AN2665

Interfacing AVR® Microcontrollers with Serial Memories

Features

- EEPROM Devices: Microchip 25xx256(25AA256/25LC256)
- DataFlash Devices: Microchip SST25VFxxxx (SST25VF010A, SST25VF020, SST25VF040B SST25VF080B)
- SPI DataFlash Driver Using 1/2/4/8 Mbit SPI Serial Flash and ATtiny817
- SPI EEPROM Driver Using 256K SPI Bus Serial EEPROM and ATtiny817
- Full Serial Memory Functions Support
- SST25VFxxxx Features:
  - Flexible erase capability
  - Fast erase and byte program
  - Auto Address Increment (AAI) programming
- 25xx256 Features:
  - 64-Byte page
  - Block write protection
  - Built-in write protection

Introduction

Author: Rupali Honrao, Microchip Technology Inc.

Serial interface memories are used in a broad spectrum of consumer, automotive, telecommunication, medical, industrial, and PC related markets. Primarily used to store personal preference data and configuration/setup data, serial memories are the most flexible type of nonvolatile memory (NVM) utilized today. Compared to other NVM solutions, serial memory devices offer lower pin count, smaller packages and lower voltages, as well as lower power consumption.

Most AVR® microcontrollers provide an SPI interface, which enables connection with serial memory devices like EEPROM 25AA256/25LC256 and DataFlash devices like SST25VF010A, SST25VF020, SST25VF040B and SST25VF080B.

To ease and accelerate SPI serial memory integration with AVR devices, basic drivers were developed to provide efficient access. This application note describes the functionality and the architecture of these drivers. This application note also provides driver source code in Atmel | START.

Table of Contents

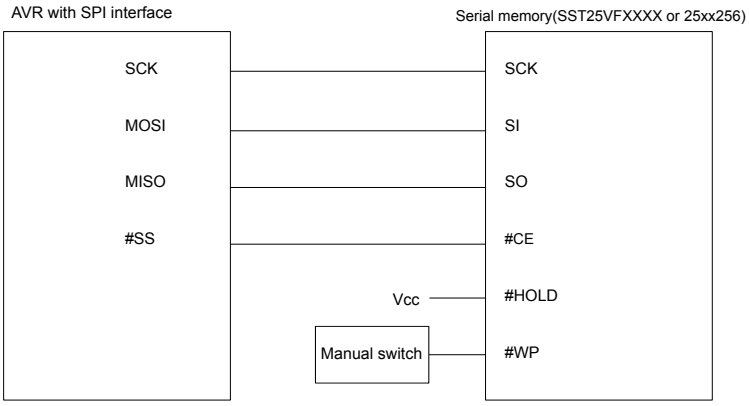
1. Serial Memories

The following sections describe connecting the memory to an AVR processor, memory access types, block write protection bits, and busy detection. For more information, refer to the device datasheet.

1.1 Serial Memory to AVR Hardware Connection

The SPI interface is configured in the Master mode. The SCK and MOSI ports are outputs and the MISO port is an input. Refer to Figure 1-1.

Figure 1-1. Hardware Connections



Note that in this application note the Slave Select signal #SS is used to control the chip enable pin #CE of the serial memory.

The #HOLD signal can be held high in its inactive state because the SPI interface always halts the clock if the data is not valid.

The user can choose the configuration of the write protect signal #WP.

Note: EEPROM's operating voltage range is from 1.8V to 5.5V whereas DataFlash devices operate from 2.7V to 3.6V.

1.2 SPI Serial Memory Access Methods

In order to understand the structure of the provided drivers, it is necessary to review the different methods of serial memory access.

All accesses follow this sequence:

1. The chip select line is driven low.
2. A number of serialized bytes are sent or received synchronously to the SCK clock on the data lines.
3. The chip select line is driven high.

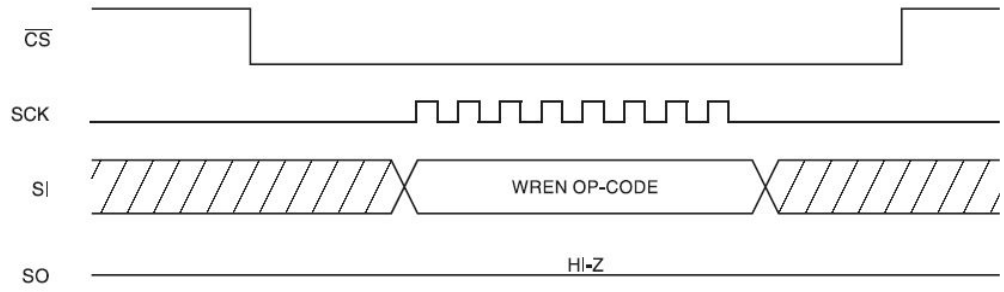
The number and values of the serialized bytes depends on the access type.

The SPI memory devices of this application note both support SPI mode 0 (0,0) and mode 3 (1,1). In the proposed driver mode 0 is selected (see the serial memory device datasheets for details).

1.2.1 Single Byte Write Commands: WREN, WRDI, CHIP ERASE

These accesses are one byte long. Only the instruction byte (later on called “op_code”) is sent on the MOSI data line.

Figure 1-2. Example WREN Timing



1.2.2 Read/Write Status Register: RDSR/WRSR

These accesses are two bytes long: The op_code byte is followed by the byte to be written (WRSR) or to be read (RDSR).

Note: The Write Status register operation must be preceded by the WREN command.

Figure 1-3. WRSR Timing

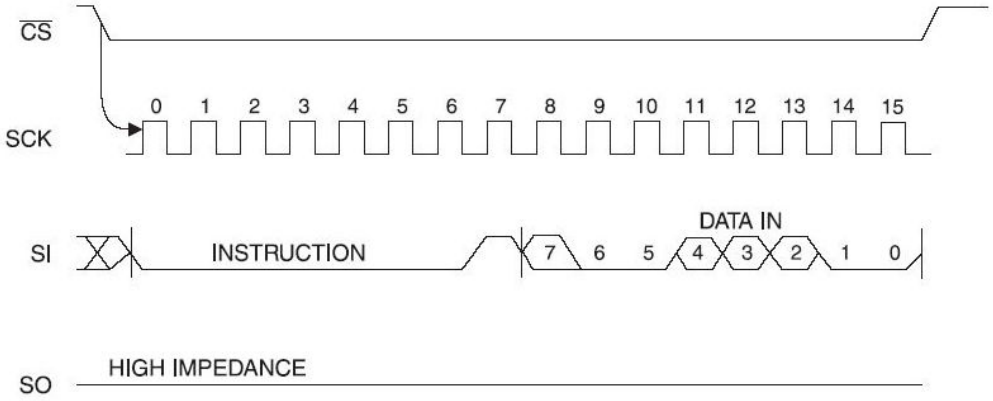
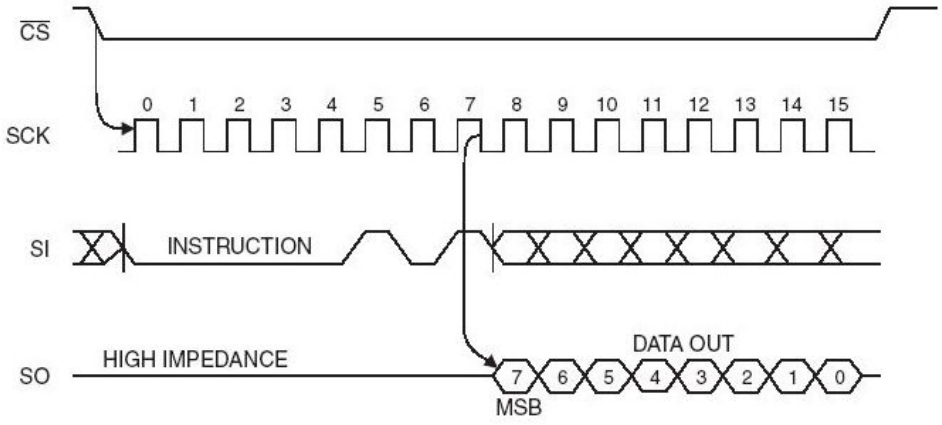


Figure 1-4. RDSR Timing



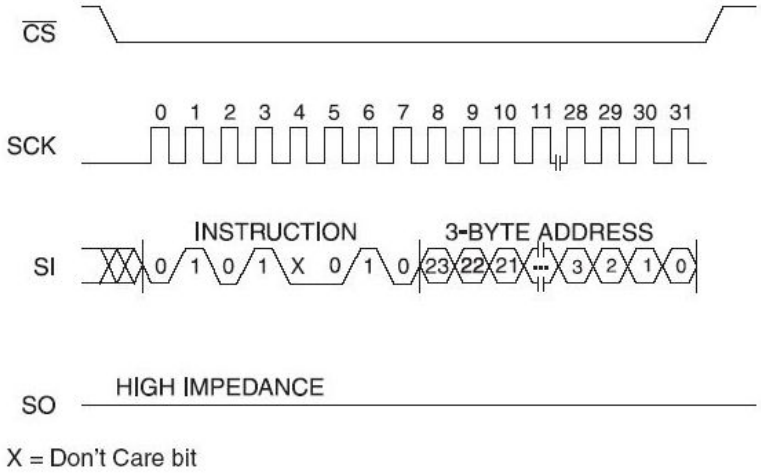
1.2.3 Sector-/Block-Erase Command

(only for SST25VFxxxx devices)

This access consists of a one-byte op_code and three address bytes. Figure 1-5 illustrates the 32-KByte Block-Erase sequence.The 32-KByte Block-Erase instruction is initiated by executing an 8-bit command, 52H, followed by three address bytes.

Note: Prior to any Write operation, the Write-Enable (WREN) instruction must be executed

Figure 1-5. 32-KByte Block-Erase Timing



1.2.4 Data Write and Data Read Accesses

These accesses consist of a one-byte op_code followed by an address phase and a data phase. The address phase is two bytes of address for 25xx256 devices and three bytes of address for the SST25VFxxxx devices.

Prior to any attempt to write data, the write enable latch must be set by issuing the WREN instruction.

1.2.4.1 Data Write for 25xx256 device

Up to 64 bytes of data can be sent to the device before a write cycle is necessary. The only restriction is that all of the bytes must reside in the same page. Page write operations are limited to writing bytes within a single physical page, regardless of the number of bytes actually being written.

Figure 1-6. Byte Write Sequence (25XX256)

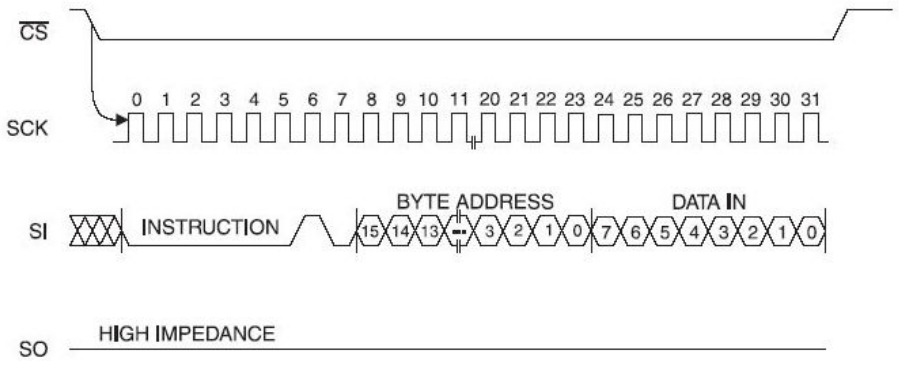
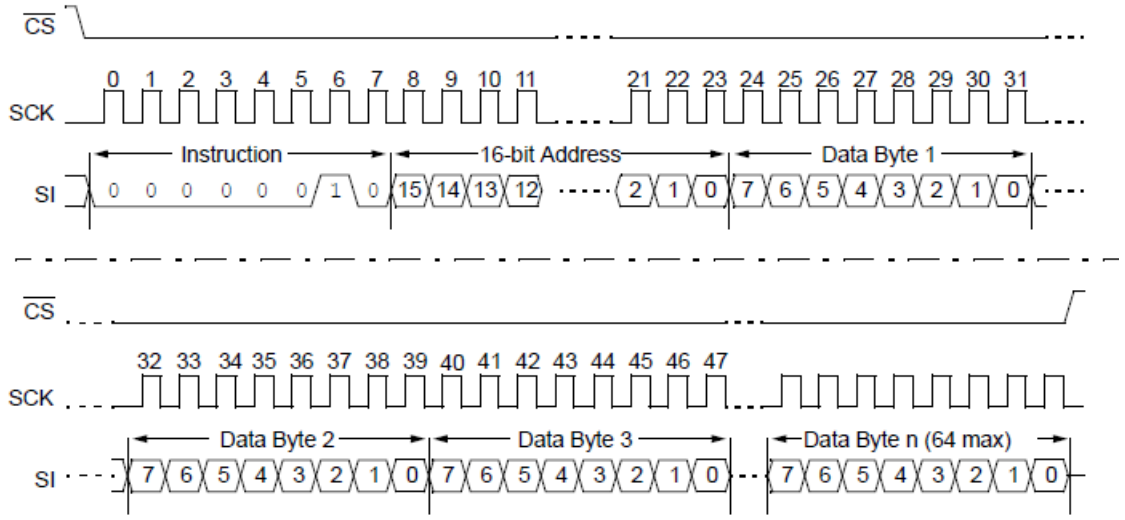


Figure 1-7. Page Write Sequence (25XX256)

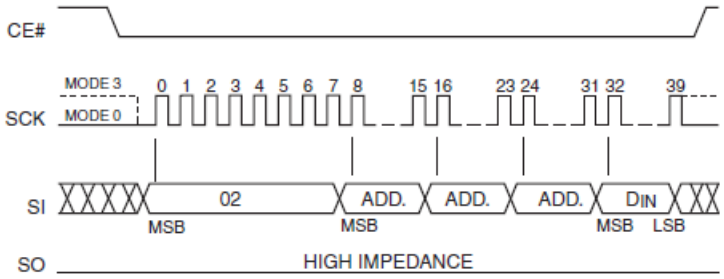


1.2.4.2 Data Write for SST25VFxxxx devices

Byte-Program:

The Byte-Program instruction programs the bits in the selected byte to the desired data, illustrated in Figure 1-8.

Figure 1-8. Byte Write Sequence (SST25VFxxxx)



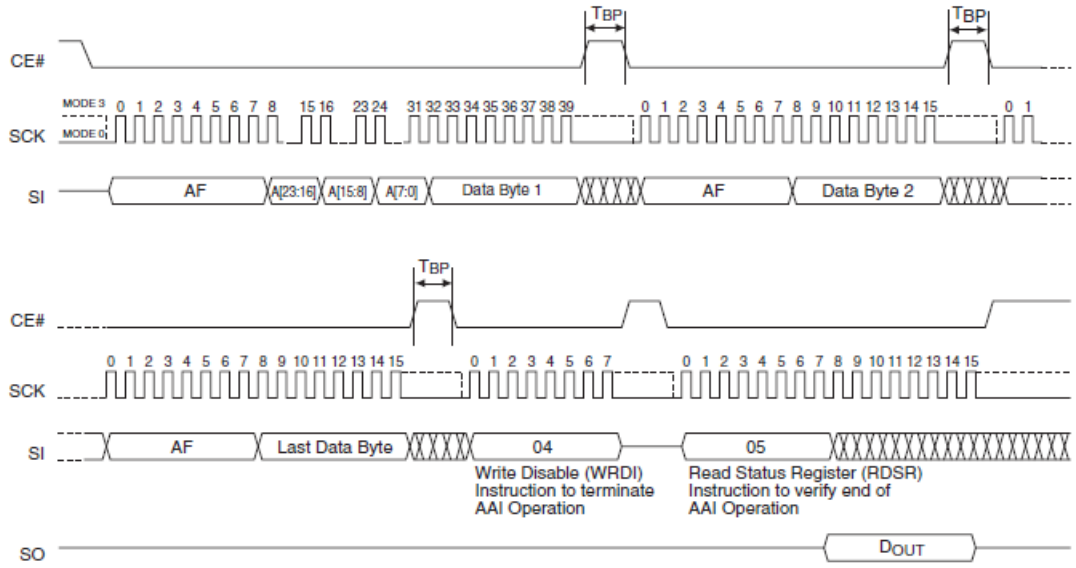
Auto Address Increment (AAI) Program:

The AAI program instruction allows multiple bytes of data to be programmed without re-issuing the next sequential address location. This feature decreases total programming time when the entire memory array is to be programmed. Prior to any write operation, the Write-Enable (WREN) instruction must be executed.

The AAI program instruction is initiated by executing an 8-bit command, AFH, followed by address bits [A23-A0]. Following the addresses, the data is input sequentially from MSB (bit 7) to LSB (bit 0). Once the device completes the programming byte, the next sequential address may be programmed. Enter the 8-bit command, AFH, followed by the data to be programmed. When the last desired byte has been programmed, and execute the Write-Disable (WRDI) instruction, 04H, to terminate AAI. After execution of the WRDI command, the user must poll the Status register to ensure the device completes programming.

Refer Figure 1-9 for AAI programming sequence. This is applicable for devices SST25VF010A and SST25VF020.

Figure 1-9. Auto Address Increment (AAI) Program Sequence (SST25VF010A/SST25VF020)



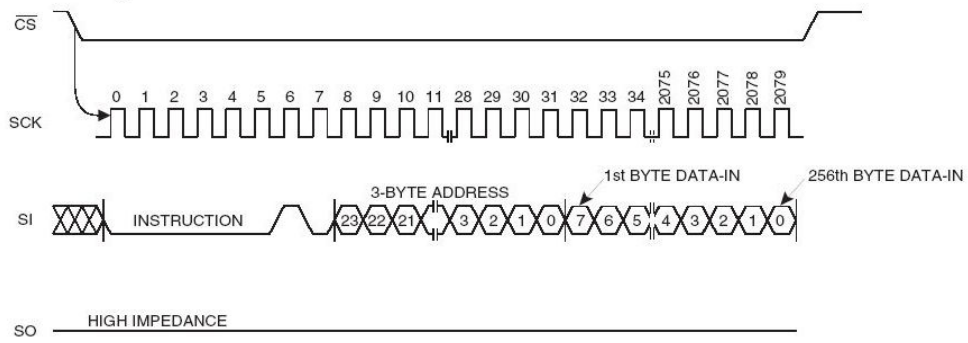
Auto Address Increment (AAI) Word-Program:

The Auto Address Increment (AAI) Word-Program Sequence is similar to Auto Address Increment (AAI) Program Sequence described above except:

- The op_code for AAI Word-Program is ADH
- Two bytes of data are written
- The sequence to terminate AAI Word_program in Hardware End-of-Write Detection mode is: Execute the Write-Disable (WRDI) instruction, 04H, followed by the 8-bit DBSY command, 80H.

Refer Figure 1-10 for Auto Address Increment (AAI) Word-Program. It is applicable for devices SST25VF040B/080B.

Figure 1-10. Auto Address Increment (AAI) Word-Program Sequence (SST25VF040B/080B)



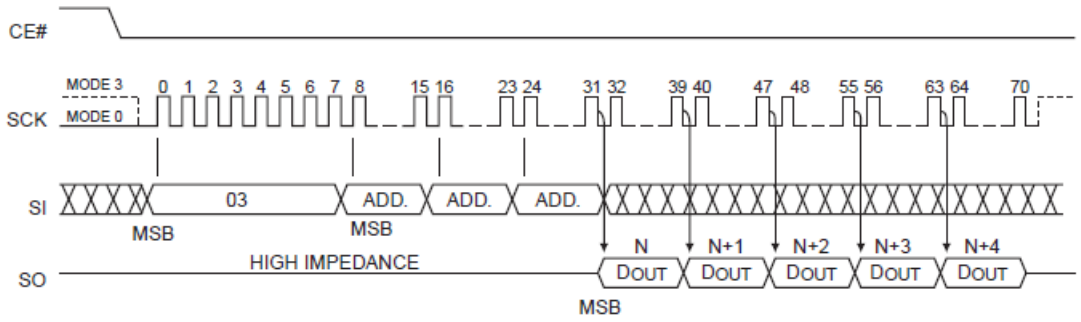
1.2.4.3 Data Read Accesses

These accesses consist of a one-byte op_code followed by an address phase. The address phase is two bytes of address for 25xx256 devices and three bytes of address for the SST25VFxxxx devices.

The device is selected by pulling CE low. The 8-bit READ instruction is transmitted to the device followed by the address. After the correct READ instruction and address are sent, the data stored in the memory at the selected address is shifted out on the SO pin. The data stored in the memory at the next address can be read sequentially by continuing to provide clock pulses. The internal address pointer is

automatically incremented to the next higher address after each byte of data is shifted out. When the highest address is reached, the address counter rolls over to address 0000h allowing the read cycle to be continued indefinitely. The read operation is terminated by raising the CE pin.

Figure 1-11. READ Sequence (SST25VFxxxx)



1.2.5 Access Type Summary

The following table summarizes the access types and their related lengths:

Table 1-1. Access Type Summary Table

Command	Op_code	Address phase length (bytes)		Data phase length (bytes)	
		25xx256	SST25VFxxxx	MOSI line	MISO line
WREN	0000 0110b (06H)	0	0	0	0
WRDI	0000 0100b (04H)	0	0	0	0
RDSR	0000 0101b (05H)	0	0	0	1(4)
WRSR	0000 0001b (01H)	0	0	1	0
READ	0000 0011b (03H)	2	3	0	1 to ∞(4)
WRITE	0000 0010b (02H)	2	N/A(1)	1 to 64	0
BYTE PROGRAM	0000 0010b (02H)	N/A(1)	3	1	0
AAI-Word-Program	1010 1101b (ADH)	N/A(1)	3(2)	2	0
AAI-Program	1010 1111b (AFH)	N/A(1)	3(3)	1	0
4 KByte Sector-Erase	0010 0000b (20H)	N/A(1)	3	0	0
32 KByte Block-Erase	0101 0010b (52H)	N/A(1)	3	0	0
64 KByte Block-Erase	1101 1000b (D8H)	N/A(1)	3(2)	0	0

Command	Op_code	Address phase length (bytes)		Data phase length (bytes)	
		25xx256	SST25VFxxxx	MOSI line	MISO line
CHIP ERASE	0110 0000b (60H) or 1100 0111b (C7H)	N/A(1)	0	0	0
RDID	1001 0000b (90H) or 1010 1011b (ABH)	N/A(1)	3	0	1 to ∞(4)

Note:

1. Command not applicable for this device.
2. Command is applicable for SST25VF040B/080B devices.
3. Command is applicable for SST25VF010A/020 devices.
4. Data phase length (bytes) for SST25VFxxxx devices is 1 to ∞. The data is read continuously with ongoing clock cycles until terminated by a low to high transition on CE# for SST25VFxxxx devices.

1.3 SPI Peripheral Handling

To perform the accesses to serial memory the SPI interface: a synchronous serial communication interface is used.

During each SPI clock cycle, the master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it. This sequence is maintained even when only one-directional data transfer is intended.

The SPI operates byte-wise. After a byte was sent or received, the flag SPIF is set in the INTFLAGS register.

There are two possible methods of SPI peripheral operation:

1. Polling the INTFLAGS register to detect if the SPIF was set, and then perform the next SPI transfer.
2. Enable the SPI interrupt and manage the next SPI transfer in the SPI interrupt handler.

Note: Polling method of SPI peripheral operation is used in the provided source code.

1.4 Busy Detection

The serial memory 25xx256 device indicates its current state with the bit WIP(Write-in-Process) in its status register.

The serial memory SST25VFxxxx device indicates the busy state with bit BUSY in its status register. When this bit set to a ‘1’, a write is in progress, when set to a ‘0’, no write is in progress. This bit is read-only.

2. Get Source Code from Atmel | START

The example code is available through Atmel | START, which is a web-based tool that enables configuration of application code through a Graphical User Interface (GUI). The code can be downloaded for both Atmel Studio and IAR Embedded Workbench® via the direct example code-link(s) below or the *BROWSE EXAMPLES* button on the Atmel | START front page.

Atmel | START web page: http://start.atmel.com/

Example Code

- SPI DataFlash:
  - http://start.atmel.com/#example/Atmel:spi_serial_memory:1.0.0::Application:SPI_DataFlash:
- SPI EEPROM:
  - http://start.atmel.com/#example/Atmel:spi_serial_memory:1.0.0::Application:SPI_EEPROM:

Press *User guide* in Atmel | START for details and information about example projects. The *User guide* button can be found in the example browser, and by clicking the project name in the dashboard view within the Atmel | START project configurator.

Atmel Studio

Download the code as an .atzip file for Atmel Studio from the example browser in Atmel | START, by clicking *DOWNLOAD SELECTED EXAMPLE*. To download the file from within Atmel | START, click *EXPORT PROJECT* followed by *DOWNLOAD PACK*.

Double-click the downloaded .atzip file and the project will be imported to Atmel Studio 7.0.

IAR Embedded Workbench

For information on how to import the project in IAR Embedded Workbench, open the Atmel | START user guide, select *Using Atmel Start Output in External Tools*, and *IAR Embedded Workbench*. A link to the Atmel | START user guide can be found by clicking *About* from the Atmel | START front page or *Help And Support* within the project configurator, both located in the upper right corner of the page.

3. Source Code Overview

The following sections describe peripherals, initialization of the SPI Interface, the polling mode functions and DataFlash/EEPROM Read/Program functions.

Peripherals:

- CPU clock : (default) 3.33 MHz.
- Peripherals used:
  - SPI
    - PA1: MOSI
    - PA2: MISO
    - PA3: SCK
    - PA4: SS

The project configured in Atmel | START generates peripheral driver functions and files, as well as a `main()` function that initializes all drivers.

- Peripheral driver header and source files are in the *src* and *include* folder.
- In the *atmel_start.c* file, the function `atmel_start_init()` initializes MCU, peripheral drivers and middleware in the project.
- DataFlash driver files: dataflash.c, dataflash.h
- EEPROM driver files: eeprom.c, eeprom.h

3.1 Initialization of the SPI Interface

The function `dataflash_init()`, initializes SPI in the DataFlash driver.

The function `eeprom_init()`, initializes SPI in the EEPROM driver.

1. The port directions are configured as the following:
   1.1. Outputs: PA3/SCK, PA1/MOSI, PA4/SS
   1.2. Inputs: PA2/MISO.
2. The SPI interface is configured as a master by configuring the CTRLA register.
3. For SPI clock speed, the clock prescaler used is DIV4 and clock speed is doubled using the CLK2X bit in CTRLA register.
4. SPI Transfer mode is selected as mode 0 and slave select is disabled by configuring the SSD bit high. By configuring the SSD bit high, #SS does not disable Master mode.
5. The slave select line is configured high.

3.2 Polling Operation Mode Functions

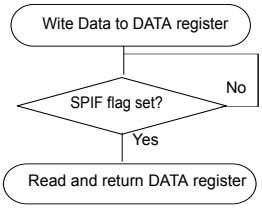
The SPI Transfer Elementary Function

The function `spi()` is the elementary function that handles the SPI interface in polling mode. It sends a byte and stores the received one. Note that this function does not control the level of the chip select line. `spi()` is called from `tx_spi()` and `rx_spi()`. The function `tx_spi`() writes a byte on SPI and rx_spi() receives the byte.

Prototype:

```
char spi(uint8_t data);
```

Figure 3-1. SPI Transfer Function Flowchart



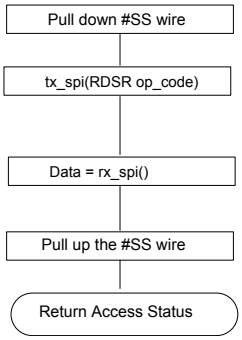
The Read Status Register Function

The functions `dataflash_read_status_register()` and `eeprom_read_status_register()` perform an RDSR access. They return the access status value.

Prototype:

```
uint8_t dataflash_read_status_register(void);
uint8_t eeprom_read_status_register(void);
```

Figure 3-2. Read Status Register Function Flowchart



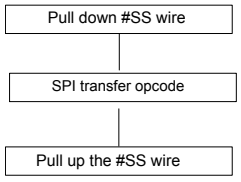
The Single Byte Write Command

The function `send_cmd()` sends the single byte op_codes, such as WREN,WRDI, etc. The serial memory must be ready before performing the access.

Prototype:

```
void send_cmd(uint8_t opcode)
```

Figure 3-3. Send Command Flowchart



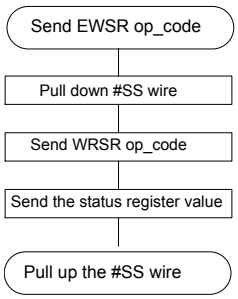
The Write Status Register Function

This function performs the RDSR access. It sends op_codes OP_EWSR: Enable Write Status Register and OP_WRSR: Write Status Register and then sends the data to be written to the status register.

Prototype:

```
void dataflash_write_status_register(uint8_t byte);
```

Figure 3-4. Write Status Register Function Flowchart



3.3 DataFlash Read & Program Functions

DataFlash functions are implemented and defined in dataflash.c and dataflash.h in the example source code.

DataFlash Read Single Byte:

The function `dataflash_read_byte()` reads one byte from the DataFlash from the given address location.

Prototype:

```
uint8_t dataflash_read_byte(uint32_t addr);
```

DataFlash Read Multiple Bytes:

Reading multiple bytes is performed using the following three functions. These functions are called from `read_dataflash()` in the example source code main.c

Prototype:

```
void dataflash_read_multiple_start(uint32_t addr);
void dataflash_read_multiple_continue(uint8_t *buff, uint8_t buff_length);
void dataflash_read_multiple_stop();
```

`dataflash_read_multiple_start()` pulls down the #SS wire, sends op_code READ and sends the start address.

`dataflash_read_multiple_continue()` continuously reads data until the number of bytes read are equal to the given `buff_length`. Data is stored in the given buffer.

`dataflash_read_multiple_stop()` pulls up the #SS wire to stop the read process.

DataFlash Program Single Byte:

The function `dataflash_program_byte()` programs a single byte to the given address.

Prototype:

```
void dataflash_program_byte(uint32_t addr, uint8_t byte);
```

DataFlash Program multiple Bytes:
Programming multiple bytes is performed using the following three functions. These function are called from `write_to_dataflash()` in the example source code main.c

Prototypes:

```
void dataflash_program_multiple_start(uint32_t addr);
uint8_t dataflash_program_multiple_continue(uint8_t *buff, uint8_t length);
void dataflash_program_multiple_stop();
```

Programming multiple bytes uses the AAI program instruction, which allows multiple bytes of data to be programmed without re-issuing the next sequential address location.

`dataflash_program_multiple_start()` calls `AAI_program_start()` function. It sends op_code EBSY and WREN, pulls down the #SS wire and sends the op_code OP_AAI_WORD_PROG. Then it sends the start address from where data needs to be read.

`dataflash_program_multiple_continue()` calls `AAI_program_continue()` to write word or byte depending on device selection. For devices SST25VF040B/SST25VF080B, a word is programmed and for devices SST25VF010A/SST25VF020 a byte is programmed. The function `AAI_program_continue()` is called until number of bytes written equals given length. When last byte is programmed, the function returns 0.

`dataflash_program_multiple_stop()` function calls `AAI_program_stop()`. It sends the stop sequence: sends op_code OP_WRDI and OP_DBSY.

Note: For op_codes description refer to dataflash.h or the device data sheet.

Data Flash read Device ID and Manufacturer's ID functions:

Prototypes:

```
uint8_t dataflash_read_id(uint32_t addr); /* Use addresse ADD_DEV_ID :0x01 to Read Device ID
*/
uint32_t dataflash_jedec_id_read(); /*Reads  Manufacturer's ID from address 0x00*/
```

Data Flash erase functions:

Prototypes:

```
void dataflash_erase_chip(void);              /* Erase entire chip*/
void dataflash_erase_sector_4k(uint32_t addr);/* Erase sector 4K*/
void dataflash_erase_block_32k(uint32_t addr);/* Erase sector 32K*/
void dataflash_erase_block_64k(uint32_t addr);/* Erase sector 64K*/
```

3.4 EEPROM Read & Program Functions

The EEPROM read and program functions are implemented and defined in eeprom.c and eeprom.h in the example source code SPI EEPROM.

EEPROM Read Single Byte:

The function `eeprom_read_byte()` reads one byte from the EEPROM from the given address location.

Prototype:

```
uint8_t eeprom_read_byte(uint16_t addr);
```

DataFlash Read Multiple Bytes:

The function `eeprom_program_multiple_read()` reads multiple bytes up to buff_length from given address and stores bytes in the given buffer.

Prototype:

```
void eeprom_program_multiple_read(uint16_t addr, uint8_t *buff, uint8_t buff_length);
```

In this function the sequence is: pull down the #SS wire, send op_code READ and send start address. Then continuously read data until the number of bytes read are equal to the given `buff_length`. Data is stored in the given buffer then the #SS wire is pulled up to stop the read process.

EEPROM Program Single Byte:

The function `eeprom_program_byte` programs a single byte to the given address.

Prototype:

```
void eeprom_program_byte(uint16_t addr, uint8_t byte);
```

EEPROM Program multiple Bytes:
This function writes multiple bytes up to buff_length from the given array. It starts writing from the given start address. In this function address and buff_length validations are done. The start address should be the start of the page address and buff_length should be within the limit. If validation fails, the function returns without writing anything.

If address and buffer length are valid, the total number of pages that needs to be written is calculated and the `eeprom_page_write()` function is called, which writes data to the page of a size up to PAGE_SIZE, 64 bytes.

Prototypes:

```
void eeprom_program_multiple_write(uint16_t addr, uint8_t *buff, uint8_t buff_length);
void eeprom_page_write(uint16_t addr, uint8_t *buff, uint8_t buff_length);
```

EEPROM erase functions:

Prototypes:

```
void eeprom_erase_chip();             /* Erase entire chip*/
void eeprom_erase_page(uint16_t addr);/* Erase page,address should be start of page address*/
```

4. Development Environment

The hardware and software used in the development process of the accompanying driver code is as follows:

Hardware

- ATtiny817 Xplained pro board
- A 8-lead SOIC socket on breadboard
- AT25AA256/AT25LC256 soldered on a SOIC package
- SST25VF010A soldered on a SOIC package
- SST25VF020 soldered on a SOIC package
- SST25VF040B soldered on a SOIC package
- SST25VF080B soldered on a SOIC package

Software

- Atmel Studio 7

5. Revision History

Doc. Rev.	Date	Comments
DS00002665A	3/2018	This document uses Microchip's DS number and replaces Atmel 2595C
2595C	11/2017	Updated using serial memories: AT25xx256/SST25VFxxxx
2595B	07/2016	New template
2595A	03/2005	Initial document release

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO. [X](1) –X /XX

Device — Tape and Reel — Temperature Range — Package

Device:	Device A, Feature A, (Package A) Device B, Feature B, (Package B)	
Tape & Reel Option:	Blank	= Tube
	T	= Tape & Reel
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
Package:	AA	= Package AA
	BB	= Package BB

Examples:

- MCPXXXXXAT-E/AA: Tape and Reel, Extended temperature, XAA package
- MCPXXXXXBT-E/BB: Tape and Reel Extended temperature, XBB package

Note:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check http://www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-2747-6

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-7289-7561 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820