can improve the overall system throughput.

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, the Tx FIFO automatically operates in the Store-and-forward mode even if the MAC is configured in Threshold (cut-through) mode.

Make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter, the reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the checksum insertion must be enabled only in the packets that are less than the number of bytes, given by the following equation:

$$\text{Packet size} < \text{TxQSize} - (\text{PBL} + 7) \times 4$$

where

TxQSize corresponds to the TQS bitfield of *Tx queue 0 operating mode register (ETH_MTLTXQ0OMR)*

PBL corresponds to the TxPBL bitfield of *Channel 0 transmit control register (ETH_DMAC0TXCR)*

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

Note: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status it in the Transmit status (bit 0 in *Table 546: TDES3 normal descriptor (write-back format)*).

This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.

- For IPv6 datagrams:
  - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in *Table 546: TDES3 normal descriptor (write-back format)*). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 525 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: Do not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Section : Transmit checksum offload engine because it might result in incorrect checksum insertion or unexpected behavior.

Table 525. Transmit checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Yes	Yes

Table 525. Transmit checksum offload engine functions for different packet types (continued)

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Authentication – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – No – No – Yes – No – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive checksum offload engine

The receive checksum offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

The receive checksum offload engine (Rx COE) can be enabled by setting the IPC bit of *Operating mode configuration register (ETH_MACCR)*. When this bit is set, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the *VLAN tag register (ETH_MACVTR)* is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when DIS_TCP_EF bit of the *Rx queue 0 operating mode register (ETH_MTLRXQ0OMR)* is reset and FEP bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 526: Receive checksum offload engine functions for different packet types describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 526. Receive checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – Yes – No – Yes – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

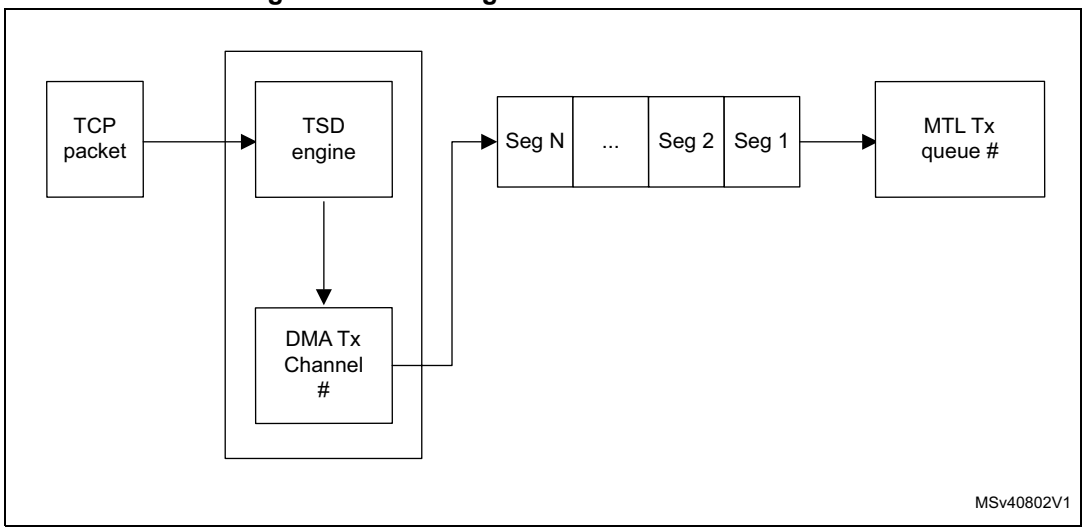
48.5.6 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in *Figure 715*.

This feature is enabled by programming the TSE bit of corresponding ETH_DMAC0TXCR register (see *Channel 0 transmit control register (ETH_DMAC0TXCR)*). It is only supported when the MAC operates in Full-duplex mode.

For detailed programming steps, refer to *Section 48.9.14: Programming guidelines for TSO*.

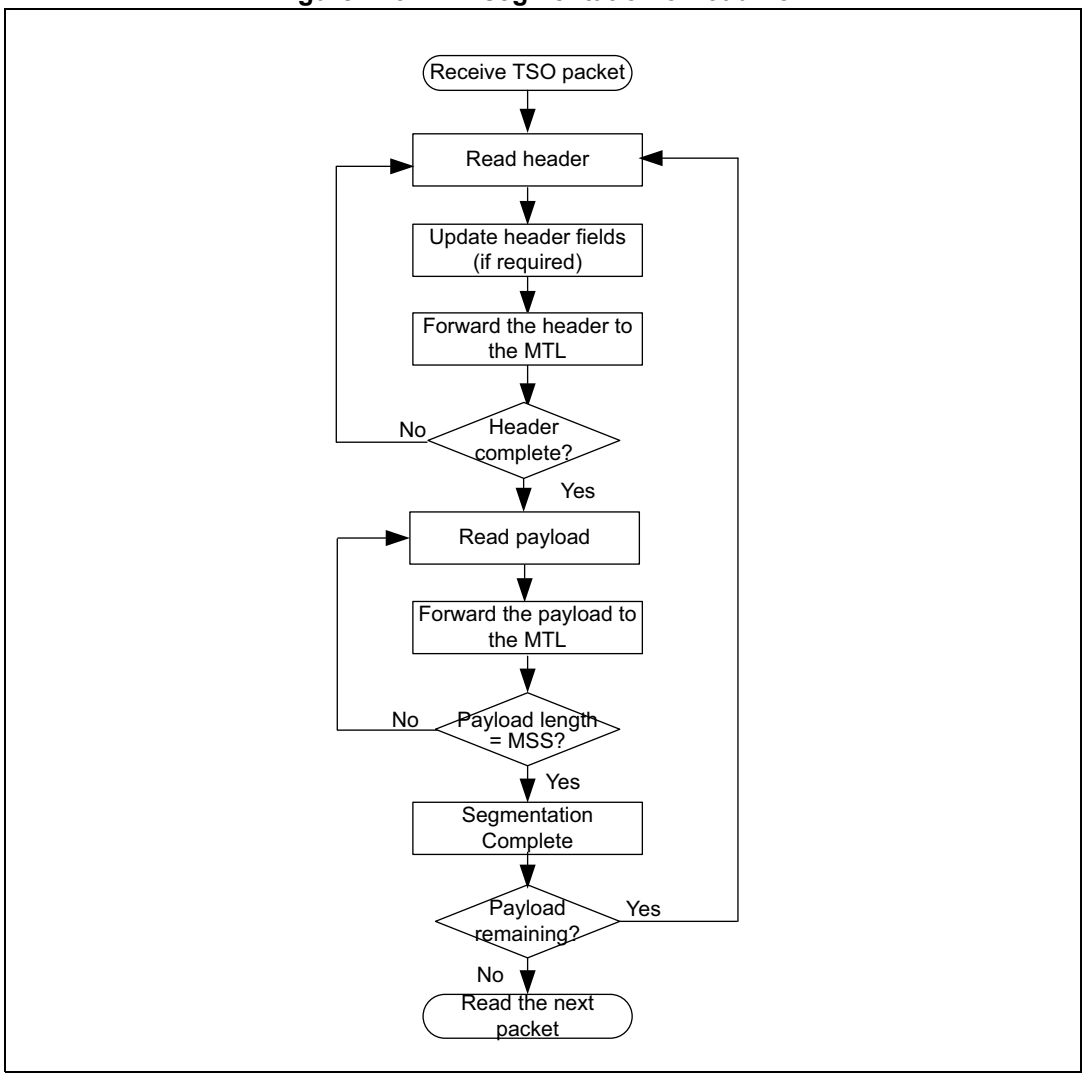
Figure 715. TCP segmentation offload overview



DMA operation with TSO feature

Figure 716 shows the TSO flow.

Figure 716. TCP segmentation offload flow



For the TSO feature, the Tx DMA operation is as follows:

1. The application sets up the Transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA Tx channel.
3. While in the Run state, the DMA fetches the next available descriptor and performs one of the following actions:
   – If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.
   – If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a Context Descriptor Error (TDES3[23]) and fetches the next descriptor.

– If the descriptor is a normal descriptor, the DMA checks the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.
5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.
6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (if present and when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, it fetches again the header buffer in system memory, as done for the first segment. In such cases, the DMA does not close the first descriptor containing the header buffer until the header for last segment is fetched.
7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL Tx queue.
8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory, and directly pushes it into the MTL Tx queue. If the buffer(s) in the descriptor do(es) not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.
9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the Tx queue.
10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory), and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no more descriptor are available, the DMA enters the suspend state.

Note: The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first Normal Tx descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.

TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. *Table 527* describes how the TCP and IP headers are updated.

Table 527. TSO: TCP and IP header fields

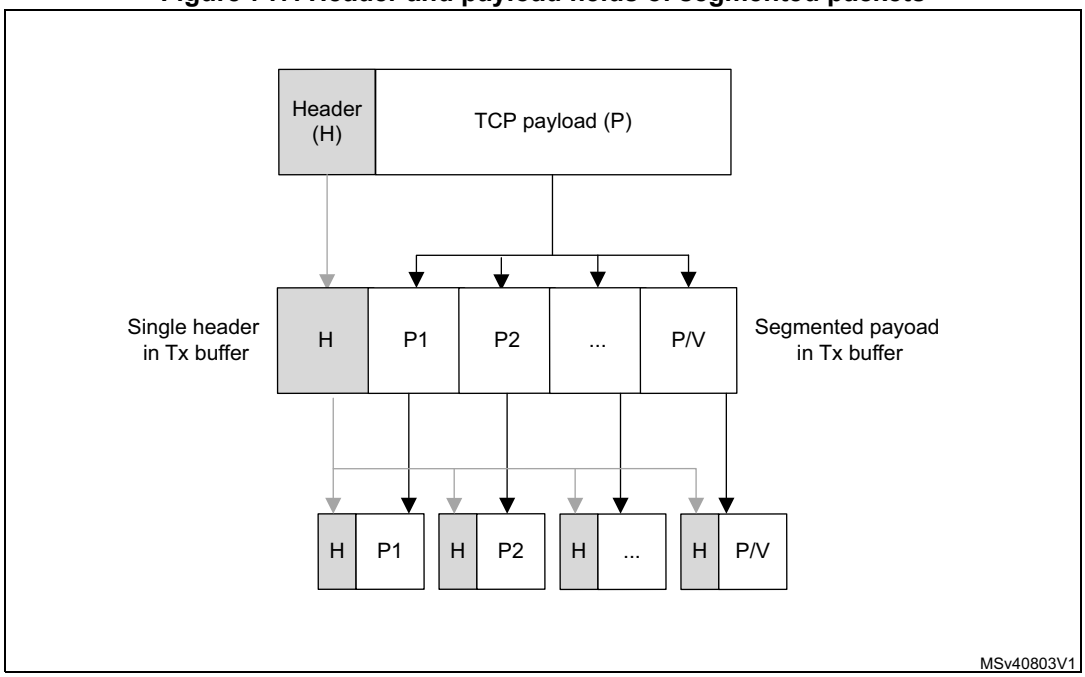
Packet sequence	TCP header	IP header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field is not modified. It is sent as per the header provided by the software. – IPv4 Header Checksum is recalculated. IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field = Previous Identification Field + 1 – IPv4 Header Checksum is recalculated IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = Remaining Payload + TCP Header Length + IP Header Length – Identification Field = Previous Identification Field + 1 – IPv4 header Checksum is recalculated IPv6 Header – Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

Header and payload fields of segmented packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in *Table 527: TSO: TCP and IP header fields*. *Figure 717: Header and payload fields of segmented packets* shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

Figure 717. Header and payload fields of segmented packets



Context descriptor sequence

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to *Section 48.10.3: Transmit descriptor*).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see *Channel 0 interrupt enable register (ETH_DMAC0IER)*).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see *Channel 0 status register (ETH_DMAC0SR)*).

Building the Descriptor and the packet for the TSO feature

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see *Section 48.10.3: Transmit descriptor*). If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMAC0CR (see *Channel 0 control register (ETH_DMAC0CR)*) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMAC0TXCR register (see *Channel 0 transmit control register (ETH_DMAC0TXCR)*) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQOMR register (see *Tx queue 0 operating mode register (ETH_MTLTXQ0OMR)*). A MSS plus header equal to half the programmed Tx queue size is recommended.

The DMA also supports segmentation of VLAN-tagged TCP/IP frames. If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning this is an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 00 (MAC does CRC and padding) and checksum insertion control bits are forced to 11 (hardware does the checksum calculation for both header and payload).

48.5.7 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets**.** ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:

1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to the application with ARP Reply Not Generated status bit set (bit 34). However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request. If the Disable CRC check (DCRCC) bit of the *Extended operating mode configuration register (ETH_MACECR)* is set, then the MAC does not check for valid CRC of a ARP

request packet. It can generate an ARP response packet if the other conditions are valid. The ARP request packet must always have a valid CRC.

Note: When the received ARP request is less than the 64-byte packet length, the MAC does not send an ARP response. It is treated as a normal packet and forwarded to the application based on the MAC filter settings.

48.5.8 Loopback

The MAC supports loopback of transmitted packets to its receiver.

Guidelines for using loopback mode

Below some guidelines for using the loopback mode:

- Enable loopback only with the Full-duplex mode. In Half-duplex mode, the carrier sense signal or collision signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip, externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets since they may get corrupted in the loopback FIFO.

The transmit and receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the transmitted data to the receive path. The FIFO is free-running to write on the write clock (eth_tx_clk) and read on every read clock (eth_rx_clk). At the start of each packet read from the FIFO, the write and read pointers are reinitialized to have an offset of four (in 10/100 Mbit/s mode). This avoids overflow or underflow during a packet transfer, and ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between MII transmit and receive clock frequencies.

Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the receive Protocol Engine module generates received packet status and sends it to the receive Packet Controller module. The control, missed packet, and filter fail status are added to the receive status in the receive Packet Controller module. The MAC does not process ARP or PMT packets that are looped back.

Enabling loopback mode

To enable this feature, program the LM bit of the *Operating mode configuration register (ETH_MACCR)*. Loopback can be enabled for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal receive MII interface, irrespective of which PHY interface is selected.

The loopback data is also passed through the corresponding transmit PHY interface block, onto the Ethernet line.

Note: During loopback, the data/packet is reflected on the mii_txd signal.

48.5.9 Flow control

The transmit flow control involves transmitting Pause packets in Full-duplex mode and backpressure in Half-duplex mode to control the flow of packets from the remote end. This section describes the flow control for transmit and receive paths.

Flow control in Full-duplex mode

In Full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. *Table 528* describes the fields of a Pause packet.

Table 528. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets
PT	Contains Pause time specified in the PT field of the *Tx Queue flow control register (ETH_MACQTXFCR)*

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Flow control in Half-duplex mode

In Half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 529 describes the flow control in the Tx path based on the setting of the following bits:

- TFE bit of *Tx Queue flow control register (ETH_MACQTXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Flow control is similar for all queues.

Table 529. Tx MAC flow control

TFE	DM	Description
0	X	The MAC transmitter does not perform the flow control or backpressure operation.
1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set.
1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set.

Transmit flow control

The transmit flow control is enabled when TFE bit is set in *Tx Queue flow control register (ETH_MACQTXFCR)*.

Flow control trigger

The application can request the MAC to send a Pause packet or initiate back-pressure by setting the FCB bit in the corresponding *Tx Queue flow control register (ETH_MACQTX-FCR)*.

Receive flow control

In the receive path, the flow control is functional only in full-duplex mode. If any Pause packet is received in half-duplex mode, the packet is considered as a normal control packet.

Description of receive flow control

Table 530 describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of *Rx flow control register (ETH_MACRXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Table 530. Rx MAC flow control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

The following sequence describes the Rx flow control:

1. The MAC checks the destination address (DA) of the received Pause packet for either of the following:
   - Multicast destination address: the DA matches the unique multicast address specified for the control packet (0x0180 C200 0001).
   - Unicast destination address: the DA matches the content of the MAC Address 0 registers (*MAC address 0 high register (ETH_MACA0HR)* and *MAC address x low register (ETH_MACAxLR)*) and the UP bit of *Rx flow control register (ETH_MACRXFCR)* is set.

If the UP bit is set, the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.

2. The MAC decodes the following fields of the received packet:
   - Type field: this field is checked for 0x8808.
   - Opcode field: this field is checked for 0x0001 (Pause packet).
   - Pause time: the Pause time (for Pause packet) is captured to determine the time for which the transmitter needs to be blocked.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value.

Enabling receive flow control

Set the RFE bit in the *Rx flow control register (ETH_MACRXFCR)* to enable the Pause flow control.

48.5.10 MAC management counters

The peripheral supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC management counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in *Section 48.11.4: Ethernet MAC and MMC registers*.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

The receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set bit 0 in the *Packet filtering control register (ETH_MACPFR)*. The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

In addition to control registers, two sets of registers are implemented:

- Six registers used for collision, error, and good packet counters:
  - *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)*
  - *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)*
  - *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)*
  - *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)*
  - *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)*
  - *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)*
- Four registers to record LPI mode transition:
  - *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)*
  - *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)*
  - *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)*
  - *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)*

Definitions

The following terminology is used in MMC register descriptions:

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber timeout
  - No carrier or loss of carrier
  - Late collision
  - Packet underflow
  - Excessive deferral
  - Excessive collision
- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbit/s only)
  - Length error (non-Type packet only)
  - Out of range (non-Type packet only, longer than 1518 bytes)
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN frame maxsize = 1,522
  - Jumbo frame maxsize = 9,018
  - JumboVLAN frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the *Table 531*.

Table 531. Size of the maximum receive packet

JE	S2KP	GPSLCE	EDVLP	Untagged frame maximum size in bytes	Single VLAN frame maximum size in bytes	Double VLAN Frame maximum size in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

48.5.11 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the *Interrupt enable register (ETH_MACIER)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

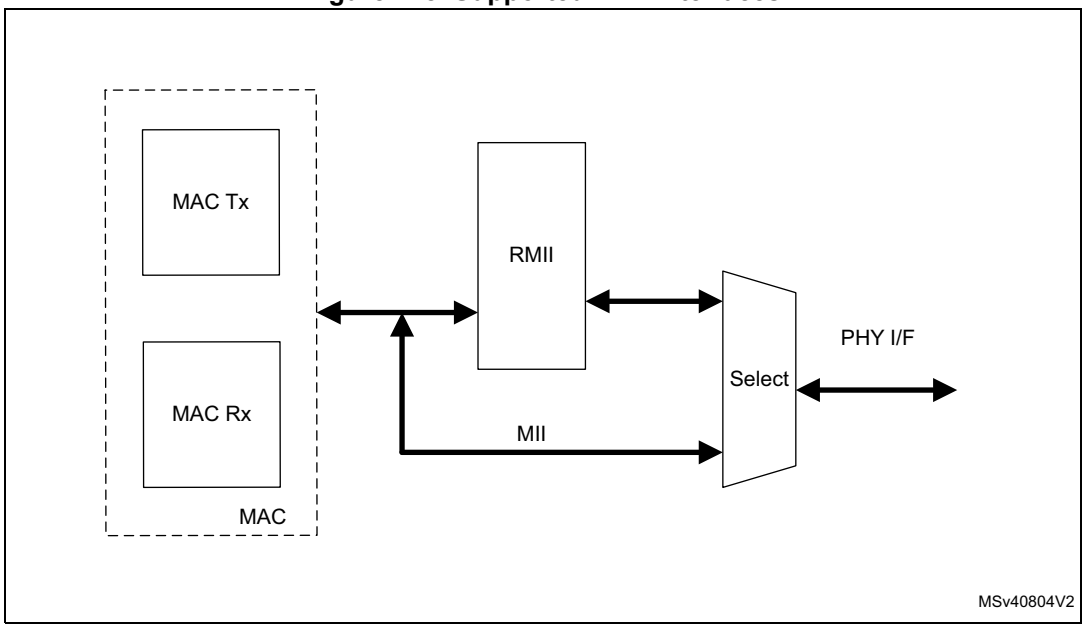
48.5.12 MAC and MMC register descriptions

Refer to *Section 48.11.4: Ethernet MAC and MMC registers*.

48.6 Ethernet functional description: PHY interfaces

The Ethernet peripheral support several PHY interfaces. The root interface is the MII one. All other interfaces are derived from it as shown in *Figure 718*.

Figure 718. Supported PHY interfaces



This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

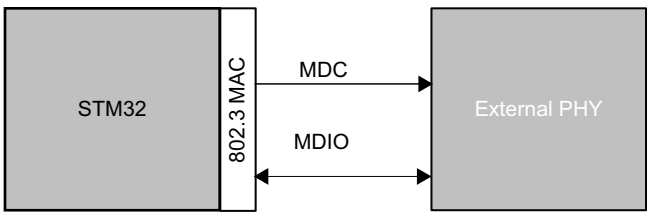
- *Station management agent (SMA)*
- *Media independent interface (MII)*
- *Reduced media independent interface (RMII)*

48.6.1 Station management agent (SMA)

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in *Figure 719*.

Figure 719. SMA Interface block



SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock (eth_hclk). The maximum operating frequency of the ETH_MDC pin is 2.5 MHz, as specified in IEEE 802.3 specifications. However, a different divider can be selected if the system supports higher clock frequencies. The division factor depends on the clock range setting through CR[3:0] in the *MDIO address register (ETH_MACMDIOAR)* register. The MDC clock is selected as follows:

Table 532. MCD clock selection

Selection	MDC clock
0000	eth_hclk/42
0001	eth_hclk/62
0010	eth_hclk/16
0011	eth_hclk/26
0100	eth_hclk/102
0101	eth_hclk/124
0110	eth_hclk/204
0111	eth_hclk/324
1000	eth_hclk/4
1001	eth_hclk/6
1010	eth_hclk/8
1011	eth_hclk/10
1100	eth_hclk/12
1101	eth_hclk/14
1110	eth_hclk/16
1111	eth_hclk/18

The data exchange between the MAC and the PHY is performed through a three-state buffer and brought out as ETH_MDIO line connected to the PHY.

Figure 720 shows the structure of a Clause 45 MDIO packet, while *Table 533* provides a detailed description of the packet fields.

Figure 720. MDIO packet structure (Clause 45)

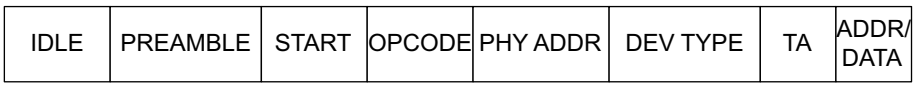
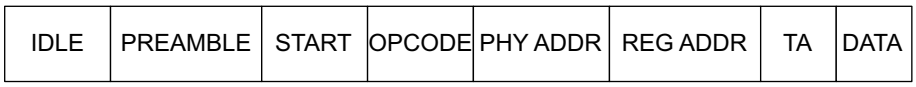


Table 533. MDIO Clause 45 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b00
OPCODE	– 0b00: Address – 0b01: Write – 0b10: Read+ Address – 0b11: Read
PHY ADDR	5-bit address select for one of 32 PHYs
DEV TYPE	5-bit device type
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	16-bit value: for an address cycle (OPCODE = 0b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data to be written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. – In address and data write cycles, the peripheral drives the ETH_MDIO line during the transfer of these 16 bits. – In read and post-read increment address cycles, the PHY drives the ETH_MDIO line during the transfer of these 16 bits.

The frame structure for Clause 22 frames is also supported. The C45E bit in *MDIO address register (ETH_MACMDIOAR)* can be programmed to enable Clause 22 or Clause 45 mode of operation. *Figure 721* shows the structure of a Clause 22 MDIO packet, while *Table 534* provides a detailed description of the packet fields.

Figure 721. MDIO packet structure (Clause 22)



MSv40806V1

Table 534. MDIO Clause 22 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b01
OPCODE	0b10 for Read and 0b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	Any 16-bit value. In a Write operation, the MAC drives ETH_MDIO. In a Read operation, the PHY drives it.

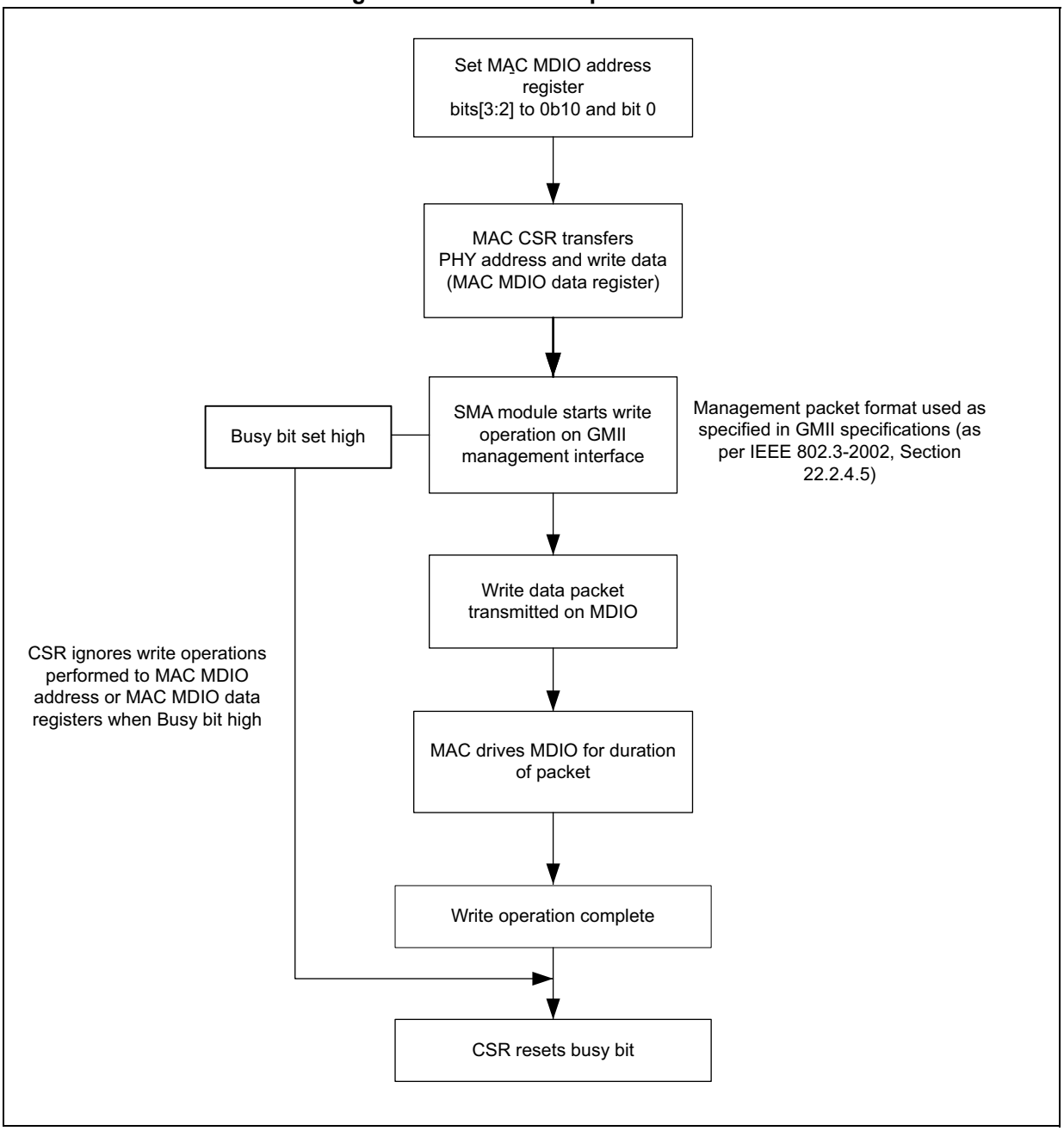
In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

MII management write operations

After the station management agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a write operation to the PHY registers.

Figure 722 illustrates the flow for a write operation from the SMA module to the PHY registers.

Figure 722. SMA write operation flow



When bits[3:2] are set to 01 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (*MDIO data register (ETH_MACMDIODR)*) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the MII management Interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations performed to the *MDIO address register (ETH_MACMDIOAR)* or the *MDIO data register*

(ETH_MACMDIODR) during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

Figure 723. Write data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

MSv40807V1

MII management read operation

When bits[3:2] are set to 11 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the MII management interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the *MDIO address register (ETH_MACMDIOAR)* or *MDIO data register (ETH_MACMDIODR)* register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the *MDIO data register (ETH_MACMDIODR)* with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

Figure 724. Read data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	10	AAAAA	RRRRR	Z0	DDD...DDD	Z

MSv40808V1

Preamble suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The peripheral provides controls to support preamble suppression. It transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the PSE bit in *MDIO address register (ETH_MACMDIOAR)*.

Trailing clocks and back-to-back transactions

The peripheral drives the ETH_MDC clock for the duration of the MDIO frame. There is no clock driven during the idle period. The trailing clock feature can be used if the PHY needs the ETH_MDC clock to be active for some cycles after the MDIO frame. The NTC[2:0] bitfield in *MDIO address register (ETH_MACMDIOAR)* allows the programming of trailing clocks from 0 to 7.

The peripheral supports back-to-back transactions which allow starting the next MDIO frame even before the trailing clocks are complete for previous MDIO frame. This feature can be enabled by setting BTB bit in *MDIO address register (ETH_MACMDIOAR)* when the trailing clock feature is also enabled. When back-to-back transactions are enabled, the GMII busy bit (GB) is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by the peripheral while trailing clocks are still on for the previous MDIO frame. When (GB) transactions are not enabled, the GMII busy bit is cleared after the trailing clocks are complete for the MDIO frame.

Interrupt for MDIO transaction completion

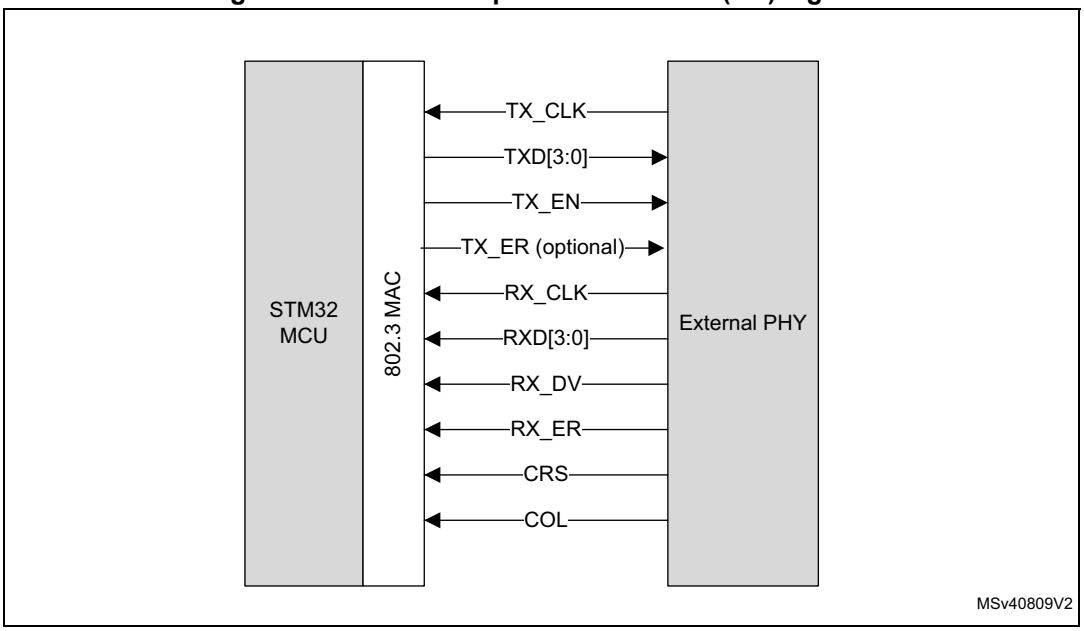
The peripheral can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy bit of the *MDIO address register (ETH_MACMDIOAR)* to know the completion of MDIO commands.

48.6.2 Media independent interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 725: Media independent interface (MII) signals*.

Figure 725. Media independent interface (MII) signals



- TX_CLK: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- TXD[3:0]: transmit data.

  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TX_ER (optional): required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- RXD[3:0]: receive data

  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.

- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.

- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.

- CRS: carrier sense.

  This signal is asserted by the PHY when either the transmit or receive medium is non idle. It is deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don't care for the MAC sublayer.

- COL: collision detection signal

  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode, the state of this signal is don't care for the MAC sublayer.

48.6.3 Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU. According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

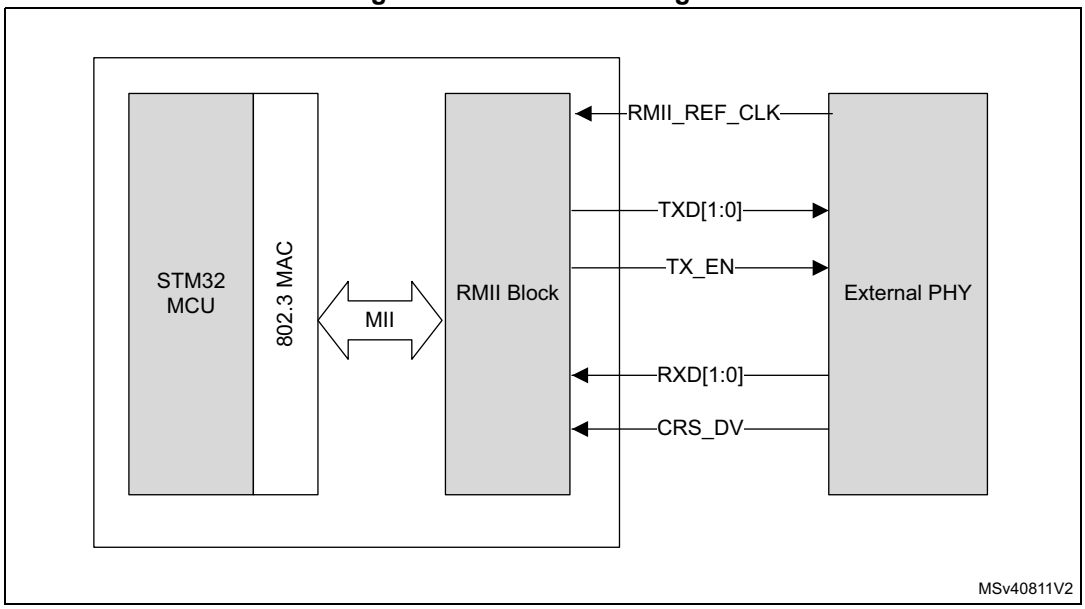
Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbit/s and 100 Mbit/s operating rates. It does not support the 1000 Mbit/s operation.
- Provides independent 2-bits wide transmit and receive paths by sourcing two clock references externally.

RMII block diagram

Figure 726: RMII block diagram shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

Figure 726. RMII block diagram

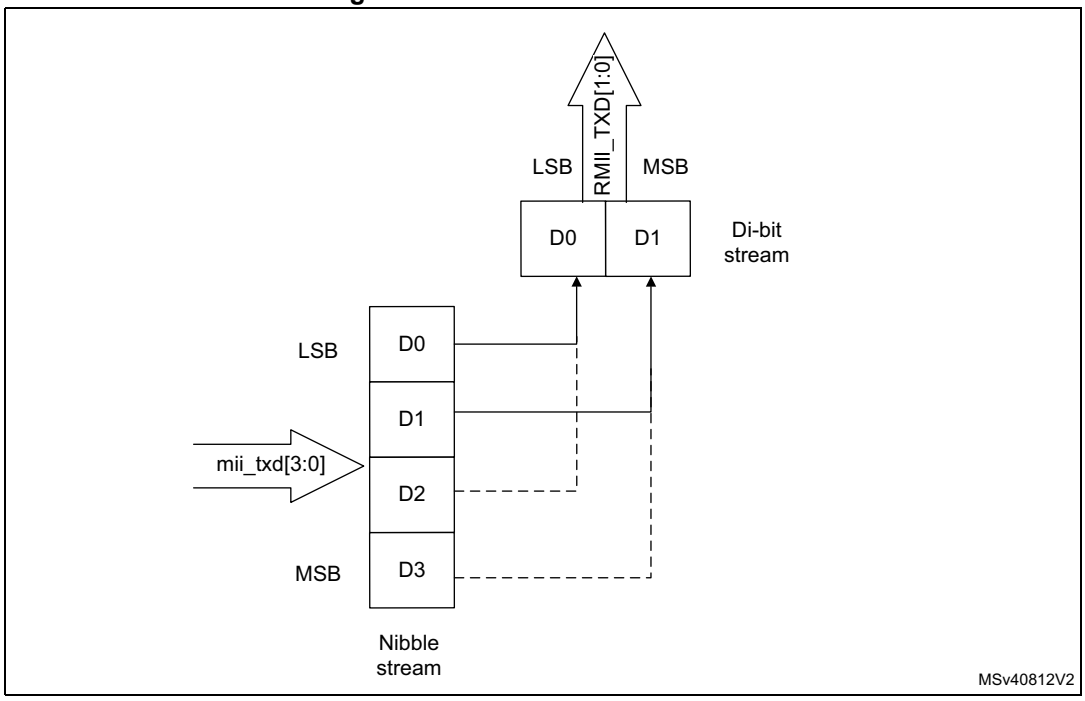


- RMII_REF_CLK: continuous 50 MHz reference clock input
- TXD[1:0]: transmit data
- TX_EN: transmit data enable.
  When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- RXD[1:0]: receive data
- CRS_DV: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.

Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in *Figure 727: Transmission bit order*. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

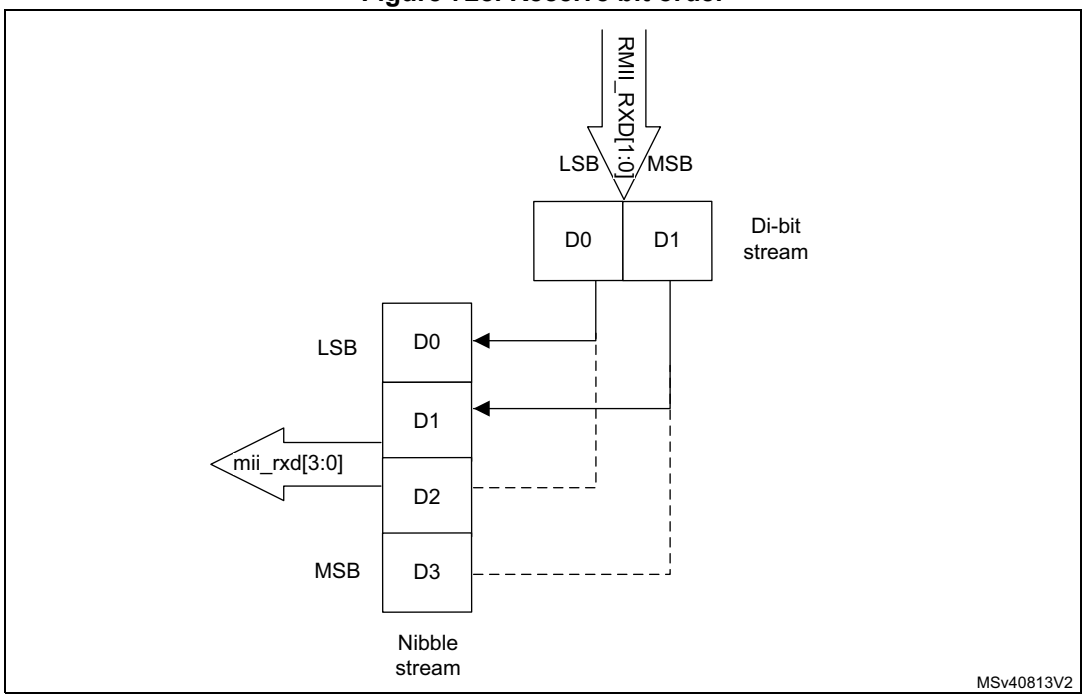
Figure 727. Transmission bit order



Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in *Figure 728: Receive bit order*. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 728. Receive bit order



48.7 Ethernet low-power modes

48.7.1 Low-power management

The Ethernet peripheral supports the following techniques to save power:

- Magic packet
- Remote wake-up

The magic packet and remove wake-up techniques are used to save power in the host system when it is in low-power mode, and has to be woken up only at the reception of specific packets from the Ethernet network. In low-power mode, the clocks on most of the host logic, along with the majority of the peripherals (except the MAC receiver logic), can be disabled. On receiving the specific packets from the network, the MAC generates the trigger to wake up the host system and come back to normal state. Refer to the power control (PWR) section of the reference manual for the list of the system operating modes that support Ethernet low-power modes.

The Energy Efficient Ethernet (EEE) mode is compliant with the IEEE 802.3az-2010 standard. It is primarily targeted to save power in the Ethernet port when there is no traffic on the line. In this mode, the host indicates to the far-end that it does not have any packets to transmit in the near future and puts the transmitter port (MAC controller, PCS and PHY layers) in low-power mode. Similarly, the receiver port can also be put in low-power mode

when the far-end host indicates that it does not have any traffic to transfer. This allows significant saving of power in the Ethernet port (mainly in the PHY) with intermittent and burst traffic profile. The triggering of entry and exit out of the EEE mode is controlled by the MAC and is supported within the peripheral.

Simultaneous operation of the EEE mode along with any or both the other power saving modes is also supported.

Description of magic packet mode

This section describes how to save power through magic packet detection.

Note: The magic packet feature is based on the magic packet technology white paper from Advanced Micro Device (AMD).

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog and jabber timeout register (ETH_MACWJBTR).

In the magic-packet-based power saving mode, the reception of a valid magic packet by the MAC receiver triggers an exit from low-power mode. The MAC enters power saving mode when PWRDWN bit of *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the magic-packet-based power saving mode is enabled by setting the MGKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The magic packet contains a unique pattern at any offset after the Destination address, Source address, and Length/Type fields. In addition to the unique pattern matching, the MAC receiver also checks for the following, to detect the received packet as a valid magic packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

Magic packet data format

The content of the unique pattern in magic packets is described as follows:

- Six bytes of all-ones (0xFF FF FF FF FF FF) called synchronization stream. There can be more than six bytes of 0xFF, but only the last six are considered.
- The synchronization stream is immediately followed by 16 repetitions of the Destination address field of the packet (*MAC address 0 high register (ETH_MACA0HR)* and the *MAC Address 0 low register (ETH_MACA0LR)*) or multicast/broadcast address.
- No break or interruption between synchronization stream and first repetition of Destination address field or within its 16 repetitions.

If the MAC address of a node is 0x00 11 22 33 44 55, the MAC scans for the following data sequence:

Destination Address Source Address Length/Type…… FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
…CRC

Description of remote wake-up packet mode

This section describes the power saving mode based on remote wake-up packet.

Note: The remote wake-up packet feature implementation is based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog and jabber timeout register (ETH_MACWJBTR).

In the remote-wake-up-magic-packet-based power saving mode, the reception of expected remote wake-up packet by the MAC receiver triggers the exit from low-power mode. The MAC enters power saving mode when PWRDWN bit in *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the remote-wake-up-magic-packet-based power saving mode is enabled by programming RWKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The MAC implements a filter lookup table (programmed through *Remote wake-up packet filter register (ETH_MACRWKPFR)* in which CRC, offset, and byte mask of the pattern embedded in remote wake-up packet and the filter operation commands are programmed.

The pattern embedded in the remote wake-up packet is located at any offset after the Destination address and Source address fields. In addition to the CRC match for the pattern, the MAC receiver also checks the following, to detect the received packet as a valid remote wake-up packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

When a valid remote wake-up packet is received, the MAC receiver sets the RWKPRCVD bit in *PMT control status register (ETH_MACPCSR)* and triggers the interrupt on pmt_intr_o output port. The PMTIS bit in *Interrupt status register (ETH_MACISR)* is set when power-gating is not enabled in low-power mode. An interrupt is triggered to the application on the sbd_intr_o output port when interrupt is enabled (PMTIE bit in *Interrupt enable register (ETH_MACIER)* is set) and CSR clock is not gated off in low-power mode.

Remote wake-up packet filters

When the remote-wake-up-based power saving mode is enabled, four remote wake-up filters can be selected. The structure of the remote wake-up filters is shown in *Table 535: Remote wake-up packet filter register*.

Table 535. Remote wake-up packet filter register

ETH_MACRWKPFR value	Field							
0	Filter 0 byte mask							
1	Filter 1 byte mask							
2	Filter 2 byte mask							
3	Filter 3 byte mask							
4	Reserved	Filter 3 command	Reserved	Filter 2 command	Reserved	Filter 1 command	Reserved	Filter 0 command
5	Filter 3 offset		Filter 2 offset		Filter 1 offset		Filter 0 offset	
6	Filter 1 CRC - 16				Filter 0 CRC - 16			
7	Filter 3 CRC - 16				Filter 2 CRC - 16			

The remote wake-up filter fields are described in *Table 536: Description of the remote wake-up filter fields*.

Table 536. Description of the remote wake-up filter fields

Register	Description
Filter *i* Byte mask	The filter *i* byte mask register defines the bytes of the packet that are examined by filter *i* (0, 1, 2 or 3) to determine whether or not a packet is a wake-up packet. – The MSB (31st bit) must be zero. – Bit j[30:0] is the byte mask. – If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter *i* Offset + j of the incoming packet; otherwise Filter *i* Offset + j is ignored.
Filter *i* Command	The 4-bit filter *i* command controls the filter *i* operation. – Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. – Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value. Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2". – Bit 1 (And_Previous) implements the Boolean logic(1). When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. – Bit 0 is the enable for filter *i*. If Bit 0 is not set, filter *i* is disabled.

Table 536. Description of the remote wake-up filter fields (continued)

Register	Description
Filter *i* Offset	This filter *i* offset register defines the offset (within the packet) from which the filter *i* examines the packets. – This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. – The minimum allowed offset is 12, which refers to the 13th byte of the packet. – The offset value 0 refers to the first byte of the packet.
Filter *i* CRC-16	This filter *i* CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block. – The 16-bit CRC calculation uses the following polynomial: $G(x) = x^{16} + x^{15} + x^2 + 1$ Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following: – 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation. – 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

1. The And_Previous bit setting is applicable within a set of four filters.
Setting And_Previous bit of a filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of four filters has no effect. For example, setting And_Previous bit of Filter 0 has no effect.
If And_Previous bit is set for a given filter to form an AND chained filter, the AND chain breaks when it finds a disabled filter. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting Filter 2 And_Previous bit has no effect, only Filter 1 result ORed with Filter 3 result is considered.
If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, then a remote wake-up frame does not pass the AND chained filter as a remote wake-up frame cannot be of both unicast and multicast address types.

The remote wake-up filter registers are implemented as eight indirect access registers (wkuppktfilter_reg#i) for four remote wake-up filters, and accessed by the application through *Remote wake-up packet filter register (ETH_MACRWKPFR)*. The entire set of wkuppktfilter_reg registers must be written to program the remote wake-up filters. The wkuppktfilter_reg register is programmed by sequentially writing the eight register values in *Remote wake-up packet filter register (ETH_MACRWKPFR)* for wkuppktfilter_reg0 to wkuppktfilter_reg3, respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in RWKPTR field of *PMT control status register (ETH_MACPCSR)*.

Note: If the Remote wake-up packet filter register (ETH_MACRWKPFR) is accessed in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when the CPU accesses Lane 3.

When Remote wake-up packet filter register (ETH_MACRWKPFR) is written, the content is transferred from CSR clock domain to PHY receive clock domain after the write operation. There should not be any further write to the Remote wake-up packet filter register (ETH_MACRWKPFR) until the first write is updated in PHY receive clock domain. Otherwise, the second write operation does not get updated to the PHY receive clock domain. Therefore, the delay between two write operations to the Remote wake-up packet filter register (ETH_MACRWKPFR) should be at least 4 cycles of the PHY receive clock.

PMT interrupt

The PMT interrupt signal is asserted when a valid remote wake-up packet is received.

Table 537 lists the remote wake-up scenarios in which PMT interrupt is generated.

Table 537. Remote wake-up packet and PMT interrupt generation(1)

Filter i Command			Frame Type and CRC Status		Interrupt Generation
CAST	INV	EN	Received Frame Cast Type	CRC Status	RWK INTR
0	0	1	Unicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
0	1	1	Unicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	0	1	Multicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	1	1	Multicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated

1. In all other combinations, the Remote Wake-up packet is not detected and PMT interrupt is not generated.

In addition to sbd_intr_o signal, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the power-down mode. It is ORed with lpi_intr_o signal (see *Section : LPI interrupt*) and tied to the EXTI peripheral.

As the pmt_intr_o signal is generated in the PHY Rx clock domain, it is not cleared immediately when the *PMT control status register (ETH_MACPCSR)* is read. This is because the resultant clear signal has to cross to the PHY Rx clock domain, and then clear the interrupt source. This delay is at least 4 clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbit/s mode. When the application clears the PWRDWN bit in *Remote wake-up packet filter register (ETH_MACRWKPFR)*, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

Power-down sequence

The software must perform the following tasks to initiate the power-down sequence:

- Disable the Transmit DMA (if applicable) by clearing the ST bit of the *Channel 0 transmit control register (ETH_DMAC0TXCR)*.
- Wait for any previous frame transmissions to complete. You can check this by reading TFCSTS[1:0] and TPESTS bits in the *Debug register (ETH_MACDR)* and TXQSTS bit in the *Tx queue 0 debug register (ETH_MTLTXQ0DR)* of all MTL Tx queues.
- Disable the MAC transmitter and MAC receiver by clearing TE and RE bits in *Operating mode configuration register (ETH_MACCR)*.
- Wait until the receive DMA empties all frames from the Rx FIFO. To check this, read PRXQ[13:0] in the *Rx queue 0 debug register (ETH_MTLRXQ0DR)*. If this bit is zero, the Rx FIFO is empty.
- Configure the magic packet (MGKPKTEN) and/or remote wake-up (RWKPKTEN) detection in the *PMT control status register (ETH_MACPCSR)*.
- Set bit 31 (ARPEN) in the *Operating mode configuration register (ETH_MACCR)*.
- Enable the MAC receiver by setting RE bit and then set PWRDWN bit in the *PMT control status register (ETH_MACPCSR)* to initiate the power-down sequence in MAC.

Note: If the feature is enabled and the MAC transmitter is in the LPI mode when it is put into the power-down mode, then the MII interface gets clamped to assert the LPI pattern. If the MAC transmitter is not in the LPI mode when it is put into the power-down mode, the GMII or MII interface gets clamped to all-zero.

Power-up sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The power-up sequence is as follows:

- The MAC asserts pmt_intr_o. When only clock-gating is employed in low-power mode, the pmt_intr_o signal can be used to start the clocks that were gated-off after entering low-power mode.
- The software performs the following tasks:
  - De-assert the pmt_intr_o by reading the *PMT control status register (ETH_MACPCSR)*.
  - Perform a write operation (with reset values) to the *PMT control status register (ETH_MACPCSR)* and the *Remote wake-up packet filter register (ETH_MACRWKPFR)* so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
  - Perform write operations to the *Operating mode configuration register (ETH_MACCR)*, *MAC address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)* to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver is on even though the Receive Enable bit is set to 0.

After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

48.7.2 Energy-efficient Ethernet (EEE)

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in Low-power idle (LPI) mode. The

EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbit/s. The peripheral supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows saving power by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit path functions

The transmit path functions include tasks that the MAC must perform to make the PHY enter the LPI state.

In the transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER LST[9:0] bitfield of *LPI timers control register (ETH_MACLTCR)*. The PHY Link Status PLS bit of the *LPI control and status register (ETH_MACLCSR)* indicates the link status of the PHY.

Note: The EEE feature is not supported when the MAC is configured to use the RMII.

According to the standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbit/s)
4. Updates the status (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

To bring the PHY out of the LPI state, that is when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:

   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the *LPI timers control register (ETH_MACLTCR)*.
3. Updates the LPI exit status (TLPIEX bit of the *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Figure 729 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Note: The MAC does not stop the TX_CLK clock. The application can stop this clock (as shown in Figure 729) if the PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o

signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay, and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal depends on the LPITCSE bit of the LPI control and status register (ETH_MACLCSR) and can be done automatically as shown on Figure 730.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.

If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.

If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

Figure 729. LPI transitions (Transmit, 100 Mbds)

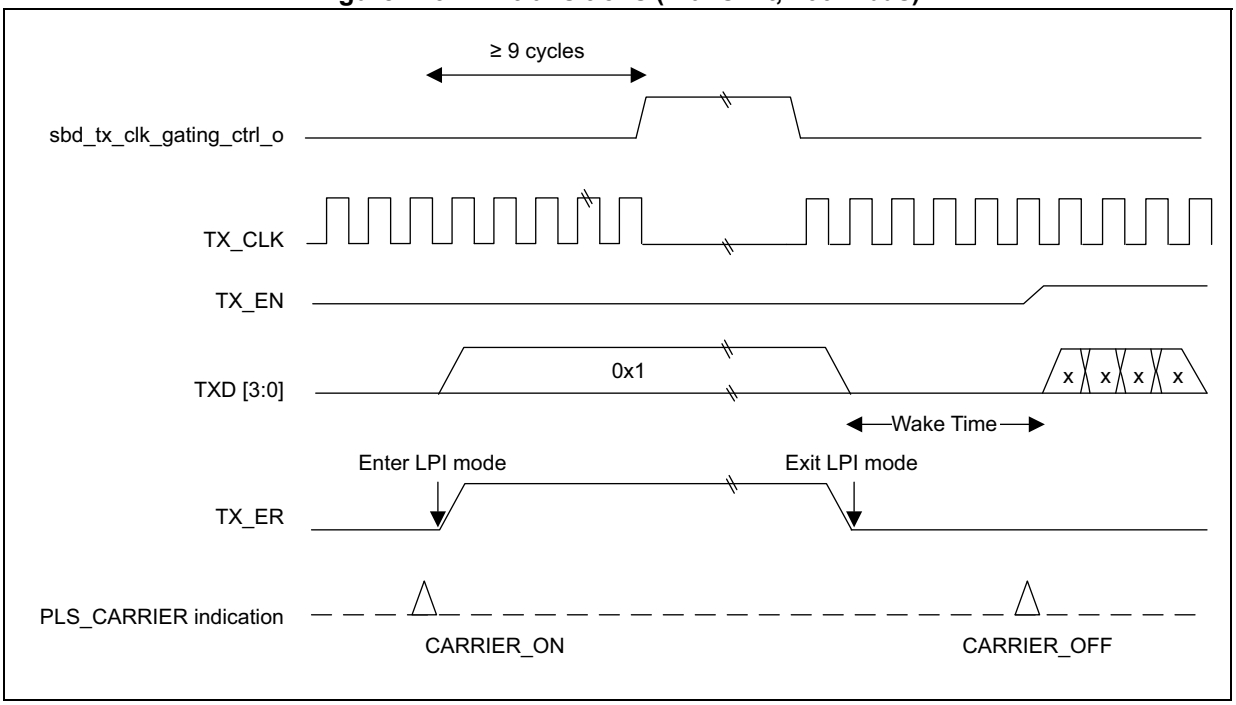
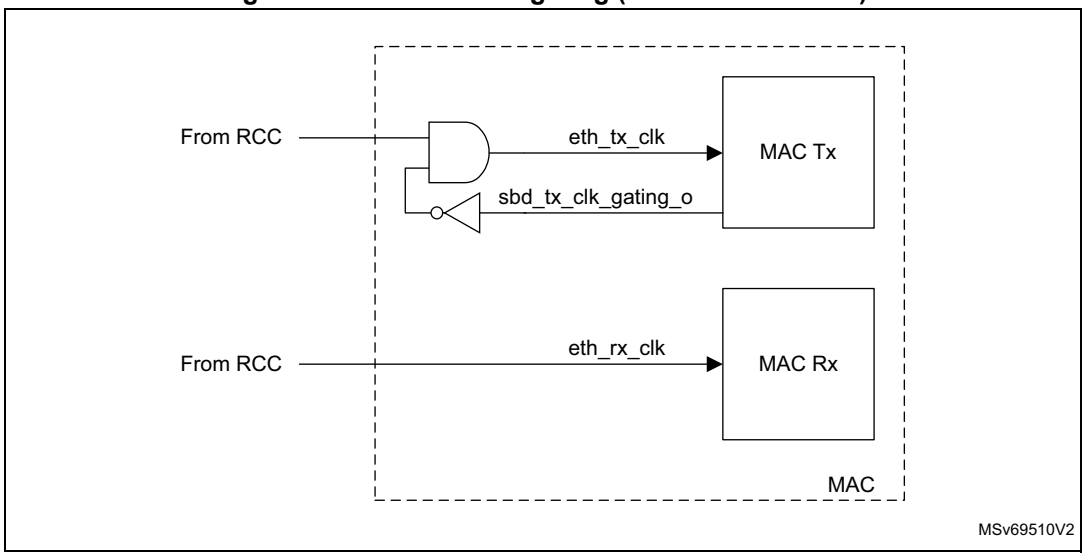


Figure 730. LPI Tx clock gating (when LPITCSE = 1)



Automated entry/exit from LPI mode in transmit path

The MAC transmitter can be programmed to enter and exit LPI Idle mode automatically based on whether it is Idle for a specific period of time or has a packet to transfer. These modes are enabled and controlled by the *LPI control and status register (ETH_MACLCSR)*.

When LPITXA and LPIEN of *LPI control and status register (ETH_MACLCSR)* are set, the MAC transmitter enters LPI Idle state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI Idle state and clears the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when LPITE is also set, the MAC transmitter enters LPI Idle state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in *LPI entry timer register (ETH_MACLETR)*. In this mode also, the MAC transmitter exits the LPI Idle state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI Idle state is possible without any software intervention when the MAC becomes idle again.

When both LPITE and LPITXA bits are cleared, the application can directly control the entry and exit of LPI Idle state by programming the LPIEN bit.

Receive path functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[3:0] to 0x1 (for 100 Mbit/s).
3. The PHY de-asserts RX_DV.
4. The MAC updates the RLPIEN bit of the *LPI control and status register (ETH_MACLCSR)* and immediately generates an interrupt.

Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.

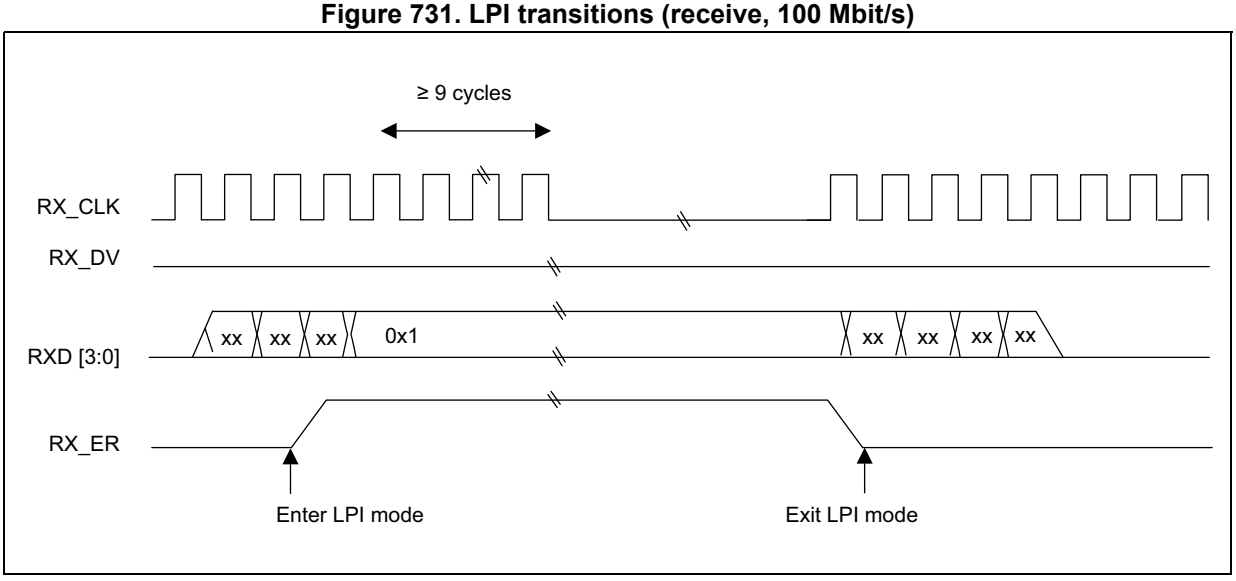
If the duration between the end of the current Rx LPI pattern and the start of the next Rx LPI pattern is very short (that is, less than two Rx clock cycles), then the MAC exits and enters again the Rx LPI mode. The MAC does not trigger the Rx LPI Exit and Entry interrupts.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the *LPI control and status register (ETH_MACLCSR)* and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 731 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 731. LPI transitions (receive, 100 Mbit/s)



Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in Figure 731.

If the MAC is in LPI mode and the application issues a soft reset or hard reset, the MAC receiver exits from LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver enters again the LPI state.

If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.

When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.

LPI timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

The following LPI timers are loaded with the respective values from the *LPI timers control register (ETH_MACLTCR)* and *LPI entry timer register (ETH_MACLETR)*:

- LPI LS TIMER

  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up.

  This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the LST[9:0] bitfield in the *LPI timers control register (ETH_MACLTCR)*. The LPI LS TIMER is 10-bit wide. The software can program up to 1023 milliseconds.
- LPI TW TIMER

  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of *LPI timers control register (ETH_MACLTCR)*. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.

  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 micro seconds.
- LPI AUTO ENTRY TIMER

  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in *LPI control and status register (ETH_MACLCSR)* is set.

LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt sbd_intr_o is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the *LPI control and status register (ETH_MACLCSR)*.

When the MAC exits the Rx LPI state, then in addition to the sbd_intr_o, the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. The lpi_intr_o signal can be used to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the LPI state. It is ORed with pmt_intr_o signal (see *Section : PMT interrupt*) and tied to the EXTI peripheral.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the *LPI control and status register (ETH_MACLCSR)* is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbit/s mode.

Programming guidelines for Energy Efficient Ethernet

For detailed guidelines on the programming guidelines, see *Section 48.9.11: Programming guidelines for Energy Efficient Ethernet (EEE) on page 2244*.

48.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the *Interrupt status register (ETH_DMAISR)* register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMAC0SR: channel 0 status (see *Channel 0 status register (ETH_DMAC0SR)*))
- ETH_MTLISR: interrupt status (see *Interrupt status register (ETH_MTLISR)*)
- ETH_MACISR: interrupt status (see *Interrupt status register (ETH_MACISR)*)

48.8.1 DMA interrupts

Interrupt registers description

The ETH_DMAC0SR: Channel 0 status register (see *Channel 0 status register (ETH_DMAC0SR)*) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMAC0IER: Channel 0 interrupt enable register (see *Channel 0 interrupt enable register (ETH_DMAC0IER)*) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts.They are indicated by Bits[15:14] of ETH_DMAC0SR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the *Interrupt status register (ETH_DMAISR)* for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of *Interrupt status register (ETH_DMAISR)* are cleared.

Receive idle timer

The Receive DMA provides multiple options to generate and regulate the occurrence of a Receive interrupt (RI) in the *Channel 0 status register (ETH_DMAC0SR)*, after an ingress packet is transferred to the host memory.

Among these is an option to generate an RI event when the Receive DMA receives and transfers a receive packet after a long gap or time interval.

The option is enabled by writing a nonzero value to the ITW[4:0] bitfield of the*Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)*. The time interval depends on the programming of this bitfield.

When the option is enabled, this timer restarts when an RI event occurs for which the following apply:

- The RIE bit is set and the RI status bit is cleared.
- The RI status bit is cleared and RIE is programmed.

Equally, the timer stops and is cleared, and no further action is taken if a new packet is received before the timer expiry due.

If the timer expires before the next Receive packet is completely transferred to the host memory, then an RI event is logged and the interrupt is asserted irrespective of the setting of the IOC bit. The other timers in the *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* continue to operate as specified. The timer starts only after a packet is transferred without a corresponding RI event.

For programming guidelines on the Receive idle timer, see *Section 48.9.16: Programming guidelines for receive idle and receive interrupt watchdog timers*

Descriptor refetch when the OWN bit is cleared

By default, the DMA engine checks the OWN bit (xDES3[31]) of the descriptor to confirm whether the descriptor content is valid and can be processed. If the OWN bit is cleared, it drops the descriptor.

If, however, bit 26 of the *Channel 0 control register (ETH_DMAC0CR)* is set, the automatic refetch operation is enabled. In this case, when the DMA engine determines that the OWN bit is cleared, it attempts to refetch the same descriptor. If the OWN bit is still cleared, the DMA engine enters Suspend mode and waits for the software to wake it up by writing into or by updating the tail pointer register, which generates the xBU interrupt.

The description refetch operation may also be required to ensure that the synchronization of the descriptor field update by the application does not generate any error.

For programming guidelines on the descriptor refetch, see *Programming guidelines for descriptor refetch when the OWN bit is cleared*.

Periodic scheduling of transmit and receive interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in *Table 541: TDES2 normal descriptor (read format)*) once for every “required” number of packets to be transmitted.
2. Similarly, set the IOC (RDES3[30] in*Table 554: RDES3 normal descriptor (read format)*) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.

In addition to above, an interrupt timer called *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* is given for flexible control and periodic scheduling of the receive interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in*Table 554: RDES3 normal descriptor (read format)*) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMAC0IER register (see *Channel 0 interrupt enable*

register (ETH_DMAC0IER)). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Receive interrupt watchdog timer

When the DMA channel receives a packet and an SOP is detected, an internal timer starts.

This timer accumulates the packet size extracted from the received packet status (expressed in bytes) over the number of packets received, starting with the packet for which the Receive DMA does not assert the Receive interrupt (RI) because the corresponding interrupt of completion (IOC), namely the DES3[30]bit, is not set.

When the number of accumulated bytes per packet equals or exceeds the programmed value, if the corresponding RIE is enabled in the *Channel 0 interrupt enable register (ETH_DMAC0IER)*, then the RI bit is set and the interrupt is asserted.

If the RI bit is set for a packet transfer whose descriptor IOC was set, or if the interrupt timer expires, the timer is stopped and cleared. The timer is reactivated automatically after the next packet transfer for which the RI event is not generated is complete.

For programming guidelines on the Receive interrupt watchdog timer, see *Section 48.9.16: Programming guidelines for receive idle and receive interrupt watchdog timers*.

Channel transfer complete interrupt

The transmit transfer complete interrupt (TI) and receive transfer complete interrupt (RI) is reflected in the channel status register (*Channel 0 status register (ETH_DMAC0SR)*). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30])).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (*Channel 0 interrupt enable register (ETH_DMAC0IER)*).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bits[17:16]) in the ETH_DMAMR register (*DMA mode register (ETH_DMAMR)*). *Table 538* explains the behavior of the Transfer Complete interrupt.

Table 538. Transfer complete interrupt behavior

Interrupt mode	Behavior of TI/RI and interrupt signal
INTM=0	The TI/RI status signals are set whenever the Transfer complete event is detected. These bits are cleared whenever the software driver writes 1 to these bits. The interrupt signal is asserted whenever the corresponding interrupts are also enabled in the ETH_DMAC0IER register.

Table 538. Transfer complete interrupt behavior

Interrupt mode	Behavior of TI/RI and interrupt signal
INTM=1	The TI/RI is set as explained above. However, the interrupt is not asserted for any RI/TI event.
INTM=2	The RI/TI status bits are set whenever the Transfer Complete event is detected and are reset whenever software driver clears them by writing 1. However, if another Transfer complete event is detected before it is cleared (serviced) by the software, then these status bits are automatically set again. However, the interrupt is not generated based on TI/RI.

48.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

The register *Interrupt status register (ETH_MTLISR)* report the queue number responsible for the event. ETH_MTLQ0ICSR: Queue 0 interrupt control status must be read for event description.

The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MTLISR)* register.

MTL interrupt signal is driven by one of these events:

- Receive queue overflow interrupt
- Transmit queue underflow

48.8.3 MAC interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MACISR)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:

- Receive status interrupt
- Transmit status interrupt
- Timestamp interrupt status
- MMC interrupt status
  - MMC receive checksum offload interrupt status
  - MMC transmit interrupt status
  - MMC Receive interrupt status
- LPI interrupt status
- PMT interrupt status
- PHY interrupt

Note: Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wake-up event detection at EXTI level.

48.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see *Section 48.9.1*)
- MTL initialization (see *Section 48.9.2*)
- MAC initialization (see *Section 48.9.3*)
- Performing normal receive and transmit operation (see *Section 48.9.4*)
- Stopping and starting transmission (see *Section 48.9.5*)
- Programming guidelines for MII link state transitions (see *Section 48.9.8*)
- Programming guidelines for IEEE 1588 timestamping (see *Section 48.9.9*)
- Programming guidelines for energy-efficient Ethernet (see *Section 48.9.11*)
- Programming guidelines for flexible pulse-per-second (PPS) output (see *Section 48.9.12*)
- Programming guidelines for IEEE 1588 auxiliary snapshot (see *Section 48.9.13*)
- Programming guidelines for TSO (see *Section 48.9.14*)
- Programming guidelines for VLAN filtering on the receiver (see *Section 48.9.15*)
- Programming guidelines for receive idle and receive interrupt watchdog timers (see *Section 48.9.16*)

48.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of *DMA mode register (ETH_DMAMR)*).
2. Wait for the completion of the reset process (poll bit 0 of the *DMA mode register (ETH_DMAMR)*, which is cleared when the reset operation is completed).
3. Program the following fields to initialize the *System bus mode register (ETH_DMASBMR)*:
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values in case of AHB bus interface.
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to *Section 48.10: Descriptors*.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMAC0TXRLR and ETH_DMAC0RXRLR registers (see *Channel 0 Tx descriptor ring length register (ETH_DMAC0TXRLR)* and *Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)*). The programmed ring length must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (*Channel 0 Tx descriptor list address register (ETH_DMAC0TXDLAR)*, *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*). In addition, program the transmit and receive tail pointer registers that inform the DMA about the available descriptors (see *Channel 0 Tx descriptor tail pointer register (ETH_DMAC0TXDTPR)* and *Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)*).
7. Program ETH_DMAC0CR, ETH_DMAC0TXCR and ETH_DMAC0RXCR registers (see *Channel 0 control register (ETH_DMAC0CR)*, *Channel 0 transmit control register (ETH_DMAC0TXCR)* and *Channel 0 receive control register (ETH_DMAC0RXCR)*) to configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ[13:0] for RxDMA, and so on.
8. Enable the interrupts by programming the ETH_DMAC0IER register (see *Channel 0 interrupt enable register (ETH_DMAC0IER)*).
9. Start the receive and transmit DMAs by setting SR (bit 0) of *Channel 0 receive control register (ETH_DMAC0RXCR)* and ST (bit 0) of the ETH_DMAC0TXCR (see *Channel 0 transmit control register (ETH_DMAC0TXCR)*).

48.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the following fields to initialize the operating mode in *Tx queue 0 operating mode register (ETH_MTLTXQ0OMR)*.
   a) Transmit store and forward (TSF) or transmit threshold control (TTC) if the Threshold mode is used.
2. Program the following fields to initialize the operating mode in the ETH_MTLRXQ0OMR register (see *Rx queue 0 operating mode register (ETH_MTLRXQ0OMR)*):
   a) Receive store and forward (RSF) or RTC if Threshold mode is used.
   b) Flow control activation and deactivation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error packet and undersized good packet forwarding enable (FEP and FUP).
   d) Receive queue size (RQS).

48.9.3 MAC initialization

The MAC configuration registers establish the operating mode of the MAC. If possible, these registers must be initialized before initializing the DMA. The following MAC Initialization operations can also be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: *MAC address x low register (ETH_MACAxLR)*, *MAC address 0 high register (ETH_MACA0HR)* and *MAC address x high register (ETH_MACAxHR)*.
2. Program the following fields to set the appropriate filters for the incoming frames in the *Packet filtering control register (ETH_MACPFR)*:
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the *Tx Queue flow control register (ETH_MACQTXFCR)*:
   a) Pause time and other Pause frame control bits.
   b) Transmit Flow control bits.
   c) Flow Control Busy.
4. Program the *Interrupt enable register (ETH_MACIER)* as required, if it is applicable for your configuration.
5. Program the appropriate fields in the *Operating mode configuration register (ETH_MACCR)* register. For example, Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in *Operating mode configuration register (ETH_MACCR)* register to start the MAC transmitter and receiver.

To support Jumbo Transmit/Receive packets, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JE bit to 1.
  b) Set JD and WD bits to 0 to avoid giant packet error reporting.
  c) Set GPSLCE bit to 1
  d) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to a value > 9026

To support Transmit/Receive packets, up to 16K, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JD and WD bits to 1 to avoid giant packet error reporting.
  b) Set GPSLCE bit to 1.
  c) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to 16383.

48.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the descriptors and writing the ETH_DMAC0TXDTPR (see *Channel 0 Tx descriptor tail pointer register (ETH_DMAC0TXDTPR)* and ETH_DMAC0RXDTPR (see *Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)*).
4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMAC0CATXDR and ETH_DMAC0CARXDR registers (see *Channel 0 current application transmit descriptor register (ETH_DMAC0CATXDR)* and *Channel 0 current application receive descriptor register (ETH_DMAC0CARXDR)*).
5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMAC0CATXDR and ETH_DMAC0CARXDR registers (see *Channel 0 current application transmit descriptor register (ETH_DMAC0CATXDR)* and *Channel 0 current application receive descriptor register (ETH_DMAC0CARXDR)*).

48.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMAC0TXCR register (see *Channel 0 transmit control register (ETH_DMAC0TXCR)*).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue 0 debug register (ETH_MTLTXQ0DR)* (TRCSTS[1:0] is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing RE and TE bits of the *Operating mode configuration register (ETH_MACCR)* Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of *Tx queue 0 debug register (ETH_MTLTXQ0DR)*, PRXQ=0 and RXQSTS[1:0] = 00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in *Tx queue 0 debug register (ETH_MTLTXQ0DR)* and RXQSTS[1:0] is set to 00.
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

Note: Do not change the configuration (such as duplex mode, speed, port, or loopback) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active.

Similarly, do not change the DMA-related configuration when transmit and receive DMA are active.

48.9.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in Suspend state, as explained below:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to Suspend state due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of an RxFIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into Stop state.

48.9.7 Programming guidelines for switching the AHB clock frequency

To dynamically change the AHB clock frequency (without applying soft reset or hard reset), follow these steps:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. When the frame transmissions is complete, the Tx FIFO becomes empty and the Tx DMA enters Stop state. The Tx FIFO status is given in the *Tx queue 0 debug register (ETH_MTLTXQ0DR)* and the status of DMA is given in *Debug status register (ETH_DMADSR)*.
2. Disable the MAC transmitter and the MAC receiver by clearing the appropriate bits in *Operating mode configuration register (ETH_MACCR)*.

3. Disable the Receive DMA (if applicable) after making sure that the data in the Rx FIFO is transferred to the system memory. The Rx FIFO empty status is given in *Rx queue 0 debug register (ETH_MTLRXQ0DR)*.
4. Ensure that the application does not perform any register read or write operation.
5. Change the frequency of the AHB clock.
6. Enable the MAC transmitter or the MAC receiver and the Transmit or Receive DMA. These steps ensure that no valid data is present in the Tx FIFO or Rx FIFO at the time of clock frequency switching and prevent any data corruption.

48.9.8 Programming guidelines for MII link state transitions

Transmit and receive clocks are running when the link is down

Complete the following steps when the link is down while the transmit and receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of *Channel 0 control register (ETH_DMAC0CR)*.
2. Disable the MAC receiver by clearing RE bit of *Operating mode configuration register (ETH_MACCR)*.
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue 0 debug register (ETH_MTLTXQ0DR)* (TRCSTS[1:0] is not 01).

   or

   Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing TE bit of the *Operating mode configuration register (ETH_MACCR)* Register.
5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in *Tx queue 0 debug register (ETH_MTLTXQ0DR)* and RXQSTS[1:0] to 00 in *Rx queue 0 debug register (ETH_MTLRXQ0DR)*).
6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.
7. Restart the operation by starting the Tx DMA. Then enable the MAC transmitter and receiver.

   The Rx DMA does not need to be enabled: since the receiver is disabled, there are no data in the Rx FIFO.

Transmit and receive clocks are stopped when the link is down

Complete the following steps when the link is down and the transmit and receive clocks are stopped:

1. Disable the MAC transmitter and receiver by clearing RE and TE bits in the *Operating mode configuration register (ETH_MACCR)*. This does not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the *Debug register (ETH_MACDR)* (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC transmitter is stopped.

4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC transmitter and receiver by setting RE and TE bits.

48.9.9 Programming guidelines for IEEE 1588 timestamping

Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the *Timestamp control register (ETH_MACTSCR)*. However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 12 of *Interrupt enable register (ETH_MACIER)*.
2. Set bit 0 of *Timestamp control register (ETH_MACTSCR)* to enable timestamping.
3. Program *Subsecond increment register (ETH_MACSSIR)* based on the PTP clock frequency.
4. If you use the Fine Correction method, program *Timestamp addend register (ETH_MACTSAR)* and set bit 5 of *Timestamp control register (ETH_MACTSCR)*.
5. Poll the *Timestamp control register (ETH_MACTSCR)* until bit 5 is cleared.
6. Program bit 1 of *Timestamp control register (ETH_MACTSCR)* to select the Fine Update method (if required).
7. Program *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)* with the appropriate time value.
8. Set bit 2 in *Timestamp control register (ETH_MACTSCR)*.

   The timestamp counter starts as soon as it is initialized with the value written in the timestamp update registers. If one-step timestamping is required:

   a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.

   b) Program *Timestamp ingress asymmetric correction register (ETH_MACTSIACR)* to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Note: If timestamp operation is disabled by clearing bit 0 of Timestamp control register (ETH_MACTSCR), repeat all these steps to restart the timestamp operation.

System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (*System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*).
2. Set bit 3 (TSUPDT) of the *Timestamp control register (ETH_MACTSCR)*.

   The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in *Section : System time register module*, calculate at which rate you intend to increment or decrement the system time.
2. Update the *Timestamp addend register (ETH_MACTSAR)* with the new value and set bit 5 of the *Timestamp control register (ETH_MACTSCR)* Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*.
5. Enable the Timestamp interrupt in bit 12 of *Interrupt enable register (ETH_MACIER)*.
6. Set bit 3 in Register *Timestamp control register (ETH_MACTSCR)*.
7. When this trigger generates an interrupt, read *Interrupt status register (ETH_MACISR)*.
8. Reprogram *Timestamp addend register (ETH_MACTSAR)* with the old value and set bit 5 again.

48.9.10 Programming guidelines for PTP offload feature

Programming guidelines to enable automatic periodic generation of PTP sync messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program SNAPTYPSEL, TSMSTRENA, and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 0, 1, and 1 respectively, to configure the node as Ordinary or Boundary Master (1, 1, and 1 for Transparent Master).
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Sync message.
3. Program the ASYNCEN bit of *PTP offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Sync message.
5. Program the LSI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Sync messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Sync message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Sync message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable periodic generation of PTP Pdelay_Req messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 1, 0, and 1 respectively to configure the node as Transparent Slave (1, 1, and 1 for Transparent Master OR 3, X, and X for Peer-to-Peer Transparent).
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Pdelay_Req message.
3. Program the APDREQEN bit of *PTP offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Pdelay_Req messages.
4. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Pdelay_Req message.
5. Program the LMPDRI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Pdelay_Req messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Pdelay_Req message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Pdelay_Req message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Master mode

Follow these steps to enable the generation of PTP response messages for Ordinary or Boundary Master mode (Periodic PTP Sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 0, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the DRSYNCR and LSI fields in *Log message interval register (ETH_MACLMIR)*. The sum of both fields is updated in logMinDelayReqInterval field of PTP Delay_Resp message.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Slave mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Slave mode (PTP Delay_Req message generated in response to PTP Sync message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 0, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the DRSYNCR field in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Slave mode

Follow these steps to enable generation of PTP response messages for Transparent Slave mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated)

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 1, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
4. Program the DRSYNCR and LMPDRI fields in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Master mode

Follow these steps to enable generation of PTP response messages for Transparent Master mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 1, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
4. Program the DRSYNCR, LSI and LMPDRI fields in *Log message interval register (ETH_MACLMIR)*, the sum of DRSYNCR and LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Peer-to-Peer Transparent mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):

1. Program the SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 3, X, and X respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
4. Program the LMPDRI field in *Log message interval register (ETH_MACLMIR)* to indicate periodicity of the PTP Pdelay_Req messages

5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

48.9.11 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-power idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol.

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).
3. Program bits 25 to 16 and bits 15 to 0 in *LPI timers control register (ETH_MACLTCR)*.
4. Read the PHY link status by using the MDIO interface and update bit 17 of *LPI control and status register (ETH_MACLCSR)*.

   Update *LPI control and status register (ETH_MACLCSR)* accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program *One-microsecond-tick counter register (ETH_MAC1USTCR)* as per the frequency of the clock used for accessing the CSR slave port.
6. Program the LPIET bit in the *LPI entry timer register (ETH_MACLETR)* with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bits 20 to 19) of *LPI control and status register (ETH_MACLCSR)* to enable LPI auto-entry and MAC auto-exit from LPI state.
8. Set bit 16 of *LPI control and status register (ETH_MACLCSR)* to put the MAC transmitter in LPI state.

   The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.
9. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.
10. The MAC transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.
11. Reset LPIEN bit if the application needs to override the auto-entry/exit modes and directly exit the MAC transmitter from LPI state.

Note: To make sure the MAC enters the LPI state only after the transmission of all the queued frames in the Tx FIFO is complete, set LPITXA bit in LPI control and status register (ETH_MACLCSR).

To switch off the CSR clock or power to the rest of the system during the LPI state, wait for the TLPIEN interrupt of LPI control and status register (ETH_MACLCSR) to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

Gating Off the CSR Clock in the LPI mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

Gating off the CSR clock in the Rx LPI mode

The following operations are performed when the MAC receives the LPI pattern from the PHY:

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status (RLPIEN interrupt of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, the CSR clock can be gated off. If the MAC TX is not in LPI mode when the CSR clock is gated off, the events on the MAC transmitter do not get reported or updated in the CSR. To restore the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when *LPI control and status register (ETH_MACLCSR)* is read.

Gating off the CSR clock in the Tx LPI mode

The following operations are performed when bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, the CSR clock can be gated off. If the MAC RX is not in LPI mode when the CSR clock is gated off, the events on the MAC receiver do not get reported or updated in the CSR. To restore the CSR clock, switch on the CSR clock when the MAC has exited TX LPI mode. After the CSR clock is resumed, reset bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* to exit the MAC from LPI mode.

48.9.12 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:

1. Program TRGTMODSEL[1:0] bit to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) as start time of PPS signal output.
2. Program the start time value in the target time registers (register *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
3. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)* Register.
4. Program PPSCMD[3:0] of *PPS control register (ETH_MACPPSCR)* to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the Target Time registers.

Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has elapsed. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be higher than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in *PPS width register (ETH_MACPPSWR)*.
3. Program PPSCMD[3:0] bits of *PPS control register (ETH_MACPPSCR)* to generate a single pulse after the previous pulse is deasserted. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in Target Time registers.

   If this command is given before the previous pulse becomes low, then the new command overwrites the previous command and the peripheral may generate only 1 extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:

1. Program TRGTMODSEL[1:0] bits to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) for start time of the PPS signal output.
2. Program the start time value in the Target Time registers (register *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
3. Program the interval value between the train of pulses on the PPS signal output in *PPS interval register (ETH_MACPPSIR)*.
4. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)*.

5. Program PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

   By default, the PPS pulse train is free-running unless it is stopped by issuing a 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands.
6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in *PPS target time nanoseconds register (ETH_MACPPSTTNR)* is reset before programming the Target Time registers again.
7. Program the PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

The pulse train can be stopped at any time by programming 0101 in the PPSCMD[3:0] field.

Similarly, the Stop Pulse train command (given in Step 7) can be canceled by programming PPSCMD[3:0] bits to 0110 before the time (programmed at step 6) has elapsed.

The pulse train generation can be stopped by programming PPSCMD[3:0] to 0011 before the start time programmed at step 2) has elapsed.

Generating an interrupt without affecting the PPS

TRGTMODSEL[1:0] bits in *PPS control register (ETH_MACPPSCR)* enable you to program the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL[1:0] bits of *PPS control register (ETH_MACPPSCR)* to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

   If TRGTMODSEL[1:0] bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

Note: The TSTRGTERR0 bit in Timestamp status register (ETH_MACTSSR) is set when the programmed target time is smaller (that is corresponds to a time in the past) compared to the system time in the System time seconds register (ETH_MACSTSR) and System time nanoseconds register (ETH_MACSTNR).

An interrupt is generated (sbd_intr_o) if the TSIE bit in the Interrupt enable register (ETH_MACIER) is set.

Therefore, to avoid unwanted interrupts, the correct writing order is as follows:

1. *PPS target time nanoseconds register (ETH_MACPPSTTNR).*
2. *PPS target time seconds register (ETH_MACPPSTTSR).*
3. *PPS interval register (ETH_MACPPSIR).*
4. *PPS width register (ETH_MACPPSWR).*
5. *PPSCTRL[3:0] and PPSCTRL[3:0] and PPSEN0 bitfields of PPS control register (ETH_MACPPSCR).*

48.9.13 Programming guidelines for IEEE 1588 auxiliary snapshot

The IEEE 1588 auxiliary snapshot function is used to capture the auxiliary snapshot for a trigger event received on the eth_ptp_aux_ts_trig_i input.

To program the IEEE 1588 auxiliary snapshot function, complete the following steps:

1. Program the TSIE bit in the *Interrupt enable register (ETH_MACIER)* to enable interrupt generation when the auxiliary snapshot is captured in the FIFO.
2. Program the ATSENx bit of the *Auxiliary control register (ETH_MACACR)* to enable the capture of the auxiliary snapshot based on the respective trigger input.
3. If the software does not want the captured auxiliary snapshot to be read, program the ATSFC bit of *Auxiliary control register (ETH_MACACR)* anytime to clear the auxiliary snapshot FIFO.
4. When TSIS is set or an interrupt is generated (due to TSIE programmed to 1), read the *Timestamp status register (ETH_MACTSSR)*, *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)* to access the captured auxiliary snapshot.

Note: When the auxiliary snapshot is captured in the FIFO for a trigger event, the following bitfields are updated:

- *AUXTSTRIG (status bit indicating that an auxiliary snapshot has been captured), ATSSTN (trigger event input source identifying the first available entry in the FIFO), and ATSNS (indicating the number of auxiliary snapshots available in the FIFO) bitfields of the Timestamp status register (ETH_MACTSSR).*
- *TSIS (indicating that an interesting event occurred in the IEEE 1588 timestamp function) bitfield in the Interrupt status register (ETH_DMAISR).*

The AUXTSTRIG bit is cleared on read (or set when the RCWE bit of the CSR software control register (ETH_MACCSRSWCR) is set).

Note: When the captured auxiliary snapshot is ignored because that the FIFO is full, the ATSSTM bit, which indicates that a trigger has been missed, is set.

48.9.14 Programming guidelines for TSO

The TCP segmentation offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet.

Follow the steps below to program TSO:

1. Program TSE bit of the corresponding *Channel 0 transmit control register (ETH_DMAC0TXCR)* to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
   – MSS[13:0] of *Channel 0 control register (ETH_DMAC0CR)*
   – or MSS in the context descriptor

   If MSS[13:0] field is programmed in both *Channel 0 control register (ETH_DMAC0CR)* and in the context descriptor, the latest software programmed sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first descriptor. This buffer must not hold any payload bytes. The payload is allocated to Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.

48.9.15 Programming guidelines to perform VLAN filtering on the receiver

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program *VLAN tag register (ETH_MACVTR)* for the following bit to select the filtering method:
   – ETV: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   – VTHM: VLAN Tag Hash Table Match Enable.
   – ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   – ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   – DOVLTC: Ignores VLAN Type for Tag Match
   – VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL bit in *VLAN tag register (ETH_MACVTR)* for the 12-bit or 16-bit VLAN tag.
3. If VLAN tag Hash filtering is enabled, program *VLAN Hash table register (ETH_MACVHTR)*.
   – When the ETV bit is reset, the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the *VLAN Hash table register (ETH_MACVHTR)*.
   – When ETV bit is set, the upper 4 bits of the calculated CRC-32 of VLAN tag are used to index the content of *VLAN Hash table register (ETH_MACVHTR)*.

   For example, when ETV bit is set, a hash value of 0b1000 selects bit 8 of the VLAN Hash table. When ETV bit is reset, a hash value of 0b1000 selects bit 7 of the VLAN Hash table.

48.9.16 Programming guidelines for receive idle and receive interrupt watchdog timers

Programming guidelines for the receive idle timer

Follow these steps to enable the receive idle timer.

1. Program the ITW[4:0] bitfield of the *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* with a nonzero value to enable the Idle timer.
2. Use the ITCU[1:0] bitfield of the *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* to define the number of application clock cycles per unit in the ITW[4:0] bitfield.
3. Program the ITW[4:0] bitfield of the *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* for a given channel to check the trigger condition for the Idle timer.
4. Program the RIE bit of the *Channel 0 interrupt enable register (ETH_DMAC0IER)* to a timer value of ITW × ITCU application clock cycles.

Programming guidelines for the receive interrupt watchdog timer

Follow these steps to enable the receive interrupt watchdog timer.

1. Set the PSEL bit of the *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* to set the receive watchdog timer to operate in received packet counter mode.
2. Clear the PSEL bit of the *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* to set the timer to operate in received byte counter mode.
3. Program the RBCT[9:0] bitfield of the *Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)* with a nonzero value to set the byte-per-packet threshold for the receive watchdog timer.

Programming guidelines for descriptor refetch when the OWN bit is cleared

The descriptor refetch checks the value of the OWN bit twice before the DMA engine enters Suspend mode.

To enable the refetch on both the transmit and receive sides, set the DRO bit of the *Channel 0 control register (ETH_DMAC0CR)*.

48.10 Descriptors

48.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- **Normal descriptors**

  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

- **Context descriptors**

  The context descriptors are used to provide control information applicable to the packet to be transmitted.

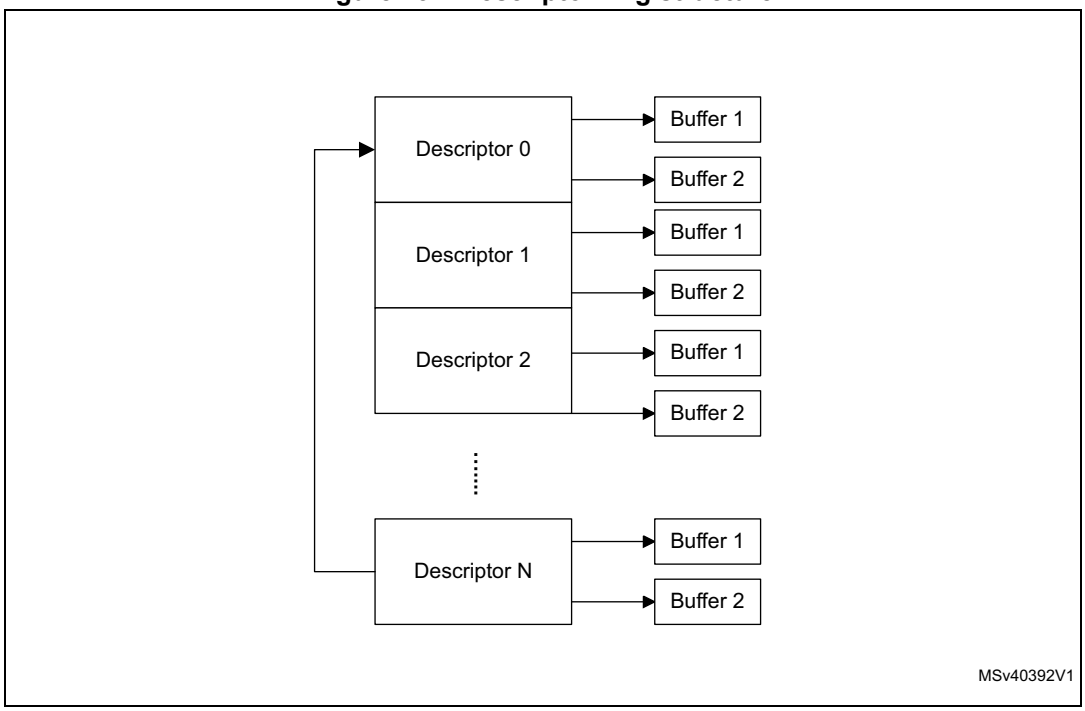
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.

48.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

Figure 732. Descriptor ring structure



In a ring structure, descriptors are separated by the 32-bit word number programmed in the DSL field of the *Channel 0 control register (ETH_DMAC0CR)*). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- *Channel 0 Tx descriptor ring length register (ETH_DMAC0TXRLR)*)
- *Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)*)

The *Channel 0 Tx descriptor tail pointer register (ETH_DMAC0TXDTPR)* or *Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)* contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the descriptor tail pointer register and update the tail pointer so that the following condition is met:

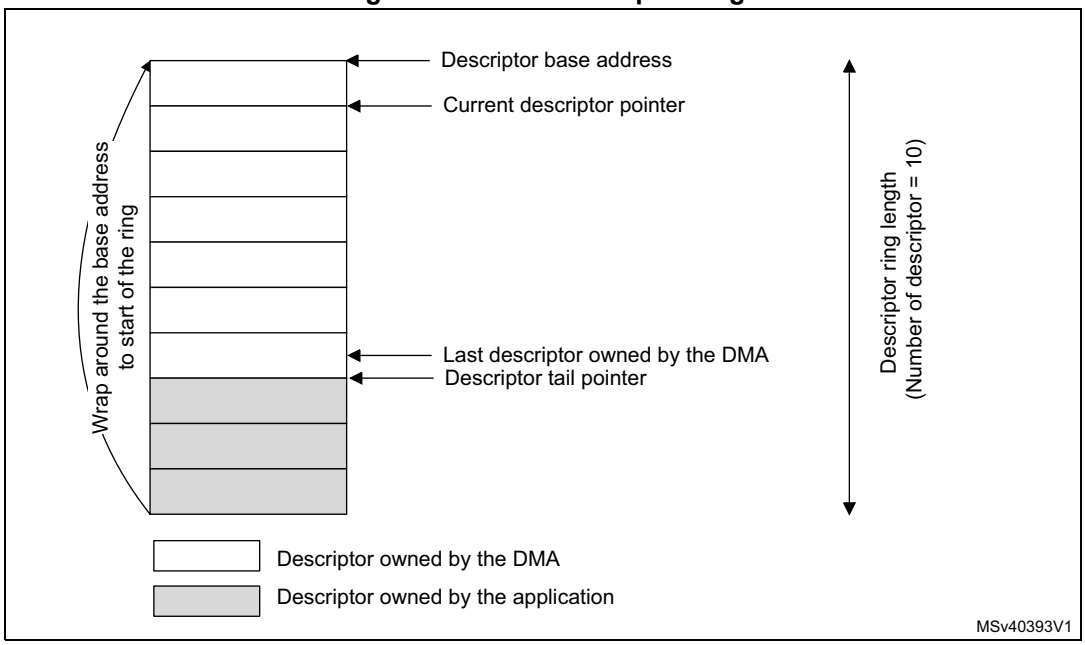
```
Descriptor Tail Pointer > Current Descriptor Pointer;
```

Note: When the application sets the descriptors into the end-of-descriptor ring, continues setting from the start of the descriptor list and updating the descriptor tail pointer, the above condition is not satisfied. The DMA detects such a case, processes the descriptors until the end of the ring and then continues processing from the start of the descriptor list until the current descriptor pointer is equal to the descriptor tail pointer.

When the application programs the descriptor tail pointer beyond the end of the descriptor ring, the DMA interprets this as the availability of one more complete ring of descriptors, and after processing the last descriptor in the ring continues to process descriptors from the start of the descriptor list. If this is not intended, the application must program the descriptor tail pointer to the start of the descriptor list after setting the last descriptor in the ring.

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 733: DMA descriptor ring*.

Figure 733. DMA descriptor ring



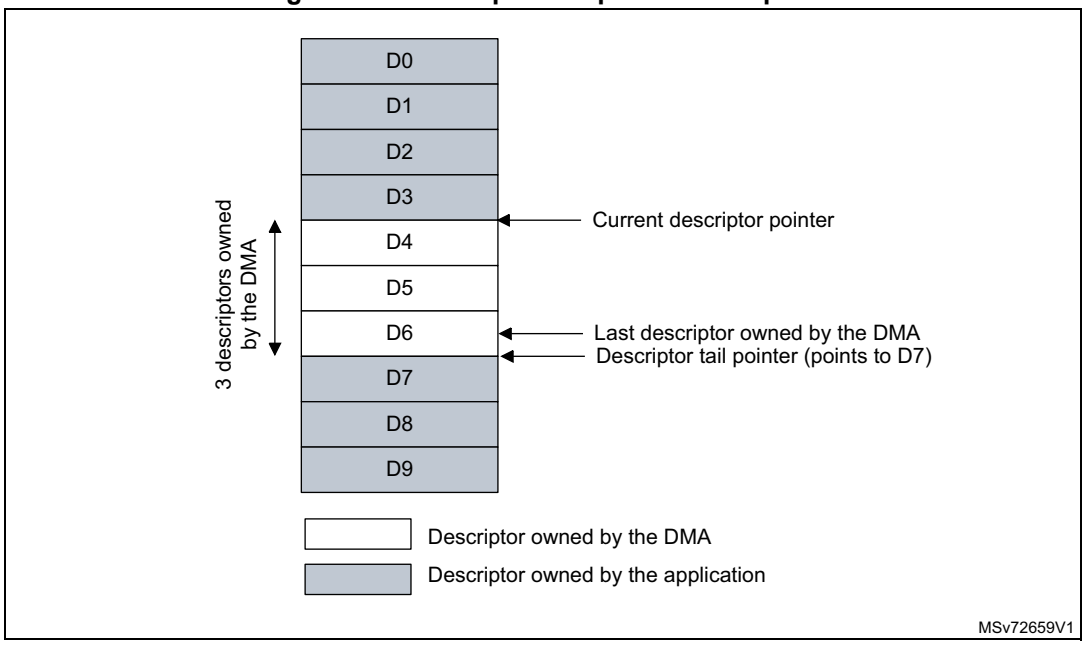
For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.

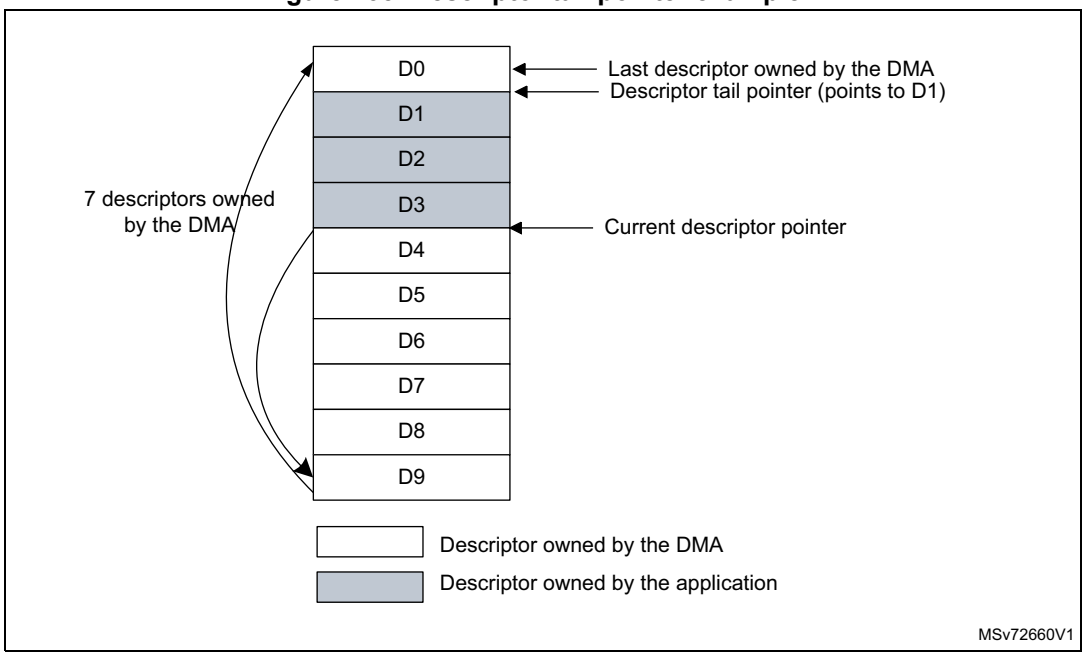
Descriptor tail pointer handling examples

Figure 734. Descriptor tail pointer example 1



When the current descriptor pointer points to descriptor 4 (D4) and the descriptor tail pointer points to descriptor 7 (D7), the last descriptor owned by the DMA is descriptor 6 (D6) and there are three descriptors (D4, D5, D6) available for the DMA, as shown in *Figure 734: Descriptor tail pointer example 1*.

Figure 735. Descriptor tail pointer example 2



As shown in *Figure 735: Descriptor tail pointer example 2*, the application frees four descriptors (D7, D8, D9 and D0) and updates the descriptor tail pointer to point to descriptor 1 (D1). The last descriptor owned by the DMA is descriptor 0 (D0). The descriptors referenced between the current descriptor pointer (D4) and the descriptor tail pointer (D1) are available to the DMA.

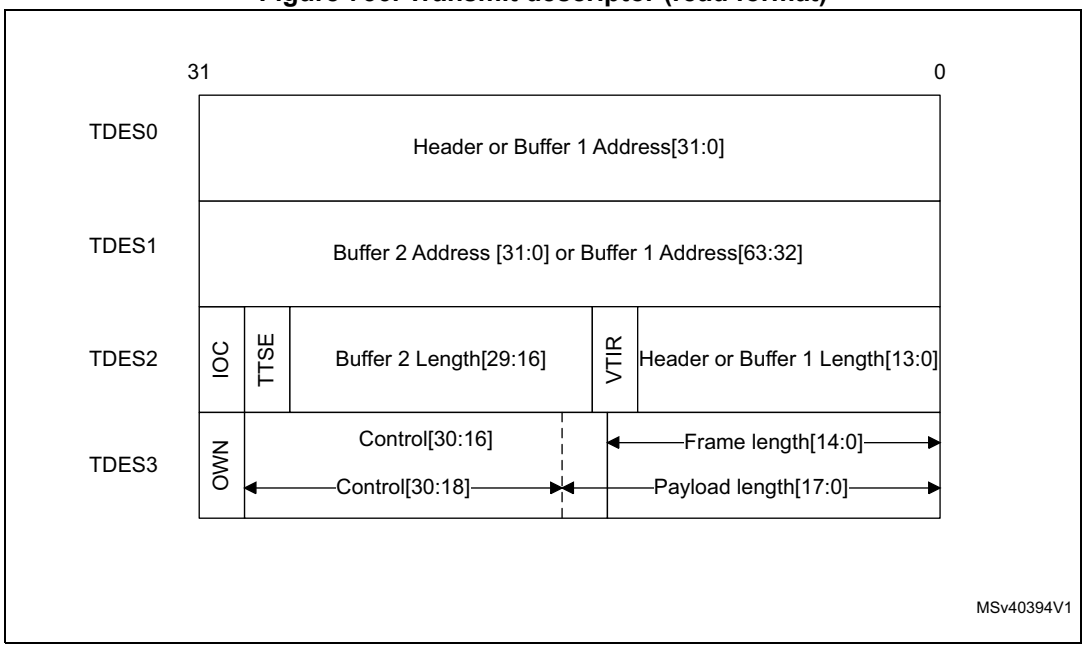
48.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format.

Transmit normal descriptor (read format)

Figure 736 shows the Read format for Transmit normal descriptor. *Table 539* to *Table 542* provide a detailed description of all Transmit normal descriptors (read format).

Figure 736. Transmit descriptor (read format)



- TDES0 normal descriptor (read format)

Table 539. TDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 address pointer or TSO header(1) address pointer** These bits indicate either the physical address of buffer 1 or the TSO header address pointer when the following bits are set: – TSE bit of TDES3 – FD bit of TDES3

1. TSO header means from the Ethernet source address to the end of the TCP header.

- TDES1 normal descriptor (read format)

Table 540. TDES1 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 or buffer 1 address pointer:** These bits indicate the physical address of buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment.

- TDES2 normal descriptor (read format)

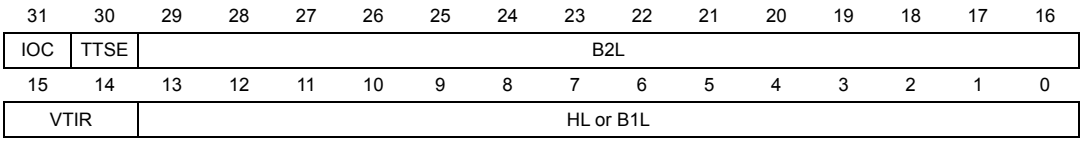


Table 541. TDES2 normal descriptor (read format)

Bits	Name	Description
31	IOC	**Interrupt on completion:** This bit sets the TI bit in the *Channel 0 status register (ETH_DMAC0SR)*) when the present packet transmission is complete.
30	TTSE	**Transmit timestamp enable** This bit enables the IEEE1588 timestamping for the transmit packet referenced by the descriptor.
29:16	B2L	**Buffer 2 length** The driver sets this field. When set, this field indicates the buffer 2 length.
15:14	VTIR	**VLAN tag insertion or replacement:** These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows: 00: Do not add a VLAN tag. 01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 10: Insert a VLAN tag with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. 11: Replace the VLAN tag in packets with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. This option should be used only with the VLAN packets.

Table 541. TDES2 normal descriptor (read format) (continued)

Bits	Name	Description
13:0	HL or B1L	**Header length or buffer 1 length** For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length. If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from the Ethernet source address until the end of the TCP header (Ethernet frame header, IP header, and TCP header). The maximum header length supported for TSO feature is 1023 bytes. If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length.

- TDES3 normal descriptor (read format)

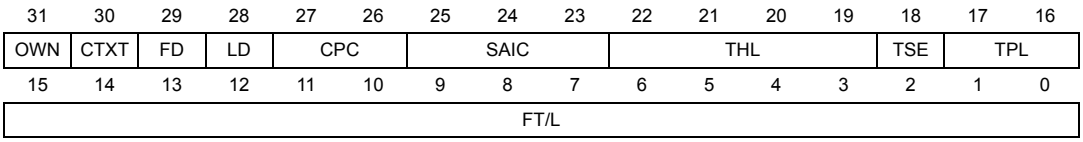


Table 542. TDES3 normal descriptor (read format)

Bits	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptor.
29	FD	**First Descriptor** When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value.

Table 542. TDES3 normal descriptor (read format) (continued)

Bits	Name	Description
27:26	CPC	**CRC pad control** This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following: 00: CRC and pad insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 01: CRC insertion (disable pad insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 10: Disable CRC insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 11: CRC replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the transmit buffer. When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.
25:23	SAIC	**SA insertion control** These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC address register (1 or 0) value that is used for source address insertion or replacement. The following list describes the values of Bits[24:23]: 00: Do not include the source address 01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. 11: Reserved These bits are valid when the First Segment control bit (TDES3 [29]) is set.
22:19	THL	**TCP header length** (in words) If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. THL value must be equal to 2 for UDP header. This field is valid only for the first descriptor.
18	TSE	**TCP segmentation enable** When this bit is set, the DMA performs the TCP/UDP segmentation for a packet. This bit is valid only if the FD bit is set.

Table 542. TDES3 normal descriptor (read format) (continued)

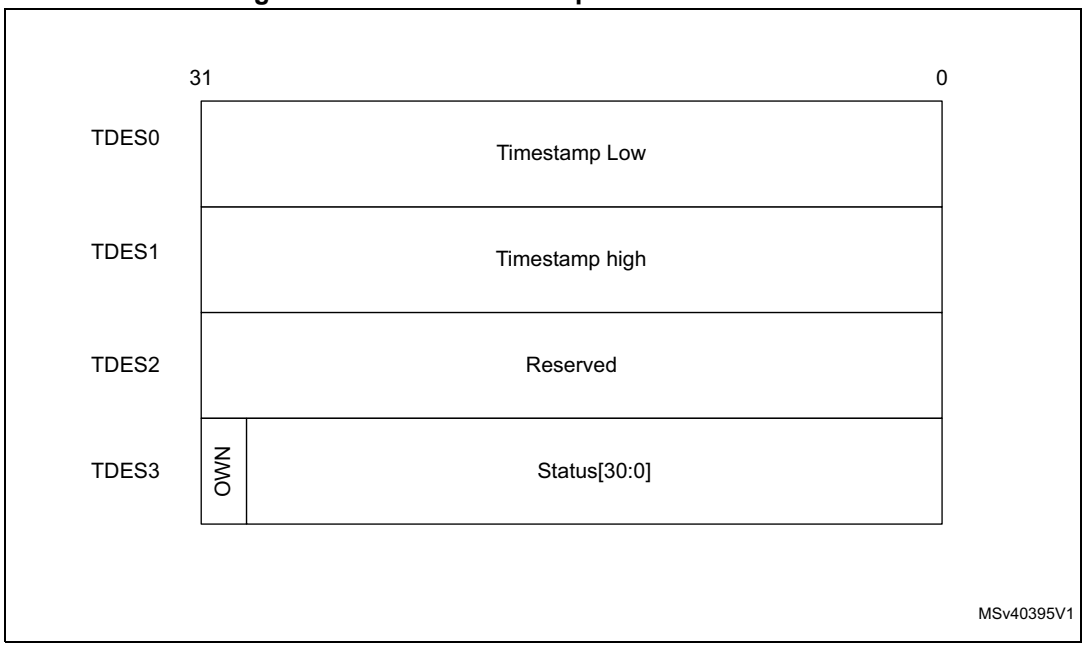
Bits	Name	Description
17:16	CIC/TPL	**Checksum insertion control or TCP payload length** These bits control the checksum calculation and insertion. They can take the following values: 00: Checksum insertion disabled. 01: Only IP header checksum calculation and insertion are enabled. 10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 11: IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the TSE bit is reset.When the TSE bit is set, it contains the upper bits [17:16] of the TCP Payload length. This allows the TCP packet length field to be spanned across TDES3[17:0] to provide 256 Kbyte packet length support.
15	TPL	**Reserved or TCP payload length** When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0].
14:0	FL/TPL	**Reserved or TCP payload length** When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length. This length does not include Ethernet header or TCP/UDP/IP header length. When the TSE bit is reset, this bit is reserved.

Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 737 shows the write-back format for Transmit normal descriptors. *Table 543* to *Table 546* provide a detailed description of all Transmit Normal descriptors (Write-Back Format).

Figure 737. Transmit descriptor write-back format



- TDES0 normal descriptor (write-back format)

Table 543. TDES0 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES1 normal descriptor (write-back format)

Table 544. TDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES2 normal descriptor (write-back format)

Table 545. TDES2 normal descriptor (write-back format)(1)

Bit	Description
31:0	Reserved

1. This format is only applicable to the last descriptor of a packet.

- TDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	Reserved										TTSS	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	PCE	LoC	NC	LC	EC	CC				ED	UF	DB	IHE

Table 546. TDES3 normal descriptor (write-back format)(1)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptors.
29	FD	**First Descriptor** This bit indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27:24	Reserved	
23	DE	**Descriptor Error** When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor errors can be: – Incorrect sequence from the context descriptor, such as a location after the first descriptor for a packet. – All 1s. – CTXT is set to 1 along with LD or FD bits set to 1. *Note: When a Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel 0 status register (ETH_DMAC0SR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.*
22:18	Reserved	

Table 546. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
17	TTSS	**Tx Timestamp Status** This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set.
16	Reserved	
15	ES	**Error Summary** This bit indicates the logical OR of the following bits: TDES3[0]: IP Header Error TDES3[14]: Jabber timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error
14	JT	**Jabber timeout** This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the *Operating mode configuration register (ETH_MACCR)* is not set.
13	FF	**Packet Flushed** This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	**Payload Checksum Error** This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.
11	LoC	**Loss of Carrier** This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.

Table 546. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
10	NC	**No Carrier** This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	**Late Collision** This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode). This bit is not valid if Underflow Error is set.
8	EC	**Excessive Collision** This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	**Collision Count** This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	**Excessive Deferral** This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times if DC bit is set in the *Operating mode configuration register (ETH_MACCR)*.
2	UF	**Underflow Error** This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty Transmit Buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register.
1	DB	**Deferred Bit** This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the Half-duplex mode.
0	IHE	**IP Header Error** When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

1. This format is only applicable to the last descriptor of a packet.

Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, and VLAN Tag ID for VLAN insertion feature. Write-back is only done on a context descriptor to reset the OWN bit.

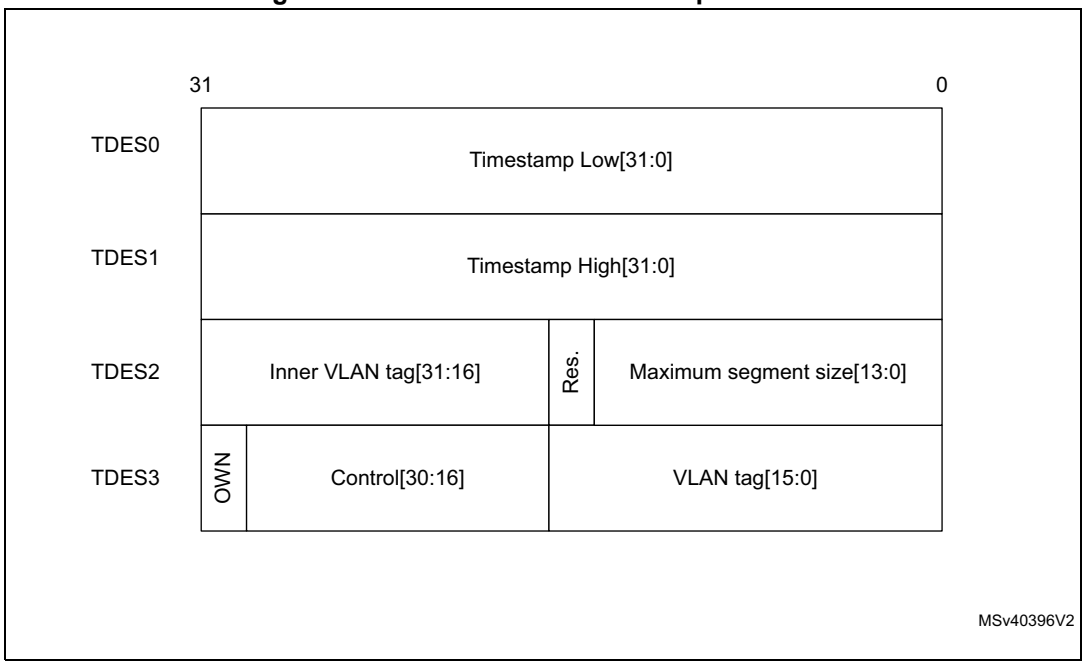
Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.

When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN tag control input.

Figure 738 shows the format for transmit context descriptors. *Table 547* to *Table 550* provide a detailed description of all transmit context descriptors.

Figure 738. Transmit context descriptor format



- TDES0 context descriptor (read format)

Table 547. TDES0 context descriptor

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)

Table 548. TDES1 context descriptor

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES2 context descriptor (read format)

Table 549. TDES2 context descriptor

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
15:14	Reserved	
13:0	MSS	**Maximum Segment Size** This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

- TDES3 context descriptor (read format)

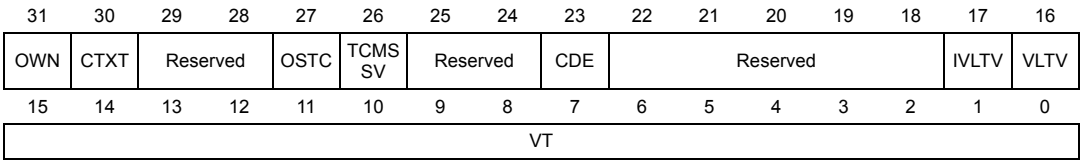


Table 550. TDES3 context descriptor

Bit	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit immediately after a read operation.
30	CTXT	**Context Type** This bit should be set to 1 for context descriptor.
29:28	Reserved	
27	OSTC	**One-Step Timestamp Correction Enable** When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
26	TCMSSV	**One-Step Timestamp Correction Input or MSS Valid** When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	Reserved	
23	CDE	**Context Descriptor Error** When this bit is set, it indicates that the context descriptor is incorrect. The DMA sets this bit during write-back while closing the context descriptor. The Context Descriptor errors can be: – Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. – All 1s. – CD, LD, and FD bits set to 1. *Note: When a Context Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel 0 status register (ETH_DMAC0SR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.* *This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported)*
22:20	Reserved	

Table 550. TDES3 context descriptor (continued)

Bit	Name	Description
19:18	IVTIR	Inner VLAN Tag Insert or Replace When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. This bitfield has the following values: 00: Do not add the inner VLAN tag. 01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. 10: Insert an inner VLAN tag with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. 11: Replace the inner VLAN tag in packets with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. This option should be used only with the VLAN frames.
17	IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.
16	VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the *VLAN inclusion register (ETH_MACVIR)* is reset.

48.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following receive descriptors are present:

- Normal descriptors with read and write-back formats
- Context descriptors

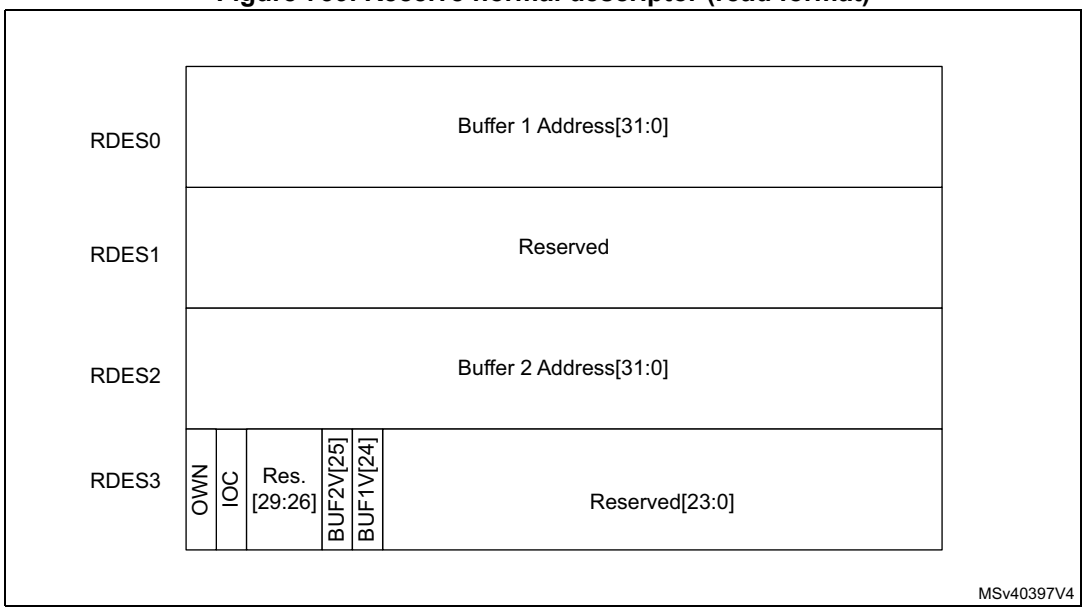
All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see *Figure 739: Receive normal descriptor (read format)* for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in *Figure 740: Receive normal descriptor (write-back format)*.

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA writes the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in *Figure 741: Receive context descriptor*.

Receive normal descriptor (read format)

Figure 739 shows the read format for receive normal descriptors. *Table 551* to *Table 554* provide a detailed description of all receive normal descriptors (read format).

Figure 739. Receive normal descriptor (read format)



Note: In the receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.

- RDES0 normal descriptor (read format)

Table 551. RDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. The application can program a byte-aligned address for this buffer, which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES2[1:0]=0 in case of 64-/128-bit configuration) as zero. However, the packet data is shifted by the actual offset as given in the buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

- RDES1 normal descriptor (read format)

Table 552. RDES1 normal descriptor (read format)

Bit	Name	Description
31:0	Reserved	Field reserved.

- RDES2 normal descriptor (read format)

Table 553. RDES2 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 Address Pointer** These bits indicate Buffer 2 physical address. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores RDES2[1:0]=0 and writes to the complete location.

- RDES3 normal descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IOC	Reserved				BUF2V	BUF1V	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

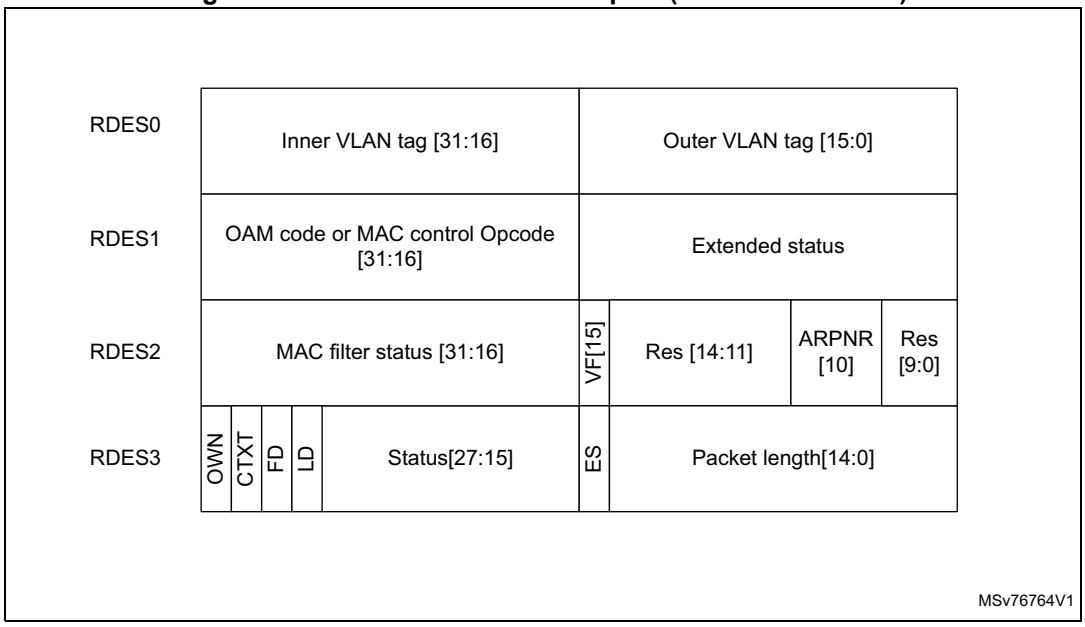
Table 554. RDES3 normal descriptor (read format)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full
30	IOC	**Interrupt Enabled on Completion** When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Reserved	
25	BUF2V	**Buffer 2 Address Valid** When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.
24	BUF1V	**Buffer 1 Address Valid** When set, this indicates to the DMA that the buffer 1 address specified in RDES0 is valid. The application must set this value if the address to which Buffer 1 address points in RDES0, can be used by the DMA to write received packet data.
23:0	Reserved	

Receive normal descriptor (write-back format)

Figure 740 shows the write-back format for receive normal descriptors. *Table 555* to *Table 558* provide a detailed description of all receive normal descriptors (write-back format).

Figure 740. Receive normal descriptor (write-back format)



- RDES0 normal descriptor (write-back format)

Table 555. RDES0 normal descriptor (write-back format)

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
15:0	OVT	**Outer VLAN Tag** This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

- RDES1 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPC															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD	TSA	PV	PFT	PMT				IPCE	IPCB	IPV6	IPV4	IPHE	PT		

Table 556. RDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:16	OPC	**OAM Subtype Code, or MAC Control Packet opcode** OAM Subtype Code If bits[18:16] of RDES3 are set to 111, this field contains the OAM subtype and code fields. MAC Control Packet opcode If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
15	TD	**Timestamp Dropped** This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.
14	TSA	**Timestamp Available** When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	**PTP Version** 1: Received PTP message in IEEE 1588 version 2 format 0: Received PTP message in IEEE 1588 version 1 format
12	PFT	**PTP Packet Type** This bit indicates that the PTP message is sent directly over Ethernet.
11:8	PMT	**PTP Message Type** These bits are encoded to give the type of the message received: 0000: No PTP message received 0001: SYNC (all clock types) 0010: Follow_Up (all clock types) 0011: Delay_Req (all clock types) 0100: Delay_Resp (all clock types) 0101: Pdelay_Req (in peer-to-peer transparent clock) 0110: Pdelay_Resp (in peer-to-peer transparent clock) 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000: Announce 1001: Management 1010: Signaling 1011–1110: Reserved 1111: PTP packet with Reserved message type
7	IPCE	**IP Payload Error** When this bit is set, it indicates either of the following: – The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. – The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. – The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.

Table 556. RDES1 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
6	IPCB	**IP Checksum Bypassed** This bit indicates that the checksum offload engine is bypassed.
5	IPV6	**IPv6 header Present** This bit indicates that an IPV6 header is detected.
4	IPV4	**IPV4 Header Present** This bit indicates that an IPV4 header is detected.
3	IPHE	**IP Header Error** – When this bit is set, it indicates either of the following: – The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. – The IP datagram version is not consistent with the Ethernet Type value. – Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.
2:0	PT	**Payload Type** These bits indicate the type of payload encapsulated in the IP datagram processed by the receive checksum offload engine (COE): 000: Unknown type or IP/AV payload not processed 001: UDP 010: TCP 011: ICMP 100: IGMP if IPV4 Header Present bit is set Others: reserved. If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

- RDES2 normal descriptor (write-back format)

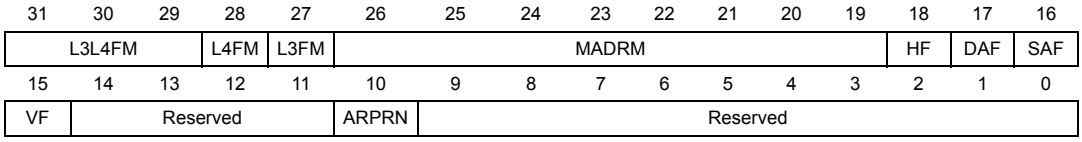


Table 557. RDES2 normal descriptor (write-back format)

Bit	Name	Description
31:29	L3L4FM	**Layer 3 and Layer 4 Filter Number Matched** These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: – 000: Filter 0 – 001: Filter 1 – 010: Filter 2 – 011: Filter 3 – 100: Filter 4 – 101: Filter 5 – 110: Filter 6 – 111: Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter.
28	L4FM	**Layer 4 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: – Layer 3 fields are not enabled and all enabled Layer 4 fields match – All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29].
27	L3FM	**Layer 3 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: – All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed – All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29].
26:19	MADRM	**MAC Address Match or Hash Value** When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register.
18	HF	**Hash Filter Status** When this bit is set, it indicates that the packet passed the MAC address Hash filter. its[26:19] indicate the Hash value.
17	DAF	**Destination Address Filter Fail** When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
16	SAF	**SA Address Filter Fail** When this bit is set, it indicates that the packet failed the SA Filter in the MAC.

Table 557. RDES2 normal descriptor (write-back format) (continued)

Bit	Name	Description
15	VF	**VLAN Filter Status** When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter.
14:11	Reserved	
10	ARPNR	**ARP Reply Not Generated** When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
9:0	Reserved	

- RDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	PL														

Table 558. RDES3 normal descriptor (write-back format)

Bit	Name	Description
31	OWN	**Own bit** 1: The DMA owns the descriptor. 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} possible values are: 00: Intermediate Descriptor 01: First Descriptor 10: Reserved 11: Descriptor error (due to all 1s) *Note: When a Descriptor error occurs, the receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel 0 status register (ETH_DMAC0SR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.*

Table 558. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
29	FD	**First Descriptor** When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. Refer to the CTXT bit description for details on how to use the CTXT bit and FD bit together.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	**Receive Status RDES2 Valid** When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	**Receive Status RDES1 Valid** When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	R**eceive Status RDES0 Valid** When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
24	CE	**CRC error** When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	**Giant packet** When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	**Receive watchdog timeout** When this bit is set, it indicates that the receive watchdog timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	**Overflow error** When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
20	RE	**Receive error** When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception.

Table 558. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
19	DE	**Dribble bit error** When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18:16	LT	**Length/type field** This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 000: The packet is a length packet 001: The packet is a type packet. 011: The packet is a ARP Request packet type 100: The packet is a type packet with VLAN Tag 101: The packet is a type packet with Double VLAN tag 110: The packet is a MAC Control packet type 111: The packet is a OAM packet type 010: Reserved
15	ES	**Error Summary** When this bit is set, it indicates the logical OR of the following bits: RDES3[19]: Dribble Error RDES3[20]: Receive Error RDES3[21]: Overflow Error RDES3[22]: Watchdog timeout RDES3[23]: Giant Packet RDES3[24]: CRC Error This field is valid only when the LD bit of RDES3 is set.
14:0	PL	**Packet Length** These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet.

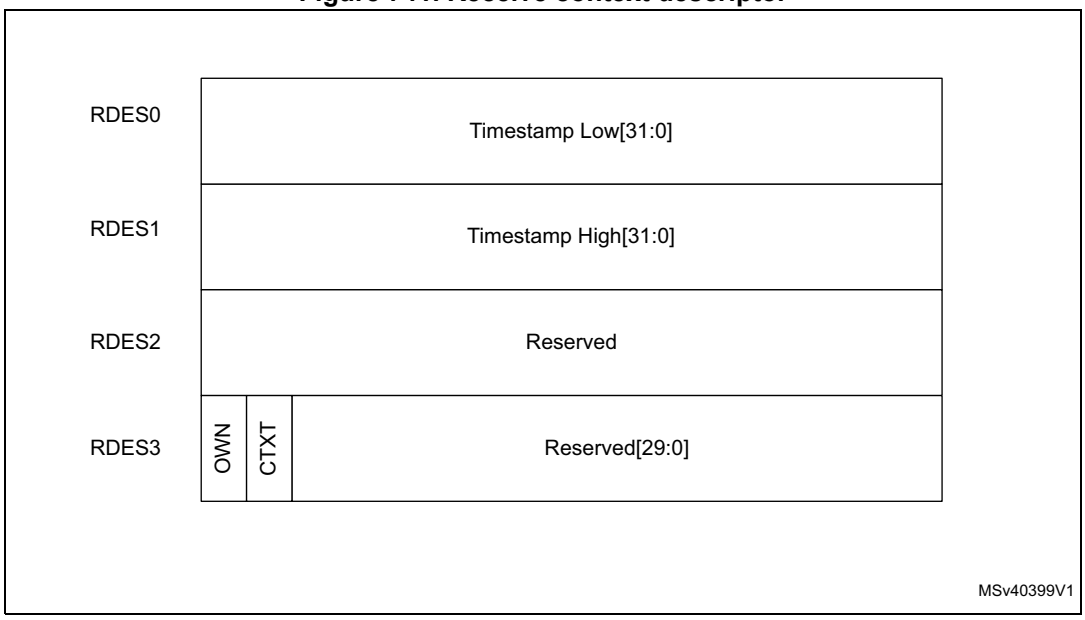
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

Figure 741 shows the format for receive context descriptors. *Table 559* to *Table 562* provide a detailed description of all receive context descriptors.

Figure 741. Receive context descriptor



- RDES0 context descriptor

Table 559. RDES0 context descriptor

Bit	Name	Description
31:0	RTSL	**Receive Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

- RDES1 context descriptor

Table 560. RDES1 context descriptor

Bit	Field	Description
31:0	RTSH	**Receive Packet Timestamp High** The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

- RDES2 context descriptor

Table 561. RDES2 context descriptor

Bit	Description
31:0	Reserved

- RDES3 context descriptor

Table 562. RDES3 context descriptor

Bit	Name	Description
31	OWN	**Own Bit** 1: The DMA owns the descriptor 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor. When CTXT and DE bits are used together, {CTXT, DE} possible values are: 00: Reserved 01: Reserved 10: Context Descriptor 11: Descriptor error When a Descriptor error occurs, the receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data The receive DMA sets the CDE bit in *Channel 0 status register (ETH_DMAC0SR)* but does not set the RI field even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.
29	DE	**Descriptor Error** Refer to the CTXT bit description for details on how to use the E bit along with CTXT bit.
28:0	Reserved	

48.11 Ethernet registers

48.11.1 Ethernet register maps

This section provides the following register maps:

- DMA registers (see *Section 48.11.2: Ethernet DMA registers*)
- MTL registers (see *Section 48.11.3: Ethernet MTL registers*)
- MAC registers including MMC registers (see *Section 48.11.4: Ethernet MAC and MMC registers*)

48.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000

Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PR[2:0]			TXPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA	SWR
	rw	rw	rw	rw										rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **INTM[1:0]**: Interrupt Mode

This field defines the interrupt mode of the Ethernet peripheral.
The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMAC0SR register changes depending on the INTM value (refer to *Table 538: Transfer complete interrupt behavior*).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PR[2:0]**: Priority ratio

These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set.
000: The priority ratio is 1:1
001: The priority ratio is 2:1
010: The priority ratio is 3:1
011: The priority ratio is 4:1
100: The priority ratio is 5:1
101: The priority ratio is 6:1
110: The priority ratio is 7:1
111: The priority ratio is 8:1

Bit 11 **TXPR**: Transmit priority

When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.

Bits 10:2 Reserved, must be kept at reset value.

Bit 1 **DA**: DMA Tx or Rx Arbitration Scheme

This bit specifies the arbitration scheme between the transmit and receive paths of all channels:

0: Weighted Round-Robin with Rx:Tx or Tx:Rx

The priority between the paths is according to the priority specified in Bits[14:12] and the priority weight is specified in the TXPR bit.

1: Fixed priority

The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path.

Bit 0 **SWR**: Software Reset

When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all clock domains. Before reprogramming any register, a value of zero should be read in this bit.

Note: The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.

System bus mode register (ETH_DMASBMR)

Address offset: 0x1004

Reset value: 0x0000 0000

The System bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB	MB	Res.	AAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FB
r	rw		rw												rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RB**: Rebuild INCRx Burst

When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.

Bit 14 **MB**: Mixed Burst

When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **AAL**: Address-Aligned Beats

When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **FB**: Fixed Burst Length

When this bit is set to 1, the AHB master initiates burst transfers of specified length (INCRx or SINGLE).

When this bit is set to 0, the AHB master initiates transfers of unspecified length (INCR) or SINGLE transfers.

Interrupt status register (ETH_DMAISR)

Address offset: 0x1008

Reset value: 0x0000 0000

The application reads this interrupt status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC0IS
															r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **MACIS**: MAC interrupt status

This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

Bit 16 **MTLIS**: MTL interrupt status

This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **DC0IS**: DMA channel interrupt status

This bit indicates an interrupt event in DMA channel. To reset this bit to 0, the software must read the corresponding register in DMA channel to get the exact cause of the interrupt and clear its source.

Debug status register (ETH_DMADSR)

Address offset: 0x100C

Reset value: 0x0000 0000

The Debug status register gives the receive and transmit process status for DMA channel for debugging purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXWHSTS
r	r	r	r	r	r	r	r								r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **TPS0[3:0]**: DMA channel Transmit Process State

This field indicates the Tx DMA FSM state for channel:
000: Stopped (Reset or Stop Transmit Command issued)
001: Running (Fetching Tx Transfer Descriptor)
010: Running (Waiting for status)
011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
100: Timestamp write state
101: Reserved for future use
110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
111: Running (Closing Tx Descriptor)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPS0[3:0]**: DMA channel receive Process State

This field indicates the Rx DMA FSM state for channel:
000: Stopped (Reset or Stop receive Command issued)
001: Running (Fetching Rx Transfer Descriptor)
010: Reserved for future use
011: Running (Waiting for Rx packet)
100: Suspended (Rx Descriptor Unavailable)
101: Running (Closing the Rx Descriptor)
110: Timestamp write state
111: Running (Transferring the received packet data from the Rx buffer to the system memory)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **AXWHSTS**: AHB Master Write channel

When high, this bit indicates that the write channel of the AHB master FMSs are in non-idle state.

Channel 0 control register (ETH_DMAC0CR)

Address offset: 0x1100

Reset value: 0x0000 0000

The DMA channel 0 control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	DRO	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8
					rw						rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MSS[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **DRO**: Descriptor refetch when OWN bit is 0

When this bit is set and the OWN bit of a descriptor is cleared, an additional attempt is made by the DMA engine to fetch the descriptor. If the OWN bit is still cleared, the transmit/receive buffer unavailable (TBU/RBU) interrupt is generated and the DMA goes into Suspend state until the tail pointer register is updated by the application.

When this bit is cleared, the DMA engine goes into Suspend mode as soon as it finds OWN bit cleared.

Bits 25:21 Reserved, must be kept at reset value.

Bits 20:18 **DSL[2:0]**: Descriptor Skip Length

This bit specifies the 32-bit word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.

When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PBLX8**: 8xPBL mode

When this bit is set, the PBL value programmed in Bits[21:16] in *Channel 0 transmit control register (ETH_DMAC0TXCR)* and in Bits[21:16] in *Channel 0 receive control register (ETH_DMAC0RXCR)* is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **MSS[13:0]**: Maximum Segment Size

This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of *Channel 0 transmit control register (ETH_DMAC0TXCR)* is set.

The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

Channel 0 transmit control register (ETH_DMAC0TXCR)

Address offset: 0x1104

Reset value: 0x0000 0000

The DMA channel 0 transmit control register controls the Tx features such as PBL, TCP segmentation, and Tx channel weights.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPBL[5:0]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	Res.	Res.	Res.	ST
			rw								rw				rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **TXPBL[5:0]**: Transmit Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in ETH_DMAC0CR.

b) Set the TXPBL[5:0].

Note: The maximum value of TXPBL must be less than or equal to half the Tx Queue size (TQS field of *Tx queue 0 operating mode register (ETH_MTLTXQ0OMR)*) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. The total locations in Tx Queue of size 2048 bytes is 512, TXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TSE**: TCP Segmentation Enabled

When this bit is set, the DMA performs the TCP segmentation for packets in channel x. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **OSF**: Operate on Second Packet

When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ST**: Start or Stop Transmission Command

When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the base address of the Transmit list set by the ETH_DMAC0TXDLAR register.

– The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMAC0SR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMAC0TXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMAC0TXDLAR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

Channel 0 receive control register (ETH_DMAC0RXCR)

Address offset: 0x1108

Reset value: 0x0000 0000

The DMA channel 0 receive control register controls the Rx features such as PBL, buffer size, and extended status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBL[5:0]					
rw										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RBSZ[13:0]														SR
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RPF**: DMA Rx Channel 0 Packet Flush

When this bit is set to 1, the Ethernet peripheral automatically flushes the packet from the Rx queues destined to DMA Rx channel 0 when the DMA Rx channel 0 is stopped after a system bus error has occurred. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is restarted are routed to the RxDMA. The flushing happens on the Read side of the Rx queue.

When this bit is set to 0 the Ethernet peripheral does not flush the packet in the Rx queue destined to DMA Rx channel 0 after the DMA is stopped due to a system bus error.

This might cause head-of-line blocking in the corresponding RxQueue.

Note: The stopping of packet flow from a Rx DMA channel to the application by setting RPF works only when there is one-to-one mapping of the Rx queues to the Rx DMA channels. In Dynamic mapping mode, setting the RPF bit in the Channel 0 receive control register (ETH_DMAC0RXCR) might flush packets from unintended Rx queues, which are destined to the stopped Rx DMA channel.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:16 **RXPBL[5:0]**: Receive Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in the ETH_DMAC0CR.

b) Set the RXPBL[5:0].

Note: The maximum value of RXPBL must be less than or equal to half the Rx Queue size (RQS field of Rx queue 0 operating mode register (ETH_MTLRXQ0OMR)) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the MTL Rx Queue Controller is transferring data to MAC.The total locations in Rx Queue of size 2048 bytes is 512, RXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 Reserved, must be kept at reset value.

Bits 14:1 **RBSZ[13:0]**: Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes.

Note: The buffer size must be a multiple of 4. This is required even if the value of the buffer address pointer is not aligned to the bus width. If the buffer size is not a multiple of 4, it may result in an undefined behavior.

The LSB bits (1:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0 **SR**: Start or Stop receive

When this bit is set, the DMA tries to acquire the descriptor from the receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the address set by the *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*.

– The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMAC0SR is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*, the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

Channel 0 Tx descriptor list address register (ETH_DMAC0TXDLAR)

Address offset: 0x1114

Reset value: 0x0000 0000

Channel 0 Tx descriptor list address register points the DMA to the start of the transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Writing to this register is permitted only when the Tx DMA has stopped, that is, the ST bit is cleared in ETH_DMAC0TXCR register. When stopped, this register can be written with a

new descriptor list address. When the ST bit is set, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is cleared, the DMA takes the descriptor address where it was stopped earlier.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **TDESLA[31:0]**: Start of Transmit List

This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)

Address offset: 0x111C

Reset value: 0x0000 0000

The channel 0 Rx descriptor list address register points the DMA to the start of receive descriptor list.

This register points to the start of the receive descriptor list. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in the ETH_DMAC0RXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **RDESLA[31:0]**: Start of receive list

This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel 0 Tx descriptor tail pointer register (ETH_DMAC0TXDTPR)

Address offset: 0x1120

Reset value: 0x0000 0000

The channel 0 Tx descriptor tail pointer register points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **TDT[31:0]**: transmit descriptor tail pointer

This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)

Address offset: 0x1128

Reset value: 0x0000 0000

The channel 0 Rx descriptor tail pointer points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **RDT[31:0]**: Receive Descriptor Tail Pointer

This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

Channel 0 Tx descriptor ring length register (ETH_DMAC0TXRLR)

Address offset: 0x112C

Reset value: 0x0000 0000

The channel 0 Tx descriptor ring length register contains the length of the transmit descriptor ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TDRL[9:0]**: Transmit descriptor ring length

This 10-bit field sets the maximum number of Tx descriptors in the circular descriptor ring.
TDRL[9:0] corresponds to the number of Tx descriptors minus one.
It can be programmed to any value up to 0x3FF (1024 descriptors).
A minimum ring descriptor length of four is recommended.
For example, program 0x9 to configure 10 descriptors.

Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)

Address offset: 0x1130

Reset value: 0x0000 0000

The channel 0 Rx descriptor ring length register contains the length of the receive descriptor circular ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **ARBS[7:0]**: Alternate receive buffer size

Indicates size in bytes for Buffer 1 when ARBS[7:0] is programmed to a non-zero value.
When ARBS[7:0] = 0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ[13:0] field of *Channel 0 receive control register (ETH_DMAC0RXCR)*.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **RDRL[9:0]**: Receive Descriptor Ring Length

This 10-bit field sets the maximum number of Rx descriptors in the circular descriptor ring.
RDRL[9:0] corresponds to the number of Rx descriptors minus one. It can be programmed to any value up to 0x3FF (1024 descriptors).
For example, program 0x9 to configure 10 descriptors.

Channel 0 interrupt enable register (ETH_DMAC0IER)

Address offset: 0x1134

Reset value: 0x0000 0000

The channel 0 interrupt enable register enables the interrupts reported by the Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NIE**: Normal Interrupt Summary Enable

When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the *Channel 0 status register (ETH_DMAC0SR)*:
Bit 0: Transmit interrupt
Bit 2: Transmit buffer unavailable
Bit 6: Receive interrupt
Bit 11: Early receive interrupt
When this bit is reset, the normal interrupt summary is disabled.

Bit 14 **AIE**: Abnormal Interrupt Summary Enable

When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the *Channel 0 status register (ETH_DMAC0SR)*:
Bit 1: Transmit process stopped
Bit 7: Rx buffer unavailable
Bit 8: Receive process stopped
Bit 9: Receive watchdog timeout
Bit 10: Early transmit interrupt
Bit 12: Fatal bus error
Bit 13: Context descriptor error
When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 **CDEE**: Context descriptor error enable

When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 **FBEE**: Fatal Bus Error enable

When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 **ERIE**: Early receive interrupt enable

When this bit is set along with the NIE bit, the Early receive interrupt is enabled. When this bit is reset, the Early receive interrupt is disabled.

Bit 10 **ETIE**: Early Transmit Interrupt enable

When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 **RWTE**: Receive watchdog timeout enable

When this bit is set along with the AIE bit, the receive watchdog timeout interrupt is enabled. When this bit is reset, the receive watchdog timeout interrupt is disabled.

Bit 8 **RSE**: Receive Stopped Enable

When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.

Bit 7 **RBUE**: Receive Buffer Unavailable enable

When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.

Bit 6 **RIE**: Receive Interrupt enable

When this bit is set along with the NIE bit, the receive interrupt is enabled. When this bit is reset, the receive interrupt is disabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBUE**: Transmit Buffer Unavailable enable

When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

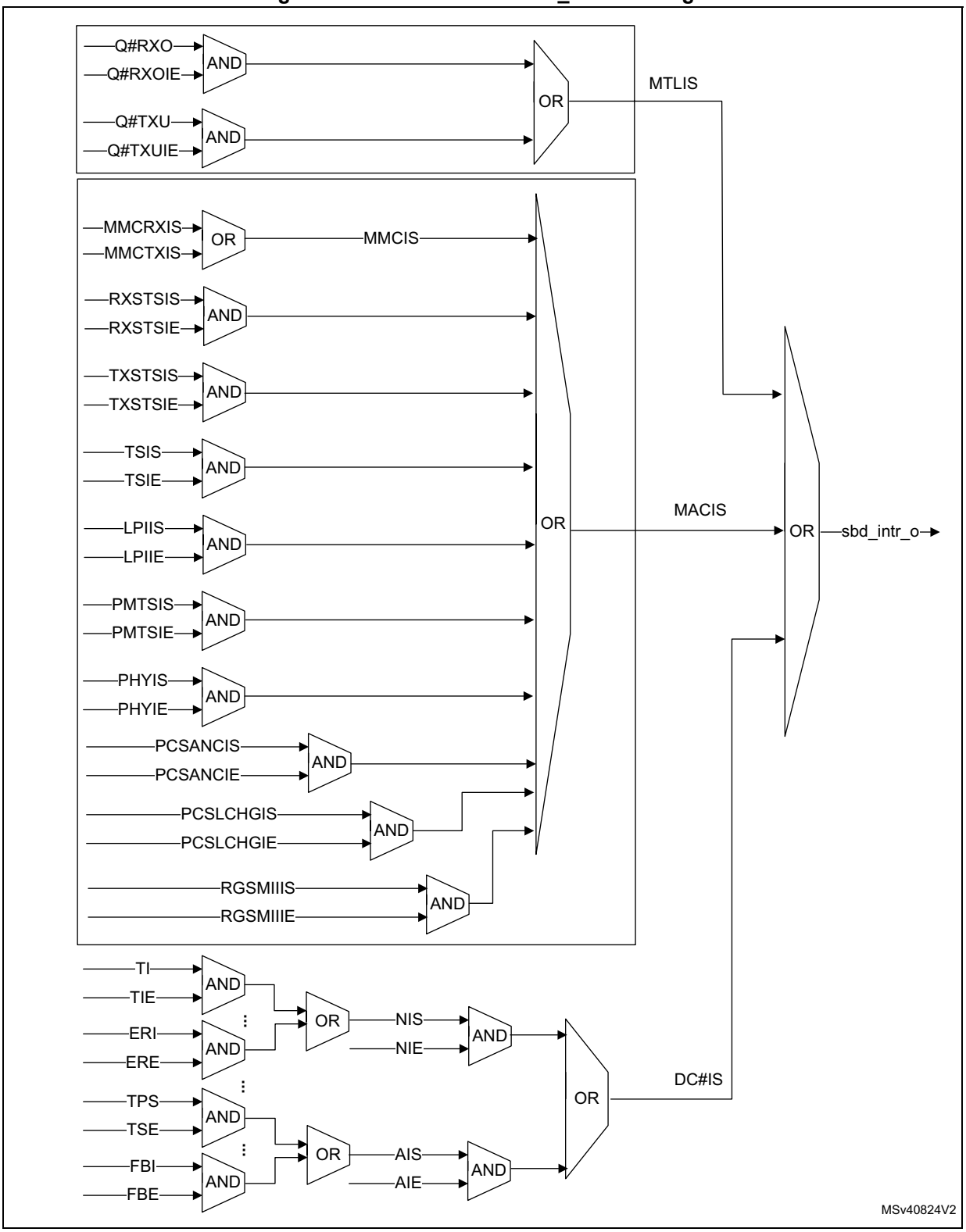
Bit 1 **TXSE**: Transmit Stopped enable

When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0 **TIE**: Transmit Interrupt enable

When this bit is set along with the NIE bit, the transmit interrupt is enabled. When this bit is reset, the transmit interrupt is disabled.

Figure 742. Generation of ETH_DMAISR flags



Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)

Address offset: 0x1138

Reset value: 0x0000 0000

The channel 0 receive interrupt watchdog timer register indicates the watchdog timeout for receive interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the *Channel 0 status register (ETH_DMAC0SR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSEL	Res.	RBCT[9:0]										ITCU[1:0]		RWTU[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ITW[4:0]					RWT[7:0]							
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **PSEL**: Packet Count Interrupt Select

When this bit is set, RBCT[9:0] indicates the number of packet threshold at which the receive interrupt (RI) is activated and the watchdog timer reset.

When this bit is cleared, RBCT[9:0] indicates the number of data byte threshold at which the receive interrupt (RI) is activated and the watchdog timer reset.

1: RBCT[9:0] is used to generate interrupt based on the number of packets received.

0: RBCT[9:0] is used to generate interrupt based on the number of bytes received.

Bit 30 Reserved, must be kept at reset value.

Bits 29:20 **RBCT[9:0]**: Receive Byte Count Threshold

This bitfield sets the limit in number of bytes or number of RX packets (based on PSEL) transferred without the assertion of receive interrupt. Once this limit is crossed, the receive interrupt is asserted and the watchdog timer (corresponding to RWT[7:0] is also reset.

This function is valid only when set to a nonzero value. When RBCT[9:0] = 0x00, RX packet/byte counter is inactive.

When PSEL = 0, this bitfield is multiplied by 1024 to get the actual number of byte threshold. When PSEL = 1, RBCT[9:0) value is directly used. Thus, the maximum limit is 511 packets or 511 x 1024 bytes, respectively.

Bits 19:18 **ITCU[1:0]**: Rx Idle Timer Count Units

This bitfield indicates the number of system clock cycles corresponding to one unit in the ITW[4:0] bitfield.

0b00: 256

0b01: 512

0b10: 1024

0b22: 2048

For example, when ITW[4:0] = 0b01010 and ITCU[1:0] = 0b01, the Rx Idle timer is set for 10 × 512 = 1024 system clock cycles.

Bits 17:16 **RWTU[1:0]**: Receive Interrupt Watchdog Timer Count Units

This bitfield indicates the number of system clock cycles corresponding to one unit in RWT[7:0] bitfield.

00: 256

01: 512

10: 1024

11: 2048

For example, when RWT[7:0] = 2 and RWTU[1:0] = 1, the watchdog timer is set for 2 × 512 = 1024 system clock cycles.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **ITW[4:0]**: Rx Idle Timer Window

When programmed with a nonzero value, this bitfield enables the Rx Idle timer operation. The timer window is expressed in number of system clock cycles when this bitfield is multiplied by the ITCU unit.

Bits 7:0 **RWT[7:0]**: Receive Interrupt Watchdog Timer Count

This bitfield indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.

The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMAC0SR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].

When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

Channel 0 current application transmit descriptor register (ETH_DMAC0CATXDR)

Address offset: 0x1144

Reset value: 0x0000 0000

The channel 0 current application transmit descriptor register points to the current transmit descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTDESAPTR[31:0]**: Application transmit descriptor address pointer

The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel 0 current application receive descriptor register (ETH_DMAC0CARXDR)

Address offset: 0x114C

Reset value: 0x0000 0000

The channel 0 current application receive descriptor register points to the current receive descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRDESAPTR[31:0]**: Application receive descriptor address pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel 0 current application transmit buffer register (ETH_DMAC0CATXBR)

Address offset: 0x1154

Reset value: 0x0000 0000

The channel 0 current application transmit buffer address register points to the current Tx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTBUFAPTR[31:0]**: Application Transmit Buffer Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel 0 current application receive buffer register (ETH_DMAC0CARXBR)

Address offset: 0x115C

Reset value: 0x0000 0000

The channel 0 current application receive buffer address register points to the current Rx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRBUFAPTR[31:0]**: Application receive buffer address pointer

The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel 0 status register (ETH_DMAC0SR)

Address offset: 0x1160

Reset value: 0x0000 0000

The software driver (application) reads the status register during interrupt service routine or polling to determine the status of the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]		
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rw	rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **REB[2:0]**: Rx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AHB interface.

Bit [2]: Error during data transfer by Rx DMA when 1, no rror during data transfer by Rx DMA when 0.

Bit[1]: Error during descriptor access when 1, error during data buffer access when 0

Bit[0]: Error during read transfer when 1, error during write transfer when 0

This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bits 18:16 **TEB[2:0]**: Tx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AHB interface.

Bit[2]: Error during data transfer by Tx DMA when 1, no error during data transfer by Tx DMA when 0

Bit[1]: Error during descriptor access when 1, error during data buffer access when 0

Bit[0]: Error during read transfer when 1, Error during write transfer when 0

This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15 **NIS**: Normal Interrupt Summary

Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMAC0IER register:
Bit 0: Transmit interrupt
Bit 2: Transmit buffer unavailable
Bit 6: Receive interrupt
Bit 11: Early receive interrupt
Only unmasked bits (interrupts for which interrupt enable is set in ETH_DMAC0IER register) affect the Normal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14 **AIS**: Abnormal Interrupt Summary

Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMAC0IER register:
Bit 1: Transmit process stopped
Bit 7: Receive buffer unavailable
Bit 8: Receive process stopped
Bit 10: Early transmit interrupt
Bit 12: Fatal bus error
Bit 13: Context descriptor error
Only unmasked bits affect the Abnormal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13 **CDE**: Context descriptor error

This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one‘s descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.

Bit 12 **FBE**: Fatal bus error

This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.

Bit 11 **ERI**: Early receive interrupt

When set, this bit indicates that the RxDMA has completed the transfer of packet data to the memory. The RI bit of this register automatically clears this bit.

Bit 10 **ETI**: Early transmit interrupt

When set, this bit indicates that the TxDMA has completed the transfer of packet data to the MTL TXFIFO memory.

Bit 9 **RWT**: Receive watchdog timeout

This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.

Bit 8 **RPS**: Receive process stopped

This bit is asserted when the Rx process enters the Stopped state.

Bit 7 **RBU**: Receive buffer unavailable

This bit indicates that the application owns the next descriptor in the receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the receive descriptor tail pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor.

Bit 6 **RI**: Receive interrupt

This bit indicates that the packet reception is complete or Rx Idle timer/Rx watchdog timer is timed out. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

The reception remains in the Running state.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBU**: Transmit buffer unavailable

This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSi field of the *Debug status register (ETH_DMADSR)* register explains the Transmit Process state transitions.

To resume processing the Transmit descriptors, the application should do the following:

1. Change the ownership of the descriptor by setting Bit 31 of TDES3.
2. Issue a Transmit Poll Demand command.

For ring mode, the application should advance the transmit descriptor tail pointer register of a channel.

Bit 1 **TPS**: Transmit process stopped

This bit is set when the transmission is stopped.

Bit 0 **TI**: Transmit interrupt

This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

Channel 0 missed frame count register (ETH_DMAC0MFCR)

Address offset: 0x1164

Reset value: 0x0000 0000

This register has the number of packet counter that got dropped by the DMA either due to bus error or due to programing RPF field in *Channel 0 receive control register (ETH_DMAC0RXCR)* register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
rc_r					rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MFCO**: Overflow status of the MFC Counter

When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MFC[10:0]**: Dropped packet counters

This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in *Channel 0 receive control register (ETH_DMAC0RXCR)*. The counter gets cleared when this register is read.

Ethernet DMA register map and reset values

Table 563. ETH_DMA common register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000	ETH_DMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]		Res.	PR[2:0]			TXPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.			DA	SWR
	Reset value															0	0		0	0	0	0										0	0
0x1004	ETH_DMASBMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RB	MB	Res.	AAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FB
	Reset value																	0	0		0												0
0x1008	ETH_DMAISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC0IS
	Reset value															0	0																0
0x100C	ETH_DMADSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS0[3:0]				RPS0[3:0]							Res.	Res.	Res.	Res.	AXWHSTS
	Reset value																	0	0	0	0	0	0	0	0								0

Table 564. ETH_DMA_CH0 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1100	ETH_DMAC0CR	Res.	Res.	Res.	Res.	Res.	DRO	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value						0						0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1104	ETH_DMAC0TXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPBL[5:0]						Res.	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	Res.			ST
	Reset value											0	0	0	0	0	0				0								0				0
0x1108	ETH_DMAC0RXCR	RPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBL[5:0]						Res.	RBSZ[13:0]											Res.	Res.	Res.	SR
	Reset value	0										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0				0
0x110C - 0x1110	Reserved																																
0x1114	ETH_DMAC0TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1118	Reserved																																

Table 564. ETH_DMA_CH0 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x111C	ETH_DMAC0RXDLAR	RDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1120	ETH_DMAC0TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1124	Reserved																																
0x1128	ETH_DMAC0RXDTPR	RDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x112C	ETH_DMAC0TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1130	ETH_DMAC0RXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value									0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x1134	ETH_DMAC0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1138	ETH_DMAC0RXIWTR	PSEL	Res.	RBCT[9:0]										ITCU[1:0]		RWTU[1:0]		Res.	Res.	Res.	ITW[4:0]						RWT[7:0]						
	Reset value	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x113C-0x1140	Reserved																																
0x1144	ETH_DMAC0CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1148	Reserved																																
0x114C	ETH_DMAC0CARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1150	Reserved																																
0x1154	ETH_DMAC0CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1158	Reserved																																
0x115C	ETH_DMAC0CARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1160	ETH_DMAC0SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0

Table 564. ETH_DMA_CH0 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1164	ETH_DMAC0MFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

48.11.3 Ethernet MTL registers

Operating mode register (ETH_MTLOMR)

Address offset: 0x0C00

Reset value: 0x0000 0000

The operating mode register establishes the transmit and receive operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CNT CLR	CNT PRST	Res.	Res.	Res.	Res.	Res.	Res.	DTX STS	Res.
						rw	rw							rw	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CNTCLR**: Counters reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRST bit, CNTPRST has precedence.

Bit 8 **CNTPRST**: Counters preset

When this bit is set:

- *Tx queue 0 underflow register (ETH_MTLTXQ0UR)* is initialized/preset to 0x7F0.
- Missed Packet and Overflow Packet counters in *Rx queue 0 missed packet and overflow counter register (ETH_MTLRXQ0MPOCR)* is initialized/preset to 0x7F0

This bit is cleared automatically.

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 Reserved, must be kept at reset value.

Bit 1 **DTXSTS**: Drop transmit status

When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

Bit 0 Reserved, must be kept at reset value.

Interrupt status register (ETH_MTLISR)

Address offset: 0x0C20

Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q0IS
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **Q0IS**: Queue 0 interrupt status

This bit indicates that an interrupt has been generated by Queue 0. To reset this bit, read ETH_MTLQ0ICSR register to identify the interrupt cause and clear the source.

Tx queue 0 operating mode register (ETH_MTLTXQ0OMR)

Address offset: 0x0D00

Reset value: 0x0007 0008

The queue 0 transmit operating mode register establishes the transmit queue operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
									rw	rw	rw	r	r	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **TQS[2:0]**: Transmit queue size

This field indicates the size of the allocated transmit queues in blocks of 256 bytes. Queue size range from 256 bytes (TQS=0b000) to 2048 bytes (TQS=0b111).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **TTC[2:0]**: Transmit threshold control

These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.

000: 32
001: 64
010: 96
011: 128
100: 192
101: 256
110: 384
111: 512

Bits 3:2 **TXQEN[1:0]**: Transmit queue enable

This field is used to enable/disable the transmit queue 0.

00: Not enabled
10: Enabled
Others: Reserved, must not be used.

Bit 1 **TSF**: Transmit store and forward

When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped.

Bit 0 **FTQ**: Flush transmit queue

When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the *Tx queue 0 operating mode register (ETH_MTLTXQ0OMR)* register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (eth_tx_clk) should be active.

Tx queue 0 underflow register (ETH_MTLTXQ0UR)

Address offset: 0x0D04

Reset value: 0x0000 0000

The queue 0 underflow counter register contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UFCNT OVF	UFFRMCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **UFCNTOVF**: Overflow Bit for Underflow packet counter

This bit is set every time the Tx queue Underflow packet counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened.

Bits 10:0 **UFFRMCNT[10:0]**: Underflow packet counter

This field indicates the number of packets aborted by the controller because of Tx queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read.

Tx queue 0 debug register (ETH_MTLTXQ0DR)

Address offset: 0x0D08

Reset value: 0x0000 0000

The queue 0 transmit debug register gives the debug status of various blocks related to the transmit queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]		
									r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTS FSTS	TXQ STS	TWC STS	TRCSTS[1:0]		TXQPA USED
										r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **STXSTSF[2:0]**: Number of status words in Tx status FIFO of queue

This field indicates the current number of status in the Tx Status FIFO of this queue.
When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PTXQ[2:0]**: Number of packets in the transmit queue

This field indicates the current number of packets in the Tx queue.
When the DTXSTS bit of *Operating mode register (ETH_MTLOMR)* register is set to 1, this field does not reflect the number of packets in the transmit queue.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TXSTSFSTS**: MTL Tx status FIFO Full status

When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.

Bit 4 **TXQSTS**: MTL Tx queue not empty status

When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission.

Bit 3 **TWCSTS**: MTL Tx queue write controller status

When high, this bit indicates that the MTL Tx queue write controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx queue read controller status
This field indicates the state of the Tx queue read controller:
00: Idle state
01: Read state (transferring data to the MAC transmitter)
10: Waiting for pending Tx Status from the MAC transmitter
11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit queue in pause
When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the Full-duplex only mode) because of the following:
– Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
– Reception of 802.3x Pause packet when PFC is disabled

Queue 0 interrupt control status register (ETH_MTLQ0ICSR)

Address offset: 0x0D2C

Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue 0 interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS
							rw								rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUNFIS
							rw								rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive queue overflow interrupt enable
When this bit is set, the receive queue overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **RXOVFIS**: Receive queue overflow interrupt status
This bit indicates that the receive queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TXUIE**: Transmit queue underflow interrupt enable
When this bit is set, the transmit queue underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **TXUNFIS**: Transmit queue underflow interrupt status

This bit indicates that the transmit queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.

Rx queue 0 operating mode register (ETH_MTLRXQ0OMR)

Address offset: 0x0D30

Reset value: 0x0070 0000

The queue 0 receive operating mode register establishes the receive queue operating modes and command.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[2:0]			Res.	Res.	Res.	Res.
									r	r	r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_ TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
									rw	rw	rw	rw		rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **RQS[2:0]**: Receive Queue Size

This field is read-only and the configured Rx FIFO size in blocks of 256 bytes is reflected in the reset value. The size of the Queue is (RQS + 1) × 256 bytes.

Bits 19:7 Reserved, must be kept at reset value.

Bit 6 **DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets

When this bit is set, the MAC does not drop the packets which only have the errors detected by the receive checksum offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.

When this bit is reset, all error packets are dropped if the FEP bit is reset.

Bit 5 **RSF**: Receive Queue Store and Forward

When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Bit 4 **FEP**: Forward Error Packets

When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped.

When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.

Bit 3 **FUP**: Forward Undersized Good Packets

When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTC[1:0]**: Receive Queue Threshold Control

These bits control the threshold level of the MTL Rx queue (in bytes):
00: 64
01: 32
10: 96
11: 128
The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred.
This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

Rx queue 0 missed packet and overflow counter register (ETH_MTLRXQ0MPOCR)

Address offset: 0x0D34

Reset value: 0x0000 0000

The queue 0 missed packet and overflow counter registers contain the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTOVF**: Missed packet counter Overflow Bit

When set, this bit indicates that the Rx Queue Missed packet counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed packet counter

This field indicates the number of packets missed by the Ethernet peripheral because the application requested to flush the packets for this queue. This counter is reset when this register is read.
This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **OVFCNTOVF**: Overflow Counter Overflow Bit
When set, this bit indicates that the Rx Queue Overflow packet counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow packet counter
This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.

Rx queue 0 debug register (ETH_MTLRXQ0DR)

Address offset: 0x0D38

Reset value: 0x0000 0000

The queue 0 receive debug register gives the debug status of various blocks related to the receive queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PRXQ[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
										r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PRXQ[13:0]**: Number of Packets in Receive Queue
This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256 Kbytes/16 bytes = 16K packets, that is, Max_Queue_Size/Min_Packet_Size.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **RXQSTS[1:0]**: MTL Rx Queue Fill-Level Status
This field gives the status of the fill-level of the Rx queue:
00: Rx queue empty
01: Rx queue fill-level below flow-control deactivate threshold
10: Rx queue fill-level above flow-control activate threshold
11: Rx queue full

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **RRCSTS[1:0]**: MTL Rx Queue Read Controller State
This field gives the state of the Rx queue Read controller:
00: Idle state
01: Reading packet data
10: Reading packet status (or timestamp)
11: Flushing the packet data and status

Bit 0 **RWCSTS**: MTL Rx Queue Write Controller Active Status
When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.

Ethernet MTL register map and reset values

Table 565. ETH_MTL register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C00	ETH_MTLOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTCLR	CNTPRST	Res.	Res.					DTXSTS	Res.
	Reset value																							0	0							0	
0x0C04 - 0x0C1C	Reserved																																
0x0C20	ETH_MTLISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q0IS
	Reset value																																0
0x0C24 - 0x0CFC	Reserved																																
0x0C40	Reserved																																
0x0D00	ETH_ MTLTXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value														1	1	1										0	0	0	1	0	0	0
0x0D04	ETH_ MTLTXQ0UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D08	ETH_ MTLTXQ0DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D0C - 0x0D28	Reserved																																
0x0D2C	ETH_MTLQ0ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUNFIS
	Reset value								0								0								0								0
0x0D30	ETH_ MTLRXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value										1	1	1														0	0	0	0		0	0
0x0D34	ETH_ MTLRXQMPOCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 565. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D38	ETH_ MTLRXQ0DR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0		0	0	0
0xD3C-0xD58	Reserved																																

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

48.11.4 Ethernet MAC and MMC registers

Operating mode configuration register (ETH_MACCR)

Address offset: 0x0000

Reset value: 0x0000 8000

The MAC Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	Res.	JD	JE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **ARPEN**: ARP Offload Enable

When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus.

When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.

Bits 30:28 **SARC[2:0]**: Source Address Insertion or Replacement Control

This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:

010:the MAC inserts the content of the MAC Address 0 registers (*MAC address 0 high register (ETH_MACA0HR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

011: the MAC replaces the content of the MAC Address 0 registers (*MAC address 0 high register (ETH_MACA0HR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

110: the MAC inserts the content of the MAC Address 1 registers (*MAC address x high register (ETH_MACAxHR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets

111: the MAC replaces the content of the MAC Address 1 registers (*MAC address x high register (ETH_MACAxHR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

Others: Reserved, must not be used.

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bit 27 **IPC**: Checksum Offload

When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.

The Layer 3 and Layer 4 Packet Filter feature automatically selects the IPC Full Checksum Offload Engine on the Receive side. When this feature is enabled, you must set the IPC bit.

Bits 26:24 **IPG[2:0]**: Inter-Packet Gap

These bits control the minimum IPG between packets during transmission.

000: 96 bit times

001: 88 bit times

010: 80 bit times

...

111: 40 bit times

This range of minimum IPG is valid in Full-duplex mode.

In the Half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.

When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.

The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23 **GPSLCE**: Giant Packet Size Limit Control Enable

When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.

When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).

The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22 **S2KP**: IEEE 802.3as Support for 2K Packets

When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.

When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 566: Giant Packet Status based on S2KP and JE Bits*.

Note: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21 **CST**: CRC stripping for Type packets

When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.

Note: For information about how the settings of the ACS bit and this bit impact the packet length, see Table 567: Packet Length based on the CST and ACS bits.

Bit 20 **ACS**: Automatic Pad or CRC Stripping

When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.

When this bit is reset, the MAC passes all incoming packets to the application, without any modification.

Note: For information about how the settings of CST bit and this bit impact the packet length, see Table 567: Packet Length based on the CST and ACS bits.

Bit 19 **WD**: Watchdog Disable

When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.

When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **JD**: Jabber Disable

When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.

When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.

Bit 16 **JE**: Jumbo Packet Enable

When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.

For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 566: Giant Packet Status based on S2KP and JE Bits*.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **FES**: MAC Speed

This bit selects the speed in the 10/100 Mbit/s mode:

0: 10 Mbit/s

1: 100 Mbit/s

Bit 13 **DM**: Duplex Mode

When this bit is set, the MAC operates in the Full-duplex mode in which it can transmit and receive simultaneously.

Bit 12 **LM**: Loopback Mode

When this bit is set, the MAC operates in the loopback mode at MII. The MII Rx clock input (eth_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Bit 11 **ECRSFD**: Enable Carrier Sense Before Transmission in Full-duplex mode

When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the Full-duplex mode. The MAC starts the transmission only when the CRS signal is low.

When this bit is reset, the MAC transmitter ignores the status of the CRS signal.

Bit 10 **DO**: Disable Receive Own

When this bit is set, the MAC disables the reception of packets when the ETH_TX_EN is asserted in the Half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.

This bit is not applicable in the Full-duplex mode. This bit is reserved and read-only (RO) with default value in the Full-duplex-only configurations.

Bit 9 **DCRS**: Disable Carrier Sense During Transmission

When this bit is set, the MAC transmitter ignores the MII CRS signal during packet transmission in the Half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.

When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

Bit 8 **DR**: Disable Retry

When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.

When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the Half-duplex mode.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **BL[1:0]**: Back-Off Limit

The back-off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbit/s) for which the MAC waits before rescheduling a transmission attempt during retries after a collision:

00: k= min (n, 10)

01: k = min (n, 8)

10: k = min (n, 4)

11: k = min (n, 1)

where n = retransmission attempt

The random integer r takes the value in the range $0 <= r < 2^k$.

This bit is applicable only in the Half-duplex mode.

Bit 4 **DC**: Deferral check

When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbit/s mode.

Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII.

The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.

When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.

This bit is applicable only in the Half-duplex mode.

Bits 3:2 **PRELEN[1:0]**: Preamble length for transmit packets

These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the Full-duplex mode.

00: 7 bytes of preamble

01: 5 bytes of preamble

10: 3 bytes of preamble

11: Reserved, must not be used

Bit 1 **TE**: Transmitter enable

When this bit is set, the Tx state machine of the MAC is enabled for transmission on the MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 **RE**: Receiver enable

When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the MII interface.

Table 566 shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

Table 566. Giant Packet Status based on S2KP and JE Bits(1)

Length/Type Field	Received Packet Length	S2KP	JE	Giant Packet Status
Untagged packet	> 1,518	0	0	1
	> 2,000	1	0	1
	> 9,018	x	1	1
VLAN tagged packet	> 1,522	0	0	1
	> 2,000	1	0	1
	> 9,022	x	1	1

1. For all other combinations, the Giant Packet status is 0.

Table 567 shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

Table 567. Packet Length based on the CST and ACS bits

Received Packet Length	CST	ACS	FCS Stripping Done
< 1,536	x	0	No
	x	1	Yes (for Ethernet packets)
≥ 1,536	0	x	No
	1	x	Yes (for Type packets)

Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004

Reset value: 0x0000 0000

The MAC Extended Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	APDIM	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC
	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GPSL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **APDIM**: ARP Packet Drop if IP Address Mismatch

When this bit is set, Packet for which Target Protocol Address does not match IPv4 address is dropped in the MTL layer.
When this bit is reset, when target Protocol Address does not match, the packet is forwarded to MTL maintaining backward compatibility.

Bits 29:25 **EIPG[4:0]**: Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits) along with IPG field in *Operating mode configuration register (ETH_MACCR)*, gives the minimum IPG greater than 96 bit times in steps of 8 bit times. For example:

- EIPG = 0 and IPG = 0 give 104 bit times
- EIPG = 0 and IPG = 1 give 112 bit times
- EIPG = 0 and IPG = 2 give 120 bit times
- ..
- EIPG = 7 and IPG = 31 give 2144 bit times

Bit 24 **EIPGEN**: Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in *Operating mode configuration register (ETH_MACCR)* together as minimum IPG greater than 96 bit times in steps of 8 bit times.
When this bit is reset, the MAC ignores EIPG field and interprets IPG field in *Operating mode configuration register (ETH_MACCR)* as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-duplex mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **USP**: Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the *MAC address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC address x low register (ETH_MACAxLR)*. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).
When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.

Bit 17 **SPEN**: Slow Protocol Detection Enable

When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid subtypes.
When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16 **DCRCC**: Disable CRC Checking for Received Packets

When this bit is set, the MAC receiver does not check the CRC field in the received packets.
When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **GPSL[13:0]**: Giant Packet Size Limit

If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.

For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. For double VLAN tagged packets, the MAC adds 8 bytes to the programmed value. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

Packet filtering control register (ETH_MACPFR)

Address offset: 0x0008

Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE
rw										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RA**: Receive All

When this bit is set, the MAC receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word.

When this bit is reset, the receiver module passes only those packets to the application that pass the SA or DA address filter.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DNTU**: Drop Non-TCP/UDP over IP Packets

When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.

Bit 20 **IPFE**: Layer 3 and Layer 4 Filter Enable

When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.

When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **VTFE**: VLAN Tag Filter Enable

When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HPF**: Hash or Perfect Filter

When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.

When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Bit 9 **SAF**: Source Address Filter Enable

When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.

When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

Note: According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8 **SAIF**: SA Inverse Filtering

When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.

When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6 **PCF[1:0]**: Pass Control Packets

These bits control the forwarding of all control packets (including unicast and multicast Pause packets).

00: The MAC filters all control packets from reaching the application.

01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.

10: The MAC forwards all control packets to the application even if they fail the Address filter.

11: The MAC forwards the control packets that pass the Address filter.

Bit 5 **DBF**: Disable Broadcast Packets

When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.

When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4 **PM**: Pass All Multicast

When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.

Bit 3 **DAIF**: DA Inverse Filtering

When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast

When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.

When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 1 **HUC**: Hash Unicast

When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.

When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 0 **PR**: Promiscuous Mode

When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

Watchdog and jabber timeout register (ETH_MACWJBTR)

Address offset: 0x000C

Reset value: 0x0000 0000

The watchdog and jabber timeout register controls the watchdog timeout for received packets and the jabber timeout limit for transmitted packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PJE	Res.	Res.	Res.	Res.	JTO[3:0]			
							rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
							rw					rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PJE**: Programmable Jabber Enable

When this bit is set and the JD bit of the *Operating mode configuration register (ETH_MACCR)* is reset, the JTO[3:0] bitfield is used as jabber timeout limit for transmitted packets.

When this bit is cleared, the jabber timeout for a received packet is controlled by setting the JD and JE bits of the *Operating mode configuration register (ETH_MACCR)*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **JTO[3:0]**: Jabber timeout

When the PJE bit is set and the JD bit of the *Operating mode configuration register (ETH_MACCR)* is cleared, this bitfield is used as jabber timeout limit for transmitted packets. If the length of a transmitted packet exceeds the value of this bitfield, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes
0x1: 3 Kbytes
0x2: 4 Kbytes
0x3: 5 Kbytes
..
0xC: 14 Kbytes
0xD: 15 Kbytes
0xE: 16383 Bytes
0xF: Reserved, must not be used

Note: When the PJE bit is set, the value in this bitfield must be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and terminated.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **PWE**: Programmable Watchdog Enable

When this bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in *Operating mode configuration register (ETH_MACCR)* register.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WTO[3:0]**: Watchdog timeout

When the PWE bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes
0x1: 3 Kbytes
0x2: 4 Kbytes
0x3: 5 Kbytes
..
0xC: 14 Kbytes
0xD: 15 Kbytes
0xE: 16383 Bytes
0xF: Reserved, must not be used

Note: When the PWE bit is set, the value in this bitfield should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

Hash Table 0 register (ETH_MACHT0R)

Address offset: 0x0010

Reset value: 0x0000 0000

The Hash Table Register 0 contains the first lower 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 0b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT31T0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT31T0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT31T0[31:0]**: MAC Hash Table First 32 Bits

This field contains the first 32 Bits [31:0] of the Hash table.

Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014

Reset value: 0x0000 0000

The Hash Table 1 register contains the upper 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT63T32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT63T32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT63T32[31:0]**: MAC Hash Table Second 32 Bits

This field contains the second 32 Bits [63:32] of the Hash table.

VLAN tag register (ETH_MACVTR)

Address offset: 0x0050

Reset value: 0x0000 0000

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV
rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EIVLRXS**: Enable Inner VLAN Tag in Rx Status

When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **EIVLS[1:0]**: Enable Inner VLAN Tag Stripping on Receive

This field indicates the stripping operation on inner VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 27 **ERIVLT**: Enable Inner VLAN Tag

When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.The ERSVLM bit and DOVLTC bit determines which VLAN type is enabled for filtering.

Bit 26 **EDVLP**: Enable Double VLAN Processing

When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).

Bit 25 **VTHM**: VLAN Tag Hash Table Match Enable

When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHTR register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table.
When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison.
When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison.
When this bit is reset, the VLAN Hash Match operation is not performed.

Bit 24 **EVLRXS**: Enable VLAN Tag in Rx status

When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EVLS[1:0]**: Enable VLAN Tag Stripping on Receive

This field indicates the stripping operation on the outer VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 20 **DOVLTC**: Disable VLAN Type Check

When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19 **ERSVLM**: Enable Receive S-VLAN Match

When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18 **ESVL**: Enable S-VLAN

When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17 **VTIM**: VLAN Tag Inverse Match Enable

When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16 **ETV**: Enable 12-Bit VLAN Tag Comparison

When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:0 **VL[15:0]**: VLAN Tag Identifier for Receive Packets

This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag
When the ETV bit is set, only the VID is used for comparison.
If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.

VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058

Reset value: 0x0000 0000

When the VTHM bit of *VLAN tag register (ETH_MACVTR)* register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of *VLAN tag register (ETH_MACVTR)* register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLHT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VLHT[15:0]**: VLAN Hash Table
This field contains the 16-bit VLAN Hash Table.

VLAN inclusion register (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit packets. It also contains the VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]	
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN priority control

When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

Bits 17:16 **VLC[1:0]**: VLAN tag control in transmit packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.

10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN tag for transmit packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Inner VLAN inclusion register (ETH_MACIVIR)

Address offset: 0x0064

Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the transmit packet. It also contains the inner VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]	
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN tag input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN priority control

When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 **VLC[1:0]**: VLAN tag control in transmit packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion

The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.

10: VLAN tag insertion

The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement

The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN tag for transmit packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.
The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Tx Queue flow control register (ETH_MACQTXFCR)

Address offset: 0x0070

Reset value: 0x0000 0000

The flow control register controls the generation and reception of the control (Pause Command) packets by the flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
								rw	rw	rw	rw			rw	rw

Bits 31:16 **PT[15:0]**: Pause Time

This field holds the value to be used in the Pause Time field in the Tx control packet. I

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DZPQ**: Disable Zero-Quanta Pause

When this bit is set, it disables the automatic generation of the zero-quanta Pause packets.
When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Bits 6:4 **PLT[2:0]**: Pause Low Threshold

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

000: Pause Time minus 4 Slot Times (PT -4 slot times)

001: Pause Time minus 28 Slot Times (PT -28 slot times)

010: Pause Time minus 36 Slot Times (PT -36 slot times)

011: Pause Time minus 144 Slot Times (PT -144 slot times)

100: Pause Time minus 256 Slot Times (PT -256 slot times)

101: Pause Time minus 512 Slot Times (PT -512 slot times)

110 to 111: Reserved, must not be used

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface.

This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TFE**: Transmit Flow Control Enable

Full-duplex mode: when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

Half-duplex mode: when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0 **FCB_BPA**: Flow Control Busy or Backpressure Activate

This bit initiates a Pause packet in the Full-duplex mode and activates the backpressure function in the Half-duplex mode if the TFE bit is set.

Full-Duplex mode: this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

Half-duplex mode: When this bit is set (and TFE bit is set) in the Half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the Full-duplex mode, the BPA is automatically disabled.

Rx flow control register (ETH_MACRXFCR)

Address offset: 0x0090

Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC transmit based on the received Pause packet.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UP**: Unicast Pause Packet Detect

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in *MAC address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC address x low register (ETH_MACAxLR)*.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01 80 C2 00 00 01) is as specified in IEEE 802.1 Qbb-2011.

Bit 0 **RFE**: Receive Flow Control Enable

When this bit is set and the MAC is operating in Full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in Half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the transmit queue, with matching priorities, for a duration of received Pause time.

Interrupt status register (ETH_MACISR)

Address offset: 0x00B0

Reset value: 0x0000 0000

The interrupt status register contains the status of interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIS	Res.	Res.
													rc_r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	Res.
	rc_r	rc_r	rc_r		r	r	r			r	r	r			

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **MDIOIS**: MDIO interrupt status

This bit indicates an interrupt event after the completion of MDIO operation. To reset this bit, the application has to read this bit (or write it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* register is set) register is set.

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIS**: Receive status interrupt

This bit indicates the status of received packets. This bit is set when the RWT bit is set in the *Rx Tx status register (ETH_MACRXTXSR)*. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 13 **TXSTSIS**: Transmit status interrupt

This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the *Rx Tx status register (ETH_MACRXTXSR)*:

- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 12 **TSIS**: Timestamp interrupt status

If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:

– The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.

– There is an overflow in the Seconds register.

– The Target Time Error occurred, that is, programmed target time already elapsed.

If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the *Timestamp status register (ETH_MACTSSR)*.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC transmit interrupt status

This bit is set high when an interrupt is generated in the *MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 9 **MMCRXIS**: MMC Receive interrupt status

This bit is set high when an interrupt is generated in the *MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 8 **MMCIS**: MMC interrupt status

This bit is set high when MMCTXIS or MMCRXIS is set high. This bit is cleared only when all these bits are low.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **LPIIS**: LPI interrupt status

This bit is set for any LPI state entry or exit in the MAC transmitter or receiver. This bit is cleared when the TLPIEN bit of *LPI control and status register (ETH_MACLCSR)* is read.

Bit 4 **PMTIS**: PMT interrupt status

This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the *PMT control status register (ETH_MACPCSR)*.

Bit 3 **PHYIS**: PHY Interrupt

This bit is set when rising edge is detected on the ETH_PHY_INTN input. This bit is cleared when this register is read.

Bits 2:0 Reserved, must be kept at reset value.

Interrupt enable register (ETH_MACIER)

Address offset: 0x00B4

Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIE	Res.	Res.
													rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	Res.
	rw	rw	rw							rw	rw	rw			

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **MDIOIE**: MDIO Interrupt Enable
When this bit is set, it enables the assertion of the interrupt when MDIOIS field is set in the *Interrupt status register (ETH_MACISR)*.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIE**: Receive Status Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 13 **TXSTSIE**: Transmit Status Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 12 **TSIE**: Timestamp Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **LPIIE**: LPI Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 4 **PMTIE**: PMT Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 3 **PHYIE**: PHY Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 2:0 Reserved, must be kept at reset value.

Rx Tx status register (ETH_MACRXTXSR)

Address offset: 0x00B8

Reset value: 0x0000 0000

The receive transmit status register contains the receive and transmit error status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
							rc_r			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RWT**: Receive watchdog timeout

This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **EXCOL**: Excessive Collisions

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after the first collision and the packet transmission is aborted.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 4 **LCOL**: Late Collision

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode.

This bit is not valid if the Underflow error occurs.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 3 **EXDEF**: Excessive Deferral

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)* and the DC bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 when Jumbo packet is enabled).

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **LCARR**: Loss of Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the loss of carrier occurred during packet transmission, that is, the ETH_CRS signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **NCARR**: No Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TJT**: Transmit Jabber timeout

This bit indicates that the transmit jabber timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

PMT control status register (ETH_MACPCSR)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw			r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
					rw	rw			r	rc_r			rw	rw	rw

Bit 31 **RWKFILTRST**: Remote Wake-up Packet Filter Register Pointer Reset

When this bit is set, the remote wake-up packet filter register pointer is reset to 0. It is automatically cleared after 1 clock cycle.

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:24 **RWKPTR[4:0]**: Remote wake-up FIFO Pointer

This field gives the current value (0 to 7) of the Remote wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wake-up Packet Filter Register are transferred to the eth_rx_clk domain when a Write occurs to that register.

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **RWKPFE**: Remote wake-up Packet Forwarding Enable

When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet.

The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high.

Note: If Magic Packet Enable and wake-up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.

Bit 9 **GLBLUCAST**: Global Unicast

When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RWKPRCVD**: Remote wake-up Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read.

Bit 5 **MGKPRCVD**: Magic Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RWKPKTEN**: Remote wake-up Packet Enable

When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.

Bit 1 **MGKPKTEN**: Magic Packet Enable

When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0 **PWRDWN**: Power Down

When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wake-up Packet Enable bit is set high.

Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

Remote wake-up packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACRWKPFR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACRWKPFR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MACRWKPFR[31:0]**: Remote wake-up packet filter

Refer to *Table 535*, *Table 536* and *Table 537* for details on register content and programming sequence.

The ETH_MACRWKPFR register at address 0x00C4 loads the wake-up Packet Filter register.

To load values in a wake-up Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

LPI control and status register (ETH_MACLCSR)

Address offset: 0x00D0

Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	Res.	PLS	LPIEN
										rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
						r	r					r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **LPITCSE**: LPI Tx Clock Stop Enable

When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.
If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.

Bit 20 **LPITE**: LPI Timer Enable

This bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPIEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.

After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again.

When LPITE is 0, the LPI Auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.

Bit 19 **LPITXA**: LPI Tx Automate

This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the transmit side.

If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTXQ0OMR, when the MAC is in the LPI mode, it exits the LPI mode.

When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PLS**: PHY Link Status

This bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.

When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16 **LPIEN**: LPI Enable

When this bit is set, it instructs the MAC transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.

This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **RLPIST**: Receive LPI State

When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.

Bit 8 **TLPIST**: Transmit LPI State

When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **RLPIEX**: Receive LPI Exit

When this bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 2 **RLPIEN**: Receive LPI Entry

When this bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 1 **TLPIEX**: Transmit LPI Exit

When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TLPIEN**: Transmit LPI Entry

When this bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

LPI timers control register (ETH_MACLTCR)

Address offset: 0x00D4

Reset value: 0x03E8 0000

The LPI timers control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	LST[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TWT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **LST[9:0]**: LPI LS Timer

This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 s) as defined in the IEEE standard.

Bits 15:0 **TWT[15:0]**: LPI TW Timer

This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

LPI entry timer register (ETH_MACLETR)

Address offset: 0x00D8

Reset value: 0x0000 0000

This register controls the Tx LPI entry timer. This counter is enabled only when LPITE bit of *LPI control and status register (ETH_MACLCSR)* register is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LPIET[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPIET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **LPIET[19:0]**: LPI Entry Timer

This field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1.

Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

One-microsecond-tick counter register (ETH_MAC1USTCR)

Address offset: 0x00DC

Reset value: 0x0000 0063

This register controls the generation of the Reference time (one-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 TIC_1US_CNTR[11:0]: 1 µs tick Counter

The application must program this counter so that the number of clock cycles of CSR clock is 1 µs (subtract 1 from the value before programming).

For example if the CSR clock is 100 MHz then this field needs to be programmed to 100 - 1 = 99 (which is 0x63).

This is required to generate the 1 µs events that are used to update some of the EEE related counters.

Version register (ETH_MACVR)

Address offset: 0x0110

Reset value: 0x0000 3354

The version register identifies the version of the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USERVER[7:0]								SNPSVER[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **USERVER[7:0]**: ST-defined version

Bits 7:0 **SNPSVER[7:0]**: IP version

Debug register (ETH_MACDR)

Address offset: 0x0114

Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
													r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **TFCSTS[1:0]**: MAC transmit packet controller status

This field indicates the state of the MAC transmit packet controller module:

00: Idle state

01: Waiting for one of the following:

- Status of the previous packet
- IPG or backoff period to be over

10: Generating and transmitting a Pause control packet (in Full-duplex mode)

11: Transferring input packet for transmission

Bit 16 **TPESTS**: MAC MII transmit protocol engine status
When this bit is set, it indicates that the MAC MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **RFCFCSTS[1:0]**: MAC receive packet controller FIFO status
When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC receive packet controller module.

Bit 0 **RPESTS**: MAC MII receive protocol engine status
When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.

HW feature 0 register (ETH_MACHWF0R)

Address offset: 0x011C

Reset value: 0x0A0D 73F5

This register indicates the presence of first set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]		MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	r	r	r			r	r	r	r	r	r	r	r	r	r

Bits 31:28 **ACTPHYSEL[3:0]**: Active PHY selected
When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion:
0x0: GMII or MII
0x1: RGMII
0x2: SGMII
0x3: TBI
0x4: RMII
0x5: RTBI
0x6: SMII
Others: Reserved, must not be used

Bit 27 **SAVLANINS**: Source Address or VLAN Insertion Enable
This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected

Bits 26:25 **TSSTSSEL[1:0]**: Timestamp System Time Source
This bit indicates the source of the Timestamp system time:
01: Internal
10: External
11: Both
00: Reserved, must not be used
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bit 24 **MACADR64SEL**: MAC Addresses 64-127 Selected
This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected

Bit 23 **MACADR32SEL**: MAC Addresses 32-63 Selected
This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected

Bits 22:18 **ADDMACADRSEL[4:0]**: MAC Addresses 1-31 Selected
This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected

Bit 17 Reserved, must be kept at reset value.

Bit 16 **RXCOESEL**: Receive checksum offload Enabled
This bit is set to 1 when the enable receive TCP/IP checksum check option is selected

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXCOESEL**: Transmit Checksum Offload Enabled
This bit is set to 1 when the enable transmit TCP/IP checksum insertion option is selected

Bit 13 **EEESEL**: Energy Efficient Ethernet Enabled
This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected

Bit 12 **TSSEL**: IEEE 1588-2008 Timestamp Enabled
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ARPOFFSEL**: ARP Offload Enabled
This bit is set to 1 when the Enable IPv4 ARP Offload option is selected

Bit 8 **MMCSEL**: RMON Module Enable
This bit is set to 1 when the Enable MAC management counters (MMC) option is selected

Bit 7 **MGKSEL**: PMT Magic Packet Enable
This bit is set to 1 when the Enable Magic Packet Detection option is selected

Bit 6 **RWKSEL**: PMT Remote Wake-up Packet Enable
This bit is set to 1 when the Enable Remote Wake-up Packet Detection option is selected

Bit 5 **SMASEL**: SMA (MDIO) Interface
This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected

Bit 4 **VLHASH**: VLAN Hash Filter Selected
This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected

Bit 3 **PCSSEL**: PCS Registers (TBI, SGMII, or RTBI PHY interface)
This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected

Bit 2 **HDSEL**: Half-duplex Support
This bit is set to 1 when the Half-duplex mode is selected

Bit 1 **GMIISEL**: 1000 Mbit/s Support
This bit is set to 1 when 1000 Mbit/s is selected as operating mode.

Bit 0 **MIISEL**: 10 or 100 Mbit/s Support
This bit is set to 1 when 10/100 Mbit/s is selected as operating mode.

HW feature 1 register (ETH_MACHWF1R)

Address offset: 0x0120

Reset value: 0x1104 1924

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN
	r	r	r	r		r	r	r		r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					SPRAM	RXFIFOSIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
This field indicates the total number of L3 or L4 filters:
0000: No L3 or L4 Filter
0001: 1 L3 or L4 Filter
0010: 2 L3 or L4 Filters
..
1000: 8 L3 or L4

Bit 26 Reserved, must be kept at reset value.

Bits 25:24 **HASHTBLSZ[1:0]**: Hash Table Size
This field indicates the size of the Hash table:
00: No Hash table
01: 64
10: 128
11: 256

Bit 23 **POUOST**: One Step for PTP over UDP/IP Feature Enable
This bit is set to 1 when the Enable one step timestamp for PTP over UDP/IP feature is selected.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **RAVSEL**: Rx Side Only AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option on Rx Side Only is selected.

Bit 20 **AVSEL**: AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option is selected.

Bit 19 **DBGMEMA**: DMA debug registers enable
This bit is set to 1 when the Debug Mode Enable option is selected

Bit 18 **TSOEN**: TCP segmentation offload enable
This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Bit 17 **SPHEN**: Split header feature enable
This bit is set to 1 when the Enable Split Header Structure option is selected

Bit 16 **DCBEN**: DCB feature enable
This bit is set to 1 when the Enable Data Center Bridging option is selected

Bits 15:14 **ADDR64[1:0]**: Address width
This field indicates the configured address width.
00: 32 bits
Others: Reserved, must not be used

Bit 13 **ADVTHWORD**: IEEE 1588 High Word Register enable
This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected

Bit 12 **PTOEN**: PTP offload enable
This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11 **OSTEN**: One-step timestamping enable
This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.

Bits 10:6 **TXFIFOSIZE[4:0]**: MTL transmit FIFO size
This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011 to 11111: Reserved, must not be used

Bit 5 **SPRAM**: Single Port RAM Enable
This bit is set to 1 when the Use single port RAM feature is selected.

Bits 4:0 **RXFIFOSIZE[4:0]**: MTL receive FIFO Size
This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011: 256 Kbytes
01100 to 11111: Reserved, must not be used

HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124

Reset value: 0x4100 0000

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]	
	r	r	r		r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
r	r	r	r			r	r	r	r			r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **AUXSNAPNUM[2:0]**: Number of auxiliary snapshot inputs
This field indicates the number of auxiliary snapshot inputs:
000: No auxiliary input
001: 1 auxiliary input
010: 2 auxiliary inputs
011: 3 auxiliary inputs
100: 4 auxiliary inputs
101 to 111: Reserved, must not be used

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **PPSOUTNUM[2:0]**: Number of PPS outputs
This field indicates the number of PPS outputs:
000: No PPS output
001: 1 PPS output
010: 2 PPS outputs
011: 3 PPS outputs
100: 4 PPS outputs
101 to 111: Reserved, must not be used

Bits 23:22 **TDCSZ[1:0]**: Tx DMA descriptor cache size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 21:18 **TXCHCNT[3:0]**: Number of DMA transmit channels
This field indicates the number of DMA transmit channels:
0000: 1 DMA Tx channel
0001: 2 DMA Tx channels
..
0111: 8 DMA Tx

Bits 17:16 **RDCSZ[1:0]**: Rx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 15:12 **RXCHCNT[3:0]**: Number of DMA receive channels
This field indicates the number of DMA receive channels:
0000: 1 DMA Rx channel
0001: 2 DMA Rx channels
..
0111: 8 DMA Rx

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **TXQCNT[3:0]**: Number of MTL transmit queues
This field indicates the number of MTL transmit queues:
0000: 1 MTL Tx queue
0001: 2 MTL Tx queues
..
0111: 8 MTL Tx

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **RXQCNT[3:0]**: Number of MTL receive queues
This field indicates the number of MTL receive queues:
0000: 1 MTL Rx queue
0001: 2 MTL Rx queues
..
0111: 8 MTL Rx

HW feature 3 register (ETH_MACHWF3R)

Address offset: 0x0128

Reset value: 0x0000 0020

This register indicates the presence of fourth set the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ASP[1:0]		TBSSEL	FPESEL	Res.	Res.	Res.	Res.	ESTWID[1:0]		ESTDEP[2:0]			ESTSEL
		r	r	r	r					r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FRPES[1:0]		FRPBS[1:0]		FRPSEL	PDUPSEL	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
	r	r	r	r	r	r				r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ASP[1:0]**: Automotive Safety Package
Following are the encoding for the different Safety features
00: None
01: ECC only
10: AS_NPPE
11: AS_PPE

Bit 27 **TBSSEL**: Time-based scheduling Enable
This bit is set to 1 when the time-based scheduling feature is selected.

Bit 26 **FPESEL**: Frame Preemption Enable
This bit is set to 1 when the Enable Frame preemption feature is selected.

Bits 25:22 Reserved, must be kept at reset value.

Bits 21:20 **ESTWID[1:0]**: Width of the Time Interval field in the Gate Control List
This field indicates the width of the Configured Time Interval Field
00: No width
01: 16
10: 20
11: 24

Bits 19:17 **ESTDEP[2:0]**: Depth of the Gate Control List
This field indicates the depth of Gate Control list
000: No depth
001: 64
010: 128
011: 256
100: 512
101: 1024
110: Reserved, must not be used

Bit 16 **ESTSEL**: Enhancements to Scheduled Traffic Enable
This bit is set to 1 when the enable enhancements to scheduling traffic feature is selected.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **FRPES[1:0]**: Flexible receive parser table entries size
This field indicates the maximum number of parser entries supported by flexible receive parser.
00: 64 entries
01: 128 entries
10: 256 entries
11: Reserved, must not be used

Bits 12:11 **FRPBS[1:0]**: Flexible receive parser buffer size
This field indicates the supported maximum number of bytes of the packet data to be parsed by flexible receive parser.
00: 64 bytes
01: 128 bytes
10: 256 bytes
11: Reserved, must not be used

Bit 10 **FRPSEL**: Flexible receive parser Selected
This bit is set to 1 when the Enable Flexible Programmable receive parser option is selected.

Bit 9 **PDUPSEL**: Broadcast/Multicast Packet Duplication
This bit is set to 1 when the Broadcast/Multicast Packet Duplication feature is selected.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **DVLAN**: Double VLAN processing enable
This bit is set to 1 when Double VLAN processing is enabled.

Bit 4 **CBTISEL**: Queue/channel based VLAN tag insertion on Tx enable
This bit is set to 1 when the Enable Queue/channel based VLAN tag insertion on Tx feature is selected.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **NRVF[2:0]**: Number of Extended VLAN Tag Filters Enabled
This field indicates the Number of Extended VLAN Tag Filters selected:
000: No Extended Rx VLAN Filters
001: 4 Extended Rx VLAN Filters
010: 8 Extended Rx VLAN Filters
011: 16 Extended Rx VLAN Filters
100: 24 Extended Rx VLAN Filters
101: 32 Extended Rx VLAN Filters
110 to 111: Reserved, must not be used

MDIO address register (ETH_MACMDIOAR)

Address offset: 0x0200

Reset value: 0x0000 0000

The MDIO address register controls the management cycles to external PHY through a management interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]				
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	MB
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PSE**: Preamble suppression enable

When this bit is set, the SMA suppresses the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.

When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.

Bit 26 **BTB**: Back-to-back transactions

When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame.

When this bit is reset, then the read/write command completion (MII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.

This bit must not be set when NTC=0.

Bits 25:21 **PA[4:0]**: Physical layer address

This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.
This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Bits 20:16 **RDA[4:0]**: Register/Device address

These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **NTC[2:0]**: Number of training cocks

This field controls the number of training clock cycles generated on ETH_MDC after the end of transmission of the MDIO frame. The valid values can be from 0 to 7. Programming the value to 011 indicates that there are three additional clock cycles on the MDC line after the end of the MDIO frame transfer.

Bits 11:8 **CR[3:0]**: CSR clock range

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency (eth_hclk):
0000: MDC clock = eth_hclk / 42
0001: MDC clock = eth_hclk / 62
0010: MDC clock = eth_hclk / 16
0011: MDC clock = eth_hclk / 26
0100: MDC clock = eth_hclk / 102
0101: MDC clock = eth_hclk / 124
0110: MDC clock = eth_hclk / 204
0111: MDC clock = eth_hclk / 324
The suggested CSR clock frequency range applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately in the 1.0 MHz to 2.5 MHz frequency range.
When Bit 11 is set, the MDC clock frequency can reach a higher value than the frequency limit of 2.5 MHz (specified in the IEEE 802.3). Therefore, a lower clock divider can be programmed. For example, when the CSR clock is of 100 MHz frequency and these bits are programmed to 1010, the resultant MDC clock is of 12.5 MHz, which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks:
1000: eth_hclk / 4
1001: eth_hclk / 6
1010: eth_hclk / 8
1011: eth_hclk / 10
1100: eth_hclk / 12
1101: eth_hclk / 14
1110: eth_hclk / 16
1111: eth_hclk / 18

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SKAP**: Skip address packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Bits 3:2 **GOC[1:0]**: MII operation command

This bit indicates the operation command to the PHY.
00: Reserved, must not be used
01: Write
10: Post Read Increment Address for Clause 45 PHY
11: Read
When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E**: Clause 45 PHY Enable

When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.

Bit 0 **MB**: MII Busy

The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in *MDIO address register (ETH_MACMDIOAR)* and *MDIO data register (ETH_MACMDIODR)* as long as this bit is set.

For write transfers, the application must first write 16-bit data in the MD field (and also RA field when C45E is set) in *MDIO data register (ETH_MACMDIODR)* register before setting this bit. When C45E is set, it should also write into the RA field of *MDIO data register (ETH_MACMDIODR)* before initiating a read transfer. When a read transfer is completed (MII busy=0), the data read from the PHY register is valid in the MD field of the *MDIO data register (ETH_MACMDIODR)*.

Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit.

MDIO data register (ETH_MACMDIODR)

Address offset: 0x0204

Reset value: 0x0000 0000

The MDIO data register stores the write data to be written to the PHY register located at the address specified in the *MDIO address register (ETH_MACMDIOAR)*. This register also stores the read data from the PHY register located at the address specified in the MDIO address register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **RA[15:0]**: Register address

This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

Bits 15:0 **MD[15:0]**: MII data

This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

ARP address register (ETH_MACARPAR)

Address offset: 0x0210

Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPPA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARPPA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARPPA[31:0]**: ARP Protocol Address

This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

CSR software control register (ETH_MACCSRSWCR)

Address offset: 0x0230

Reset value: 0x0000 0000

This register contains software-programmable controls for changing the CSR access response and status bit clearing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
															rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RCWE**: Register Clear on Write 1 Enable

When this bit is set, the access mode to some register fields changes to rc_w1 (clear on write) meaning that the application needs to set that respective bit to 1 to clear it.
When this bit is reset, the access mode to these register fields remains rc_r (clear on read).

MAC presentation time register (ETH_MACPRSTIMR)

Address offset: 0x0240

Reset value: 0x0000 0000

This register contains the 32-bit binary rollover equivalent time of the PTP System Time expressed in ns.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPTN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPTN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MPTN[31:0]**: MAC 1722 Presentation Time in ns

These bits indicate the value of the 32-bit binary rollover equivalent time of the PTP System Time in ns.

MAC presentation time update register (ETH_MACPRSTIMUR)

Address offset: 0x0244

Reset value: 0x0000 0000

This register holds the 32-bit value of MAC 1722 Presentation Time in ns, that should be added to the current presentation time counter value. The initialization happens when TSINIT is set, and the update when the TSUPDT bit is set (TSINIT or TSUPDT bits in *Timestamp control register (ETH_MACTSCR)*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPTU[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPTU[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MPTU[31:0]**: MAC 1722 Presentation Time Update

This field holds the initialization value or the update value for the presentation time. When used for update, this field holds the 32-bit value, in ns, that should be added to the current presentation time counter value. The initialization happens when TSINIT is set, and the update when the TSUPDT bit is set (TSINIT or TSUPDT bits in *Timestamp control register (ETH_MACTSCR)*). When ADDSUB field of *System time nanoseconds update register (ETH_MACSTNUR)* is set, this value is directly used for subtraction.

MAC address 0 high register (ETH_MACA0HR)

Address offset: 0x0300

Reset value: 0x8000 FFFF

The MAC address0 high register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x11 22 33 44 55 66 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x66 55 44 33 22 11.

If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

This bit is always set to 1.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address0[47:32]

This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the transmit flow control (Pause) packets.

MAC address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)

Reset value: 0xFFFF FFFF

The MAC address x low register holds the lower 32 bits of the 6-byte first MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)

This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the transmit flow control (Pause) packets.

MAC address x high register (ETH_MACAxHR)

Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)

Reset value: 0x0000 FFFF

The MAC address x high register holds the upper 16 bits of the second 6-byte MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30 **SA**: Source Address

When this bit is set, the MAC Addressx[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.

0: DA

1: SA

Bits 29:24 **MBC[5:0]**: Mask Byte Control

These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:

Bit 29: ETH_MACAxHR[15:8]

Bit 28: ETH_MACAxHR[7:0]

Bit 27: ETH_MACAxLR[31:24]

Bit 26: ETH_MACAxLR[23:16]

Bit 25: ETH_MACAxLR[15:8]

Bit 24: ETH_MACAxLR[7:0]

You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address1 [47:32]

This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MMC control register (ETH_MMC_CONTROL)

Address offset: 0x0700

Reset value: 0x0000 0000

This register configures the MMC operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
							rw			rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets

The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of *Packet filtering control register (ETH_MACPFR)*.

When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset

When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF FFF0 (Half 16).

When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF F800 (Full 2Kbytes) and all packet-counters get preset to 0xFFFF FFF0 (Full 16).

For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.

Bit 4 **CNTPRST**: Counters Preset

When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.

This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze

When this bit is set, it freezes all MMC counters to their current value.

Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Bit 2 **RSTONRD**: Reset on Read

When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover
When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset
When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)

Address offset: 0x0704

Reset value: 0x0000 0000

This register maintains the interrupts generated from all receive statistics counters.

The MMC receive interrupt register maintains the interrupts that are generated when the following occur:

- Receive statistic counters reach half of their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, interrupts are set but the counter remains at all-ones. The MMC receive interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.
				rc_r	rc_r									rc_r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
									rc_r	rc_r					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIS**: MMC receive LPI transition counter interrupt status
This bit is set when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIS**: MMC receive LPI microsecond counter interrupt status
This bit is set when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIS**: MMC receive Unicast good packet counter interrupt status
This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIS**: MMC receive alignment error packet counter interrupt status
This bit is set when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIS**: MMC receive CRC error packet counter interrupt status
This bit is set when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)

Address offset: 0x0708

Reset value: 0x0000 0000

This register maintains the interrupts generated from all transmit statistics counters.

The MMC transmit interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones.

The MMC transmit interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.
				rc_r	rc_r					rc_r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rc_r	rc_r														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIS**: MMC transmit LPI transition counter interrupt status
This bit is set when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIS**: MMC transmit LPI microsecond counter interrupt status
This bit is set when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIS**: MMC transmit good packet counter interrupt status
This bit is set when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIS**: MMC transmit multiple collision good packet counter interrupt status
This bit is set when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIS**: MMC transmit single collision good packet counter interrupt status
This bit is set when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

MMC Rx interrupt mask register (ETH_MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C

Reset value: 0x0000 0000

The MMC receive interrupt mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.
				rw	rw									rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
									rw	rw					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIM**: MMC receive LPI transition counter interrupt Mask
Setting this bit masks the interrupt when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIM**: MMC receive LPI microsecond counter interrupt Mask
Setting this bit masks the interrupt when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIM**: MMC receive Unicast good packet counter interrupt mask
Setting this bit masks the interrupt when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIM**: MMC receive Alignment Error packet counter interrupt mask
Setting this bit masks the interrupt when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIM**: MMC receive CRC Error packet counter interrupt mask
Setting this bit masks the interrupt when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt mask register (ETH_MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710

Reset value: 0x0000 0000

The MMC transmit interrupt mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32-bit wide.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.
				rw	rw					rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIM**: MMC transmit LPI transition counter interrupt mask
Setting this bit masks the interrupt when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIM**: MMC transmit LPI microsecond counter interrupt mask
Setting this bit masks the interrupt when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIM**: MMC transmit good packet counter interrupt mask
Setting this bit masks the interrupt when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIM**: MMC transmit multiple-collision good packet counter interrupt mask
Setting this bit masks the interrupt when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIM**: MMC transmit single-collision good packet counter interrupt mask
Setting this bit masks the interrupt when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXSNGLCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXSNGLCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXSNGLCOLG[31:0]**: Tx Single Collision Good Packets
This field indicates the number of successfully transmitted packets after a single collision in the Half-duplex mode.

Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMULTCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMULTCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMULTCOLG[31:0]**: Tx Multiple Collision Good Packets
This field indicates the number of successfully transmitted packets after multiple collisions in the Half-duplex mode.

Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)

Address offset: 0x0768

Reset value: 0x0000 0000

This register provides the number of good packets transmitted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPKTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPKTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPKTG[31:0]**: Tx Packet Count Good
This field indicates the number of good packets transmitted.

Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)

Address offset: 0x0794

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with CRC error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRCERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRCERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRCERR[31:0]**: Rx CRC Error Packets
This field indicates the number of packets received with CRC error.

Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)

Address offset: 0x0798

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXALGNERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXALGNERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXALGNERR[31:0]**: Rx Alignment Error Packets
This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)

Address offset: 0x07C4

Reset value: 0x0000 0000

This register provides the number of good unicast packets received by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXUCASTG[31:0]**: Rx Unicast Packets Good
This field indicates the number of good unicast packets received.

Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)

Address offset: 0x07EC

Reset value: 0x0000 0000

This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPIUSC[31:0]**: Tx LPI Microseconds Counter

This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)

Address offset: 0x07F0

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPITRC[31:0]**: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate mode (because of LPITXA bit set in the *LPI control and status register (ETH_MACLCSR)*), the counter is incremented.

Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)

Address offset: 0x07F4

Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPIUSC[31:0]**: Rx LPI Microseconds Counter

This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)

Address offset: 0x07F8

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPITRC[31:0]**: Rx LPI Transition counter

This field indicates the number of times Rx LPI Entry has occurred.

L3 and L4 control 0 register (ETH_MACL3L4C0R)

Address offset: 0x0900

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0
										rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM0**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM0 bit is set high.

Bit 20 **L4DPM0**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM0**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM0 bit is set high.

Bit 18 **L4SPM0**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN0**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of higher bits of IP Destination Address that are masked in the IPv4 packets:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. Number of bits masked is given by concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP source address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP source or destination address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.

Bit 2 **L3SAM0**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN0**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

Layer4 address filter 0 register (ETH_MACL4A0R)

Address offset: 0x0904

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP0[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP0[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 0 register (ETH_MACL3A00R)

Address offset: 0x0910

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A00[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A00[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A00[31:0]**: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A10[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A10[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A10[31:0]**: Layer 3 Address 1 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 0 register (ETH_MACL3A20R)

Address offset: 0x0918

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A20[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A20[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A20[31:0]**: Layer 3 Address 2 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

Layer3 address 3 filter 0 register (ETH_MACL3A30R)

Address offset: 0x091C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A30[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A30[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A30[31:0]**: Layer 3 Address 3 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

L3 and L4 control 1 register (ETH_MACL3L4C1R)

Address offset: 0x0930

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 1 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1
										rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM1**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM1 bit is set high.

Bit 20 **L4DPM1**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM1**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM1 bit is set high.

Bit 18 **L4SPM1**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN1**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM1 or L4DPM1 bit is set.

Bits 15:11 **L3HDBM1[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP Destination Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits:

0: No bits are masked

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

127: All bits except MSb are masked

This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set.

Bits 10:6 **L3HSBM1[4:0]**: Layer 3 IP SA Higher Bits Match

Condition: IPv4 packets

This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

This field contains Bits[4:0] of L3HSBM1. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set high.

Bit 5 **L3DAIM1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM1 bit is set high.

Bit 4 **L3DAM1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN1 bit is set, you should set either this bit or the L3SAM1 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM1**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM1 bit is set.

Bit 2 **L3SAM1**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN01 bit is set, you should set either this bit or the L3DAM1 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN1**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM1 or L3DAM1 bit is set.

Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP1[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP1[15:0]**: Layer 4 Source Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 1 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A01[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A01[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A01[31:0]**: Layer 3 Address 0 Field

When the L3PEN1 and L3SAM1bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 1 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A11[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A11[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 1 Register (ETH_MACL3A21R)

Address offset: 0x0948

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A21[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A21[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A21[31:0]**: Layer 3 Address 2 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

Layer3 address 3 filter 1 register (ETH_MACL3A31R)

Address offset: 0x94C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A31[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A31[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A31[31:0]**: Layer 3 Address 3 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

MAC indirect access control register (ETH_MAC_IACR)

Address offset: 0x0A70

Reset value: 0x0000 0000

This register provides the Indirect Access control and status for the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AOFF[7:0]								Res.	Res.	AUTO	Res.	Res.	Res.	COM	OB
rw	rw	rw	rw	rw	rw	rw	rw			rw				rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **MSEL[3:0]**: Mode Select

This field is used for indirect accesses to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*. It must be set when initiating a read/write to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* and should not be changed until the OB bit is reset.

0000:Typ_RXQ_ (Type-based RXQ mapping)

0001-1111: Reserved for future use

Bits 15:8 **AOFF[7:0]**: Address Offset

This field is used for indirect accesses to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*. It must be set when initiating a read/write to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* and should not be changed until the OB bit is reset.

00000000: IndReg0 (Indirect register 0)

00000001: IndReg1 (Indirect register 1)

...

00000111: IndReg7 (Indirect register 7)

00001000-11111111: Reserved, must not be used

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **AUTO**: Auto-increment

1: AOFF is incremented by 1. The software should ensure not to cause a wrap condition. Byte-wise read/write is not supported when auto-increment is enabled.
0: AOFF is not automatically incremented. The software should program the correct address offset for each access.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **COM**: Command type

This bit indicates the register access type:
1: Indicates a read operation.
0: Indicates a write operation.

Bit 0 **OB**: Operation Busy.

This bit is set along with a read or write command to initiate an indirect access to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*. This bit is reset when the read or write command to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* is complete. The next indirect register access can be initiated only after this bit is reset. During a write operation, the bit is reset only after the data has been written to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* register. During a read operation, the data should be read from the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* only after this bit is reset.

MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)

Address offset: 0x0A74

Reset value: 0x0000 0000

This register contains the type, associated queue number related to Type-based RXQ mappings. It holds the data for eight indirect registers, defined by the AOFF[7:0] bitfield in the *MAC indirect access control register (ETH_MAC_IACR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMRQ[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:16 **TMRQ[2:0]**: Type Match Rx Queue Number

Indicates the receive queue number to which the packet needs to be forwarded on detecting a type match.

Bits 15:0 **TYP[15:0]**: Type field Value

Indicates the type value of packet that needs to be compared with the received Ethernet packet. This field is valid when programmed to a value greater than or equal to 0x0600.

Timestamp control register (ETH_MACTSCR)

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the receiver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]	
			rw				rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	PTGE	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **AV8021ASMEN**: AV 802.1AS Mode Enable

When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS operating mode.

When the PTP offload feature is enabled, for PTP offload, the transport-specific field in the PTP header is generated and checked based on the value of this bit.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TXTSSTSM**: Transmit Timestamp Status Mode

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering

When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.

When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.

For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.

For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTYPSEL[1:0]**: Select PTP packets for Taking Snapshots

These bits, along with Bits 15 and 14, define the set of PTP packet types for which snapshot needs to be taken. The encoding is given in *Table 521: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master

When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages

When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see *Table 521: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 13 **TSIPV4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPV6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets

When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format

When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control

When this bit is set, the Timestamp Low register rolls over after 0x3B9A C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of subsecond register is 0x7FFF FFFF. The subsecond increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets

When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PTGE**: Presentation Time Generation Enable

When this bit is set the Presentation Time generation is enabled.

Bit 5 **TSADDREG**: Update Addend Register

When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TSUPDT**: Update Timestamp

When this bit is set, the system time is updated (added or subtracted) with the value specified in *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.
When Media Clock Generation and Recovery is enabled, ETH_MACPRSTIMUR should also be updated before setting this field.
This bit should be zero before updating it. This bit is reset when the update is complete in hardware.

Bit 2 **TSINIT**: Initialize Timestamp

When this bit is set, the system time is initialized (overwritten) with the value specified in the *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.
When Media Clock Generation and Recovery is enabled, *MAC presentation time update register (ETH_MACPRSTIMUR)* should also be updated before setting this field.
This bit should be zero before it is updated. This bit is reset when the initialization is complete.

Bit 1 **TSCFUPDT**: Fine or Coarse Timestamp Update

When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0 **TSENA**: Enable Timestamp

When this bit is set, the timestamp is added for transmit and receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.
On the receive side, the MAC processes the 1588 packets only if this bit is set.

Subsecond increment register (ETH_MACSSIR)

Address offset: 0x0B04

Reset value: 0x0000 0000

This register specifies the value to be added to the internal system time register every cycle of clk_ptp_ref_i clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SSINC[7:0]**: Subsecond Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the subsecond register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [TSCTRLSSR bit is set in *Timestamp control register (ETH_MACTSCR)*]. When TSCTRLSSR is cleared, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:0 Reserved, must be kept at reset value.

System time seconds register (ETH_MACSTSR)

Address offset: 0x0B08

Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TSS[31:0]**: Timestamp Second

The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

System time nanoseconds register (ETH_MACSTNR)

Address offset: 0x0B0C

Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSSS[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field has the subsecond representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set in *Timestamp control register (ETH_MACTSCR)*, each bit represents 1 ns. The maximum value is 0x3B9A C9FF after which it rolls-over to zero.

System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10

Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in *Timestamp control register (ETH_MACTSCR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSS[31:0]**: Timestamp Seconds

The value in this field is the seconds part of the update.
When ADDSUB is reset, this field must be programmed with the seconds part of the update value.
When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value.
For example, to subtract 2.000000001 seconds from the system time, the TSS field in the ETH_MACSTSUR register must be 0xFFFF FFFE (that is, $2^{32} – 2$).

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0 in *System time nanoseconds update register (ETH_MACSTNUR)*. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, $2^{32} – 1$) and the *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] field = 0x7FFF FFFF)

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD SUB	TSSS[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADDSUB**: Add or Subtract Time

When this bit is set, the time value is subtracted with the contents of the update register.
When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field is the subseconds part of the update.

– ADDSUB is 1: This field must be programmed with the complement of the subseconds part of the update value as described.

– ADDSUB is 0: This field must be programmed with the subseconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the *Timestamp control register (ETH_MACTSCR)*.

– TSCTRLSSR field in the *Timestamp control register (ETH_MACTSCR)*is 1:
- The programmed value must be 10^9 – <subsecond value>.
- Each bit represents 1 ns and the programmed value should not exceed 0x3B9A C9FF.

– TSCTRLSSR field in the *Timestamp control register (ETH_MACTSCR)* is 0:
- The programmed value must be 2^{31} - <subsecond_value>.
- Each bit represents an accuracy of 0.46 ns.

For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the ETH_MACSTNUR register must be 0x7FFF FFFF (that is, 2^{31} – 1), when TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is reset and 0x3B9A C9FF (that is, 10^9 –1), when TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is set.

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] = 0) and the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, 2^{32} – 1).

Timestamp addend register (ETH_MACTSAR)

Address offset: 0x0B18

Reset value: 0x0000 0000

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the *Timestamp control register (ETH_MACTSCR)*). The content of this register is added to a 32-bit accumulator in every clock cycle of clk_ptp_ref_i and the system time is updated whenever the accumulator overflows.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSAR[31:0]**: Timestamp Addend Register

This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Timestamp status register (ETH_MACTSSR)

Address offset: 0x0B20

Reset value: 0x0000 0000

All bits except Bits[27:25] gets cleared when the application reads this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]			
		r	r	r	r	r	rc_r					rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
rc_r												rc_r	rc_r	rc_r	rc_r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **ATSNS[4:0]**: Number of Auxiliary Timestamp Snapshots

This field indicates the number of Snapshots available in the FIFO. A value equal to the depth of FIFO (4) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.

Bit 24 **ATSSTM**: Auxiliary Timestamp Snapshot Trigger Missed

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATSSTN[3:0]**: Auxiliary Timestamp Snapshot Trigger Identifier

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

Bit 16: Auxiliary trigger 0

Bit 17: Auxiliary trigger 1

Bit 18: Auxiliary trigger 2

Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

Bit 15 **TXTSSIS**: Tx Timestamp Status interrupt status

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the *Tx timestamp status seconds register (ETH_MACTXTSSSR)* is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 14:4 Reserved, must be kept at reset value.

Bit 3 **TSTRGTERR0**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR elapses (see *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **AUXTSTRIG**: Auxiliary Timestamp Trigger Snapshot

This bit is set high when the auxiliary snapshot is written to the FIFO.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **TSTARGT0**: Timestamp Target Time Reached

When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR registers (see *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
This bit is cleared when the application reads this bit (or writes of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Bit 0 **TSSOVF**: Timestamp Seconds Overflow

When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 0xFFFF FFFF.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Rx domain time increment register (ETH_MACRXDTI)

Address offset: 0x0B24

Reset value: 0x0000 0000

This register indicates the increment value of the PTP time maintained in the receive clock domain that is used for timestamping.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXNS[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **RXNS[15:0]**: Receive domain time increment value in nanoseconds

This bitfield indicates the nanosecond part of the increment value of the PTP time maintained in the receive clock domain that is used for timestamping.
For example, when operating in 2.5G speed, the clock period is 3.2 ns, so in digital rollover mode, the RXNS[15:0] bitfield must be programmed to 3. In binary rollover mode, 3 / 0.466 = 6.44, so the RXNS[15:0] bitfield must be programmed to 6. When RXNS[15:0] is zero, the fixed value based on the operating speed is used for PTP time increment in receive clock domain.

Bits 15:0 Reserved, must be kept at reset value.

Tx domain time increment register (ETH_MACTXDTI)

Address offset: 0x0B28

Reset value: 0x0000 0000

This register indicates the increment value of the PTP time maintained in the transmit clock domain that is used for timestamping.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXNS[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **TXNS[15:0]**: Transmit domain time increment value in nanoseconds

This bitfield indicates the nanosecond part of the increment value of the PTP time maintained in the transmit clock domain that is used for timestamping.

For example, when operating in 2.5G speed, the clock period is 3.2 ns, so in digital rollover mode, the TXNS[15:0] bitfield must be programmed to 3. In binary rollover mode, 3 / 0.466 = 6.44, so the TXNS[15:0] bitfield must be programmed to 6. When TXNS[15:0] is zero, the fixed value based on operating speed is used for PTP time increment in transmit clock domain.

Bits 15:0 Reserved, must be kept at reset value.

Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)

Address offset: 0x0B30

Reset value: 0x0000 0000

This register contains the nanosecond part of timestamp captured for transmit packets when Tx status is disabled.The *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register, along with *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, gives the 64-bit timestamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* is read (bits[31:24]).

If the application does not read these registers and timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the setting of the TXTSSTSM bit of the *Timestamp control register (ETH_MACTSCR)*. The status bit TXTSSIS bit [15] in *Timestamp status register (ETH_MACTSSR)* register is set when the MAC transmitter captures the timestamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSMIS	TXTSSLO[30:16]														
r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSLO[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bit 31 **TXTSSMIS**: Transmit Timestamp Status Missed

When this bit is set, it indicates one of the following:

– The timestamp of the current packet is ignored if TXTSSTSM bit of the *Timestamp control register (ETH_MACTSCR)* is reset

– The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the *Timestamp control register (ETH_MACTSCR)* is set.

Bits 30:0 **TXTSSLO[30:0]**: Transmit Timestamp Status Low

This field contains the 31 bits of the Nanoseconds field of the transmit packet's captured timestamp.

Tx timestamp status seconds register (ETH_MACTXTSSSR)

Address offset: 0x0B34

Reset value: 0x0000 0000

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXTSSHI[31:0]**: Transmit Timestamp Status High

This field contains the lower 32 bits of the Seconds field of transmit packet's captured timestamp.

Auxiliary control register (ETH_MACACR)

Address offset: 0x0B40

Reset value: 0x0000 0000

The auxiliary timestamp control register controls the auxiliary timestamp snapshot.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATSEN3**: Auxiliary Snapshot 3 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg3 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 6 **ATSEN2**: Auxiliary Snapshot 2 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg2 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 5 **ATSEN1**: Auxiliary Snapshot 1 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg1 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg0 input is enabled. When this bit is reset, the events on this input are ignored.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear

When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

Auxiliary timestamp nanoseconds register (ETH_MACATSNR)

Address offset: 0x0B48

Reset value: 0x0000 0000

The *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*, along with *Auxiliary timestamp seconds register (ETH_MACATSSR)*, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in *Timestamp status register (ETH_MACTSSR)* indicate the fill-level of the FIFO. The top of the FIFO is removed only when *Auxiliary timestamp seconds register (ETH_MACATSSR)* is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXTSLO[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSLO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp

Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

Auxiliary timestamp seconds register (ETH_MACATSSR)

Address offset: 0x0B4C

Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUXTSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUXTSHI[31:0]**: Auxiliary Timestamp

Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

Timestamp ingress asymmetric correction register (ETH_MACTSIACR)

Address offset: 0x0B50

Reset value: 0x0000 0000

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTIAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTIAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction

This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000.

The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Timestamp egress asymmetric correction register (ETH_MACTSEACR)

Address offset: 0x0B54

Reset value: 0x0000 0000

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTEAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTEAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction

This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}.

For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD 8000, which is the 2's complement of 0x0002 8000(2.5×2^{16}). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003 4CCC (3.3×2^{16}).

Timestamp ingress correction nanosecond register (ETH_MACTSICNR)

Address offset: 0x0B58

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSIC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSIC[31:0]**: Timestamp Ingress Correction

This field contains the ingress path correction value as defined by the Ingress Correction expression.

Timestamp egress correction nanosecond register (ETH_MACTSECNR)

Address offset: 0x0B5C

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSEC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSEC[31:0]**: Timestamp Egress Correction

This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

Timestamp ingress latency register (ETH_MACTSILR)

Address offset: 0x0B68

Reset value: 0x0000 0000

This register holds the ingress MAC latency.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ITLNS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ITLNS[11:0]**: Ingress Timestamp Latency, in nanoseconds

This register holds the average latency (in nanoseconds) between the MAC input ports and the actual point (MII) where the ingress timestamp is taken. Ingress correction value is computed as described in *Section : Ingress correction*.

Bits 15:8 **ITLSNS[7:0]**: Ingress Timestamp Latency, in subnanoseconds

This register holds the average latency (in subnanoseconds) between the MAC input ports and the actual point (MII) where the ingress timestamp is taken. ingress correction value is computed as described in *Section : Ingress correction*.

Bits 7:0 Reserved, must be kept at reset value.

Timestamp egress latency register (ETH_MACTSELR)

Address offset: 0x0B6C

Reset value: 0x0000 0000

This register holds the Egress MAC latency.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ETLNS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ETLNS[11:0]**: Egress Timestamp Latency, in nanoseconds

This register holds the average latency in nanoseconds between the actual point (MII) where the egress timestamp is taken and the output ports of the MAC. Egress correction value is computed as described in *Section : Egress correction*.

Bits 15:8 **ETLSNS[7:0]**: Egress Timestamp Latency, in subnanoseconds

This register holds the average latency in subnanoseconds between the actual point (MII) where the egress timestamp is taken and the output ports of the MAC. Egress correction value is computed as described in *Section : Egress correction*.

Bits 7:0 Reserved, must be kept at reset value.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TIMESEL**: Time Select

When this bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input
When this bit is reset, presentation time is used to capture time at trigger input, maintaining backward compatibility

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **MCGREN0**: MCGR Mode Enable for PPS0 Output

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set it operates in MCGR mode and on reset it operates in PPS mode.

0: PPS mode

1: MCGR mode

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Enables MCGR Interrupt whose status bit is indicated by TSTARGT0 bit in *Timestamp status register (ETH_MACTSSR)* register

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, PPSCTRL[3:0] function as PPSCMD[3:0]. When this bit is reset, PPSCTRL[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCTRL[3:0]**: PPS Output Frequency Control

This field controls the frequency of the PPS output (eth_ptp_pps_out) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:

0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.

0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.

0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.

0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

..

1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

Note: In the binary rollover mode, the PPS output (eth_ptp_pps_out) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- *When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.*
- *When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of*
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high).
- *When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of*
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.

PPS control register [alternate] (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TIMESEL**: Time Select

When this bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input
When this bit is reset, presentation time is used to capture time at trigger input, maintaining backward compatibility

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **MCGREN0**: MCGR Mode Enable for PPS Output

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set it operates in MCGR mode and on reset it operates in PPS mode.
0: PPS mode
1: MCGR mode

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) mode for PPS output signal:
00: Target Time registers are programmed only for generating the interrupt event.
01: Enables MCGR Interrupt whose status bit is indicated by TSTARGT0 bit in *Timestamp status register (ETH_MACTSSR)* register
10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.
11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD[3:0]. When this bit is reset, Bits[3:0] function as PPSCTRL(Fixed PPS mode).

Bits 3:0 **PPSCMD[3:0]:** Flexible PPS Output (eth_ptp_pps_out) Control

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are 'all zero'. The following list describes the values of PPSCMD0:

0000: No Command

0001: START Single Pulse.

This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.

0010: START Pulse Train.

This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.

0011: Cancel START.

This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.

0100: STOP Pulse Train at time.

This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010) after the time programmed in the Target Time registers elapses.

0101: STOP Pulse Train immediately.

This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010).

0110: Cancel STOP Pulse train.

This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.

0111 to 1111: Reserved, must not be used

PPS target time seconds register (ETH_MACPPSTTSR)

Address offset: 0x0B80

Reset value: 0x0000 0000

The PPS output Target Time Seconds register, along with *PPS target time nanoseconds register (ETH_MACPPSTTNR)*, is used to schedule an interrupt event (Bit TSSOVF of *Timestamp status register (ETH_MACTSSR)*) when the system time exceeds the value programmed in these registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSTRH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTRH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSTRH0[31:0]**: PPS Target Time Seconds Register

This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the *PPS control register (ETH_MACPPSCR)*.

If the PTGE field of *Timestamp control register (ETH_MACTSCR)* is set with Presentation time control set in Recovery mode, then these bits indicate the TPT being programmed by the application and in Generation mode it indicates the CPT generated at the sampled trigger.

PPS target time nanoseconds register (ETH_MACPPSTTNR)

Address offset: 0x0B84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRGTBUSY0	TTSL0[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TRGTBUSY0**: PPS Target Time Register Busy

The MAC sets this bit when the PPSCMD0 field in the *PPS control register (ETH_MACPPSCR)* is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers with the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

Bits 30:0 **TTSL0[30:0]**: Target Time Low for PPS Register

This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in *PPS control register (ETH_MACPPSCR)*.

When the TSCTRLSSR bit is reset in the *Timestamp control register (ETH_MACTSCR)*, this value should be (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of subsecond increment value.

When the TSCTRLSSR bit is set in the *Timestamp control register (ETH_MACTSCR)*, this value should not exceed 0x3B9A C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of subsecond increment value.

PPS interval register (ETH_MACPPSIR)

Address offset: 0x0B88

Reset value: 0x0000 0000

The PPS Interval register contains the number of units of subsecond increment value between the rising edges of PPS output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSINT0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSINT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSINT0[31:0]**: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of subsecond increment value), you should program value 4 (5-1) in this register.

PPS width register (ETH_MACPPSWR)

Address offset: 0x0B8C

Reset value: 0x0000 0000

The PPS Width register contains the number of units of subsecond increment value between the rising and corresponding falling edges of PPS output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSWIDTH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSWIDTH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSWIDTH0[31:0]**: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of subsecond increment value), you should program value 3 (4-1) in this register.

Note: The value programmed in this register must be lesser than the value programmed in PPS interval register (ETH_MACPPSIR).

PTP offload control register (ETH_MACPOCR)

Address offset: 0x0BC0

Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DN[7:0]								PDRDIS	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DN[7:0]**: Domain Number

This field indicates the domain Number in which the PTP node is operating.

Bit 7 **PDRDIS**: Disable Peer Delay Response response generation

When this bit is set, the Peer Delay Response (Pdelay_Resp) response is not generated for received Peer Delay Request (Pdelay_Req) request packet, as required by the programmed mode.

Setting this bit to 1 affects the normal PTP Offload operation and the time synchronization. So, this bit must be set only if there is problem with Pdelay_Resp generation in hardware and/or Pdelay_Resp generation is handled by software.

Bit 6 **DRRDIS**: Disable PTO Delay Request/Response response generation

When this bit is set, the Delay Request and Delay response are not generated for received SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 **APDREQTRIG**: Automatic PTP Pdelay_Req message Trigger

When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the APDREQEN bit for this operation.

Bit 4 **ASYNCTRIG**: Automatic PTP SYNC message Trigger

When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the ASYNCEN bit for this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **APDREQEN**: Automatic PTP Pdelay_Req message Enable

When this bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer Transparent mode.

Bit 1 **ASYNCEN**: Automatic PTP SYNC message Enable

When this bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Clock Master mode.

Bit 0 **PTOEN**: PTP Offload Enable

When this bit is set, the PTP Offload feature is enabled.

PTP source port identity 0 register (ETH_MACSPI0R)

Address offset: 0x0BC4

Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI0[31:0]**: Source Port Identity 0
This field indicates bits [31:0] of sourcePortIdentity of PTP node.

PTP Source port identity 1 register (ETH_MACSPI1R)

Address offset: 0x0BC8

Reset value: 0x0000 0000

This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI1[31:0]**: Source Port Identity 1
This field indicates bits [63:32] of sourcePortIdentity of PTP node.

PTP Source port identity 2 register (ETH_MACSPI2R)

Address offset: 0x0BCC

Reset value: 0x0000 0000

This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SPI2[15:0]**: Source Port Identity 2
This field indicates bits [79:64] of sourcePortIdentity of PTP node.

Log message interval register (ETH_MACLMIR)

Address offset: 0x0BD0

Reset value: 0x0000 0000

This register contains the periodic intervals for automatic PTP packet generation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LMPDRI[7:0]**: Log Min Pdelay_Req Interval
This field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11 Reserved, must be kept at reset value.

Bits 10:8 **DRSYNCR[2:0]**:
Delay_Req to SYNC Ratio
In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.
0: DelayReq generated for every received SYNC
1: DelayReq generated every alternate reception of SYNC
2: for every 4 SYNC messages
3: for every 8 SYNC messages
4: for every 16 SYNC messages
5: for every 32 SYNC messages
Others: Reserved, must not be used
The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0 **LSI[7:0]**:
Log Sync Interval
This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Ethernet MAC register map and reset values

Table 568. Ethernet MAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	ETH_MACCR	ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	Res.	JD	JE	Res.	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0004	ETH_MACECR	Res.	APDIM	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC	Res.	Res.	GPSL[13:0]														
	Reset value		0	0	0	0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	ETH_MACPFR	RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE	Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
	Reset value	0										0	0				0						0	0	0	0	0	0	0	0	0	0	0
0x000C	ETH_MACWJBTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PJE	Res.	Res.	Res.	Res.	JTO[3:0]					Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
	Reset value								0					0	0	0	0								0					0	0	0	0
0x0010	ETH_MACHT0R	HT31T0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	ETH_MACHT1R	HT63T32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018 - 0x004C	Reserved																																
0x0050	ETH_MACVTR	EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV	VL[15:0]																
	Reset value	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	Reserved																																
0x0058	ETH_MACVHTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLHT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	Reserved																																
0x0060	ETH_MACVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	ETH_MACiVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068 - 0x006C	Reserved																																

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0070	ETH_ MACQTXFCR	PT[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0			0	0
0x0074 - 0x008C	Reserved																																
0x0090	ETH_MACRXFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
	Reset value																															0	0
0x0094- 0x00AC	Reserved																																
0x00B0	ETH_MACISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIS	Res.	Res.	Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	Res.
	Reset value														0				0	0	0		0	0	0			0	0	0			
0x00B4	ETH_MACIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIE	Res.	Res.	Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	Res.
	Reset value														0				0	0	0							0	0	0			
0x00B8	ETH_ MACRXTXSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
	Reset value																								0			0	0	0	0	0	0
0x00BC	Reserved																																
0x00C0	ETH_MACPCSR	RWKFILTRS	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
	Reset value	0			0	0	0	0	0														0	0			0	0			0	0	0
0x00C4	ETH_ MACRWKPFR	MACRWKPFR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8 - 0x00CC	Reserved																																
0x00D0	ETH_MACLCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	Res.	PLS	LPIEN	Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
	Reset value											0	0	0		0	0							0	0					0	0	0	0
0x00D4	ETH_MACLTCR	Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]										TWT[15:0]															
	Reset value							1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	ETH_MACLETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00DC	ETH_ MAC1USTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
	Reset value																					0	0	0	0	0	1	1	0	0	0	1	1
0x00E0 - 0x010C	Reserved																																
0x0110	ETH_MACVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERVER[7:0]								SNPSVER[7:0]							
	Reset value	0x0000 3354																															
0x0114	ETH_MACDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
	Reset value														0	0	0														0	0	0
0x0118	Reserved																																
0x011C	ETH_MACHWF0R	ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]			MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL	Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	Reset value	0x0A0D 73F5																															
0x0120	ETH_MACHWF1R	Res.	L3L4FNUM[3:0]			Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN	ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					SPRAM	RXFIFOSIZE[4:0]				
	Reset value	0x1104 1924																															
0x0124	ETH_MACHWF2R	Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]			RDCSZ[1:0]		RXCHCNT[3:0]			Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]				
	Reset value	0x4100 0000																															
0x0128	ETH_MACHWF3R	Res.	Res.	ASP[1:0]		TBSSEL	FPESEL	Res.	Res.	Res.	Res.	ESTWID[1:0]		ESTDEP[2:0]			ESTSEL	Res.	FRPES[1:0]		FRPBS[1:0]		FRPSEL	PDUPSEL	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
	Reset value	0x0000 0020																															
0x012C - 0x01FC	Reserved																																
0x0200	ETH_ MACMDIOAR	Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]					Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	MB
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0204	ETH_ MACMDIODR	RA[15:0]																MD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0208 - 0x020C	Reserved																																
0x0210	ETH_MACARPAR	ARPPA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0214 - 0x022C	Reserved																																
0x0230	ETH_ MACCSRSWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
	Reset value																																0
0x0234- 0x23C	Reserved																																
0x0240	ETH_ MACPRSTIMR	MPTN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0244	ETH_ MACPRSTIMUR	MPTU[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0248 - 0x02FC	Reserved																																
0x0300	ETH_MACA0HR	AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	1																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0304	ETH_MACA0LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0308	ETH_MACA1HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30C	ETH_MACA1LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0310	ETH_MACA2HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0314	ETH_MACA2LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0318	ETH_MACA3HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x031C	ETH_MACA3LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0320 - 0x06FC	Reserved																																
0x0700	ETH_MMC_ CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
	Reset value																								0			0	0	0	0	0	0
0x0704	ETH_MMC_RX_ INTERRUPT	Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0708	ETH_MMC_TX_ INTERRUPT	Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x070C	ETH_MMC_RX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0710	ETH_MMC_TX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x0714 - 0x0748	Reserved																																
0x074C	ETH_TX_SINGLE_ COLLISION_ GOOD_PACKETS	TXSNGLCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0750	ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS	TXMULTCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0754 - 0x0764	Reserved																																
0x0768	ETH_TX_PACKET_COUNT_GOOD	TXPKTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x076C - 0x0790	Reserved																																
0x0794	ETH_RX_CRC_ERROR_PACKETS	RXCRCERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0798	ETH_RX_ALIGNMENT_ERROR_PACKETS	RXALGNERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x079C - 0x07C0	Reserved																																
0x07C4	ETH_RX_UNICAST_PACKETS_GOOD	RXUCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C8 - 0x07E8	Reserved																																
0x07EC	ETH_TX_LPI_USEC_CNTR	TXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F0	ETH_TX_LPI_TRAN_CNTR	TXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F4	ETH_RX_LPI_USEC_CNTR	RXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F8	ETH_RX_LPI_TRAN_CNTR	RXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07FC- 0x8FC	Reserved																																
0x0900	ETH_MACL3L4C0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0	L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0904	ETH_MACL4A0R	L4DP0[15:0]																L4SP0[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0908 - 0x090C	Reserved																																
0x0910	ETH_MACL3A00R	L3A00[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0914	ETH_MACL3A10R	L3A10[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0918	ETH_MACL3A20R	L3A20[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x091C	ETH_MACL3A30R	L3A30[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0920 - 0x092C	Reserved																																
0x0930	ETH_ MACL3L4C1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1	L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0934	ETH_MACL4A1R	L4DP1[15:0]																L4SP1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0938 - 0x093C	Reserved																																
0x0940	ETH_MACL3A01R	L3A01[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0944	ETH_MACL3A11R	L3A11[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0948	ETH_MACL3A21R	L3A21[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094C	ETH_MACL3A31R	L3A31[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0950 - 0x0A6C	Reserved																																
0x0A70	ETH_MAC_IACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSEL[3:0]				AOFF[7:0]								Res.	Res.	AUTO	Res.	Res.	Res.	COM.	OB
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0			0				0	0
0x0A74	ETH_MAC_ TMRQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMRW[2:0]			TYP[15:0]																	
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A78 - 0x0AFC	Reserved																																

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B00	ETH_MACTSCR	Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]		TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	PTGE	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
	Reset value				0				0						0	0	0	0	0	1	0	0	0	0	0		0	0		0	0	0	0
0x0B04	ETH_MACSSIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0																
0x0B08	ETH_MACSTSR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0C	ETH_MACSTNR	Res.	TSSS[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B10	ETH_MACSTSUR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B14	ETH_MACSTNUR	ADDSUB	TSSS[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B18	ETH_MACTSAR	TSAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B1C	Reserved																																
0x0B20	ETH_MACTSSR	Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]				TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
	Reset value			0	0	0	0	0	0					0	0	0	0	0												0	0	0	0
0x0B24	ETH_MACRXDTI	RXNS[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x0B28	ETH_MACTXDTI	TXNS[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x0B2C	Reserved																																
0x0B30	ETH_ MACTXTSSNR	TXTSSMIS	TXTSSLO[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B34	ETH_ MACTXTSSR	TXTSSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B38 - 0x0B3C	Reserved																																
0x0B40	ETH_MACACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
	Reset value																									0	0	0	0				0
0x0B44	Reserved																																
0x0B48	ETH_MACATSNR	Res.	AUXTSLO[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4C	ETH_MACATSSR	AUXTSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B50	ETH_MACTSIACR	OSTIAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B54	ETH_MACTSEACR	OSTEAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B58	ETH_ MACTSICNR	TSIC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B5C	ETH_ MACTSECNR	TSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B60 - 0x0B64	Reserved																																
0x0B68	ETH_ MACTSILR	Res.	Res.	Res.	Res.	ITLNS[11:0]												ITLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0B6C	ETH_ MACTSELR	Res.	Res.	Res.	Res.	ETLNS[11:0]												ETLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0B70	ETH_MACPPSCR	Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
	Reset value				0																					0	0	0	0	0	0	0	0

Table 568. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B70	ETH_MACPPSCR (alternate)	Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
	Reset value				0																					0	0	0	0	0	0	0	0
0x0B74 - 0x0B7C	Reserved																																
0x0B80	ETH_ MACPPSTTSR	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B84	ETH_ MACPPSTTNR	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B88	ETH_MACPPSIR	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8C	ETH_MACPPSWR	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B90 - 0x0BBC	Reserved																																
0x0BC0	ETH_MACPOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DN[7:0]								PRDIS	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0BC4	ETH_MACSPI0R	SPI0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC8	ETH_MACSPI1R	SPI1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BCC	ETH_MACSPI2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BD0	ETH_MACLMIR	LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
	Reset value	0	0	0	0	0	0	0	0														0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

49 Debug support (DBG)

49.1 DBG introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components.

- SWJ-DP: JTAG/serial-wire debug port
- AHB-AP: AHB access port
- ROM table
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watch-point and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace Macrocell™ (ETM)
- Cross trigger interface (CTI)
- Trace port interface unit (TPIU)

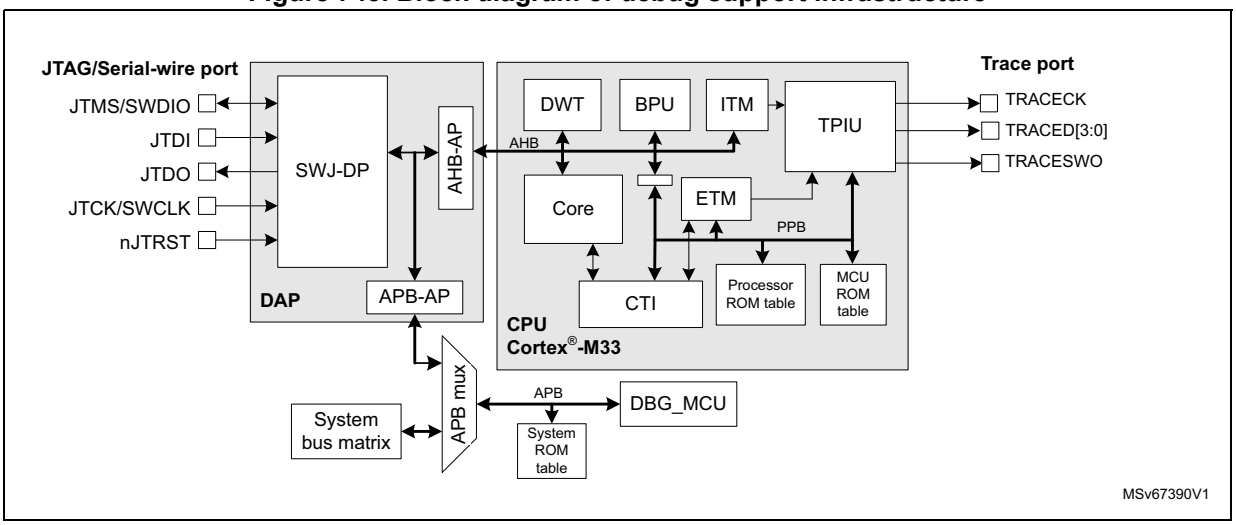
The debug features are accessible by the debugger via the AHB-AP.

Additional information can be found in the Arm documents referenced in *Section 49.13*.

49.2 DBG functional description

49.2.1 DBG block diagram

Figure 743. Block diagram of debug support infrastructure



49.2.2 DBG pins and internal signals

Table 569. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		Serial-wire debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial-wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial-wire clock	PA14
JTDI	I	JTAG test data input	-	-	PA15
JTDO	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

Table 570. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to the datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

Table 571. Single-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single-wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the serial-wire debug interface, and not when using JTAG.

49.2.3 DBG reset and clocks

The debug port (SWJ-DP) is reset by a power-on reset and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. This clock must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DCLK domain that covers the rest of the SWJ-DP and the access port.

The DCLK is a gated version of the system clock.

The DCLK domain is enabled by the debugger using CDBGPWRUPREQ. The DCLK is disabled at power-up, and must be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

Caution: To avoid a lockup situation when accessing the TPIU registers (except ID registers), it is necessary to activate the TRCENA bit (bit 24, trace enable) of the DEMCR register (0xE000EDFC) of the Cortex M33.

49.2.4 DBG power domains

The debug components are located in the core power domain. This means that the debugger connection is not possible in standby low-power mode. To avoid losing the connection when the device enters standby mode, the power can be maintained to the core by setting a bit in DBGMCU_CR. This also keeps the processor clocks active and holds off the reset, so that the debug session is maintained.

49.2.5 Debug and low-power modes

The devices include power-saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off or if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power-saving mode, but the debugger does not lose the connection.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information, refer to *Section 49.12: Microcontroller debug unit (DBGMCU)*.

49.2.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. To protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope.

Debugger access is disabled while the processor is booting from system flash memory.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features
  0: All debug features are disabled.
  1: Debug features are enabled.
- **niden**: enables trace and performance monitoring (noninvasive debug).
  0: Trace generation is disabled.
  1: Trace generation is enabled.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter or to the relevant Arm technical documentation.

The state of the signals is set according to the debug state as shown in the table below.

Table 572. Authentication signal states of the CPU

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 niden = 1	Debug and trace is enabled whatever the state of the processor. All memory and resources are accessible to the debugger.
CLOSED	dbgen = 1 spiden = 1 niden = 0 spniden = 0	Debug and trace is disabled.

The state of the authentication signals can be read from the DAUTHSTATUS register in the system control space (SCS) of the Cortex-M33.

Table 573. Authentication signal states of the DAP

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 (AHB-AP) deviceen = 1 (APB-AP)	Debug and trace is enabled whatever the state of the processor. All memory and resources are accessible to the debugger.
CLOSED	dbgen = 0 spiden = 1 deviceen = 1 (under reset)	Debug and trace is disabled. Debug usable under reset for RDP regression, transition and OEMKEY validation.

The state of the DAP AP authentication signals can be read from the MEM-AP CSW registers (AP0_CSWR and AP1_CSWR).

The debug state depends on the product lifecycle state (see *Section 3.7: Product life cycle*).

Table 574. Lifecycle state and debug states

Product lifecycle state	Debug state
RDP Level 0	OPEN
RDP Level 2 with Boundary Scan	CLOSED
RDP Level 2	CLOSED

49.2.7 OEMKEY and BSKEY password management

Readout protection (RDP) L2 and L2_wBS can be changed to L0 under the condition that a 128-bit OEMKEY was provisioned under RDP L0. That is an RDP regression.

Read-out protection L2_wBS can be changed to L2 thanks to the BSKEY. This is an RDP transition.

Figure 744. Product life cycle states and regression management

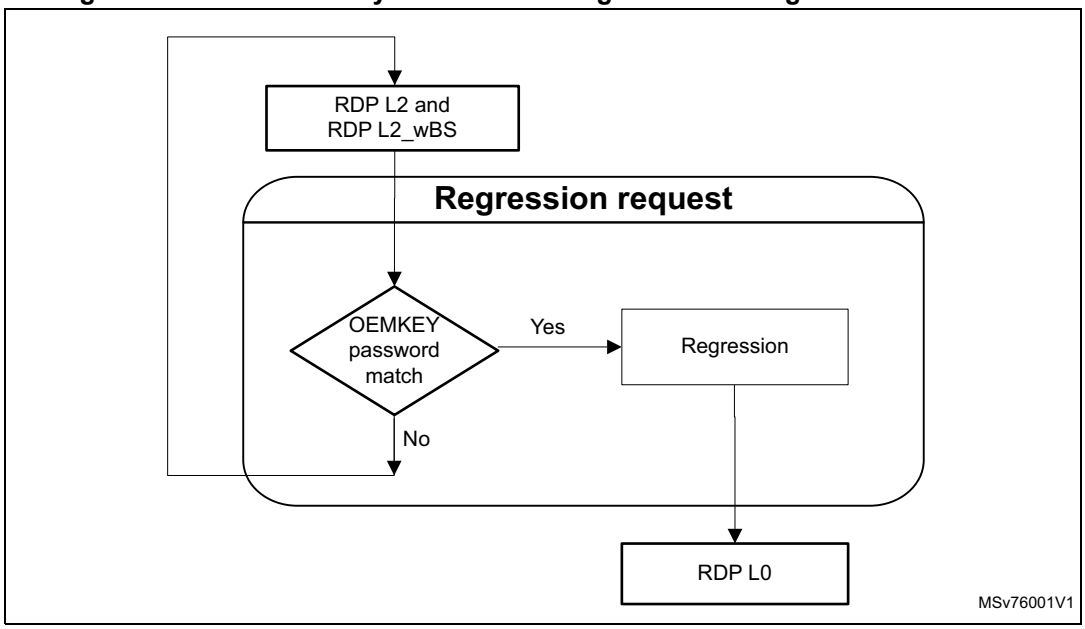


Figure 744 shows the product life cycle and regression mechanism. If the device is in RDP_STATE “L2” or “L2_BS”, the debug state is CLOSED. The regression procedure allows a full-regression if the adequate password (128-bit OEMKEY) is provided under reset to the DBGMCU_DBG_AUTH_HOST register, thanks to JTAG or SWD.

RDP regression erases the entire user flash memory area.

OEMKEY regression principle

The RDP regression protocol is based on a password (OEMKEY) provisioned under RDP_L0 to trigger the full regression action. The OEMKEY must be provisioned under RDP L0. After option-byte loading, the OEMLOCK bit in FLASH_SR indicates if an

OEMKEY has been provisioned. If reset, no OEMKEY is provisioned under RDP L0, and the regression is impossible.

The host can write this password to the DBGMCU_DBG_AUTH_HOST register via the JTAG/SWD interface. It can get information on product identification through the DBGMCU_DBG_AUTH_DEVICE register.

If the provided password matches the populated key, the regression is automatically started by hardware.

OEMKEY validation principle

The OEMKEY password is write only. Transitioning to RDP L2 or L2_wBS with an incorrect password loaded, means that it becomes impossible for the user to regress to RDP L0. In order to verify OEMKEY under RDP L0 prior transitioning to RDP L2 or RDP L2_wBS, a validation mechanism exists.

To validate the OEMKEY the host can write the password in DBGMCU_DBG_AUTH_HOST via the JTAG/SWD interface under reset and RDP L0.

Once the provided password has been compared to the OEMKEY, the VAL_RDY status flag is set in DBGMCU_DBG_VALR. The VAL_OEMKEY bit indicates then if the OEMKEY password provided for validation matches the provisioned one.

BSKEY transition principle

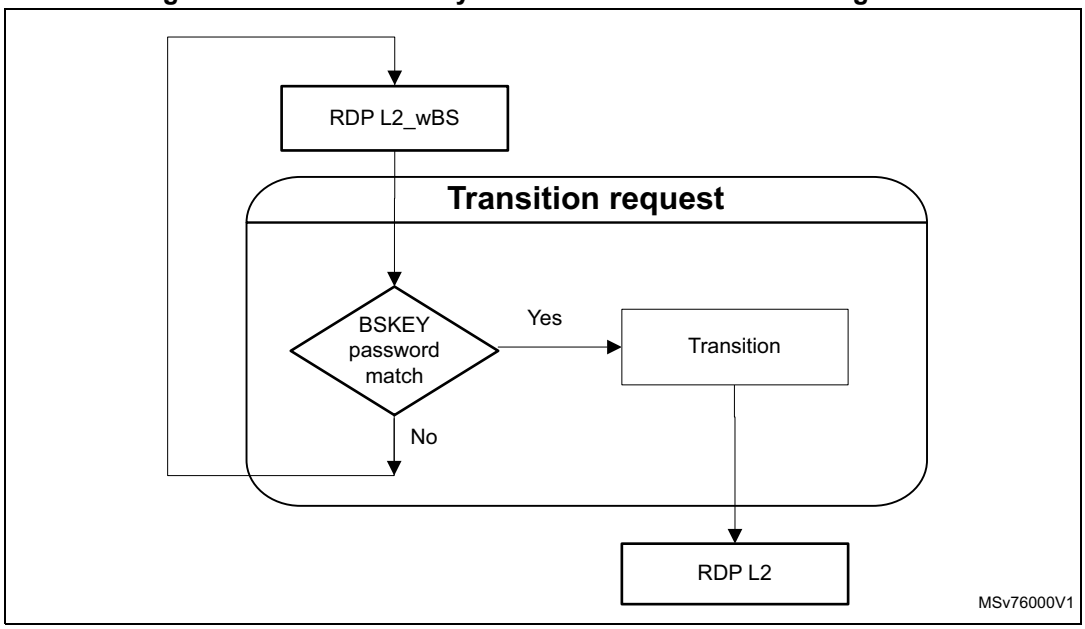
The RDP transition protocol to go from RDP L2_wBS to RDP_L2 (to close the boundary scan) is based on a password (BSKEY) populated under RDP_L0 to trigger the full regression action. A default value (0xAAAA AAAA) is preloaded during manufacturing.

The host can write this password to DBGMCU_BSKEY_HOST via the JTAG/SWD interface. It can get information on product identification through DBGMCU_DBG_AUTH_DEVICE.

If the provided password matches the provisioned key, the transition is automatically started by hardware.

Figure 745 shows the product life cycle states during a transition stage from RDP L2_wBS to L2. The debug state is CLOSED in both of these product lifecycles.

Figure 745. Product life cycle states and transition management



49.3 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- 5-pin standard JTAG interface (JTAG-DP)
- 2-pin (clock + data) serial-wire debug port (SW-DP)

These two modes are mutually exclusive, since they share the same I/O pins.

By default, the JTAG-DP is selected after a system or a power-on reset. The five I/O pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates the pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

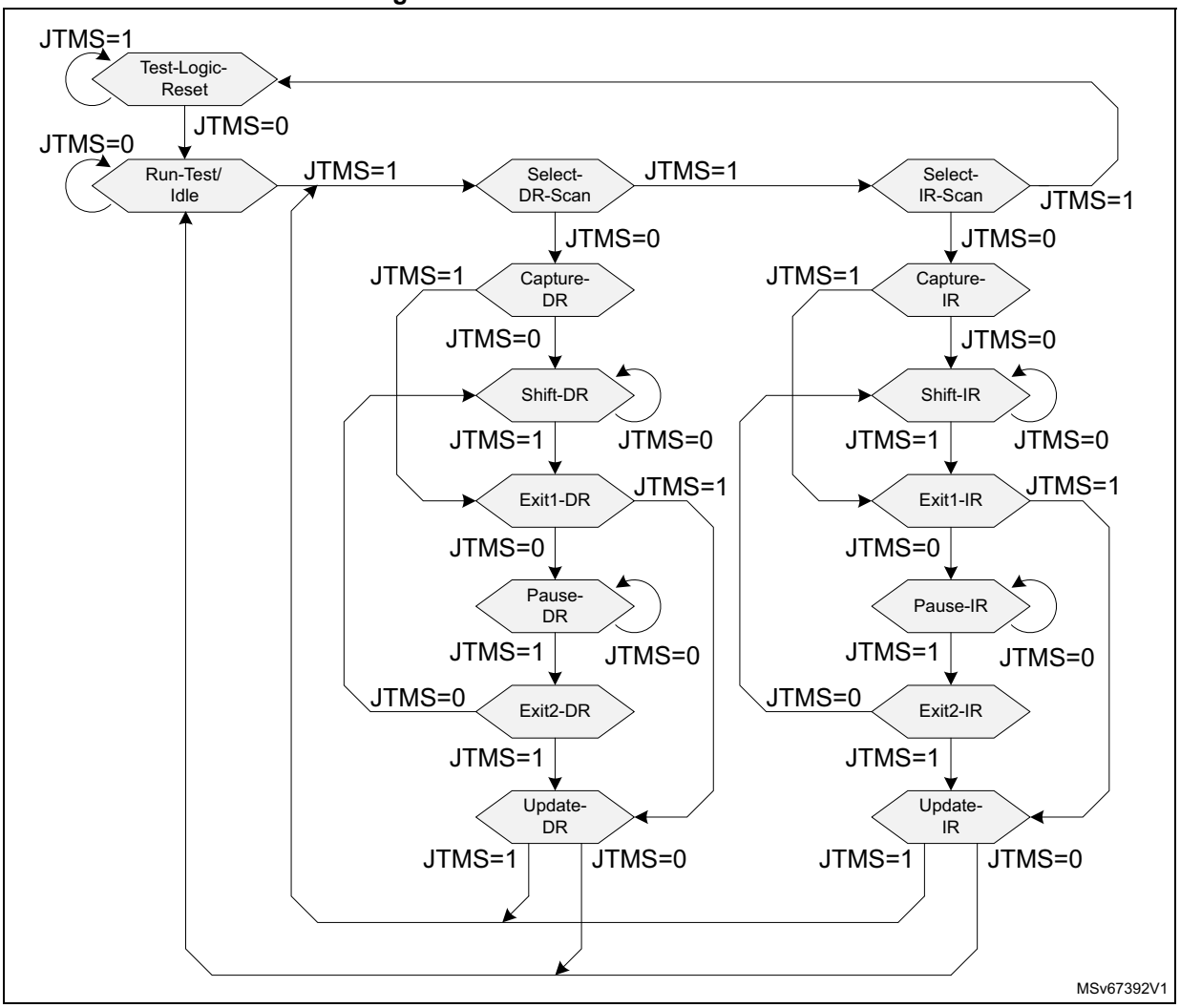
In SW-DP mode, the unused JTAG pins JTDI, JTDO, and nJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software, but debugging is no longer possible.

49.3.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in the figure below, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR) and the other one with a number of data registers (DR).

Figure 746. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred to the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
   - The LSB of the IR scan chain is output on JTDO.
   - Bit[n] of the IR scan chain is transferred to bit[n-1].
   - The value on JTDI is transferred to the MSB of the IR scan chain.

3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred to the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers to one of the DR scan chains, connected between JTDI and JTDO.
5. The value held in the instruction register determines which data register, and associated DR scan chain, are selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shifts in the Shift-IR state.
7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred to the selected data register.
8. When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the test-logic-reset state.

The data registers corresponding to the 4-bit IR instructions are listed in the table below.

Table 575. JTAG-DP data registers

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – bits 31:1 = reserved – bit 0 = APABORT: write 1 to generate an AP abort.
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request bits 2:1 = A[3:2] = 2-bit address of a debug port register bit 0 = RnW = read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: – 010 = OK/FAULT – 001 = WAIT – Other = Reserved

Table 575. JTAG-DP data registers (continued)

IR instruction	DR register	Scan chain length	Description
1011	APACC	35	Access port access register Initiates an access port and gives access to an access port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request bits 2:1 = A[3:2] = 2-bit sub-address of an access port register bit 0 = RnW= Read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: – 010 = OK/FAULT – 001 = WAIT – Other = Reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	Identification code 0x6BA0 0477: Cortex-M33 JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO.

The DR registers are described in more detail in the Arm Debug Interface Architecture Specification (see *Section 49.13*).

49.3.2 Serial-wire debug port

The serial-wire debug protocol uses the following pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional serial data

Serial data is transferred LSB first, synchronously with the clock.

A transfer comprises three phases:

1. Packet request (8 bits) transmitted by the host (see *Table 576*).
2. Acknowledge response (3 bits) transmitted by the target (see *Table 577*).
3. Data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see *Table 578*).

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock-cycle turn-around time is inserted.

Table 576. Packet request

Bit field	Name	Description
0	Start	Must be 1
1	APnDP	– 0: DP register access - see *Section 49.3.3: Debug port registers* – 1: AP register access - see *Section 49.4: Access ports*
2	RnW	– 0: write request – 1: read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 580* or *Table 582*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by target

Table 577. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 578. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single-bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case, the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the serial-wire debug protocol, refer to the Arm Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

49.3.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in *Table 580*.

The debugger can access the DP registers as follows:

1. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the data field with the write data. If using SWD, the A(3:2) and RnW fields are

part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 576*). The write data are sent in the data phase.

2. To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECTR register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the contents of the DP_SELECTR register.

Table 579. Debug port registers

Address	A(3:2) value	R/W	Description
0x0	00	R	DP_DPIDR contains the IDCODE for the debug port.
		W	DP_ABORTR(1) aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTATR register.
0x4	01	R/W	If DP_SELECTR.DPBANKSEL[3:0] = 0x0, DP_CTRLSTATR controls the DP and provides status information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x1, DP_DLCR(2) controls the operating mode of the SWD data link.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x2, DP_TARGETIDR provides target identification information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x3, DP_DLPIDR(2) provides the SWD protocol version.
0x8	10	R	DP_RESENDR(2) returns the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer.
		W	DP_SELECTR selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	DP read buffer register (DP_RDBUFFR): – Via JTAG-DP, this register allows the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation) – Via SW-DP, this register contains the result of the preceding AP read access, allowing a new AP access to be avoided.

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: Revision code
0x6: Rev 6

Bits 27:20 **PARTNO[7:0]**: Part number for the debug port
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: Minimal debug port (MINDP) implementation
0x0: MINDP not implemented

Bits 15:12 **VERSION[3:0]**: Debug port architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm JEDEC code

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORTR)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: Overrun error clear
0: No effect
1: STICKYORUN bit cleared in DP_CTRL/STATR register

Bit 3 **WDERRCLR**: Write data error clear
0: No effect
1: WDATAERR bit cleared in DP_CTRL/STATR register

Bit 2 **STKERRCLR**: Sticky error clear
0: No effect
1: STICKYERR bit cleared in DP_CTRL/STATR register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT**: Current AP transaction aborted if excessive number of WAIT responses returned
This bit indicates that the transaction is stalled.
0: No effect
1: Transaction aborted

DP control and status register (DP_CTRLSTATR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	r				r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK**: Debug power-up acknowledge

See description in *Section 49.2.5: Debug and low-power modes*.

0: DCLK gated

1: DCLK enabled

Bit 28 **CDBGPWRUPREQ**: Debug power-up request

This bit controls the DCLK enable request signal.

0: Requests DCLK gating

1: Requests DCLK enable

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR**: Write data error (read-only) in SW-DP

This bit indicates that there is a parity or framing error on the data phase of a write, or a write accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the ABORT.WDERRCLR bit.

0: No error

1: An error occurred.

Note: This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response (read-only) in SW-DP

This bit indicates the response to the last AP read access.

0: Read not OK

1: Read OK

Note: This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: Transaction error (read-only in SW-DP, read/write in JTAG-DP)

This bit indicates that an error occurred in an AP transaction.It is reset by writing 1 to STKERRCLR bit in DP_ABORTR (in SW-DP and JTAG-DP).

0: No error

1: An error occurred.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN**: Overrun (read-only in SW-DP, read/write in JTAG-DP)

This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. it is reset by writing 1 to the ORUNERRCLR bit in DP_ABORTR (in SW-DP and JTAG-DP).

0: No overrun

1: An overrun occurred.

Bit 0 **ORUNDETECT**: Overrun detection mode enable

0: Disabled

1: Enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
						r	r	r	r						

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**:Tristate period for SWDIO

0x0: 1 data bit period

Bits 7:6 **WIREMODE[1:0]**: SW-DP mode

0x0: Synchronous mode

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETIDR)

Address offset: 0x4

Reset value: 0xXXXX 0041

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: Target revision

Bits 27:12 **TPARTNO[15:0]**: Target part number
0x44E0: STM32C55x/562
0x44F0: STM32C53x/542
0x45A0: STM3259x/5A3

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: Target instance number
This bitfield defines the instance number for the device in a multi-drop system.
0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial-wire debug protocol version
0x1: version 2

DP event status register (DP_RESENDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: Value returned by the last AP read or DP_RDBUFF read
This register is used in the event of a corrupted read transfer.

DP access port select register (DP_SELECTR)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: Access port select

This bitfield selects the access port for the next transaction.
0x00: AP0 - System debug access port (APB-AP)
0x01: AP1 - Cortex-M33 debug access port (AHB-AP)
Other: Reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select

This bitfield selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select

This bitfield selects the register at address 0x4 of the debug port.
0x0: DP_CTRLSTAT register
0x1: DP_DLCR register
0x2: DP_TARGETID register
0x3: DP_DLPIDR register
Other: Reserved

DP read buffer register (DP_RDBUFFR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: Value returned by the last AP read access

The value returned by an AP read access can either be obtained using a second read access to the same address that initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

Debug port register map

These registers are not on the CPU memory bus. They are only accessed through the SW-DP and JTAG-DP debug interface.

The debug port address offset is 4-bit wide, where the two most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 580. Debug port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	DP_DPIDR	REVISION [3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION [3:0]				DESIGNER[10:0]											Res.
	Reset value	0	1	1	0	1	0	1	1	1	0	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1
0x0	DP_ABORTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
	Reset value																												0	0	0		0
0x4(1)	DP_CTRLSTATR	Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
	Reset value			0	0																					0	0	0				0	0
0x4(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND [1:0]		WIREMODE [1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0	0	0						
0x4(3)	DP_TARGETIDR	TREVISION [3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x4(4)	DP_DLPIDR	TINSTANCE [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN [3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x8	DP_RESENDR	RESEND[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	DP_SELECTR	APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL [3:0]				DPBANKSEL[3:0]			
	Reset value	x	x	x	x	x	x	x	x																	x	x	x	x	x	x	x	x
0xC	DP_RDBUFFR	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DP_SELECTR.DPBANKSEL[3:0] = 0x0.
2. DP_SELECTR.DPBANKSEL[3:0] = 0x1.
3. DP_SELECTR.DPBANKSEL[3:0] = 0x2.
4. DP_SELECTR.DPBANKSEL[3:0] = 0x3.

49.4 Access ports

There are two access ports (AP) attached to the DP.

- System debug access port (AP0): Enables access to the DBGMCU and the system ROM table via an APB bus.
- Cortex-M33 debug access port (AP1): Enables access to the debug and trace features integrated in the Cortex-M33 processor core via its internal AHB bus.

49.4.1 Access port registers

The access ports are of type MEM-AP: the debug and trace component registers are mapped in the address space of the AHB. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 582*.

The address of the AP registers is composed of the following fields:

- Bits [7:4]: content of APBANKSEL[3:0] in DP_SELECTR
- Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 575*), or of the SW-DP packet request (see *Table 576*), depending on the debug interface used
- Bits [1:0]: always set to 0

The content of the DP_SELECTR.APSEL[3:0] field defines which MEM-AP is being accessed.

Table 581. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	CSWR	Control/status word register
0x04	0x0	1	TAR	Transfer address register Target address for the bus transaction.
0x08	-	-	-	Reserved
0x0C	0x0	3	DRWR	Data read/write register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0]
0x10	0x1	0	BD0R	Banked data 0 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x0.
0x14	0x1	1	BD1R	Banked data 1 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x4.
0x18	0x1	2	BD2R	Banked data 2 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x8.
0x1C	0x1	3	BD3R	Banked data 3 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0xC.

Table 581. MEM-AP registers (continued)

Address	APBANKSEL	A(3:2)	Name	Description
0x20	-	-	-	Reserved
0x24 to 0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	0xF	1	CFGR	Configuration register (read only)
0xF8	0xF	2	BASER	Debug base address register (read only) Base address of the ROM table
0xFC	0xF	3	IDR	Identification register (read only)

The debugger can access the AP registers as follows:

1. Program APSEL[3:0] in DP_SELECTR to choose the AP, and APBANKSEL[3:0] in DP_SELECTR to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC data register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 576*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the *APx transfer address register (APx_TAR) (x = 0, 1)*.
2. Program AP1_CSWR, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from APx_DRWR (x = 0, 1) to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to APx_BDnR (x = 0, 1) triggers access to the TAR[31:4] + n address, allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register.

For more detailed information on the MEM-AP, refer to the Arm Debug Interface Architecture Specification *[1]*.

AP0 control/status word register (AP0_CSWR)

Address offset: 0x0

Reset value: 0x8000 0042

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBGSWEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		r	r	r

Bit 31 **DBGSWEN**: Software access enable
This bit enables or disables software access to the APB bus.
0: Software access disabled
1: Software access enabled

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: Mode of operation
0b0000: Normal download or upload
Other: Reserved

Bit 7 **TRINPROG**:Transfer in progress (read only)
This bitfield indicates whether a transfer is currently in progress on the APB master port.
0: No APB transfer in progress
1: APB transfer in progress

Bit 6 **DEVICEEN**: Device enable status (read only)
1: APB transfers always enabled

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode
This bitfield defines whether TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address incremented by the size in bytes of the transaction (SIZE field)
Other: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction
0x2: word (32-bit)

AP1 control/status word register (AP1_CSWR)

Address offset: 0x0

Reset value: 0x43X0 00X2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw	rw	rw	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
								r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: Secure transfer request
This bit sets the protection attribute HPROT[6] of the bus transfer.
0: Reserved
1: Nonsecure transfer, HPROT[6] = high

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **PROT[3:0]**: Bus transfer protection
This bitfield sets the protection attributes HPROT[3:0] of the bus transfer.
0bXXX0: Instruction access
0bXXX1: Data access
0bXX0X: User mode
0bXX1X: Privilege mode
0bX0XX: Non-bufferable
0bX1XX: Bufferable
0b0XXX: Non-shareable, no look-up, non-modifiable
0b1XXX: Shareable, look-up, modifiable

Bit 23 **SPISTATUS**: Secure debug authentication status
This bit indicates the state of the SPIDEN signal.
0: No secure AHB transfers allowed
1: Reserved

Bits 22:8 Reserved, must be kept at reset value.

Bit 7 **TRINPROG**: Transfer in progress (read only)
This bit indicates whether a transfer is currently in progress on the APB master port.
0: No AHB transfer in progress
1: AHB transfer in progress

Bit 6 **DBGSTATUS**: Debug enable (DBGEN) status
0: AHB transfers blocked
1: AHB transfers enabled

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode
This bitfield defines whether TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address incremented by the size in bytes of the transaction (SIZE field). Single transfer.
0x2: Address incremented by the size in bytes of the transaction (SIZE field). Packs four 8-bit transfers, or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface.
Other: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction
0x0: Byte (8-bit)
0x1: Half-word (16-bit)
0x2: Word (32-bit)
Other: Reserved

APx transfer address register (APx_TAR) (x = 0, 1)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TA[31:0]**: Address of current transfer. In AP0
TA[1:0] are fixed at 0.

APx data read/write register (APx_DRWR) (x = 0, 1)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TD[31:0]**: Data of current transfer

APx banked data n register (APx_BDnR) (x = 0, 1)

Address offset: 0x10 + 0x4 * n, (n = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]:** Banked data of current transfer to address TA [31:4] + 4 * n
Auto address incrementing is not performed on APx_BD0-3R. Banked transfers are only supported for word transfers.

APx base address register (APx_BASER) (x = 0, 1)

Address offset: 0xF8

Reset value: AP 0: 0xE00E 0003

Reset value: AP 1: 0xE00F E003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]**: Base address (bits 31 to 12) of the first ROM table
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
0xE00E0: AP0
0xE00FE: AP1

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base-address register format
1: Arm debug interface v5

Bit 0 **ENTRYPRESENT**: Debug components presence
This bit indicates that debug components are present on the access port bus.
1: Debug components present

APx identification register (APx_IDR) (x = 0, 1)

Address offset: 0xFC

Reset value: AP 0: 0x5477 0002

Reset value: AP 1: 0x8477 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS [3]
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLASS[2:0]			Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
r	r	r						r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: Revision number
0x5: r1p0 (AP0)
0x8: r0p9 (AP1)

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm

Bits 16:13 **CLASS[3:0]**:
0x1: MEM-AP

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**:
0x1: AHB-AP
0x2: APB-AP

Access port register map

These registers are not on the CPU memory bus. They are only accessed through SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 582. Access port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP0_CSWR	DBGSWEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value	1																				0	0	0	0	0	1	0	0		0	1	0
0x00	AP1_CSWR	Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		1			0	0	1	1	X												0	0	0	0	0	X	0	0		0	1	0
0x04	APx_TAR	TA[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	Reserved	Reserved																															
0x0C	APx_DRWR	TD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x10	APx_BD0R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	APx_BD1R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	APx_BD2R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	APx_BD3R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0xF4	Reserved	Reserved																															
0xF8	AP0_BASER	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0										1	1
0xF8	AP1_BASER	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0										1	1

Table 582. Access port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFC	AP0_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	1	0
0xFC	AP0_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	0	1

49.5 ROM tables

The ROM table is a CoreSight component that contains the base addresses of the CoreSight debug components accessible via the access port to which it is attached. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one top level ROM table behind each access port, APn. The base address of this ROM table can be obtained by reading the APn_BASER register of the access port. The top-level ROM table may point in turn to other ROM tables.

The system ROM table is pointed to by the AP0 base register, AP0_BASER. It contains the base address pointer for the DBGMCU.

The system ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00E 0000 to 0xE00E 0FFC, when accessed by the debugger.

Table 583. System ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	DBGMCU	0xE00E 4000 (debugger with AP0) 0x4402 4000 (CPU)	0x0000 4000	4	0x0000 4003
0x004	Top of table	-	-	-	0x0000 0000
0x008 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 586*

There are two ROM tables in the CPU subsystem. The MCU ROM table is pointed to by the AP1 base register, AP1_BASER. It contains the base-address pointer for the processor ROM table and for the TPIU registers.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

Table 584. MCU ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	Processor ROM table	0xE00F F000	0x0000 1000	4	0x0000 1003
0x004	TPIU	0xE004 0000	0xFFF4 2000	4	0xFFF4 2003
0x008	Reserved	-	-	-	0x1FF0 2002
0x00C	Reserved	-	-	-	0x1FF0 2002
0x010	Top of table	-	-	-	0x0000 0000
0x014 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 587*

The processor ROM table contains the base-address pointer for the system control space (SCS) registers that allow the debugger to identify the CPU core, as well as for the BPU, DWT, ITM, ETM, and CTI.

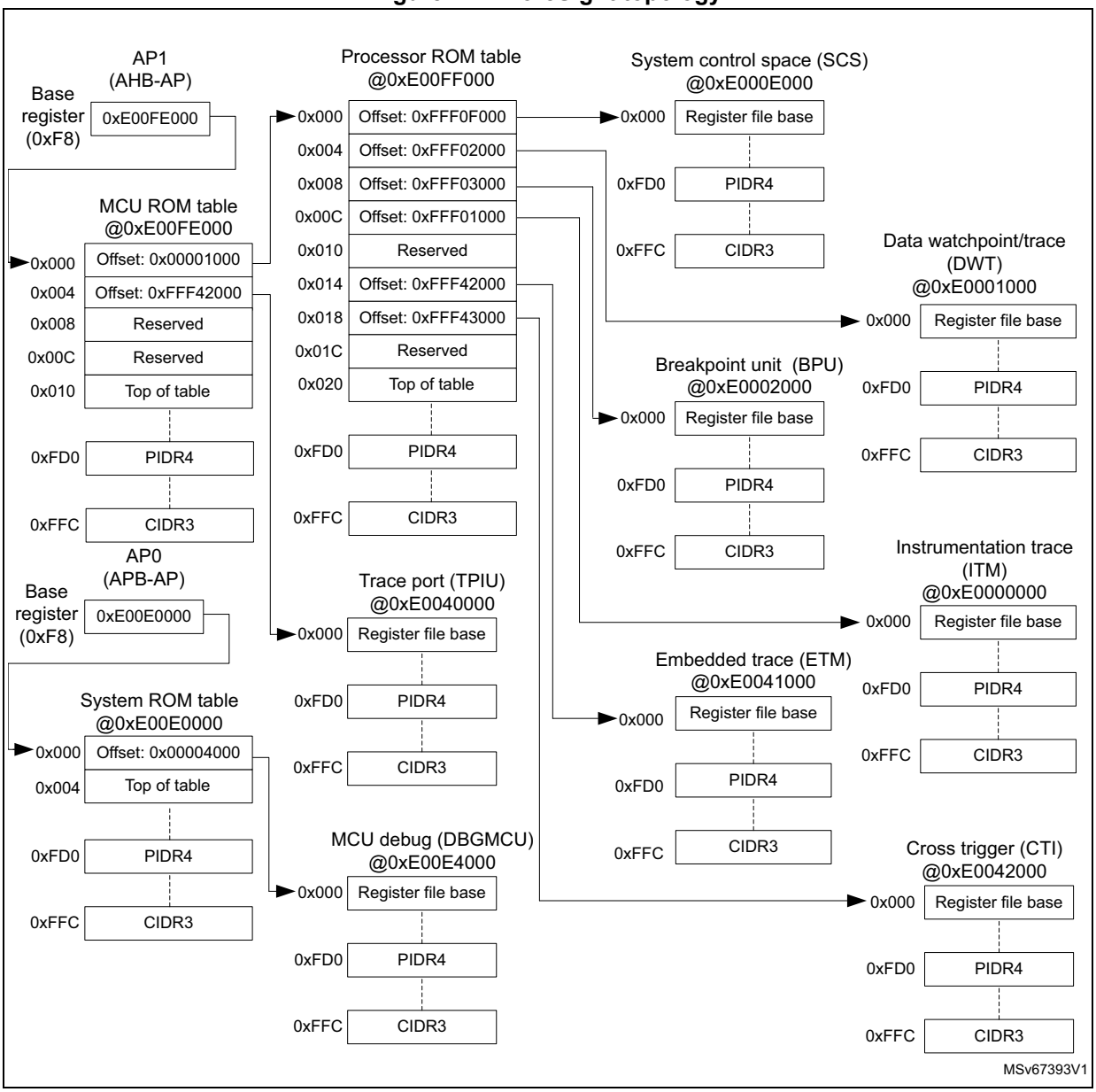
The processor ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

Table 585. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000	4	0xFFF0 2003
0xE00F F008	BPU	0xE000 2000	0xFFF0 3000	4	0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000	4	0xFFF0 1003
0xE00F F010	Reserved	-	-	-	0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000	4	0xFFF4 2003
0xE00F F018	CTI	0xE004 2000	0xFFF4 3000	4	0xFFF4 3003
0xE00F F01C	Reserved	-	-	-	0xFFF4 4002
0xE00F F020	Top of table	-	-	-	0x0000 0000
0xE00F F024 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	See *Table 588*

The topology for the CoreSight components is shown in the figure below.

Figure 747. CoreSight topology



49.5.1 System ROM table registers

System ROM memory type register (SYSROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory
0x1: System memory present on this bus

System ROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

System ROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x4E: STM32C5

System ROM CoreSight peripheral identity register 1(SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: STM32C5

System ROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

System ROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

System ROM CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

System ROM CoreSight peripheral identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

System ROM CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: common identification value

System ROM CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

System ROM table register map

Table 586. System ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	SYSROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1

Table 586. System ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-FDC	Reserved																																
0xFE0	SYSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0xFE4	SYSROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	X	X	X	X
0xFE8	SYSROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	SYSROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SYSROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SYSROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	SYSROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SYSROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 583: System ROM table* for register boundary addresses.

49.5.2 MCU ROM table registers

MCU ROM memory type register (MCUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory
0x1: System memory present on this bus

MCU ROM CoreSight peripheral identity register 4 (MCUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

MCU ROM CoreSight peripheral identity register 0 (MCUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x4E: STM32C5

MCU ROM CoreSight peripheral identity register 1(MCUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: STM32C5

MCU ROM CoreSight peripheral identity register 2 (MCUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

MCU ROM CoreSight peripheral identity register 3 (MCUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

MCU ROM CoreSight component identity register 0 (MCUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

MCU ROM CoreSight peripheral identity register 1 (MCUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

MCU ROM CoreSight component identity register 2 (MCUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: common identification value

MCU ROM CoreSight component identity register 3 (MCUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

MCU ROM table register map

Table 587. MCU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	MCUROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1

Table 587. MCU ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	MCUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-FDC	Reserved																																
0xFE0	MCUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0xFE4	MCUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	X	X	X	X
0xFE8	MCUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	MCUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	MCUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	MCUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	MCUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	MCUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 584: MCU ROM table* for register boundary addresses.

49.5.3 Processor ROM table registers

CPU ROM memory type register (CPUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory
1: System memory present on this bus

CPU ROM CoreSight peripheral identity register 4 (CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC continuation code

CPU ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xC9: Cortex-M33

CPU ROM CoreSight peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: Cortex-M33

CPU ROM CoreSight peripheral identity register 2 (CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

CPU ROM CoreSight peripheral identity register 3 (CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CPU ROM CoreSight component identity register 0 (CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

CPU ROM CoreSight peripheral identity register 1 (CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CPU ROM CoreSight component identity register 2 (CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: common identification value

CPU ROM CoreSight component identity register 3 (CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

Processor ROM table register map

Table 588. CPU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	CPUROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1

Table 588. CPU ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD4-0xFDC	Reserved																																
0xFD0	CPUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	CPUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	0	1	0	0	1
0xFE4	CPUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	CPUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CPUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CPUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CPUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	CPUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CPUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 585: Processor ROM table* for register boundary addresses.

49.6 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following:

- Watchpoint
- ETM trigger
- PC sampling trigger
- Data address sampling trigger
- Data comparator (COMP 1 only)
- Clock cycle counter comparator (COMP 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction

- Interrupt overhead

A DWT comparator compares the value held in its DWT_COMPxR with one of the following:

- A data address
- An instruction address
- A data value
- The cycle-count value, for COMP 0 only

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter debug state.

For more details on how to use the DWT, refer to the Armv8-M Architecture Reference Manual *[4]*.

49.6.1 DWT registers

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: Number of comparators implemented (read only)
0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read only)
0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read only)
0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read only)
0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read only)
0: Supported

Bit 23 **CYCDISS**: Cycle counter disabled secure
This bit controls whether the cycle counter is disabled in secure mode.
0: No effect
1: Not Applicable (STM32C5 always in non-secure state)

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA**: Enable for folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA**: Enable for sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA**: Enable for CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA**: Enable for exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: Enable for POSTCNT counter to be used as a timer for periodic PC sample packet generation
0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Position of the synchronization packet counter tap on the CYCCNT counter
This bitfield determines the synchronization packet rate.
00: Disabled, no synchronization packets
01: Tap at CYCCNT[24]
10: Tap at CYCCNT[26]
11: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Selection of the position of the POSTCNT tap on the CYCCNT counter
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter

Writes to this field are ignored if POSTCNT counter is enabled. CYCEVTENA or PCSAMPLENA bits must be reset prior to writing POSTINIT.

Bits 4:1 **POSTRESET[3:0]**: Reload value of the POSTCNT counter

Bit 0 **CYCCNTENA**: Enable of CYCCNT counter

0: Disabled

1: Enabled

DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CYCCNT[31:0]**: Processor clock-cycle counter

DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter

This bitfield counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNTR, and counts any instruction fetch stalls.

DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter
This bitfield counts the number of cycles spent in exception processing.

DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter
This bitfield counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: Load store counter
This bitfield counts additional cycles required to execute load and store instructions.

DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter
This bitfield increments on each instruction that takes 0 cycles.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EIASAMPLE[31:0]**: Executed instruction address sample value.
This bitfield samples the current value of the program counter.

DWT comparator x register (DWT_COMPxR)

Address offset: 0x020 + 0x10 * x, (x = 0 to 3)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COMP[31:0]**: Reference value for comparison

DWT function register 0 (DWT_FUNCTR0)

Address offset: 0x028

Reset value: 0x5800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

This bitfield identifies the capability for match for comparator 0.

0b01011: Cycle-counter, instruction-address, data-address, and data-address-with-value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

This bit indicates if a comparator match occurred since the register was last read.

0: No match

1: A match occurred.

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

This bitfield defines the size of the object being watched for by data-value and data-address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Debug event generated

0x2: For a cycle-counter, instruction-address, data-address, data-value, or linked-data-value comparator, generate a data-trace-match packet. For a data-address-with-value comparator, generate a data-trace-data-value packet.

0x3: For a data-address-limit comparator, generate a data-trace-data-address packet. For a cycle-counter, instruction-address-limit, or data-address comparator, generate a data-trace-PC-value packet. For a data-address-with-value comparator, generate both a data-trace-PC-value packet, and a data-trace-data-value packet.

Bits 3:0 **MATCH[3:0]**: Match type

This bitfield controls the type of match generated by comparator 0.

For possible values of this bitfield, refer to *[4]*.

DWT function register 1 (DWT_FUNCTR1)

Address offset: 0x038

Reset value: 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

This bitfield identifies the capability for match for comparator 1.

0b11010: Instruction-address, instruction-address-limit, data-address, data-address-limit, and data-address-with-value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

This bit indicates if a comparator match occurred since the register was last read.

0: No match

1: A match occurred.

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

This bitfield defines the size of the object being watched for by data value and data address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Debug event generated

0x2: For a cycle-counter, instruction-address, data-address, data-value, or linked-data-value comparator, generate a data-trace-match packet. For a data-address-with-value comparator, generate a data-trace-data-value packet.

0x3: For a data-address-limit comparator, generate a data-trace-data-address packet. For a cycle-counter, instruction-address-limit, or data-address comparator, generate a data-trace-PC-value packet. For a data-address-with-value comparator, generate both a data-trace-PC-value packet and a data-trace-data-value packet.

Bits 3:0 **MATCH[3:0]**: match type

This bitfield controls the type of match generated by comparator 1.

For possible values of this bitfield, refer to *[4]*.

DWT function register 2 (DWT_FUNCTR2)

Address offset: 0x048

Reset value: 0x5000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

This bitfield identifies the capability for match for comparator 2

0b01010: Instruction-address, data-address, and data-address-with-value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

This bit indicates if a comparator match occurred since the register was last read.

0: No match

1: A match occurred.

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

This bitfield defines the size of the object being watched for by data-value and data-address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Debug event generated

0x2: For a cycle-counter, instruction-address, data-address, data-value, or linked-data-value comparator, generate a data-trace-match packet. For a data-address-with-value comparator, generate a data-trace-data-value packet.

0x3: For a data-address-limit comparator, generate a data-trace-data-address packet. For a cycle-counter, instruction-address-limit, or data-address comparator, generate a data-trace-PC-value packet. For a data-address-with-value comparator, generate both a data-trace-PC-value packet and a data-trace-data-value packet.

Bits 3:0 **MATCH[3:0]**: match type

This bitfield controls the type of match generated by comparator 2.

For possible values of this bitfield, refer to *[4]*

DWT function register 3 (DWT_FUNCTR3)

Address offset: 0x058

Reset value: 0xF000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

This bitfield identifies the capability for match for comparator 3.

0b11110: Instruction-address, instruction-address-limit, data-address, data-address-limit, data-value, linked-data-value, and data-address-with-value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match

This bit indicates if a comparator match occurred since the register was last read.

0: No match

1: A match occurred.

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

This bitfield defines the size of the object being watched for by data-value and data-address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Debug event generated

0x2: For a cycle-counter, instruction-address, data-address, data-value, or linked-data-value comparator, generate a data-trace-match packet. For a data-address-with-value comparator, generate a data-trace-data-value packet.

0x3: For a data-address-limit comparator, generate a data-trace-data-address packet. For a cycle-counter, instruction-address-limit, or data-address comparator, generate a data-trace-PC-value packet. For a data-address-with-value comparator, generate both a data-trace-PC-value packet and a data-trace-data-value packet.

Bits 3:0 **MATCH[3:0]**: match type

This bitfield controls the type of match generated by comparator 3.

For possible values of this bitfield, refer to *[4]*

DWT device type architecture register (DWT_DEVARCHR)

Address offset: 0xFC8

Reset value: 0x4770 1A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DWT_DEVARCH register present
0x1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: DWT architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x1: DWT architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA02: DWT architecture

DWT device type register (DWT_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Subtype
0x0: Other

Bits 3:0 **MAJOR[3:0]**: Major type
0x0: Miscellaneous

DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Rgister file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: Cortex-M33 DWT part number

DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: Cortex-M33 DWT part number

DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

DWT register map

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

Table 589. DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DWT_CTRLR	NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DWT_CYCCNTR	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x00C	DWT_EXCCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x010	DWT_SLPCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	DWT_LSUCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x018	DWT_FOLDCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 589. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	DWT_COMP0R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x024	Reserved																																
0x028	DWT_FUNCTR0	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	1			0													0	0					0	0	0	0	0	0
0x02C	Reserved																																
0x030	DWT_COMP1R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x034	Reserved																																
0x038	DWT_FUNCTR1	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	1	1	0	1	0			0													0	0					0	0	0	0	0	0
0x03C	Reserved																																
0x040	DWT_COMP2R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x044	Reserved	Reserved																															
0x048	DWT_FUNCTR2	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x04C	Reserved																																
0x050	DWT_COMP3R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x054	Reserved	Reserved																															
0x058	DWT_FUNCTR3	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	1	1	1	1	0			0													0	0					0	0	0	0	0	0
0x05C-0xFC4	Reserved																																
0xFC8	DWT_DEVARCHR	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHVER[3:0]				ARCHPART[11:0]												
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0
0xFCC	DWT_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 589. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved																																
0xFE0	DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 585: Processor ROM table* for register boundary addresses.

49.7 Instrumentation trace macrocell (ITM)

The ITM generates trace information in packets. Three sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are the following:

- Software trace

  The software can write directly to any 32 x 32-bit ITM stimulus register to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.
- Hardware trace

  The DWT generates trace packets in response to a data trace event, a PC sample, or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.
- Local time-stamping

  The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

For more information on the ITM and how to use it, refer to *[4]*.

49.7.1 ITM registers

The ITM registers are located at the address range 0xE000 0000 to 0xE000 0FFC.

ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x4 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when writing

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	rw	rw

Bits 31:0 **STIMULUS[31:0]**: Trace output data

Write data is output on the trace bus as a software event packet.

ITM stimulus register x [alternate] (ITM_STIMRx)

Address offset: 0x000 + 0x4 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when reading

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DISABLE	FIFO_READY
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DISABLE**: Disable flag
- 0: Stimulus port and ITM enabled
- 1: Stimulus port and ITM disabled

Bit 0 **FIFO_READY**: FIFO ready indicator
- 0: Stimulus port buffer is full (or port is disabled).
- 1: Stimulus port can accept new write data.

ITM trace enable register (ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable

Each bit x(0 to 31) enables the stimulus port associated with the ITM_STIMRx register.
- 0: Port disabled
- 1: Port enabled

ITM trace privilege register (ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: Disable unprivileged access to ITM stimulus ports
Each bit controls eight stimulus ports.
XXX0: Unprivileged access permitted on ports 0 to 7
XXX1: Only privileged access permitted on ports 0 to 7
XX0X: Unprivileged access permitted on ports 8 to 15
XX1X: Only privileged access permitted on ports 8 to 15
X0XX: Unprivileged access permitted on ports 16 to 23
X1XX: Only privileged access permitted on ports 16 to 23
0XXX: Unprivileged access permitted on ports 24 to 31
1XXX: Only privileged access permitted on ports 24 to 31

ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw			rw	r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: ITM status
This bit indicates whether the ITM is currently processing events.
0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this bitfield.
Note: Different identifiers must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]**: Local timestamp prescaler, used with the trace packet reference clock
0x0: No prescaling
0x1: Divide by 4.
0x2: Divide by 16.
0x3: Divide by 64.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STALLENA**: Stall enable
0: Drop hardware source packets, and generate an overflow if the ITM output is stalled.
1: Stall the processor to guarantee delivery of data trace packets.

Bit 4 **SWOENA**: SWO enable
This bit enables asynchronous clocking of the timestamp counter (read only).
0: Timestamp counter uses processor clock.

Bit 3 **TXENA**: Transmit enable
This bit enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA**: Synchronization packet transmission enable
The debugger setting this bit must also configure DWT_CTRLR.SYNCTAP for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1 **TSENA**: Local timestamp generation enable
0: Disabled
1: Enabled

Bit 0 **ITMENA**: ITM enable
0: Disabled
1: Enabled

ITM device type architecture register (ITM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x1: ITM architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA01: ITM architecture

ITM device type register (ITM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Subtype
0x4: Associated with a bus, stimulus derived from bus activity

Bits 3:0 **MAJOR[3:0]**: Major type
0x3: Trace source

ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: ITM part number

ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: ITM part number

ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

ITM register map

The ITM registers are located at the address range 0xE000 0000 to 0xE000 0FFC.

Table 590. ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	ITM_STIMR0 to ITM_STIMR31	STIMULUS[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x07C-0xDFC	Reserved																																
0xE00	ITM_TER	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04-0xE3C	Reserved																																
0xE40	ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK [3:0]			
	Reset value																													0	0	0	0
0xE44-0xE7C	Reserved																																
0xE80	ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE [1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0							0	0			0	0	0	0	0	0
0xE84-0xFB8	Reserved																																
0xFBC	ITM_DEVARCHR	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[3:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0xFC0-0xFC8	Reserved																																
0xFCC	ITM_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0xFD0	ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved																																
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1

Table 590. ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFEC	ITM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ITM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 585: Processor ROM table* for register boundary addresses.

49.8 Breakpoint unit (BPU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

For more information on the breakpoint unit and how to use it, refer to *[4]*.

49.8.1 BPU registers

The BPU registers are located at address range 0xE0002000 to 0xE0002FFC.

BPU control register (BPU_CTRLR)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			Res.	Res.	Res.	Res.	NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r					r	r	r	r			rw	rw

Bits 31:28 **REV[3:0]**: Revision number
0x1: BPU version 2

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:12, 7:4 **NUM_CODE[6:0]**: Number of instruction address comparators supported
0x08: Eight instruction comparators supported

Bits 11:8, 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key
A write to FPB_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: FPB enable
0: Disabled
1: Enabled

BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPADDR[15:1]															BE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]**: Breakpoint address

Bit 0 **BE**: Breakpoint enable
0: Disabled
1: Enabled

BPU device type architecture register (BPU_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register present
0x1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: BPU architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x1: BPU architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA03: BPU architecture

BPU device type register (BPU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Subtype
0x0: Other

Bits 3:0 **MAJOR[3:0]**: Major type
0x0: Miscellaneous

BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: BPU part number

BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: BPU part number

BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

BPU CoreSight peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

BPU CoreSight component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

BPU CoreSight component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

BPU register map

The BPU registers are located at address range 0xE000 2000 to 0xE000 2FFC.

Table 591. BPU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	BPU_CTRLR	REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE [6:4]			Res.	Res.	Res.	Res.	NUM_CODE [3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0					1	0	0	0			0	0
0x004	Reserved	Reserved																															
0x008 to 0x024	BPU_COMP0R to BPU_COMP7R	BPADDR[31:1]																															BE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028 to 0xFB8	Reserved	Reserved																															
0xFBC	BPU_DEVARCHR	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
0xFC0 to 0xFC8	Reserved	Reserved																															
0xFCC	BPU_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	BPU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	BPU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	BPU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	BPU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	BPU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	BPU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	BPU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	BPU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1

Table 591. BPU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	BPU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 585: Processor ROM table* for register boundary addresses.

49.9 Embedded trace macrocell (ETM)

The ETM is a CoreSight component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M33 core to be traced. In the STM32C5, the ETM is configured for instruction trace only. Data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- Number of instructions executed in the same cycle
- Changes in program flow
- Current processor instruction state
- Addresses of memory locations accessed by load and store instructions
- Type, direction, and size of a transfer
- Condition code information
- Exception information
- Wait-for-interrupt state information

For more information, refer to the Arm CoreSight ETM-M33 Technical Reference Manual *[6]*.

49.9.1 ETM registers

The ETM registers are located at the address range 0xE004 1000 to 0xE004 1FFC.

ETM programming control register (ETM_PRGCTLR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Trace unit enable
0: Disabled
1: Enabled

ETM status register (ETM_STATR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: Stability status
This bit indicates that the ETM-M33 registers are stable and can be read.
0: Not stable
1: Stable

Bit 0 **IDLE**: Trace unit status
This bit indicates that the trace unit is inactive.
0: Not idle
1: Idle

ETM configuration register (ETM_CONFIGR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
			rw		rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RS**: Return stack enable
0: Disabled
1: Enabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:5 **COND[5:0]**: Conditional instruction tracing
0x0: Conditional instruction tracing disabled
0x1: Conditional load instructions traced
0x2: Conditional store instructions traced
0x3: Conditional load and store instructions traced
0x7: All conditional instructions traced

Bit 4 **CCI**: Cycle counting in instruction trace
0: Disabled
1: Enabled

Bit 3 **BB**: Branch broadcast mode
0: Disabled
1: Enabled

Bits 2:0 Reserved, must be kept at reset value.

ETM event control 0 register (ETM_EVENTCTL0R)

Address offset: 0x020

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: Resource type for event1
0: Single selected resource
1: Boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: Resource number based on TYPE1
This bitfield selects the resource number, based on the value of TYPE1.
When TYPE1 = 0, a single resource from 0-15 defined by SEL1[3:0] is selected.
When TYPE1 = 1, a boolean combined resource pair defined by SEL1[2:0] is selected.

Bit 7 **TYPE0**: Resource type for event0
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: Resource number based on TYPE0
This bitfield selects the resource number, based on the value of TYPE0.
When TYPE0 = 0, a single resource from 0-15 defined by SEL0[3:0] is selected.
When TYPE0 = 1, a boolean combined resource pair defined by SEL0[2:0] is selected.

ETM event control 1 register (ETM_EVENTCTL1R)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
			rw	rw										rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: Low-power state behavior override
0: Normal low-power state behavior
1: The resources and event trace generation are not affected by entry to a low-power state.

Bit 11 **ATB**: ATB trigger enable
0: Disabled
1: Enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: Instruction event generation
This bitfield enables generation of an event element in the instruction stream.
0bX0: Event0 does not cause an event element.
0bX1: Event0 causes an event element when it occurs.
0b0X: Event1 does not cause an event element.
0b1X: Event1 causes an event element when it occurs.

ETM stall control register (ETM_STALLCTLR)

Address offset: 0x02C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
					rw		rw					rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **INSTPRIORITY**: Instruction trace priority
This bit prioritizes instruction trace if instruction trace buffer space is less than LEVEL[3:0].
0: The ETM must not prioritize instruction trace.
1: The ETM can prioritize instruction trace.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ISTALL**: Processor stalling

This bit stalls processor based on instruction-trace-buffer space.

0: The ETM must not stall the processor.

1: The ETM can stall the processor.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LEVEL[3:0]**: Threshold at which stalling becomes active

This bitfield provides four levels. This level can be varied to optimize the level of invasion caused by stalling, balanced against the risk of a FIFO overflow.

0x0: Zero invasion, but greater risk of FIFO overflow

...

0xF: Maximum invasion but less risk of FIFO overflow

ETM synchronization period register (ETM_SYNCPR)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: Synchronization period

This bitfield defines the number of bytes of trace between trace synchronization requests as a total of the number of bytes generated by the instruction stream.

0xA: 1024 bytes

ETM cycle count control register (ETM_CCCTLR)

Address offset: 0x038

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: Instruction-trace cycle-count threshold

This bitfield sets the threshold value for instruction-trace cycle counting. The threshold represents the minimum interval between cycle-count trace packets.

ETM trace identification register (ETM_TRACEIDR)

Address offset: 0x040

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace identification to output onto the trace bus

This bitfield must be programmed with a unique value to differentiate it from other trace sources in the system.

Values 0x00 and 0x70-0x7F are reserved.

ETM ViewInst main control register (ETM_VICTLR)

Address offset: 0x080

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Exception level in secure state

This bitfield controls whether instruction tracing is enabled for the corresponding exception level, in secure state.

0bXXX0: Instruction trace not generated in secure state, for exception level 0.

0bXXX1: Instruction trace generated in secure state, for exception level 0.

0b0XXX: Instruction trace not generated in secure state, for exception level 3.

0b1XXX: Instruction trace generated in secure state, for exception level .3

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: Trace-system error exception

0: The system error exception is traced only if the instruction or exception immediately before the system error exception is traced.

1: The system error exception is always traced.

Bit 10 **TRCRESET**: Trace reset exception

0: The reset exception is traced only if the instruction or exception immediately before the reset exception is traced.

1: The reset exception is always traced.

Bit 9 **SSSTATUS**: Start/stop logic status
0: Stopped
1: Started

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: Event selector

ETM counter reload value register 0 (ETM_CNTRLDVR0)

Address offset: 0x140

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: Counter reload value
This value is loaded in to the counter each time the reload event occurs.

ETM identification register 8 (ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXSPEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXSPEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MAXSPEC[31:0]**: Maximum speculation depth
This bitfield indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.
0x0: The maximum trace speculation depth is zero.

ETM identification register 9 (ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: Number of P0 right-hand keys used
0x0: No P0 right-hand keys used in instruction trace

ETM identification register 10 (ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: Number of P1 right-hand keys used (including normal and special keys)
0x0: No P1 right-hand keys used in instruction trace

ETM identification register 11 (ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: Number of special P1 right-hand keys used
0x0: No special P1 right-hand keys used in any configuration

ETM identification register 12 (ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDKEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDKEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDKEY[31:0]**: Number of conditional instruction right-hand keys used (including normal and special keys)

0x1: One conditional instruction right-hand key implemented

ETM identification register 13 (ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDSPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDSPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDSPC[31:0]**: Number of special conditional instruction right-hand keys used

0x0: No special conditional instruction right-hand keys implemented

ETM implementation specific register 0 (ETM_IMSPECR0)

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: Implementation specific extension support

0x0: No implementation specific extensions are supported

ETM identification register 0 (ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x2800 06E1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1]
		r												r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
r		r	r	r	r	r		r	r	r	r	r	r	r	

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: Commit field meaning

This bit indicates the meaning of the commit field in some packets.

1: Commit mode 1

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **TRCEXDATA**: Trace data transfers for exceptions

This bit indicates support for the tracing of data transfers for exceptions and exception returns.

0: Not implemented

Bits 16:15 **QSUPP[1:0]**: Q element support

0: Not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: Conditional results tracing

This bitfield indicates how conditional results are traced.

0: The trace unit indicates only if a conditional instruction passes or fails its condition code check.

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported

0x1: Two events

Bit 9 **RETSTACK**: Return stack support

1: Two entry return stacks

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: Cycle-counting support

1: Cycle counting implemented

Bit 6 **TRCCOND**: Conditional instruction support

1: Conditional-instruction tracing implemented

Bit 5 **TRCBB**: Branch broadcast support

1: Branch-broadcast tracing implemented

Bits 4:3 **TRCDATA[1:0]**: Data tracing support

0x0: Data tracing not supported

Bits 2:1 **INSTP0[1:0]**: Support for tracing of load and store instructions as P0 elements

0x0: Not supported

Bit 0 Reserved, must be kept at reset value.

ETM identification register 1 (ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F421

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Trace unit designer
0x41: Arm

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: Major trace unit architecture version number
0x4: ETMv4

Bits 7:4 **TRCARCHMIN[3:0]**: Minor trace unit architecture version number
0x2: Minor revision 2

Bits 3:0 **REVISION[3:0]**: Implementation revision number
0x1: Implementation revision 1

ETM identification register 2 (ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: Cycle counter size
0x0: 12 bits

Bits 24:20 **DVSIZE[4:0]**: Data value size
0x0: Data value size not supported

Bits 19:15 **DASIZE[4:0]**: Data address size.
0x0: Data address size not supported

Bits 14:10 **VMIDSIZE[4:0]**: Virtual machine ID size
0x0: Virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: Context ID size
0x0: Context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: Instruction address size
0x4: Maximum 32-bit address size

ETM identification register 3 (ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0F09 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: ETM_STALLCTLR.NOOVERFLOW implementation
0: Not implemented

Bits 30:28 **NUMPROC[2:0]**: Number of processors available for tracing
0x0: One processor

Bit 27 **SYSSTALL**: System support for stall control of the processor
1: System supports stall control.

Bit 26 **STALLCTL**: Stall control support
1: ETM_STALLCTLR implemented

Bit 25 **SYNCPR**: Trace synchronization period support
1: ETM_SYNCPR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24 **TRCERR**: ETM_VICTLR.TRCERR implementation
0x1: Implemented

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Privilege levels implementation
0x9: Privilege levels thread and handler implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: Minimum value that can be programmed to TRCCCCTLR.THRESHOLD
This bitfield defines the minimum cycle-counting threshold.
0x4: Minimum of four-instruction trace cycles

ETM identification register 4 (ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0011 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: Number of virtual machine ID (VMID) comparators
0x0: VMID comparators not implemented

Bits 27:24 **NUMCIDC[3:0]**: Number of context ID comparators
0x0: Context ID comparators not supported

Bits 23:20 **NUMSSCC[3:0]**: Number of single-shot comparator controls
0x1: One single-shot comparator control implemented

Bits 19:16 **NUMRSPAIR[3:0]**: Number of resource selection pairs
0x1: Two resource selection pairs implemented

Bits 15:12 **NUMPC[3:0]**: Number of processor comparator inputs for the DWT
0x4: Four processor comparator inputs implemented

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPDAC**: Data address comparisons
0: Data address comparisons not supported

Bits 7:4 **NUMDVC[3:0]**: Number of data value comparators
0x0: No data value comparators implemented

Bits 3:0 **NUMACPAIRS[3:0]**: Number of address comparator pairs
0x0: No address comparator pairs implemented

ETM identification register 5 (ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: Reduced function counter
1: Counter 0 implemented as a reduced function counter

Bits 30:28 **NUMCNTR[2:0]**: Number of counters
0x1: One counter implemented.

Bits 27:25 **NUMSEQSTATE[2:0]**: Number of sequencer states
0x0: No sequencer states implemented.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: Low-power state override support
1: Low-power state override support implemented

Bit 22 **ATBTRIG**: ATB trigger support
1: ATB trigger support implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: Number of bits of trace identification
0x7: 7-bit trace identification implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: Number of external input selectors
0x0: No external input selectors implemented.

Bits 8:0 **NUMEXTIN[8:0]**: Number of external inputs
0x004: Four external inputs implemented.

ETM resource register 2 (ETM_RSCTLR2)

Address offset: 0x208

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: Result of a combined pair of resources inversion
0: Not inverted
1: Inverted

Bit 20 **INV**: selected resources inversion
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Group of resources selection
0x0: External input selectors (select 0-3)
0x1: Inputs from processor DWT comparators element (select 0-3)
0x2: Counter at zero (select 0)
0x3: Single-shot comparator (select 0)
Other: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: More resources selection
This bitfield selects one or more resources from the group selected in GROUP[2:0].

ETM resource register 3 (ETM_RSCTLR3)

Address offset: 0x20C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: Selected resources inversion
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Group of resources selection
0x0: External input selectors (select 0-3)
0x1: Inputs from processor DWT comparators element (select 0-3)
0x2: Counter at zero (select 0)
0x3: Single-shot comparator (select 0)
Other: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: More resources selection
This bitfield selects one or more resources from the group selected in GROUP[2:0].

ETM single-shot comparator control register 0 (ETM_SSCCR0)

Address offset: 0x280

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: Single-shot comparator reset
This bit enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.
1: Reset enabled

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (ETM_SSCSR0)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
												r	r	r	r

Bit 31 **STATUS**: Single-shot comparator status
This bit indicates whether any of the selected comparators have matched.
0: No match occurred
1: At least one match occurred

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **PC**: Processor comparator input sensitivity
1: Single-shot comparator sensitive to processor comparator inputs

Bit 2 **DV**: Data value comparator support
0: Single-shot data value comparisons not supported

Bit 1 **DA**: Data address comparator support
0: Single-shot data address comparisons not supported

Bit 0 **INST**: Instruction address comparator support
0: Single-shot instruction address comparisons not supported

ETM single-shot processor comparator input control register 0 (ETM_SSPCICR0)

Address offset: 0x2C0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PC[3:0]**: Processor comparator inputs selection for single-shot control
0xXXX0: Processor comparator input 0 not selected
0xXXX1: Processor comparator input 0 selected
0xXX0X: Processor comparator input 1 not selected
0xXX1X: Processor comparator input 1 selected
0xX0XX: Processor comparator input 2 not selected
0xX1XX: Processor comparator input 2 selected
0x0XXX: Processor comparator input 3 not selected
0x1XXX: Processor comparator input 3 selected

ETM power-down control register (ETM_PDCR)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: Power-up request
0: Power-up not requested
1: Power-up requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: Sticky power-down state
0: Trace register power has not been removed since the ETM_PDSR was last read.
1: Trace register power has been removed since the ETM_PDSR was last read.

Bit 0 **POWER**: ETM power-up status
1: ETM powered up

ETM claim tag set register (ETM_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Claim tag bits setting

Write:
0000: No effect
xxx1: Set bit 0.
xx1x: Set bit 1.
x1xx: Set bit 2.
1xxx: Set bit 3.
Read:
0xF: Indicate there are four bits in claim tag.

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Claim tag bits reset

Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Return current value of claim tag.

ETM authentication status register (ETM_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure noninvasive debug
0x2: Secure noninvasive debug disabled

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure noninvasive debug
0x2: Nonsecure noninvasive debug disabled
0x3: Nonsecure noninvasive debug enabled

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

ETM device type architecture register (ETM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4772 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x2: ETM architecture v4.2

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x4: ETM architecture v4.2

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA13: ETM architecture

ETM CoreSight device type register (ETM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device subtype identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: ETM part number

ETM CoreSight peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: ETM part number

ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ETM CoreSight peripheral identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ETM CoreSight component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

ETM register map

The ETM registers are accessed by the debugger at the address range 0xE0041000 to 0xE0041FFC.

Table 592. ETM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETM_PRGCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	Reserved	Reserved																															
0x00C	ETM_STATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value																															X	X
0x010	ETM_CONFIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
	Reset value																				X		X	X	X	X	X	X	X	X			
0x014 to 0x01C	Reserved	Reserved																															
0x020	ETM_ EVENTCTL0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value																	X				X	X	X	X	X				X	X	X	X
0x024	ETM_ EVENTCTL1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
	Reset value																				X	X										X	X
0x028	Reserved	Reserved																															
0x02C	ETM_STALLCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
	Reset value																						X		X					X	X	X	X
0x030	Reserved	Reserved																															
0x034	ETM_SYNCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
	Reset value																												0	1	0	1	0
0x038	ETM_CCCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					X	X	X	X	X	X	X	X	X	X	X	X
0x03C	Reserved	Reserved																															
0x040	ETM_TRACEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										X	X	X	X	X	X	X
0x044 to 0x07C	Reserved	Reserved																															

Table 592. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	ETM_VICTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
	Reset value													X	X	X	X					X	X	X		X	X	X	X	X	X	X	X
0x084 to 0x13C	Reserved	Reserved																															
0x140	ETM_CNTRLDVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x144 to 0x17C	Reserved	Reserved																															
0x180	ETM_IDR8	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	ETM_IDR9	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	ETM_IDR10	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	ETM_IDR11	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	ETM_IDR13	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198 to 0x1BC	Reserved	Reserved																															
0x1C0	ETM_IMSPECR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT [3:0]			
	Reset value																													0	0	0	0
0x1C4 to 0x1DC	Reserved	Reserved																															
0x1E0	ETM_IDR0	Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1:0]		Res.	CONDTYPE [1:0]		NUMEVENT [1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
	Reset value			1												0	0	0		0	0	0	1	1		1	1	1	0	0	0	0	
0x1E4	ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ [3:0]				TRCARCHMIN [3:0]				REVISION [3:0]			
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	0	1	0	0	0	0	1
0x1E8	ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]				VMIDSIZE[4:0]				CIDSIZE[4:0]				IASIZE[4:0]							
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]											
	Reset value	0	0	0	0	1	1	1	1					1	0	0	1					0	0	0	0	0	0	0	0	0	1	0	0

Table 592. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1F0	ETM_IDR4	NUMVMIDC [3:0]				NUMCIDC [3:0]				NUMSSCC [3:0]				NUMRSPAIR [3:0]				NUMPC [3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC [3:0]				NUMACPAIRS [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0				0	0	0	0	0	0	0	0	0
0x1F4	ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE [2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE [5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL [2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F8 to 0x204	Reserved	Reserved																															
0x208	ETM_RSCTLR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											X	X		X	X	X									X	X	X	X	X	X	X	X
0x20C	ETM_RSCTLR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												X		X	X	X									X	X	X	X	X	X	X	X
0x210 to 0x27C	Reserved	Reserved																															
0x280	ETM_SSCCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								X																								
0x284 to 0x29C	Reserved	Reserved																															
0x2A0	ETM_SSCSR0	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
	Reset value	X																												X	X	X	X
0x2A4 to 0x2BC	Reserved	Reserved																															
0x2C0	ETM_SSPCICR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
	Reset value																													X	X	X	X
0x2C4 to 0x30C	Reserved	Reserved																															
0x310	ETM_PDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
	Reset value																													0			
0x314	ETM_PDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318 to 0xF9C	Reserved	Reserved																															
0xFA0	ETM_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0

Table 592. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA8 to 0xFB4	Reserved	Reserved																															
0xFB8	ETM_AUTHSTATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									X	X	X	X	X	X	X	X
0xFBC	ETM_DEVARCHR	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0to 0xFC8	Reserved	Reserved																															
0xFCC	ETM_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYP [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4to 0xFDC	Reserved	Reserved																															
0xFE0	ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

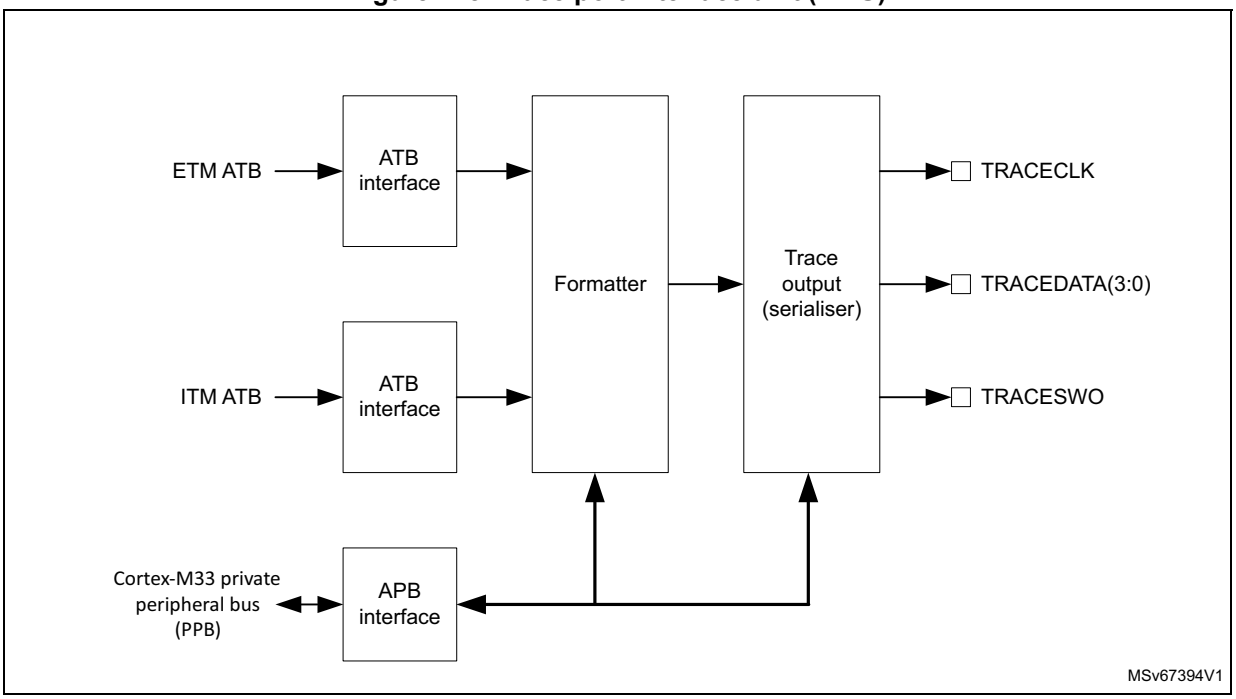
Refer to *Table 585: Processor ROM table* for register boundary addresses.

49.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in *Figure 748*, the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is a synchronous parallel port, comprising a clock

output, TRACECLK, and four data outputs, TRACEDATA(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up I/Os for other purposes, at the expenses of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

Figure 748. Trace port interface unit (TPIU)



Trace data can also be output on the serial-wire output, TRACESWO.

For more information on the trace port interface in the Cortex-M33, refer to the Arm Cortex-M33 Technical Reference Manual *[5]*.

49.10.1 TPIU registers

TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Trace port sizes, from 1 to 32 pins
Bit n-1 when set, indicates that port size n is supported.
0x0000 000F: Port sizes 1 to 4 supported

TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Current trace port size

Bit n-1 when set, indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register must only be modified when the formatter is stopped.

TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: Baudrate for the asynchronous output, TRACESWO

The baudrate is given by the TRACELKIN frequency divided by (PRESCALER +1).

TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]**: Protocol used for trace output
0x0: Parallel trace port mode
0x1: Asynchronous SWO using Manchester encoding
0x2: Asynchronous SWO using NRZ encoding
0x3: Reserved

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: Formatter stop
This bit indicates whether the formatter can be stopped or not.
1: The formatter cannot be stopped.

Bit 2 **TCPRESENT**: TRACECTL output pin availability
This bit indicates whether the optional TRACECTL output pin is available for use.
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: Formatter stop
The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
0: The formatter has not stopped.

Bit 0 **FLINPROG**: Flush in progress
This bit indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in TPIU_FFCR register.
0: No flush in progress
1: Flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
							r		rw					rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN**: Trigger on trigger in
1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: Flush on manual
0: Flush completed
1: Generate a flush.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** Continuous formatting enable
Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output. ETM trace is discarded.
0: Continuous formatting disabled
1: Continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

TPIU periodic synchronization counter register (TPIU_PSCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PSCOUNT[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PSCOUNT[12:0]**: Formatter frames counter
This bitfield enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word synchronization frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Claim tag bits setting

Write:
0000: No effect
xxx1: Set bit 0.
xx1x: Set bit 1.
x1xx: Set bit 2.
1xxx: Set bit 3.
Read:
0xF: Indicate there are four bits in claim tag.

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Claim tag bits reset

Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Return current value of claim tag.

TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Serial-wire output, NRZ support
0x1: Supported

Bit 10 **SWOMAN**: Serial-wire output, Manchester encoded format, support
0x1: Supported

Bit 9 **TCLKDATA**: Trace clock plus data support
0x0: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relationship (synchronous or asynchronous)
0x1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x1: Two input ports

TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Subclassification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

TPIU register map

Table 593. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0x004	TPIU_CSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008	Reserved	Reserved																															
0x010	TPIU_ACPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014-0x0EC	Reserved																																
0x0F0	TPIU_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
	Reset value																															0	1
0x0F4-0x2FC	Reserved																																
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
	Reset value																								1		0					1	

Table 593. TPIU register map and reset values (continued)

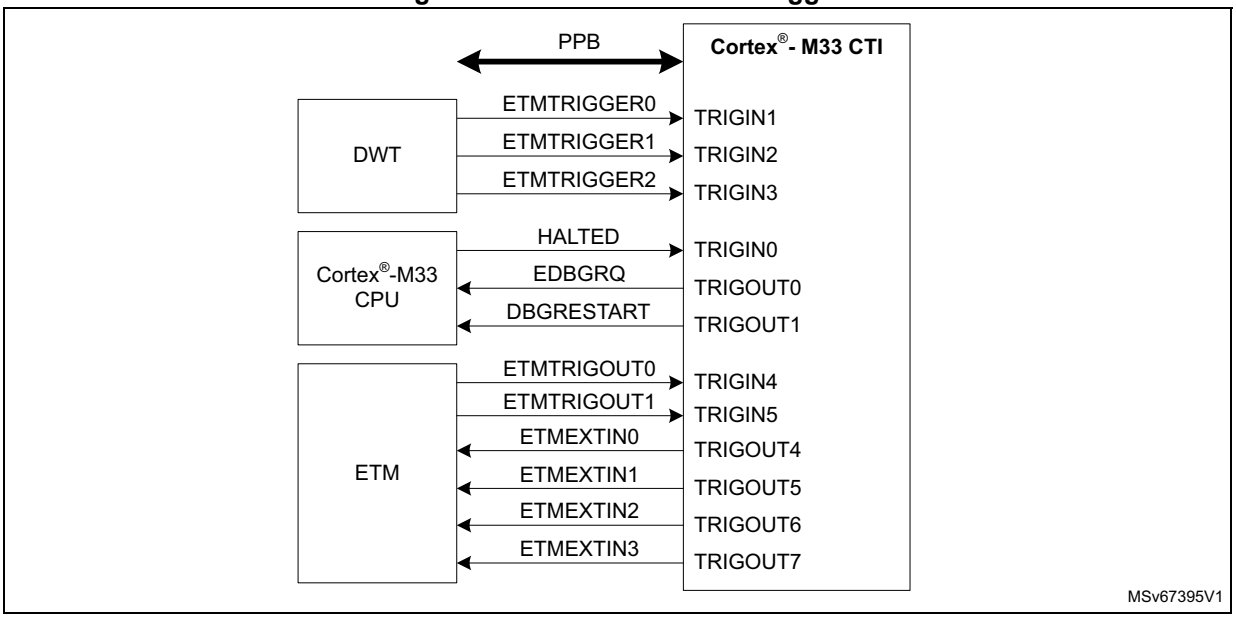
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	TPIU_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x30C-0xF9C	Reserved																																
0xFA0	TPIU_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0FA8-0xFC4	Reserved																																
0xFC8	TPIU_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ:	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MaXNUM[4:0]				
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	1
0xFCC	TPIU_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	TPIU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	TPIU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 584: MCU ROM table* for register boundary addresses.

49.11 Cross-trigger interface (CTI)

The CTI allows cross triggering between the processor and the ETM (see the figure below).

Figure 749. Embedded cross trigger



The CTI enables events from various sources to trigger debug and/or trace activity. For example, a watchpoint reached in the processor can start or stop the code trace, or a trace comparator can halt the processor.

The trigger input and output signals for the CTI are listed in the tables below.

Table 594. CTI inputs

Number	Source signal	Source component	Comments
0	HALTED	CPU	Processor halted - CPU is in debug mode
1	ETMTRIGGER0	DWT	DWT comparator output 0
2	ETMTRIGGER1	DWT	DWT comparator output 1
3	ETMTRIGGER2	DWT	DWT comparator output 2
4	ETMTRIGOUT0	ETM	ETM event output 0
5	ETMTRIGOUT1	ETM	ETM event output 1
6	-	-	Not used
7	-	-	Not used

Table 595. CTI outputs

Number	Source signal	Destination component	Comments
0	EDBGRQ	CPU	CPU halt request - Puts CPU in debug mode
1	DBGRESTART	CPU	CPU restart request - CPU exits debug mode
2	ETMEXTIN0	ETM	ETM event input 0
3	ETMEXTIN1	ETM	ETM event input 1
4	ETMEXTIN2	ETM	ETM event input 2
5	ETMEXTIN3	ETM	ETM event input 3
6	-	-	Not used
7	-	-	Not used

For more information on the cross-trigger interface CoreSight component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

49.11.1 CTI registers

The register file base address for the CTI is 0xE004 2000.

CTI control register (CTI_CONTROLR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: Global CTI enable

0: Disabled

1: Enabled

CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: Trigger acknowledge

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSETR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: Channel event setting
Read:
XXX0: Channel 0 event inactive
XXX0: Channel 0 event active
XX0X: Channel 1 event inactive
XX1X: Channel 1 event active
X0XX: Channel 2 event inactive
X1XX: Channel 2 event active
0XXX: Channel 3 event inactive
1XXX: Channel 3 event active
Write:
XXX0: No effect
XXX0: Set event on channel 0.
XX0X: No effect
XX1X: Set event on channel 1.
X0XX: No effect
X1XX: Set event on channel 2.
0XXX: No effect
1XXX: Set event on channel 3.

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: Channel event clear
0000: No effect
XXX1: Clear event on channel 0.
XX1X: Clear event on channel 1.
X1XX: Clear event on channel 2.
1XXX: Clear event on channel 3.

CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: Pulse channel event
This register clears itself immediately.
0000: No effect
XXX1: Generate pulse on channel 0.
XX1X: Generate pulse on channel 1.
X1XX: Generate pulse on channel 2.
1XXX: Generate pulse on channel 3.

CTI trigger input x enable register (CTI_INENxR)

Address offset: 0x020 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: Trigger input event enable
This bitfield enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).
0000: Trigger does not generate events on channels.
XXX1: Trigger x generates events on channel 0.
XX1X: Trigger x generates events on channel 1.
X1XX: Trigger x generates events on channel 2.
1XXX: Trigger x generates events on channel 3.

CTI trigger output x enable register (CTI_OUTENxR)

Address offset: 0x0A0 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: Trigger output event enable

For each channel, this bitfield defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on trigger outputs.

XXX1: Channel 0 events generate triggers on trigger output x.

XX1X: Channel 1 events generate triggers on trigger output x.

X1XX: Channel 2 events generate triggers on trigger output x.

1XXX: Channel 3 events generate triggers on trigger output x.

CTI trigger input status register (CTI_TRGISTSR)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: Trigger input status

There is one bit of the register for each CTITRIGINx input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger output status register (CTI_TRGOSTSR)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel input status register (CTI_CHINSTSR)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: Channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel output status register (CTI_CHOUTSTSR)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: Channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATER)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: Channel output enable

For each channel, this bitfield defines whether an event on that channel can propagate over the CTM to other CTIs.
0000: Channels events do not propagate.
XXX1: Channel 0 events propagate.
XX1X: Channel 1 events propagate.
X1XX: Channel 2 events propagate.
1XXX: Channel 3 events propagate.

CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available

0x4: Four channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available

0x8: Eight trigger inputs and eight trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers
0x0: None

CTI device type identifier register (CTI_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Subclassification
0x1: Cross-triggering component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

CTI register map

Table 596. CTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x004-0x00C	Reserved																																
0x010	CTI_INTACKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	CTI_APPSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR [3:0]			
	Reset value																													0	0	0	0
0x01C	CTI_APPPULSER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE [3:0]			
	Reset value																													X	X	X	X
0x020-0x03C	CTI_INEN0R to CTI_INEN7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN [3:0]			
	Reset value																													0	0	0	0
0x040-0x09C	Reserved																																
0x0A0 to 0x0BC	CTI_OUTEN0R to CTI_OUTEN7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN [3:0]			
	Reset value																													0	0	0	0
0x0C0-0x12C	Reserved																																
0x130	CTI_TRGISTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRGOSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	CTI_CHINSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS [3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
	Reset value																													0	0	0	0

Table 596. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x140	CTI_GATER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144-0xFC4	Reserved																																
0xFC8	CTI_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM [4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	CTI_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved																																
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	0	0	1
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 585: Processor ROM table* for register boundary addresses.

49.12 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the software):

- To maintain the clock and power to the processor cores when in low-power modes (Sleep, Stop, or Standby mode).
- To maintain the clock and power to the system debug and trace components when in low-power modes.

- To stop the clock to certain peripherals (SMBUS timeout, watchdogs, timers, RTC) when either processor core is stopped in debug mode

49.12.1 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. Debug tools can locate this register via the CoreSight discovery procedure described in *Section 49.5: ROM tables*.

49.12.2 Low-power mode emulation

When the device enters either Stop mode (clocks are stopped) or Standby mode (core power is switched off), the debugger can no longer access the debug access port, and looses the connection with the device. To avoid this, the debugger (or software) can set the DBG_STANDBY and/or DBG_STOP bits in DBGMCU_CR. These bits, when set, maintain the clock and power to the processor while the device is in the corresponding low-power mode. The processor remains in Sleep mode, and exits the low-power mode in the normal way. However, peripheral devices continue to operate, so the device behavior may not be identical to that of the actual low-power mode.

49.12.3 Peripheral clock freeze

The DBGMCU peripheral clock-freeze registers allow the operation of certain peripherals to be suspended in debug mode. The peripheral units, which support this feature are listed in the table below.

Table 597. Peripheral clock-freeze control bits

Bus	Control register	Peripheral	Description
APB1L	DBGMCU_APB1LFZR	I3C1	I3C1 SCL stall timeout counter
		I2C2	I2C2 SMBUS timeout
		I2C1	I2C1 SMBUS timeout
		IWDG	Independent watchdog
		WWDG	Window watchdog
		TIM12	General purpose timer 12
		TIM7	General purpose timer 7
		TIM6	General purpose timer 6
		TIM5	General purpose timer 5
		TIM4	General purpose timer 4
		TIM3	General purpose timer 3
		TIM2	General purpose timer 2
APB2	DBGMCU_APB2FZR	TIM17	General purpose timer 17
		TIM16	General purpose timer 16
		TIM15	General purpose timer 15
		TIM8	General purpose timer 8
		TIM1	General purpose timer 1

Table 597. Peripheral clock-freeze control bits (continued)

Bus	Control register	Peripheral	Description
APB3	DBGMCU_APB3FZR	RTC	Real time clock
		LPTIM1	Low power timer 1
AHB1	DBGMCU_AHB1FZR	LPDMA2 0 to 7	Low-power DMA2 channels 0 to 7
		LPDMA1 0 to 7	Low-power DMA1 channels 0 to 7

Each peripheral unit or DMA channel has a corresponding control bit, DBG_xxx_STOP, where xxx is the acronym of the peripheral (or DMA channel). The control bits are organized in DBGMCU_zzzFZR registers, where zzz corresponds to the name of the bus (AHB or APB). For example, DBGMCU_APB1LFZR contains the control bits for peripherals on the APB1L bus.

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1), according to the table below:

Table 598. Peripheral behavior in debug mode

HALTED	DBG_xxx_STOP	Peripheral behavior
0	X	The operation continues.
1	0	The operation continues.
1	1	The operation is suspended.

49.12.4 DBGMCU registers

The DBGMCU registers are not reset by a system reset, only by a power-on reset. They are accessible to the debugger via the AHB access port at base address 0xE00E 4000, and to software at base address 0x4402 4000.

For the actual availability of some peripherals and their related bits, check the datasheet or *Table 3: Memory map and peripheral register boundary addresses*. If not present, consider them reserved, and keep them at their reset value.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000

Reset value: 0xXXXX 6XXX

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Revision of the device
0x1000: Revision A
0x1001: Revision Z
0x1003: Revision Y

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identification
0x44E: STM32C55xx/C56xx
0x44F: STM32C53xx/C54xx
0x45A: STM32C59xx/C5Axx

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is accessible to the debugger in RDP L0. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
								rw	rw	rw	rw		rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **TRACE_MODE[1:0]**: Trace pin assignment
0x0: Trace pins assigned for asynchronous mode (TRACESWO)
0x1: Trace pins assigned for synchronous mode with a port width of 1 (TRACECK, TRACED0)
0x2: Trace pins assigned for synchronous mode with a port width of 2 ((TRACECK, TRACED0-1)
0x3: Trace pins assigned for synchronous mode with a port width of 4 ((TRACECK, TRACED0-3)

Bit 5 **TRACE_EN**: Trace port and clock enable.
This bit enables the trace port clock, TRACECK.
0: Disabled
1: Enabled

Bit 4 **TRACE_IOEN**: Trace pin enable
0: Disabled. Trace pins not assigned.
1: Enabled. Trace pins assigned according to the value of TRACE_MODE bitfield.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Debug in Standby mode
0: Normal operation. All clocks are disabled and the core is powered down automatically in Standby mode.
1: Automatic clock stop/power down disabled. All active clocks and oscillators continue to run during Standby mode. The core supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Debug in Stop mode

0: Normal operation. All clocks are disabled automatically in Stop mode.

1: Automatic clock stop disabled. All active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 **DBG_SLEEP**: Debug in Sleep mode

0: Normal operation. Clocks to CPU and buses are disabled automatically in Sleep mode.

1: Automatic clock stop disabled. All active clocks and oscillators continue to run during Sleep mode, allowing full debug capability. On exit from Sleep mode, the clock settings are set to the Sleep mode exit state.

DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZR)

Address offset: 0x008

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
								rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw					rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_I3C1_STOP**: I3C1 SCL stall counter stop in debug

0: Normal operation. I3C1 SCL stall timeout counter continues to operate while CPU is in debug mode.

1: Stop in debug. I3C1 SCL stall timeout counter is frozen while CPU is in debug mode.

Bit 22 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug

0: Normal operation. I2C2 SMBUS timeout continues to operate while CPU is in debug mode.

1: Stop in debug. I2C2 SMBUS timeout is frozen while CPU is in debug mode.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug

0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.

1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP**: IWDG stop in debug

0: Normal operation. IWDG continues to operate while CPU is in debug mode.

1: Stop in debug. IWDG is frozen while CPU is in debug mode.

Bit 11 **DBG_WWDG_STOP**: WWDG stop in debug

0: Normal operation. WWDG continues to operate while CPU is in debug mode.

1: Stop in debug. WWDG is frozen while CPU is in debug mode.

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **DBG_TIM12_STOP**: TIM12 stop in debug
0: Normal operation. TIM12 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM12 is frozen while CPU is in debug mode.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: Normal operation. TIM7 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM7 is frozen while CPU is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: Normal operation. TIM6 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM6 is frozen while CPU is in debug mode.

Bit 3 **DBG_TIM5_STOP**: TIM5 stop in debug
0: Normal operation. TIM5 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM5 is frozen while CPU is in debug mode.

Bit 2 **DBG_TIM4_STOP**: TIM4 stop in debug
0: Normal operation. TIM4 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM4 is frozen while CPU is in debug mode.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: Normal operation. TIM3 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM3 is frozen while CPU is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug
0: Normal operation. TIM2 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM2 is frozen while CPU is in debug mode.

DBGMCU APB1H peripheral freeze register (DBGMCU_APB1HFZR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw		rw											

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in debug
0: Normal operation. TIM17 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM17 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in debug
0: Normal operation. TIM16 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM16 is frozen while CPU is in debug mode.

Bit 16 **DBG_TIM15_STOP**: TIM15 stop in debug
0: Normal operation. TIM15 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM15 is frozen while CPU is in debug mode.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DBG_TIM8_STOP**: TIM8 stop in debug
0: Normal operation. TIM8 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM8 is frozen while CPU is in debug mode.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in debug
0: Normal operation. TIM1 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.
	rw													rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DBG_RTC_STOP**: RTC stop in debug
0: Normal operation. RTC continues to operate while CPU is in debug mode.
1: Stop in debug. RTC is frozen while CPU is in debug mode.

Bits 29:18 Reserved, must be kept at reset value.

Bit 17 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: Normal operation. LPTIM1 continues to operate while CPU is in debug mode.
1: Stop in debug. LPTIM1 is frozen while CPU is in debug mode.

Bits 16:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPDMA2_7_STOP	DBG_LPDMA2_6_STOP	DBG_LPDMA2_5_STOP	DBG_LPDMA2_4_STOP	DBG_LPDMA2_3_STOP	DBG_LPDMA2_2_STOP	DBG_LPDMA2_1_STOP	DBG_LPDMA2_0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPDMA1_7_STOP	DBG_LPDMA1_6_STOP	DBG_LPDMA1_5_STOP	DBG_LPDMA1_4_STOP	DBG_LPDMA1_3_STOP	DBG_LPDMA1_2_STOP	DBG_LPDMA1_1_STOP	DBG_LPDMA1_0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_LPDMA2_7_STOP**: LPDMA2 channel 7 stop in debug
0: Normal operation. LPDMA2 channel 7 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 7 is frozen while CPU is in debug mode.

Bit 22 **DBG_LPDMA2_6_STOP**: LPDMA2 channel 6 stop in debug
0: Normal operation. LPDMA2 channel 6 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 6 is frozen while CPU is in debug mode.

Bit 21 **DBG_LPDMA2_5_STOP**: LPDMA2 channel 5 stop in debug
0: Normal operation. LPDMA2 channel 5 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 5 is frozen while CPU is in debug mode.

Bit 20 **DBG_LPDMA2_4_STOP**: LPDMA2 channel 4 stop in debug
0: Normal operation. LPDMA2 channel 4 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 4 is frozen while CPU is in debug mode.

Bit 19 **DBG_LPDMA2_3_STOP**: LPDMA2 channel 3 stop in debug
0: Normal operation. LPDMA2 channel 3 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 3 is frozen while CPU is in debug mode.

Bit 18 **DBG_LPDMA2_2_STOP**: LPDMA2 channel 2 stop in debug
0: Normal operation. LPDMA2 channel 2 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 2 is frozen while CPU is in debug mode.

Bit 17 **DBG_LPDMA2_1_STOP**: LPDMA2 channel 1 stop in debug
0: Normal operation. LPDMA2 channel 1 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 1 is frozen while CPU is in debug mode.

Bit 16 **DBG_LPDMA2_0_STOP**: LPDMA2 channel 0 stop in debug
0: Normal operation. LPDMA2 channel 0 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA2 channel 0 is frozen while CPU is in debug mode.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DBG_LPDMA1_7_STOP**: LPDMA1 channel 7 stop in debug
0: Normal operation. LPDMA1 channel 7 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 7 is frozen while CPU is in debug mode.

Bit 6 **DBG_LPDMA1_6_STOP**: LPDMA1 channel 6 stop in debug
0: Normal operation. LPDMA1 channel 6 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 6 is frozen while CPU is in debug mode.

Bit 5 **DBG_LPDMA1_5_STOP**: LPDMA1 channel 5 stop in debug
0: Normal operation. LPDMA1 channel 5 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 5 is frozen while CPU is in debug mode.

Bit 4 **DBG_LPDMA1_4_STOP**: LPDMA1 channel 4 stop in debug
0: Normal operation. LPDMA1 channel 4 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 4 is frozen while CPU is in debug mode.

Bit 3 **DBG_LPDMA1_3_STOP**: LPDMA1 channel 3 stop in debug
0: Normal operation. LPDMA1 channel 3 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_LPDMA1_2_STOP**: LPDMA1 channel 2 stop in debug
0: Normal operation. LPDMA1 channel 2 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_LPDMA1_1_STOP**: LPDMA1 channel 1 stop in debug
0: Normal operation. LPDMA1 channel 1 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_LPDMA1_0_STOP**: LPDMA1 channel 0 stop in debug
0: Normal operation. LPDMA1 channel 0 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA1 channel 0 is frozen while CPU is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0x0FC

Reset value: 0x0001 XX03

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AP_ENABLED[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP_PRESENT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **AP_ENABLED[15:0]**: Access port enable
Bit n identifies whether access port AP n is open (can be accessed via the debug port), or locked (debug access to the AP is blocked).
Bit n = 0: APn locked
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Access port present
Bit n identifies whether access port AP n is present in device.
Bit n = 0: APn absent
Bit n = 1: APn present

DBGMCU debug authentication mailbox host register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

This register is read only when accessed by the CPU. Writes have no effect. This register can be written and read by an external debugger.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUTH_KEY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTH_KEY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AUTH_KEY[31:0]**: Device authentication key

The device specific 128-bit authentication key (OEM key) must be written to this register (in four successive 32-bit writes, least significant word first) to permit RDP regression.

DBGMCU debug authentication mailbox device register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

This register is read only when accessed via the debug port. Writes have no effect. This register can be read by the CPU.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUTH_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTH_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUTH_ID[31:0]**: Device specific ID

DBGMCU boundary-scan key password register (DBGMCU_DBG_BSKEY_PWD)

Address offset: 0x10C

Reset value: 0xXXXX XXXX

This register is read only when accessed by the CPU. Writes have no effect. This register can be written and read by an external debugger.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_BSKEY_PWD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_BSKEY_PWD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DBG_BSKEY_PWD[31:0]**: Boundary-scan key (BS key)

The device 32-bit BS key must be written to this register to permit RDP transition from L2_wBS to L2, closing the boundary-scan access. Writing a wrong key locks access to the device, and prevents code execution from the flash memory.

DBGMCU debug OEMKEY validation register (DBGMCU_DBG_VALR)

Address offset: 0x110

Reset value: 0x0000 000X

This register is always accessible. It can be read by an external debugger.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VAL_OEMKEY	VAL_RDY
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VAL_OEMKEY**: OEMKEY validation.

This bit is set by hardware once the validation request of the OEMKEY is performed, and if the provided password match the provisioned OEMKEY.
This bit is reset by hardware on power-on reset.

Bit 0 **VAL_RDY**: Validation ready

This bit is set by hardware once the validation request of the OEMKEY is performed. Results of the validation can be read in the VAL_OEMKEY bit. This VAL_RDY bit is reset by hardware upon system reset exit if VAL_OEMKEY is set or by power-on reset.

DBGMCU CoreSight peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size

0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x0: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x00: DBGMCU part number

DBGMCU CoreSight peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x0: DBGMCU part number

DBGMCU CoreSight peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DBGMCU CoreSight component identity register 0 (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

DBGMCU CoreSight component identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0xF: Non-CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

DBGMCU CoreSight component identity register 2 (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

DBGMCU CoreSight component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

DBGMCU register map

Table 599. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x

Table 599. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	DBGMCU_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value																									0	0	0	0		0	0	0
0x008	DBGMCU_APB1LFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value									0	0	0									0	0					0	0	0	0	0	0	0
0x00C	DBGMCU_APB1HFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x010	DBGMCU_APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0			0		0											
0x014	DBGMCU_APB3FZR	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0													0																	
0x018-0x01C	Reserved																																
0x020	DBGMCU_AHB1FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPDMA2_7_STOP	DBG_LPDMA2_6_STOP	DBG_LPDMA2_5_STOP	DBG_LPDMA2_4_STOP	DBG_LPDMA2_3_STOP	DBG_LPDMA2_2_STOP	DBG_LPDMA2_1_STOP	DBG_LPDMA2_0_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPDMA1_7_STOP	DBG_LPDMA1_6_STOP	DBG_LPDMA1_5_STOP	DBG_LPDMA1_4_STOP	DBG_LPDMA1_3_STOP	DBG_LPDMA1_2_STOP	DBG_LPDMA1_1_STOP	DBG_LPDMA1_0_STOP
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x024 - 0x0F8	Reserved																																
0x0FC	DBGMCU_SR	AP_ENABLED[15:0]																AP_PRESENT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	x	x	x	x	x	x	x	x	0	0	0	0	0	0	1	1

Table 599. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	DBGMCU_DBG_AUTH_HOST	AUTH_KEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x104	DBGMCU_DBG_AUTH_DEVICE	AUTH_ID[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x108	Reserved																																
0x10C	DBGMCU_DBG_BSKEY_PWD	DBG_BSKEY_PWD[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x110	DBGMCU_DBG_VALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VAL_OEMKEY	VAL_RDY
	Reset value																															x	x
0x114 - 0xFCC	Reserved																																
0xFD0	DBGMCU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 - 0xFDC	Reserved																																
0xFE0	DBGMCU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE4	DBGMCU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFE8	DBGMCU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[2:0]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	DBGMCU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DBGMCU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1

Table 599. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF4	DBGMCU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[3:0]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	DBGMCU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DBGMCU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.2: Memory organization* for register boundary addresses.

49.13 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16th March 2015
3. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
4. DDI 0553A (ID092917) - Arm® v8-M Architecture Reference Manual, Issue A.f, 29 September 2017
5. 100230_0002_00_en - Arm® Cortex®-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017
6. 100230_0004_00_en - Arm® Cortex®-M33 Processor r0p4 Technical Reference
7. Manual, Issue 0004-00, 18 April 2018

50 Device electronic signature

The device electronic signature is stored in the read only (RO) area of the flash memory module and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the devices.

50.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited for:

- Use as serial numbers (for example, USB string serial numbers or other end applications)
- Use as part of the security keys to increase the security of code in flash memory by using and combining this unique ID with software cryptographic primitives and protocols before programming the internal flash memory
- Activating secure boot processes

The 96-bit unique device identifier provides a reference number, which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x08FF F800

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24]
Lot number (ASCII encoded)

50.2 Flash size data register

Base address: 0x08FF F80C

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
This field indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x200 corresponds to 512 Kbytes.

50.3 Package data register

Base address: 0x08FF F80E

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[4:0]				
											r	r	r	r	r

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type
00000: LQFP64
00010: LQFP100
00100: LQFP144
00101: LQFP48
01001: UFQFPN32
10000: UFQFPN48
10010: LQFP80
10110: LQFP32
11000: TSSOP20
11001: QFN20
11010: QFN24
Others: reserved

Note: Refer to the product datasheet for availability of packages on a specific device.

51 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers must re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers must not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

52 Revision history

Table 600. Document revision history

Date	Revision	Changes
11-Feb-2026	1	Initial release.

Index

A

ADC_AWD1HTR 660
ADC_AWD1LTR 659
ADC_AWD2CR 658
ADC_AWD2HTR 661
ADC_AWD2LTR 660
ADC_AWD3CR 659
ADC_AWD3HTR 661
ADC_AWD3LTR 661
ADC_CALFACT 662
ADC_CFGR1 641
ADC_CFGR2 644
ADC_CR 638
ADC_DR 653
ADC_GCOMP 657
ADC_IER 636
ADC_ISR 634
ADC_JDRy 658
ADC_JSQR 653
ADC_OFCFGRy 655
ADC_OFRy 656
ADC_OR 663
ADC_PCSEL 649
ADC_SMPR1 647
ADC_SMPR2 648
ADC_SQR1 649
ADC_SQR2 650
ADC_SQR3 651
ADC_SQR4 652
ADCC_CCR 665
ADCC_CDR 666
ADCC_CDR2 667
ADCC_CSR 663
AES_CR 825
AES_DINR 828
AES_DOUTR 829
AES_ICR 836
AES_IER 834
AES_ISR 835
AES_IVR0 831
AES_IVR1 831
AES_IVR2 831
AES_IVR3 832
AES_KEYR0 829
AES_KEYR1 830
AES_KEYR2 830
AES_KEYR3 830
AES_KEYR4 832
AES_KEYR5 832
AES_KEYR6 833
AES_KEYR7 833
AES_SR 827
AES_SUSPxR 833
AP0_CSWR 2432
AP1_CSWR 2433
APx_BASER 2435
APx_BDnR 2435
APx_DRWR 2435
APx_IDR 2436
APx_TAR 2434

B

BPU_CIDR0 2485
BPU_CIDR1 2486
BPU_CIDR2 2486
BPU_CIDR3 2486
BPU_COMPxR 2482
BPU_CTRLR 2481
BPU_DEVARCHR 2482
BPU_DEVTYPER 2483
BPU_PIDR0 2484
BPU_PIDR1 2484
BPU_PIDR2 2484
BPU_PIDR3 2485
BPU_PIDR4 2483

C

C1ROM_CIDR3 . . . . . 2444, 2449, 2454
CCB_CR 768
CCB_REFTAGRx 771
CCB_SR 769
COMP_CFGR1 721
COMP_CFGR2 723
COMP_ICFR 721
COMP_SR 720
CORDIC_CSR 493
CORDIC_RDATA 496
CORDIC_WDATA 495
CPUROM_CIDR0 2453
CPUROM_CIDR1 2453
CPUROM_CIDR2 2454
CPUROM_CIDR3 2454
CPUROM_MEMTYPER 2450
CPUROM_PIDR0 2451
CPUROM_PIDR1 2452

CPUROM_PIDR2 2452
CPUROM_PIDR3 2453
CPUROM_PIDR4 2451
CRC_CR 476
CRC_DR 475
CRC_IDR 475
CRC_INIT 477
CRC_POL 477
CRS_CFGR 334
CRS_CR 332
CRS_ICR 337
CRS_ISR 335
CTI_APPCLEAR 2529
CTI_APPPULSER 2530
CTI_APPSETR 2528
CTI_CHINSTSR 2532
CTI_CHOUTSTSR 2532
CTI_CIDR0 2536
CTI_CIDR1 2537
CTI_CIDR2 2537
CTI_CIDR3 2537
CTI_CONTROLR 2527
CTI_DEVIDR 2533
CTI_DEVTYPER 2534
CTI_GATER 2533
CTI_INENxR 2530
CTI_INTACKR 2528
CTI_OUTENxR 2531
CTI_PIDR0 2535
CTI_PIDR1 2535
CTI_PIDR2 2535
CTI_PIDR3 2536
CTI_PIDR4 2534
CTI_TRGISTSR 2531
CTI_TRGOSTSR 2532

D

DAC_CCR 704
DAC_CR 693
DAC_DHR12L1 698
DAC_DHR12L2 699
DAC_DHR12LD 701
DAC_DHR12R1 697
DAC_DHR12R2 699
DAC_DHR12RD 700
DAC_DHR8R1 698
DAC_DHR8R2 700
DAC_DHR8RD 701
DAC_DOR1 702
DAC_DOR2 702
DAC_MCR 705
DAC_SHHR 707
DAC_SHRR 708
DAC_SHSR1 706
DAC_SHSR2 707
DAC_SR 703
DAC_SWTRGR 696
DBGMCU_AHB1FZR 2546
DBGMCU_APB1HFZR 2544
DBGMCU_APB1LFZR 2543
DBGMCU_APB2FZR 2545
DBGMCU_APB3FZR 2546
DBGMCU_CIDR0 2553
DBGMCU_CIDR1 2553
DBGMCU_CIDR2 2554
DBGMCU_CIDR3 2554
DBGMCU_CR 2542
DBGMCU_DBG_AUTH_DEVICE 2549
DBGMCU_DBG_AUTH_HOST 2548
DBGMCU_DBG_BSKEY_PWD 2549
DBGMCU_DBG_VALR 2550
DBGMCU_IDCODE 2541
DBGMCU_PIDR0 2551
DBGMCU_PIDR1 2551
DBGMCU_PIDR2 2552
DBGMCU_PIDR3 2552
DBGMCU_PIDR4 2550
DBGMCU_SR 2548
DLYB_CFGR 559
DLYB_CR 558
DP_ABORTR 2425
DP_CTRLSTATR 2426
DP_DLCR 2427
DP_DLPIDR 2428
DP_DPIDR 2424
DP_RDBUFFR 2429
DP_RESENDR 2428
DP_SELECTR 2429
DP_TARGETIDR 2427
DWT_CIDR0 2468
DWT_CIDR1 2468
DWT_CIDR2 2468
DWT_CIDR3 2469
DWT_COMPxR 2460
DWT_CPICNTR 2458
DWT_CTRLR 2456
DWT_CYCCNTR 2458
DWT_DEVARCHR 2465
DWT_DEVTYPER 2465
DWT_EXCCNTR 2459
DWT_FOLDCNTR 2460
DWT_FUNCTR0 2461
DWT_FUNCTR1 2462

DWT_FUNCTR2 2463
DWT_FUNCTR3 2464
DWT_LSUCNTR 2459
DWT_PCSR 2460
DWT_PIDR0 2466
DWT_PIDR1 2466
DWT_PIDR2 2467
DWT_PIDR3 2467
DWT_PIDR4 2466
DWT_SLPCNTR 2459

E

ETH_DMACxCARXBR 2297
ETH_DMACxCARXDR 2296
ETH_DMACxCATXBR 2296
ETH_DMACxCATXDR 2295
ETH_DMACxCR 2284
ETH_DMACxIER 2291
ETH_DMACxMFCR 2299
ETH_DMACxRXCR 2286
ETH_DMACxRXDLAR 2288
ETH_DMACxRXDTPR 2289
ETH_DMACxRXIWTR 2294
ETH_DMACxRXRLR 2290
ETH_DMACxSR 2297
ETH_DMACxTXCR 2285
ETH_DMACxTXDLAR 2287
ETH_DMACxTXDTPR 2289
ETH_DMACxTXRLR 2290
ETH_DMADSR 2283
ETH_DMAISR 2282
ETH_DMAMR 2280
ETH_DMASBMR 2281
ETH_MAC_IACR 2378
ETH_MAC_TMRQR 2379
ETH_MAC1USTCR 2341
ETH_MACA0HR 2356
ETH_MACACR 2389
ETH_MACARPAR 2354
ETH_MACATSNR 2390
ETH_MACATSSR 2391
ETH_MACAxHR 2357
ETH_MACAxLR 2356
ETH_MACCR 2311
ETH_MACCSRSWCR 2354
ETH_MACDR 2342
ETH_MACECR 2316
ETH_MACHT0R 2322
ETH_MACHT1R 2323
ETH_MACHWF0R 2343
ETH_MACHWF1R 2345
ETH_MACHWF2R 2347
ETH_MACHWF3R 2349
ETH_MACIER 2334
ETH_MACISR 2332
ETH_MACIVIR 2328
ETH_MACL3A00R 2371
ETH_MACL3A01R 2376
ETH_MACL3A10R 2371
ETH_MACL3A11R 2376
ETH_MACL3A20R 2372
ETH_MACL3A21R 2377
ETH_MACL3A30R 2372
ETH_MACL3A31R 2377
ETH_MACL3L4C0R 2368
ETH_MACL3L4C1R 2373
ETH_MACL4A0R 2370
ETH_MACL4A1R 2375
ETH_MACLCSR 2338
ETH_MACLETR 2341
ETH_MACLMIR 2402
ETH_MACLTCR 2340
ETH_MACMDIOAR 2351
ETH_MACMDIODR 2353
ETH_MACPCSR 2336
ETH_MACPFR 2318
ETH_MACPOCR 2400
ETH_MACPPSCR 2394, 2396
ETH_MACPPSIR 2399
ETH_MACPPSTTNR 2398
ETH_MACPPSTTSR 2397
ETH_MACPPSWR 2399
ETH_MACPRSTIMR 2355
ETH_MACPRSTIMUR 2355
ETH_MACQTXFCR 2329
ETH_MACRWKPFR 2338
ETH_MACRXDTI 2387
ETH_MACRXFCR 2331
ETH_MACRXTXSR 2335
ETH_MACSPI0R 2401
ETH_MACSPI1R 2401
ETH_MACSPI2R 2401
ETH_MACSSIR 2382
ETH_MACSTNR 2383
ETH_MACSTNUR 2384
ETH_MACSTSR 2383
ETH_MACSTSUR 2384
ETH_MACTSAR 2385
ETH_MACTSCR 2380
ETH_MACTSEACR 2392
ETH_MACTSECNR 2393
ETH_MACTSELR 2394
ETH_MACTSIACR 2391

ETH_MACTSICNR 2392
ETH_MACTSILR 2393
ETH_MACTSSR 2386
ETH_MACTXDTI 2388
ETH_MACTXTSSNR 2388
ETH_MACTXTSSSR 2389
ETH_MACVHTR 2326
ETH_MACVIR 2327
ETH_MACVR 2342
ETH_MACVTR 2324
ETH_MACWJBTR 2320
ETH_MMC_CONTROL 2358
ETH_MMC_RX_INTERRUPT 2359
ETH_MMC_RX_INTERRUPT_MASK 2361
ETH_MMC_TX_INTERRUPT 2360
ETH_MMC_TX_INTERRUPT_MASK 2362
ETH_MTLISR 2303
ETH_MTLOMR 2302
ETH_MTLQxICSR 2306
ETH_MTLRXQxDR 2309
ETH_MTLRXQxMPOCR 2308
ETH_MTLRXQxOMR 2307
ETH_MTLTXQxDR 2305
ETH_MTLTXQxOMR 2303
ETH_MTLTXQxUR 2304
ETH_RX_ALIGNMENT_ERROR_PACKETS . . . 2365
ETH_RX_CRC_ERROR_PACKETS 2364
ETH_RX_LPI_TRAN_CNTR 2367
ETH_RX_LPI_USEC_CNTR 2367
ETH_RX_UNICAST_PACKETS_GOOD . . .2365
ETH_TX_LPI_TRAN_CNTR 2366
ETH_TX_LPI_USEC_CNTR 2366
ETH_TX_MULTIPLE_COLLISION_-
GOOD_PACKETS 2364
ETH_TX_PACKET_COUNT_GOOD 2364
ETH_TX_SINGLE_COLLISION_GOOD_PACK-
ETS 2363
ETM_AUTHSTATR 2506
ETM_CCCTLR 2492
ETM_CIDR0 2509
ETM_CIDR1 2510
ETM_CIDR2 2510
ETM_CIDR3 2510
ETM_CLAIMCLR 2505
ETM_CLAIMSETR 2505
ETM_CNTRLDVR0 2494
ETM_CONFIGR 2489
ETM_DEVARCHR 2506
ETM_DEVTYPER 2507
ETM_EVENTCTL0R 2490
ETM_EVENTCTL1R 2491
ETM_IDR0 2497
ETM_IDR1 2498
ETM_IDR10 2495
ETM_IDR11 2495
ETM_IDR12 2496
ETM_IDR13 2496
ETM_IDR2 2498
ETM_IDR3 2499
ETM_IDR4 2500
ETM_IDR5 2500
ETM_IDR8 2494
ETM_IDR9 2495
ETM_IMSPECR0 2496
ETM_PDCR 2504
ETM_PDSR 2504
ETM_PIDR0 2508
ETM_PIDR1 2508
ETM_PIDR2 2508
ETM_PIDR3 2509
ETM_PIDR4 2507
ETM_PRGCTLR 2488
ETM_RSCTLR2 2501
ETM_RSCTLR3 2502
ETM_SSCCR0 2502
ETM_SSCSR0 2503
ETM_SSPCICR0 2503
ETM_STALLCTLR 2491
ETM_STATR 2489
ETM_SYNCPR 2492
ETM_TRACEIDR 2493
ETM_VICTLR 2493
EXTI_EMR1 468
EXTI_EMR2 469
EXTI_EXTICR1 461
EXTI_EXTICR2 463
EXTI_EXTICR3 464
EXTI_EXTICR4 466
EXTI_FPR1 455
EXTI_FPR2 460
EXTI_FTSR1 453
EXTI_FTSR2 457
EXTI_IMR1 468
EXTI_IMR2 469
EXTI_PRIVCFGR1 455
EXTI_PRIVCFGR2 461
EXTI_RPR1 454
EXTI_RPR2 459
EXTI_RTSR1 452
EXTI_RTSR2 456
EXTI_SWIER1 454
EXTI_SWIER2 458

F

FDCAN_CCCR 2057
FDCAN_CKDIV 2082
FDCAN_CREL 2054
FDCAN_DBTP 2054
FDCAN_ECR 2062
FDCAN_ENDN 2054
FDCAN_HPMS 2073
FDCAN_IE 2068
FDCAN_ILE 2071
FDCAN_ILS 2070
FDCAN_IR 2065
FDCAN_NBTP 2058
FDCAN_PSR 2063
FDCAN_RWD 2056
FDCAN_RXF0A 2075
FDCAN_RXF0S 2074
FDCAN_RXF1A 2076
FDCAN_RXF1S 2075
FDCAN_RXGFC 2071
FDCAN_TDCR 2065
FDCAN_TEST 2055
FDCAN_TOCC 2061
FDCAN_TOCV 2062
FDCAN_TSCC 2060
FDCAN_TSCV 2060
FDCAN_TXBAR 2078
FDCAN_TXBC 2076
FDCAN_TXBCF 2080
FDCAN_TXBCIE 2081
FDCAN_TXBCR 2079
FDCAN_TXBRP 2077
FDCAN_TXBTIE 2080
FDCAN_TXBTO 2079
FDCAN_TXEFA 2082
FDCAN_TXEFS 2081
FDCAN_TXFQS 2077
FDCAN_XIDAM 2073
FLASH_ACR 174
FLASH_BL_COM_CFG_CUR 192
FLASH_BL_COM_CFG_PRG 193
FLASH_BOOTR_CUR 190
FLASH_BOOTR_PRG 190
FLASH_BSKEYR_PRG 195
FLASH_CCR 183
FLASH_CR 180
FLASH_ECCCORR 197
FLASH_ECCDETR 198
FLASH_ECCDR 199
FLASH_HDP1R_CUR 196
FLASH_HDP1R_PRG 197
FLASH_HDP2R_CUR 201
FLASH_HDP2R_PRG 201
FLASH_HDPEXTR 184
FLASH_KEYR 175
FLASH_OEMKEYR1_PRG 193
FLASH_OEMKEYR2_PRG 194
FLASH_OEMKEYR3_PRG 194
FLASH_OEMKEYR4_PRG 194
FLASH_OPSR 176
FLASH_OPTCR 177
FLASH_OPTKEYR 175
FLASH_OPTSR_CUR 185
FLASH_OPTSR_PRG 186
FLASH_OPTSR2_CUR 188
FLASH_OPTSR2_PRG 189
FLASH_OTPBLR_CUR 191
FLASH_OTPBLR_PRG 192
FLASH_PRIVCFGR 183
FLASH_SR 178
FLASH_WRP1R_CUR 195
FLASH_WRP1R_PRG 196
FLASH_WRP2R_CUR 200
FLASH_WRP2R_PRG 200

G

GPIOx_AFRH 353
GPIOx_AFRL 352
GPIOx_BRR 354
GPIOx_BSRR 351
GPIOx_IDR 350
GPIOx_LCKR 351
GPIOx_MODER 348
GPIOx_ODR 350
GPIOx_OSPEEDR 349
GPIOx_OTYPER 348
GPIOx_PUPDR 349

H

HASH_CR 906
HASH_CSRx 914
HASH_DIN 908
HASH_HRAx 911
HASH_HRx 911
HASH_IMR 912
HASH_SR 912
HASH_STR 909

I

I2C_CR1 1673
I2C_CR2 1676
I2C_ICR 1684

I2C_ISR 1681
I2C_OAR1 1678
I2C_OAR2 1678
I2C_PECR 1685
I2C_RXDR 1685
I2C_TIMEOUTR 1680
I2C_TIMINGR 1679
I2C_TXDR 1686
I3C_BCR 1788
I3C_CEVR 1775
I3C_CFGR 1754
I3C_CR 1750, 1752
I3C_CRCAPR 1791
I3C_DCR 1789
I3C_DEVR0 1778
I3C_DEVRx 1780
I3C_EPIDR 1794
I3C_EVR 1769
I3C_GETCAPR 1790
I3C_GETMXDSR 1792
I3C_IBIDR 1763
I3C_IER 1773
I3C_MAXRLR 1782
I3C_MAXWLR 1783
I3C_MISR 1777
I3C_RDR 1759
I3C_RDWR 1760
I3C_RMR 1768
I3C_SER 1766
I3C_SR 1765
I3C_TDR 1761
I3C_TDWR 1762
I3C_TGTTDR 1764
I3C_TIMINGR0 1784
I3C_TIMINGR1 1785
I3C_TIMINGR2 1787
ICACHE_CR 218
ICACHE_CRRx 221
ICACHE_FCR 220
ICACHE_HMONR 221
ICACHE_IER 220
ICACHE_MMONR 221
ICACHE_SR 219
ITM_CIDR0 2478
ITM_CIDR1 2479
ITM_CIDR2 2479
ITM_CIDR3 2479
ITM_DEVARCHR 2475
ITM_DEVTYPER 2476
ITM_PIDR0 2477
ITM_PIDR1 2477
ITM_PIDR2 2477
ITM_PIDR3 2478
ITM_PIDR4 2476
ITM_STIMRx 2472-2473
ITM_TCR 2474
ITM_TER 2473
ITM_TPR 2474
IWDG_EWCR 1541
IWDG_ICR 1542
IWDG_KR 1538
IWDG_PR 1538
IWDG_RLR 1539
IWDG_SR 1539
IWDG_WINR 1541

L

LPDMA_CxBR1 433
LPDMA_CxCR 426
LPDMA_CxDAR 436
LPDMA_CxFCR 424
LPDMA_CxLBAR 424
LPDMA_CxLLR 437
LPDMA_CxSAR 435
LPDMA_CxSR 425
LPDMA_CxTR1 429
LPDMA_CxTR2 431
LPDMA_MISR 423
LPDMA_PRIVCFGR 422
LPDMA_RCFGLOCKR 423
LPTIM_ARR 1520
LPTIM_CCMR1 1523
LPTIM_CCR1 1519
LPTIM_CCR2 1525
LPTIM_CFGR 1516
LPTIM_CFGR2 1521
LPTIM_CNT 1520
LPTIM_CR 1518
LPTIM_RCR 1522
LPTIM1_DIER 1513-1514
LPTIM1_ICR 1510-1511
LPTIM1_ISR 1506, 1508
LPUART_BRR 1931
LPUART_CR1 . . . . . 1918, 1921
LPUART_CR2 1924
LPUART_CR3 . . . . . 1926, 1929
LPUART_ICR 1940
LPUART_ISR 1932, 1937
LPUART_PRESC 1942
LPUART_RDR 1941
LPUART_RQR 1931
LPUART_TDR 1941

M

MCUROM_CIDR0 2448
MCUROM_CIDR1 2448
MCUROM_CIDR2 2449
MCUROM_CIDR3 2449
MCUROM_MEMTYPER 2445
MCUROM_PIDR0 2446
MCUROM_PIDR1 2447
MCUROM_PIDR2 2447
MCUROM_PIDR3 2448
MCUROM_PIDR4 2446

O

OPAMP_CSR 741
OPAMP_TCMR 743

P

PKA_CLRFR 952
PKA_CR 948
PKA_SR 950
PWR_IORETR 248
PWR_PMCR 243
PWR_PMSR 244
PWR_PRIVCFGR 249
PWR_RTCCR 245
PWR_VMCR 245
PWR_VMSR 246
PWR_WUCR 247
PWR_WUSCR 246
PWR_WUSR 247

R

RAMCFG_M2DEAR 121
RAMCFG_M2ECCKEYR 122
RAMCFG_M2ICR 121
RAMCFG_M2IER 120
RAMCFG_M2SEAR 120
RAMCFG_M2WPR1 122
RAMCFG_M2WPR2 122
RAMCFG_M2WPR3 123
RAMCFG_M2WPR4 123
RAMCFG_MxCR 118
RAMCFG_MxERKEYR 119
RAMCFG_MxISR 119
RCC_AHB1ENR 296
RCC_AHB1LPENR 305
RCC_AHB1RSTR 288
RCC_AHB2ENR 297
RCC_AHB2LPENR 306
RCC_AHB2RSTR 289
RCC_AHB4ENR 299
RCC_AHB4LPENR 308
RCC_AHB4RSTR 291
RCC_APB1HENR 302
RCC_APB1HLPENR 311
RCC_APB1HRSTR 293
RCC_APB1LENR 300
RCC_APB1LLPENR 309
RCC_APB1LRSTR 291
RCC_APB2ENR 303
RCC_APB2LPENR 312
RCC_APB2RSTR 294
RCC_APB3ENR 304
RCC_APB3LPENR 313
RCC_APB3RSTR 295
RCC_CCIPR1 314
RCC_CCIPR2 315
RCC_CCIPR3 317
RCC_CFGR1 279
RCC_CFGR2 281
RCC_CICR 286
RCC_CIER 283
RCC_CIFR 285
RCC_CR1 274
RCC_CR2 277
RCC_PRIVCFGR 322
RCC_RSR 320
RCC_RTCCR 318
RNG_CR 785
RNG_DR 788
RNG_HTCRx 789
RNG_NSCR 789
RNG_SR 787, 790-791
RTC_ALRABINR 1594
RTC_ALRBBINR 1594
RTC_ALRMAR 1587
RTC_ALRMASSR 1588
RTC_ALRMBR 1589
RTC_ALRMBSSR 1590
RTC_CALR 1582
RTC_CR 1576
RTC_DR 1572
RTC_ICSR 1573
RTC_MISR 1592
RTC_PRER 1575
RTC_PRIVCFGR 1580
RTC_SCR 1593
RTC_SHIFTR 1584
RTC_SR 1591
RTC_SSR 1573
RTC_TR 1571

RTC_TSDR 1586
RTC_TSSSR 1586
RTC_TSTR 1585
RTC_WPR 1582
RTC_WUTR 1576

S

SAES_CR 878
SAES_DINR 882
SAES_DOUTR 882
SAES_ICR 890
SAES_IER 887
SAES_ISR 888
SAES_IVR0 884
SAES_IVR1 885
SAES_IVR2 885
SAES_IVR3 885
SAES_KEYR0 883
SAES_KEYR1 883
SAES_KEYR2 884
SAES_KEYR3 884
SAES_KEYR4 886
SAES_KEYR5 886
SAES_KEYR6 886
SAES_KEYR7 887
SAES_SR 881
SAES_SUSPxR 887
SBS_CCCSR 365
SBS_CCSWCR 367
SBS_CCVALR 366
SBS_CFGR2 367
SBS_CLCKR 368
SBS_ECCNMIR 369
SBS_FPUIMR 364
SBS_HDPLCR 361
SBS_HDPLSR 362
SBS_MESR 365
SBS_NEXTHDPLCR 362
SBS_PMCR 363
SPI_CFG1 2003
SPI_CFG2 2007
SPI_CR1 2001
SPI_CR2 2003
SPI_CRCPOLY 2015
SPI_I2SCFGR 2017
SPI_IER 2009
SPI_IFCR 2013
SPI_RXCRC 2016
SPI_RXDR 2014
SPI_SR 2010
SPI_TXCRC 2015
SPI_TXDR 2014
SPI_UDRDR 2017
SYSROM_CIDR0 2443
SYSROM_CIDR1 2443
SYSROM_CIDR2 2444
SYSROM_CIDR3 2444
SYSROM_MEMTYPER 2441
SYSROM_PIDR0 2441
SYSROM_PIDR1 2442
SYSROM_PIDR2 2442
SYSROM_PIDR3 2443
SYSROM_PIDR4 2441

T

TAMP_BKPxR 1621
TAMP_CFGR 1612
TAMP_CR1 1606
TAMP_CR2 1608
TAMP_CR3 1610
TAMP_FLTCR 1611
TAMP_IER 1614
TAMP_MISR 1617
TAMP_OR 1620
TAMP_PRIVCFGR 1613
TAMP_SCR 1619
TAMP_SR 1616
TIM_AF1 . . . . . 1440, 1471
TIM_AF2 . . . . . 1444, 1475
TIM_ARR 1432, 1465
TIM_BDTR 1435, 1467
TIM_CCER 1429, 1461
TIM_CCMR1 . . . . . 1424, 1426, 1458-1459
TIM_CCR1 . . . . . 1433, 1466
TIM_CCR2 1435
TIM_CNT 1432, 1464
TIM_CR1 . . . . . 1412, 1452
TIM_CR2 . . . . . 1414, 1454
TIM_DCR 1447, 1478
TIM_DIER 1420, 1455
TIM_DMAR 1448, 1479
TIM_DTR2 . . . . . 1438, 1470
TIM_EGR 1423, 1457
TIM_MPR1 . . . . . 1445, 1476
TIM_MPR2 . . . . . 1445, 1477
TIM_OOR 1446, 1477
TIM_PSC 1432, 1464
TIM_RCR 1433, 1465
TIM_SMCR 1417
TIM_SR 1421, 1456
TIM_TISEL 1439, 1471
TPIU_ACPR 2516

TPIU_CIDR0 2523
TPIU_CIDR1 2523
TPIU_CIDR2 2523
TPIU_CIDR3 2524
TPIU_CLAIMCLR 2519
TPIU_CLAIMSETR 2519
TPIU_CSPSR 2516
TPIU_DEVIDR 2520
TPIU_DEVTYPER 2520
TPIU_FFCR 2517
TPIU_FFSR 2517
TPIU_PIDR0 2521
TPIU_PIDR1 2521
TPIU_PIDR2 2522
TPIU_PIDR3 2522
TPIU_PIDR4 2521
TPIU_PSCR 2518
TPIU_SPPR 2516
TPIU_SSPSR 2515

U

USART_BRR 1868
USART_CR1 . . . . . 1850, 1854
USART_CR2 1857
USART_CR3 . . . . . 1861, 1865
USART_GTPR 1869
USART_ICR 1883
USART_ISR 1872, 1878
USART_PRESC 1885
USART_RDR 1884
USART_RQR 1871
USART_RTOR 1870
USART_TDR 1885
USB_BCDR 2117
USB_CHEPnR 2118
USB_CNTR 2108
USB_DADDR 2115
USB_FNR 2115
USB_ISTR 2111
USB_LPMCSR 2116
USBSRAM_CHEP_RXTXBD_n 2130-2131
USBSRAM_CHEP_TXRXBD_n 2128

W

WWDG_CFR 1549
WWDG_CR 1548
WWDG_SR 1550

XYZ

XSPI_ABR 543
XSPI_AR 537
XSPI_CCR 540
XSPI_CR 528
XSPI_DCR1 531
XSPI_DCR2 533
XSPI_DCR3 534
XSPI_DCR4 534
XSPI_DLR 537
XSPI_DR 538
XSPI_FCR 536
XSPI_HLCR 551
XSPI_IR 542
XSPI_LPTR 543
XSPI_PIR 539
XSPI_PSMAR 539
XSPI_PSMKR 538
XSPI_SR 535
XSPI_TCR 542
XSPI_WABR 550
XSPI_WCCR 547
XSPI_WIR 550
XSPI_WPABR 547
XSPI_WPCCR 544
XSPI_WPIR 546
XSPI_WPTCR 546
XSPI_WTCR 549

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers’ market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved