MICRONAS

Application Note

CUR 4000

Programming Guide

curSENS Technology

Edition Jan. 12, 2021
APN000178_001EN

Copyright, Warranty, and Limitation of Liability

The information and data contained in this document are believed to be accurate and reliable. The software and proprietary information contained therein may be protected by copyright, patent, trademark and/ or other intellectual property rights of TDK-Micronas. All rights not expressly granted remain reserved by TDK-Micronas.

TDK-Micronas assumes no liability for errors and gives no warranty representation or guarantee regarding the suitability of its products for any particular purpose due to these specifications.

By this publication, TDK-Micronas does not assume responsibility for patent infringements or other rights of third parties which may result from its use. Commercial conditions, product availability and delivery are exclusively subject to the respective order confirmation.

Any information and data which may be provided in the document can and do vary in different applications, and actual performance may vary over time.

All operating parameters must be validated for each customer application by customers' technical experts. Any mention of target applications for our products is made without a claim for fit for purpose as this has to be checked at system level.

Any new issue of this document invalidates previous issues. TDK-Micronas reserves the right to review this document and to make changes to the document's content at any time without obligation to notify any person or entity of such revision or changes. For further advice please contact us directly.

Do not use our products in life-supporting systems, military, aviation, or aerospace applications! Unless explicitly agreed to otherwise in writing between the parties, TDK-Micronas' products are not designed, intended or authorized for use as components in systems intended for surgical implants into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the product could create a situation where personal injury or death could occur.

No part of this publication may be reproduced, photocopied, stored on a retrieval system or transmitted without the express written consent of TDK-Micronas.

TDK-Micronas Trademarks

– curSENS

Third-Party Trademarks

All other brand and product names or company names may be trademarks of their respective companies.

Contents

1. General Information

This document is intended as guidance for programming the CUR 4000 sensor using own programming hardware and software. The programming interface as well as all programming procedures for reading and writing data to the sensor's memory are described in detail. In combination with the respective data sheets and the application notes "CUR 4000 User Manual" and "CUR 4000 Programming Environment" it represents the complete customer documentation of the CUR 4000 linear and differential magnetic sensor for current measuring applications.

1.1. Certification

TDK-Micronas GmbH fulfills the requirements of the international automotive standard IATF 16949 and is certified according to ISO 9001. This ISO standard is a worldwide accepted quality standard.

1.2. Support

We kindly ask you to register on https://service.micronas.com in order to obtain access to the workgroups for our various product families. Here you are able to get support by opening a support ticket.

TDK-Micronas GmbH – Application Engineering

Hans-Bunte-Strasse 19

D-79108 Freiburg im Breisgau

2. Programming Interface

2.1. Physical Layer

The CUR 4000 is equipped with a SPI interface (Serial Peripheral Interface) for memory programming and register reading to transmit the sensor measurement data. SPI uses four wires and a master-subordinate-architecture for synchronous serial communication. The CUR 4000 is always acting as the subordinate and the ECU is the master. The SPI bus configuration with one subordinate is shown in Fig. 2–1.

Fig. 2–1: Description of the SPI Bus

On the ‘Master Out subordinate In’ (MOSI) wire the master sends data to the subordinate. On the ‘Master In subordinate Out’ (MISO) wire the subordinate sends data to the master. The ‘Chip Select’ (CSN) is driven by the master and grants the subordinate permission to read from and write to the bus. The CSN signal is active low. The ‘Serial Clock’ (SCK) is used by the master to specify the communication speed.

If there is more than one subordinate the master has to select the desired subordinate by pulling down the corresponding CSN line. The SPI bus configuration based on an example with two subordinates is shown in Fig. 2–2.

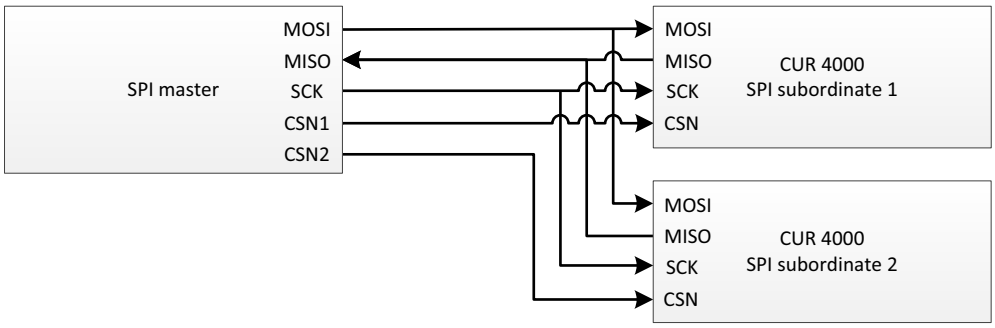

Fig. 2–2: Configuration of the SPI Bus with Two subordinates

For the CUR 4000 the typical clock frequency is 1 MHz, the signal level is 3.3 V, the ‘Clock Polarity’ (CPOL) is 0 (SCK is idle low) and the ‘Clock Phase’ (CPHA) is also 0, which means the signals (MOSI & MISO) change on the falling edge of SCK and are captured on the rising edge.

Each transfer is full duplex for simultaneously sending read/write commands to the sensor while collecting the response from the previous command. As part of the SPI protocol CUR 4000 defines a status byte, which delivers error and status information about the sensor with each SPI transfer. Additionally the protocol immanent CRC secures the correct transfer of bits as well as the correct execution of the requested command.

An example of the SPI communication protocol is shown in Fig. 2–3.

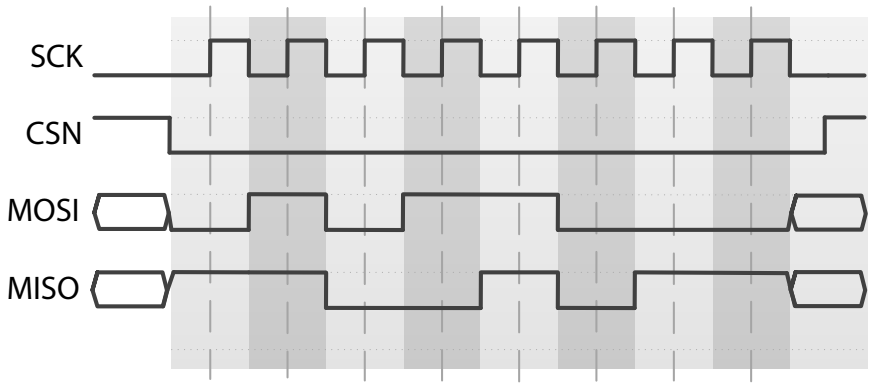

Fig. 2–3: Example of the SPI Communication Protocol

2.1.1. SPI frame

The CUR 4000 SPI frame format is as follows:

1. SPI master pulls CSN to low,
2. SPI master sends one command byte followed by two master data bytes,
3. SPI master sends a 8-bit CRC,
4. CUR 4000 replies in the next frame with one status byte and two subordinate data bytes followed by a 8-bit subordinate CRC.

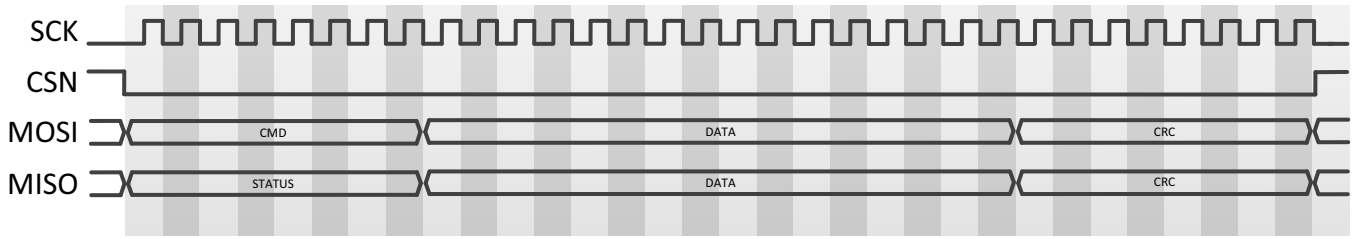

Fig. 2–4: Communication Frame Structure via SPI

Two communication frames are defined:

– Write Frame (MOSI): 8-bit command (CMD), 16-bit data and 8-bit CRC (total: 32-bit)
– Read Frame (MISO): 8-bit status (STATUS), 16-bit data and 8-bit CRC (total: 32-bit)

Write commands execute internally after the master CRC is verified in order to guarantee no unintended writing of registers.

2.1.2. Command Byte (CMD)

The command byte of the MOSI frame contains a seven bit word address (ADR) and a RWN flag as shown in Fig. 2–5. The RWN flag is the least significant bit (LSB) of the CMD byte and decides if it is a read or write command. In case of a read command (RWN is 1) the provided data bytes from MOSI are ignored. If it is a write command (RWN is 0) the data from MOSI is going to be written to ADR after the CRC byte is successfully checked.

In case of

– a read command, the data content from the requested address is sent as data bytes on MISO in the next frame.

– a write command, the subordinate responds with a status byte, the 16-bit subordinate data and the subordinate CRC byte to confirm the correct receipt of the write frame.

– power up and a write command first (there was no read command before), the master receives zero data (reset value).

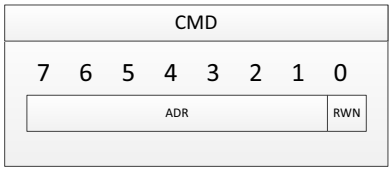

Fig. 2–5: SPI Command Byte

2.1.3. Status Byte (STATUS)

The status byte includes the following system and SPI information:

– RC[3:0]: the rolling counter keeps track of the communication frames being sent between SPI master and sensor. It is incremented by 1 with each communication frame from 1 to 15. Then it restarts at 1 again (reset value = 0),

– DIAG0: ORed bits of DIAG_0[15:0] register,

– DIAG1: ORed bits of DIAG_1[15:9] register,

– CRCERR: is set to one in case an error has been detected during CRC-check of previous MOSI frame,

– NEW: new sample indication (in case of an already received sample is sent multiple times the bit is set to 0).

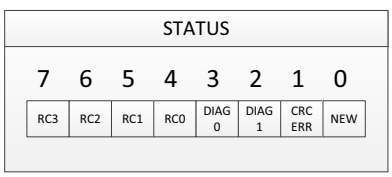

Fig. 2–6: SPI Status Byte

Note: The DIAG0 and DIAG1 bits are only updated while reading the IPC_CHANx registers. Reading EEPROM content or RAM in programming mode will not trigger the DIAG_x registers.

2.1.4. CRC Byte (CRC)

The CRC is the last byte of any transmission and covers the preceding number of bytes. A received and transmitted stream have their own CRC byte. CRC check of the MOSI frame is done every time, independent of a read/write command. Write commands are executed internally after the master CRC is verified. This guarantees that no unintended register write happens. Read commands are executed internally before the master CRC is verified. An invalid CRC indicates a detected transmission error (signaled by CRCERR = 1 in the STATUS byte). In case of a transmission error the status byte (transmitted in the next frame) gives feedback to the master via this CRCERR bit.

For details on the CRC calculation see Section 3.5.

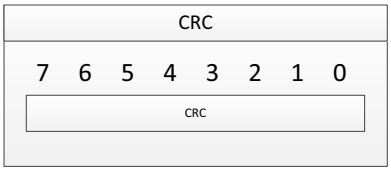

Fig. 2–7: SPI CRC Byte

2.2. Data Link and Transport Layer

2.2.1. Read Command Frame

A read command frame starts with the SPI master sending the CMD byte (7-bit address, RWN flag set to 1), followed by 16-bit master data and the 8-bit master CRC. The master data is meaningless for the read command, but relevant for the CRC calculation. It is recommended to keep the master data at 0x0000. The total frame length is 32-bit. Next, the subordinate responds with a status byte, 16-bit subordinate data and the subordinate CRC byte. The structure of a read command frame and its response is shown in Fig. 2–8.

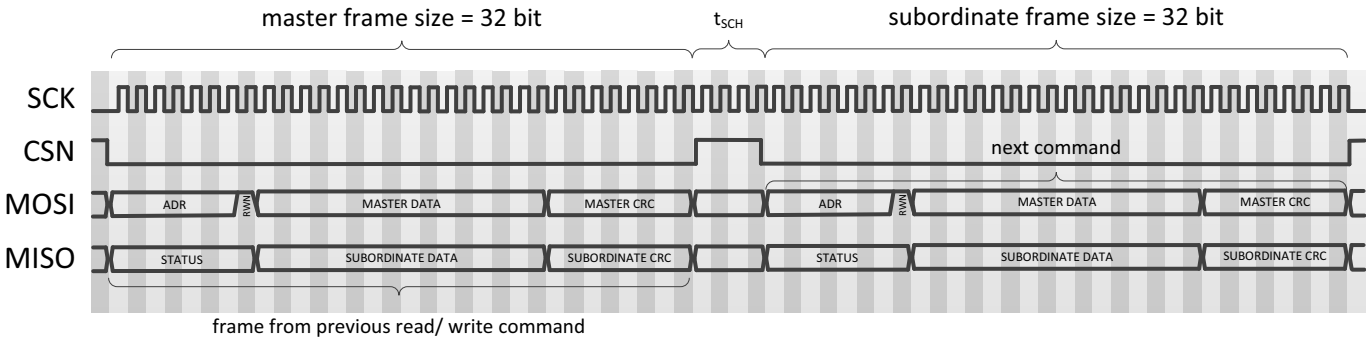

Fig. 2–8: Read Frame Structure

2.2.2. Write Command Frame

A successful mode switch into the programming mode is required to send one or more write command frames to registers 0x00 to 0x6F. It is always possible to write to registers 0x70 to 0x7F. A write command frame starts with the SPI master sending the CMD byte (7-bit address, RWN flag set so 0), followed by 16-bit master data and the 8-bit master CRC. The total frame length is 32-bit. If more than one write frame is needed, they can be set up in a pipeline directly after the first write frame (see Fig. 2–10). After the last write frame a dummy read frame is sent to the subordinate so the subordinate can respond one more time with a status byte, the 16-bit subordinate data of the last read access (the subordinate data is meaningless for the write command and also irrelevant for the master CRC calculation) and the subordinate CRC byte to confirm the correct receipt of the last write frame. The structure of one write communication frame and its response is shown in Fig. 2–9.

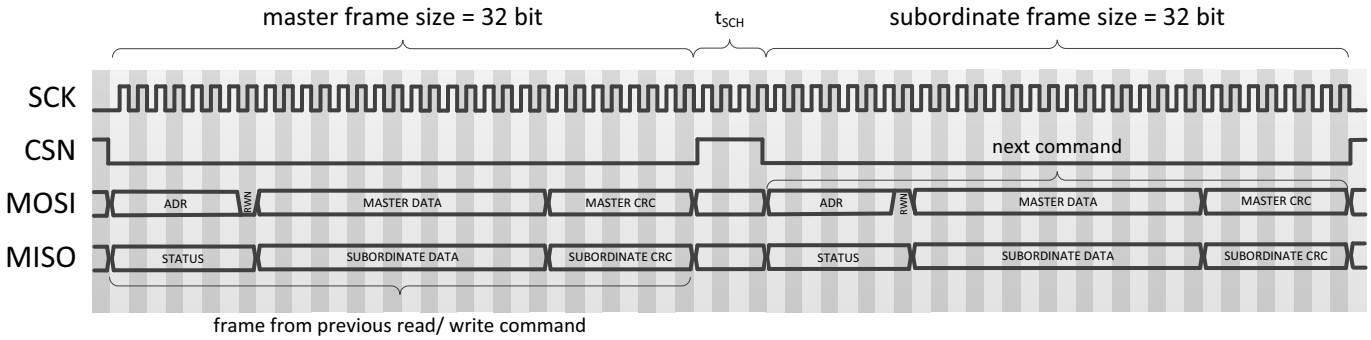

Fig. 2–9: Write Frame Structure

An example of a command pipeline is shown in Fig. 2–10:

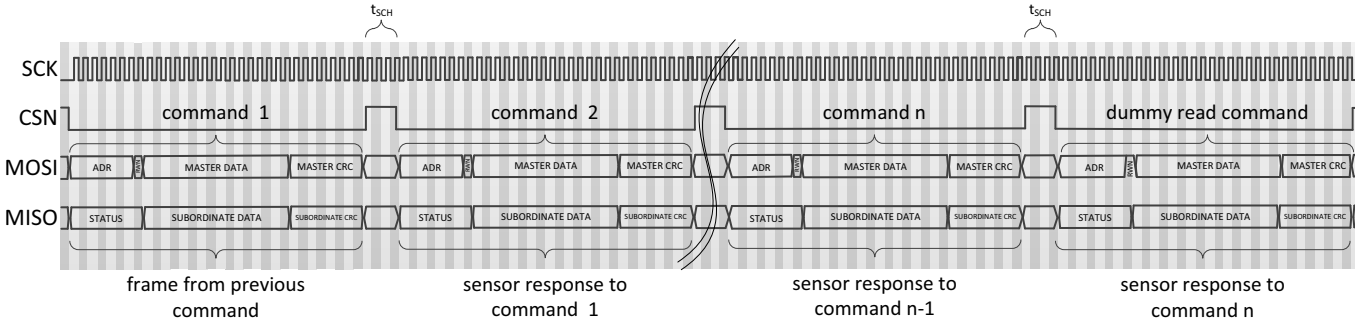

Fig. 2–10: Example of a Command Pipeline

2.3. Telegram Parameters

The SPI timing is illustrated in the following Fig. 2–11, Fig. and Table 2–1. The characteristics are given in Table 2–2.

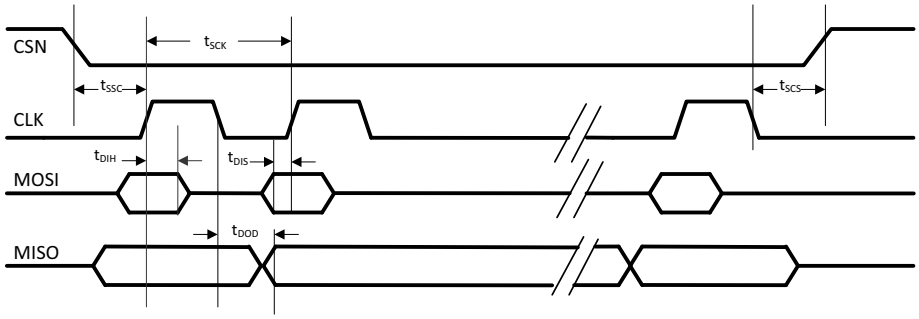

Fig. 2–11: SPI Timing Diagram

An SPI clock period of up to 10 MHz can be used within one byte. However, the SPI clock period must be halved at the byte borders between two bytes (see Fig.).

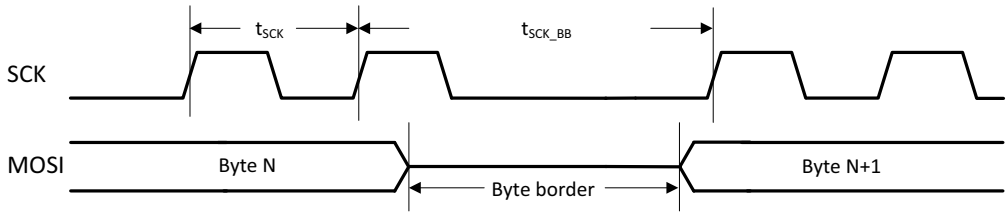

Fig. 2–12: SPI Timing Diagram Byte Border

Table 2–1: SPI Timing Parameters

Parameter	Description	Pin	Min.	Typ.	Max.	Unit	Condition
t_{SCK}	SPI clock period	SCK	100	1000	–	ns	Max. frequency 10 MHz 100 ns is only possible within one SPI byte
t_{SCK_BB}	SPI clock period byte border	SCK	200	–	–	ns	At byte border
t_{DIS}	SPI data input setup	MOSI, SCK	10	–	–	ns	Data sampling with rising SCK edge
t_{DIH}	SPI data input hold	MOSI, SCK	15	–	–	ns	
t_{DOD}	SPI data output delay	MISO, SCK	–	–	44	ns	Data output changes with falling SCK edge
t_{SSC}	SPI CSN setup time	CSN, SCK	80	–	–	ns	With respect to falling CSN edge
t_{SCS}	SPI CSN hold time	CSN, SCK	12	–	–	ns	With respect to rising CSN edge
t_{SCH}	SPI CSN high time	CSN	2	–	–	t_{SCK}	CSN high time between two consecutive SPI frames
t_{SET}	SPI settling time	–	–	4	–	ms	
t_{LISTEN}	Waiting time for the programming mode command	–	–	–	110	ms	Waiting for data 0x2EAE to address 0x75 after power on

Table 2–2: SPI characteristics

Parameter	Description	Pin	Min.	Typ.	Max.	Unit	Condition
V_{IL}	SPI input low level	MOSI SCK CSN	–	0.5	0.8	V	
V_{IH}	SPI input high level	MOSI SCK CSN	2.4	3.0	–	V	
V_{OH}	SPI output high level	MISO	V_{SUP} - 0.6 V	–	–	–	I_{OUT} = –10 mA
V_{OL}	SPI output low level	MISO	–	–	0.6	V	I_{OUT} = 20 mA
R_{PD}	Pull-down resistor	MOSI SCK	34.3	–	120	kΩ	internal pull-down resistor referred to GND
R_{PU}	Pull-up resistor	SCN	34.3	–	120	kΩ	internal pull-up resistor referred to VSUP
R_{SPI_LOAD}	Total load resistance	MISO	10	–	–	kΩ	Pull-down
C_{SPI_LOAD}	Total load capacitance	MISO	6	–	100	pF	$f_{SPI} \leq 10$ MHz
I_{OLEAK}	Leakage current	MISO	-2	–	2	µA	

3. Programming of the Sensor

3.1. Memory Access Commands

The CUR 4000 supports two commands which provide read and write access to the whole memory (RAM and EEPROM except EXT EEPROM). Table 3–1 gives an overview of all memory access commands (described in detail in the following sections).

Table 3–1: Memory Access Commands

Command	RWN bit	frame type	ADDRESS 7 bits	DATA 2 Bytes
read	0b1	read	0x00 to 0x7F	MOSI: dummy data 0x0000
write	0b0	write	0x00 to 0x7F	MOSI: data to be written to ADDRESS

3.2. Sensor Commands

The CUR 4000 supports internal programming commands (IPCs) by writing one of the following sensor commands as data bits to address 0x75 (IPC_CHAN5 register).

Table 3–2: Commands on IPC_CHAN5 Register

Command	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Start listening mode 0x22A2	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0
Start programming mode 0x2EAE	0	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0
Download EEPROM[1] 0x0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Read EXT EEPROM	1	1	0	1	0	0	0	0	0	1	1	1	4 bit address			
Write EXT EEPROM	1	1	1	1	0	0	0	0	0	1	1	1	4 bit address			
Calculate EEPROM CRC[2]	1	1	0	0	0	0	0	0	0	7-bit word for CRC						
Program EEPROM[3]	0	1	end register number							start register number						
Read Micronas ID[4] 0x000B	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
Read product and release ID[4] 0xE03F	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1
Generate Reset[5] 0x0006	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

[1] The data is copied from the EEPROM to the corresponding shadow registers.

[2] With the calculate EEPROM CRC command the CRC from bottom to the specified address is calculated. The result is given back on PROG_BUFFER0. If the whole EEPROM is to be evaluated, the residuum is returned (= 0 if stored CRC is correct) and the applicative CRC check is returned on PROG_BUFFER1. The Calculate EEPROM CRC command does not update the diagnosis registers if CRC comparison failed. Additional error codes in PROG_BUFFER1 (high byte) of the calculate EEPROM CRC command are: 0x03 (address out of range) and 0x04 (CRC mismatch).

[3)]The execution status of Program EEPROM IPC command can be checked in the registers PROG_BUFFER0 and PROG_BUFFER1 (see Section 3.3.6).

[4)]The response can be read in the registers PROG_BUFFER0 and PROG_BUFFER1.

[5)]The command Generate reset triggers a full reset. It is intended for scenarios, where an external power-on-reset is not desired, e.g. in bus configurations.

3.3. Programming Procedure

3.3.1. Listen Mode

The first step to communicate with the sensor is to set the sensor into listen mode by writing the data 0x22A2 to address 0x75. After 4 ms (t_{set}) the sensor switches and remains in listen mode for max. 110 ms (t_{listen}), during which the sensor can be switched to programming mode (see Section 3.3.2). After max. 110 ms without receiving the set programming mode command the sensor will go into reset.

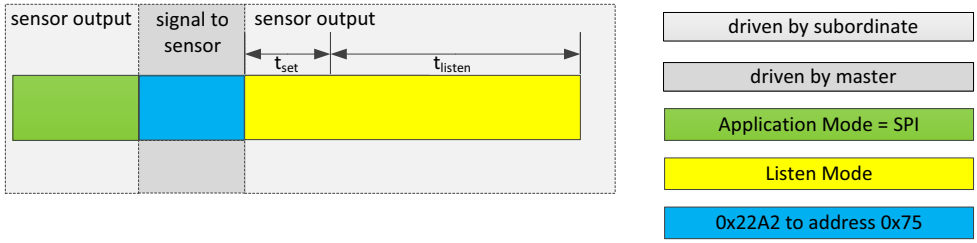

Fig. 3–1: Listen Mode Switch

3.3.2. Programming Mode

In programming mode it is possible to change the register settings of the sensor. This means for writing data into the shadow RAM it is necessary to set the sensor in programming mode. By writing the data 0x2EAE to address 0x75 when the sensor is in listen mode, the sensor will be set into programming mode. The sensor will remain in programming mode until the next power-down. After power-up the sensor is always operating in application mode.

Note: After the correct customer checksum has been stored and the customer lock is activated (setting the LOCK bit by write command and power-on reset), it is not possible to program the sensor anymore.

3.3.3. Application Mode

The sensor provides a digital output signal via SPI interface. In application mode only the registers IPC_CHAN0...3 must be used. Only those registers are secured with CRCs and error reporting (DIAG_0...1 registers are only updated while reading the IPC_CHAN0...3 registers). Furthermore, the NEW flag is only visible in IPC_CHAN0...2. For more details about the available data, please refer to the memory table of the application note “CUR 4000 User Manual”.

3.3.4. Write Data into Shadow RAM

It is not possible to write and store data directly into the EEPROM. By using the write command the data will be written into the shadow RAM first. In case of a successful write the data can be stored to the EEPROM as described in the following section.

Note: In case the data has only been written into the shadow RAM and not additionally into the EEPROM, the data will be lost in the event of a power-down of the sensor.

3.3.5. Store Data to the EEPROM

To store data from shadow RAM to the EEPROM it is necessary to send a write command frame to address 0x75 (Program EEPROM IPC command) which contains the start register number and end register number of the area to be stored. The structure of this write command frame 'program EEPROM' is shown in Table 3–2.

Note: Only during storing the EXT EEPROM register numbers follow directly on the EEPROM register numbers. This means the total register number is 96. The EEPROM register numbers start at 0x00 and end at 0x4F. The EXT EEPROM register numbers start at 0x50 and end at 0x5F. This is not valid for reading or writing the EXT EEPROM.

Storing One Register to the EEPROM

For storing only one register to the EEPROM the start register number and end register number have to be identical. When the write command frame 'program EEPROM' is sent to the sensor, the command and the CRC are correctly received, then the STATUS byte transmitted in the next communication frame indicates whether the sensor has executed the command.

Storing multiple Registers to the EEPROM

Regardless how many registers shall be stored from the shadow RAM to the EEPROM, only one write command frame has to be sent including start and end register number.

3.3.6. Check Programming Success

To ensure that the programming of the EEPROM was successful, the register PROG_BUFFER0 shall be read afterwards.

A countdown is set in the PROG_BUFFER0 register, when a 'program EEPROM' command is sent to the sensor. The number of registers which should be stored with the 'program EEPROM' command defines the countdown. The countdown decreases with every stored register.

If there is any problem during the storing of the registers and the programming is stopped, the remaining countdown can be read in the PROG_BUFFER0 register, so that it is possible to check which registers are written and which are not. In this case the 'program EEPROM' sequence shall be repeated.

Error Status and Codes

Status and error information is given in the PROG_BUFFER0 and IPC_CHAN5 registers. Generally, IPC_CHAN5 indicates the status and error codes of commands executed by the sensor to give a universal register to be read out by the customer. The standard status flag (0x0001) is written to IPC_CHAN5 when the command starts.

An overview of the error codes is given in Table 3–3 and the content of PROG_BUFFER0 and PROG_BUFFER1 after IPC_CHAN5 programming in Table 3–4.

Table 3–3: Status Codes at Register IPC_CHAN5

Status code	Description
0x00A8	No error (‘success’ in Table 3–4).
0x0001	Sensor busy executing the last command
0x0002	Error of unspecified source
0x0102	Command is locked
0x0202	Invalid command
0x0302	Register number out of range (start register number > end register number or end register number > 95)
0x0402	Mismatch content
0x0502	Time-out (hardware failure)

Table 3–4: Content of PROG_BUFFER0, PROG_BUFFER1 and IPC_CHAN5 after IPC_CHAN5 Programming

Sensor Command	PROG_BUFFER0	PROG_BUFFER1	IPC_CHAN5
Program EEPROM	0/Countdown status at error	–	success/error
Start programming mode	–	–	success
Read EXT EEPROM	16 bit result	–	success
Write EXT EEPROM	–	–	success
Read Micronas ID	16 bit result (MIC ID1)	16 bit result (MIC ID2)	success/error
Download EEPROM	–	–	success
Generate reset	–	–	–

3.4. Flowcharts

3.4.1. Read

The read EEPROM telegram uses the read command frame (see Section 2.2.1). The sensor transmits the data of the effective address after the master read frame has been successfully received. Otherwise the STATUS byte (CRCERR flag) of the subordinate frame gives feedback to the master that an error occurred. To read an EXT EEPROM register (word addresses between 0x0 and 0xF) the Read EXT EEPROM command must be used. The desired address is added to the command 0xD070. This gives the final sensor command (see Table 3–2). After the command has been sent to IPC_CHAN5 the requested 16 bit register data can be read in PROG_BUFFER0.

An exemplary flowchart for reading EEPROM register 0x11 (which contains the data 0x2000) is given in Fig. 3–2.

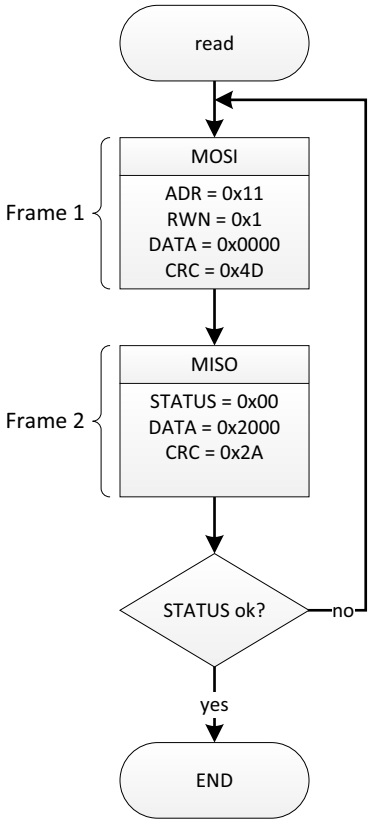

Fig. 3–2: Flowchart - Read from Address 0x11

3.4.2. Write Data to Shadow RAM

The write telegram uses the write command frame (see Section 2.2.2). The subordinate (sensor) transmits a 32-bit frame (STATUS byte ok) after the command and the CRC have been checked, the data has been successfully received, and the effective address is permitted (only in programming mode it is possible to write to the shadow RAM). In case the sensor is not in programming mode a write to shadow RAM will be ignored, but not indicated in the STATUS byte. So if an error occurs, no frame will be transmitted. To write an EXT EEPROM register (word addresses between 0x0 and 0xF) first write the desired data to the PROG_BUFFER0 register using the Write command frame and then the Write EXT EEPROM command must be used. The desired address is added to the command 0xF070. This gives the final sensor command (see Table 3–2).

An exemplary flowchart for writing 0x2000 to the shadow RAM register 0x11 is given in Fig. 3–3.

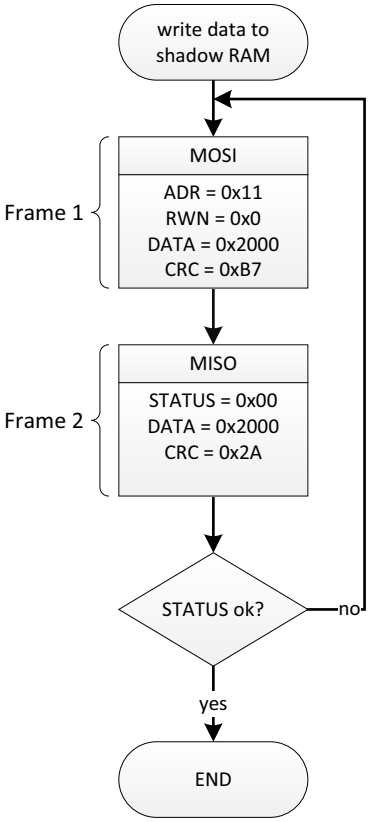

Fig. 3–3: Flowchart - Write Data 0x2000 to Shadow RAM Register 0x11

3.4.3. Store Data from Shadow RAM into EEPROM

To store data to the EEPROM the ‘Program EEPROM’ IPC command has to be used (see Section 3.2). The subordinate (sensor) transmits a 32-bit frame (STATUS byte ok) after the command and the CRC have been checked and the data has been successfully received.

The sensor additionally transmits a 32-bit frame (STATUS byte ok) after each correctly received EEPROM write command. Otherwise the command is aborted and no frame is transmitted. Any new write telegram to the sensor is discarded during the EEPROM programming.

If undefined EEPROM addresses have been used in the command, it is not indicated in the STATUS byte. This can only be seen in the PROG_BUFFER1 register.

Note: Only during storing the EXT EEPROM register numbers follow directly the EEPROM register numbers. This means the total register number is 96. The EEPROM register numbers start at 0x00 and end at 0x4F. The EXT EEPROM register numbers start at 0x50 and end at 0x5F. This is not valid for reading or writing the EXT EEPROM.

Note: An exemplary flowchart for the store EEPROM command (storing address 0x00 to 0x09) is given in Fig. 3–4.

Note: In case of an error the PROG_BUFFER0 register holds the countdown value indicating which EEPROM address has failed (see Section 3.3.6).

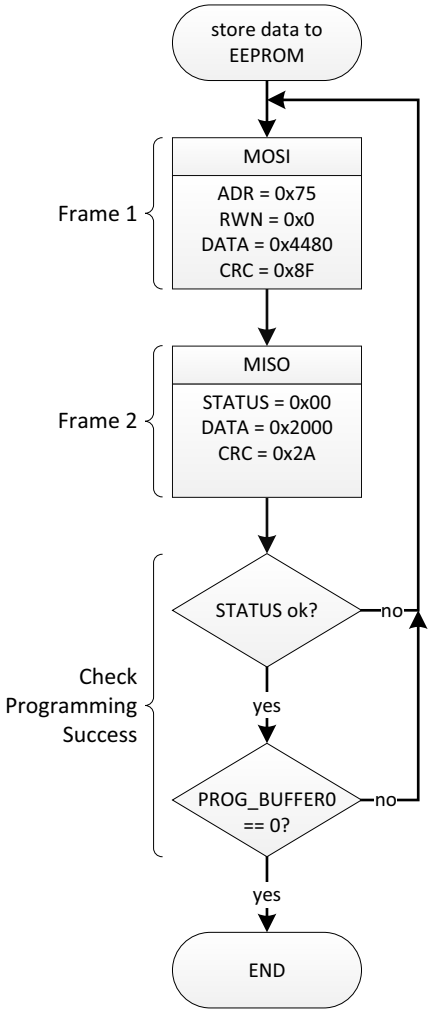

Fig. 3–4: Flowchart - Store Data from Shadow RAM Register Address 0x00 to 0x09 to EEPROM

3.5. CRC Calculation

DATA bits are always followed by 8 CRC bits. There are two different cases of the calculation of the CRC byte:

1. Master CRC: for the read and the write command on MOSI, the CRC is calculated based on the following 3 bytes:

CMD[7:0]	MASTER DATA[15:8]	MASTER DATA[7:0]

2. subordinate CRC: the byte-order (of 4 bytes) on which the CRC calculation for MISO is based on is shown below:

STATUS[7:0]	COMMAND[7:0]	DATA[15:8]	DATA[7:0]

The second byte after the actual STATUS byte is the COMMAND byte (last address respectively address from previous frame). This byte is structured as follows:

LAST_ADDRESS[6:0]	RWN

The DATA bytes (Byte 3 = DATA[15:8], Byte 4 = DATA[7:0]) always correspond to the last requested read data on MISO line (initially zero after power up), which is also used for the CRC calculation.

The polynomial for the CRC calculation is $X^8 + X^4 + X^3 + X^2 + 1$ (0x1D), with a seed value of 0xFFU and a final XOR value of 0xFFU (CRC-8-SAE-J1850).

An invalid CRC indicates a detected transmission error.

An example of the CRC calculation is shown below:

```
const uint8_t CRC_POLY = 0x1DU; // x^8+x^4+x^3+x^2+1
uint8_t crc8 (uint8_t *data, uint8_t length){
   uint8_t i, bit;
   uint8_t crc = 0xFFU;
   for (i = 0; i < length; i++)
   {
        crc = crc ^ data[i];
        for (bit = 0; bit < 8; bit++)
        {
             if (crc & 0x80U)
             {
                crc = crc << 1;
                crc = crc ^ CRC_POLY;
             }
             else
                crc = crc << 1;
        }
   }
   return ~crc;
}
```

Note: The first frame of MISO after power up consists of DATA[15:0] = 0 and CMD[7:0] = 0 (because there was no MISO frame so far). These bytes are used for the first CRC calculation of the subordinate answer. As long as no read command is used, the response (of MISO) remains at DATA[15:0] = 0 and CMD[7:0] = 0 for the CRC calculation. With the first read command the subordinate response changes: DATA[15:0] = READ-Data and CMD[7:0] = READ-command and stays that way until the next read command.

4. Application Note History

1. CUR 4000 Programming Guide, Jan. 12, 2021; APN000178_001EN.
   First release of the application note.

TDK-Micronas GmbH
Hans-Bunte-Strasse 19 · D-79108 Freiburg · P.O. Box 840 · D-79008 Freiburg, Germany
Tel. +49-761-517-0 · Fax +49-761-517-2174 · www.micronas.tdk.com