AN89659

Interfacing SPI F-RAM with PSoC® 4

Author: Shivendra Singh

Associated Project: Yes

Associated Part Family: SPI F-RAM – FM25XXX

Software Version: PSoC® Creator™ 4.0 Service Pack 1 and Above

Related Application Notes: AN304, AN79953, AN87352

AN89659 shows how to interface Serial Peripheral Interface (SPI) F-RAM with Cypress's PSoC® 4 (Programmable System-on-Chip) device with the help of example circuits, timing diagrams, and pseudocode. You can also use this application note as a reference design guide to interface SPI F-RAM with other standard SPI master controllers. This application note includes an associated PSoC 4 example project.

Contents

1 Introduction

The SPI F-RAM is a serial nonvolatile memory employing an advanced ferroelectric process. A ferroelectric random access memory (F-RAM) is a nonvolatile RAM that eliminates the complexities, overhead, and system-level reliability problems caused by serial EEPROM and other nonvolatile memories. Unlike serial EEPROM and flash memories, the F-RAM performs write operations at bus speed without incurring any write delays (NoDelay™). Data is directly written into the memory array, and the next bus cycle can begin immediately without the need for data polling. The F-RAM products offer a very high endurance of 10^{14}, orders of magnitude higher than serial EEPROM and flash memories. Also, the F-RAM exhibits lower power consumption than a serial EEPROM or flash memory. For more details on the benefits of serial F-RAM over serial EEPROM, refer to the application note AN87352 – F-RAM for Smart E-Meters.

Cypress's PSoC is a true programmable embedded system-on-chip, integrating configurable analog and programmable digital peripheral functions, memory, and a microcontroller on a single chip. The PSoC 4 is an ARM Cortex-M0 based PSoC family device. You can obtain details on PSoC 4 from AN79953 – Getting Started with PSoC® 4.

This application note elaborates on the SPI F-RAM connections and functionalities within PSoC 4. The hardware recommendations made here are not mandatory; however, their adoption will lead to a more robust overall design.

To describe the SPI F-RAM behavior at the system level, this application note explains a few opcodes with the help of timing diagrams and PSoC 4-based pseudocode.

This application note includes an associated project demonstrating the use of the PSoC 4 SPI F-RAM custom Component in PSoC Creator™. Appendix A describes integrating the SPI F-RAM component into a new project in PSoC Creator. PSoC Creator is a powerful integrated development environment (IDE) from Cypress for PSoC 3, PSoC 4, and PSoC 5LP.

All connection details and recommendations with respect to SPI F-RAM are applicable to other standard SPI master controllers interfacing with a SPI F-RAM memory.

This application note covers the following topics:

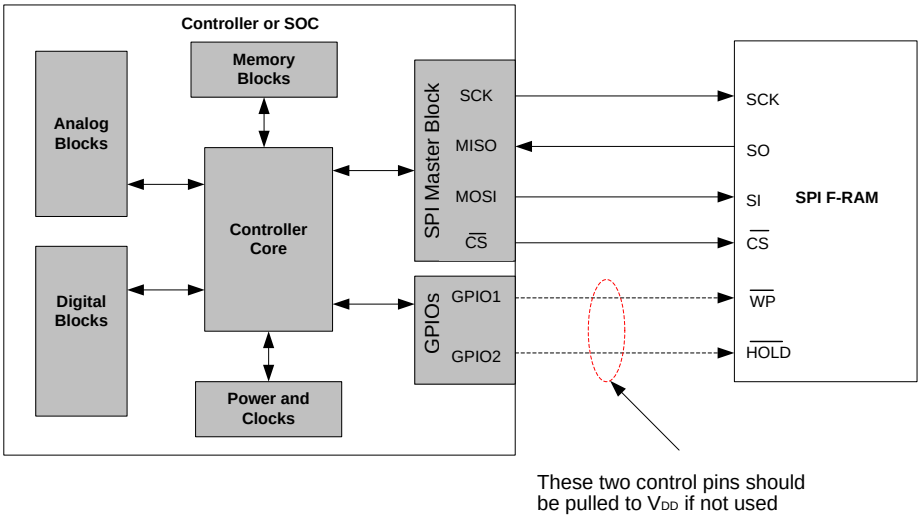
- SPI F-RAM interface
- SPI operating modes
- SPI master implementation with PSoC 4
- SPI F-RAM opcodes
- Addressing in SPI F-RAM
- SPI F-RAM operations example

2 SPI F-RAM Interface

The hardware connection between a SPI host controller (master) and SPI F-RAM (slave) remains identical across all densities. However, the firmware has some differences depending upon the density options and features that SPI devices offer. For example, 1 Mb and higher density SPI F-RAMs require 3-byte addressing, while 512 Kb and lower densities require only 2-byte addressing. In this case, the firmware change between the two density options is to incorporate either 3-byte or 2-byte addressing, while the hardware connections do not change. Similarly, the Serial Number Read (SNR) feature is not available across all SPI F-RAM devices.

Figure 1 illustrates a typical system-level configuration of the SPI F-RAM device.

Figure 1. A Typical SPI F-RAM Interface with a Controller



The acronyms used for the pin names in Figure 1 are as follows:

SCK: SPI Clock, **MISO**: Master In Slave Out, **MOSI**: Master Out Slave In; **CS**: Chip Select, **GPIO**: General Purpose I/O, **SO**: Serial Output, **SI**: Serial Input, **WP**: Write Protect, **HOLD**: Hold

3 SPI Operating Modes

The SPI standard supports four different modes of operation, which is determined by the SPI clock (SCK) polarity (CPOL) and clock phase (CPHA). The SPI clock polarity (CPOL) is the base value of the SPI clock (LOW or HIGH), and the clock phase (CPHA) is the SPI clock edge, either rising or falling. The SPI master sets the SPI mode before starting the SPI communication. Table 1 summarizes all four SPI modes with respect to the SPI clocking, data drive and capture edges on the MOSI and MISO lines.

The SPI F-RAM always uses the rising edge of the input clock (SCK) to latch data bits in the SI line, and the falling edge of the input clock to send data bits out on the SO line. This makes SPI F-RAM compatible with SPI Mode 0 and Mode 3 without any configuration or setting in the SPI F-RAM.

Table 1. SPI Operating Modes

SPI Data Driving and Capture Edges	Mode 0 (CPOL=0; CPHA=0)	Mode 1 (CPOL=0; CPHA=1)	Mode 2 (CPOL=1; CPHA=0)	Mode 3 (CPOL=1; CPHA=1)
SPI Clock (SCK) Start Logic Level	LOW	LOW	HIGH	HIGH
Data Latched In by F-RAM on MOSI	SCK Rising Edge (↑)	SCK Falling Edge (↓)	SCK Falling Edge (↓)	SCK Rising Edge (↑)
Data Driven Out by F-RAM on MISO	SCK Falling Edge (↓)	SCK Rising Edge (↑)	SCK Rising Edge (↑)	SCK Falling Edge (↓)
SPI F-RAM Support	Yes	No	No	Yes

4 SPI Master Implementation with PSoC 4

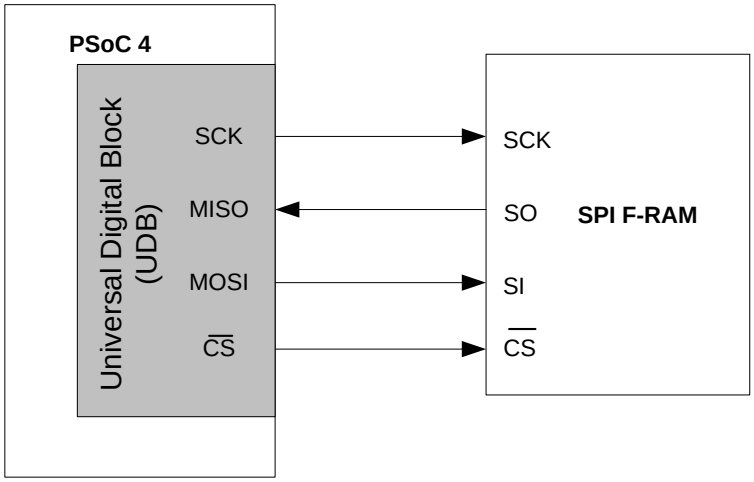
The programmable and reconfigurable digital blocks of PSoC 4 with flexible automatic routing allow you to select any four general-purpose I/Os (GPIOs) to be used as a SPI interface. PSoC 4 can implement the SPI master by using one of the following methods described in sections 4.1 to 4.3.

4.1 SPI Master Implementation Using UDB

The universal digital blocks (UDBs) of PSoC 4 can be configured as the SPI master. Since UDBs are accessible by all GPIOs, they allow the use of any four GPIOs of the PSoC 4 as the SPI master control pins. Refer to the PSoC 4 Architecture TRM and device datasheet for more details on I/O pin configuration and UDB details. Figure 2 illustrates the SPI F-RAM interface with a PSoC 4 SPI master implemented using UDBs.

The attached example project uses UDBs to implement the SPI master in PSoC 4.

Figure 2. SPI Master Implementation in PSoC 4 UDB



4.2 SPI Master Implementation Using SCB

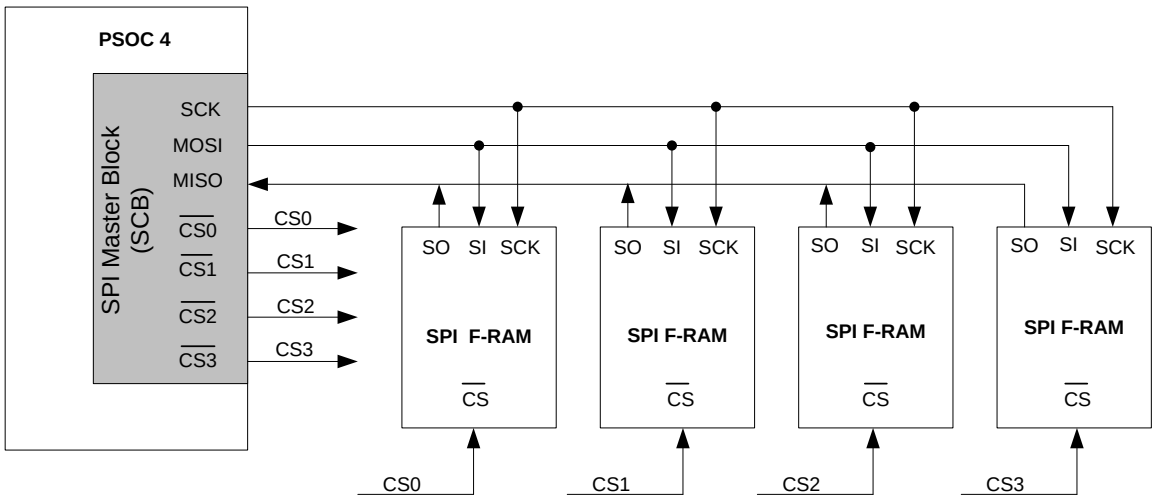
PSoC 4 devices can have up to four dedicated serial communication blocks (SCBs) that can be configured as SPI masters. Each SCB assigns a set of fixed GPIOs for specific control pins; only those pins can be used to set up a serial communication link between PSoC 4 and a peripheral device. Table 2 shows an example of PSoC 4200 family part no CY8C4245AXI showing dedicated PSoC 4 SPI control pins available for each of two SCBs. For specific PSoC 4 pin selections, refer to target PSoC 4 Datasheet or PSoC Creator software.

Table 2. SPI Pin Configuration in PSoC (CY8C4245) SCBs

PSoC 4 Pins		SCB Block	Pin Functions	Pin Description
Name	Type			
P4.0	GPIO	SCB0	MOSI	Master output slave input of SCB0
P4.1	GPIO	SCB0	MISO	Master input slave output of SCB0
P4.2	GPIO	SCB0	SCK	SPI clock of SCB0
P4.3	GPIO	SCB0	CS0	SPI slave select 0 of SCB0
P0.0	GPIO	SCB0	CS1	SPI slave select 1 of SCB0
P0.1	GPIO	SCB0	CS2	SPI slave select 2 of SCB0
P0.2	GPIO	SCB0	CS3	SPI slave select 3 of SCB0
P0.4 / P3.0	GPIO	SCB1	MOSI	Master output slave input of SCB1
P0.5 / P3.1	GPIO	SCB1	MISO	Master input slave output of SCB1
P0.6 / P3.2	GPIO	SCB1	SCK	SPI clock of SCB1
P0.7 / P3.3	GPIO	SCB1	CS0	SPI slave select 0 of SCB1
P3.4	GPIO	SCB1	CS1	SPI slave select 1 of SCB1
P3.5	GPIO	SCB1	CS2	SPI slave select 2 of SCB1
P3.6	GPIO	SCB1	CS3	SPI slave select 3 of SCB1

Each SCB can implement one SPI master block. The SPI master can communicate with up to four slave SPI devices connected to the same SPI bus. The slave select ($\overline{CS}$) pin selects the individual SPI slave connected to the SPI bus. Figure 3 illustrates the SPI F-RAM interface with the PSoC 4 SPI configured using an SCB. The figure also shows how to interface more than one SPI slave on the same SPI bus.

Figure 3. SPI Master Implementation in PSoC 4 Using an SCB



The PSoC 4 Example Project section (Appendix A) discusses the SPI NVRAM custom Component using PSoC 4 SCBs. The SPI NVRAM custom Component essentially uses standard PSoC 4 SCB as discussed above and creates a wrapper around it to make it specific to SPI F-RAM. The SPI NVRAM custom Component, once configured as SPI F-RAM, provides high-level APIs that simplify the SPI F-RAM coding and integration with PSoC 4. Refer to Section PSoC 4 Example Project for more details.

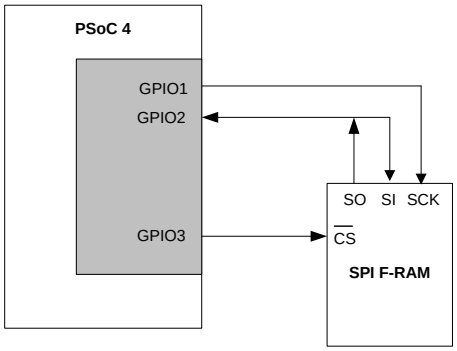
4.3 SPI Master Configuration Using Bit Banging

The third and the least preferred method is to use I/O bit-banging to implement the SPI master controller in PSoC 4. In a bit-banging implementation, the SPI protocol is generated by driving I/O pins through the firmware. The firmware implementation adds a significant amount of code overhead in comparison with the hardware implementations that use either UDBs or SCBs. The bit-banging method of the SPI interface also reduces the total data throughput and consumes more CPU resources, which makes the CPU less available for other functions.

The only advantage of the bit-banging method is that it can implement the SPI interface using only three GPIOs instead of four GPIOs pins. This is also called the half-duplex form of a SPI interface. The half- duplex form of the SPI interface requires the PSoC 4 controller to execute either a write or a read operation at any given time; therefore, a single GPIO can be used for either sending or receiving data, unlike in case of a standard SPI interface where data can be transmitted and received simultaneously.

Figure 4 illustrates the SPI F-RAM interface with PSoC 4 using its GPIOs. This application note does not cover the implementation of a SPI master controller in PSoC 4 using a bit-banging method.

Figure 4. Half-Duplex SPI Master Implementation Using PSoC 4 Bit Banging

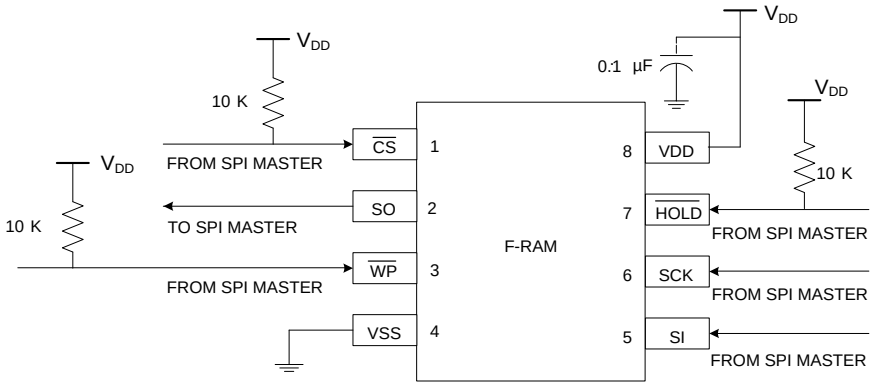


4.4 SPI F-RAM Input Pin Configuration

The SPI F-RAM has control input pins that should be properly biased to a fixed logic state (HIGH or LOW) for proper device operation. If an input pin is not properly biased and is left floating, then it can either assume an intermediate logic state that can cause high device current or it can float to a logic level LOW or HIGH, which can trigger an undesirable action. The direction in which a floating input signal assumes its logic state to be in depends upon a number of factors, such as noise in the system, capacitive coupling, and leakage.

Therefore, the logic level seen by a floating input's circuitry is relatively random and can change during operation. Such unpredictable input levels can severely impact device operation. Therefore, any unused input pin should always be tied to a proper logic level, such as HIGH for an active LOW input. A 10-kΩ resistor can be used to pull up or pull down an unused input pin.

Figure 5. SPI F-RAM Interface with a Standard SPI Master



$\overline{\text{HOLD}}$ pin: The $\overline{\text{HOLD}}$ pin is an active LOW input and allows you to suspend the clock midstream to pause an ongoing communication. If this pin is LOW, the device no longer reacts to any clock pulse received, communication is disrupted, and data is potentially lost or corrupted. If not used, this pin should be connected to a 10-kΩ pull-up resistor to avoid any undesired event.

$\overline{\text{WP}}$ pin: The Write Protect ($\overline{\text{WP}}$) pin is an active LOW input and is used to prevent writing into the Status Register by pulling this pin LOW externally. The WPEN bit in the Status Register determines the functionality of the $\overline{\text{WP}}$ pin.

If the WPEN bit is set to '1', it enables $\overline{\text{WP}}$ pin control; if it is set to '0', then the $\overline{\text{WP}}$ pin is disabled. This pin should be connected to a 10-kΩ pull-up resistor to avoid any undesired event.

$\overline{\text{CS}}$ pin: The SPI master drives the chip select ($\overline{\text{CS}}$) pin during normal operation. The external pull-up on $\overline{\text{CS}}$ is optional. However, a 10-kΩ pull-up resistor can be used to keep the $\overline{\text{CS}}$ pin HIGH when the SPI master is not driving this line.

Refer to the device datasheet for details on input control pin functionalities and their behavior during power-up, power-down, and normal operation.

4.5 SPI F-RAM Operating Voltage

The SPI F-RAM devices offer a wide range of operating voltage (V_{DD}) from 2.0 V to 5.5 V. However, this wide operating voltage range is not applicable to all F-RAM devices. The most typical F-RAM operating voltages are 2.0 V to 3.6 V, 2.7 V to 3.6 V, and 4.5 V to 5.5 V. Refer to the device datasheet for allowed voltage levels on its V_{DD} and input pins to ensure that applied voltages do not exceed datasheet limits.

5 SPI F-RAM Opcodes

All SPI opcodes, addresses, and data are 8-bit data transfers. All transactions occur with $\overline{\text{CS}}$ LOW. The opcode, address, and data-in are clocked in on the SI pin, and data-out is clocked out on the SO pin. Opcodes provide control over the device. The SPI F-RAM supports industry-standard opcodes for all read and write operations. A unique opcode is assigned for each specific operation in the SPI F-RAM. Table 3 provides the opcode list. Table 4 explains each opcode and the associated data bytes required for proper operation.

Table 3. SPI F-RAM Opcodes

Instruction Category	Instruction Name	Description	Opcode	
			Hexadecimal	Binary
Status Register Control Instructions	WREN	Set the Write Enable Latch (WEL) bit in the Status Register.	06H	0000 0110b
	WRDI	Clear the WEL bit in the Status Register; this disables all write operations.	04H	0000 0100b
	RDSR	Read Status Register	05H	0000 0101b
	WRSR	Write Status Register	01H	0000 0001b
F-RAM Read and Write Instructions	READ (16Kb and above)	Read Memory Data	03H	0000 0011b
	READ	Read Memory Data	0BH	0000 0011b
	READ (4Kb parts) (1)	Read Memory Data (from lower 256 Bytes)	03H	0000 0011b
		Read Memory Data (from upper 256 Bytes)	0BH	0000 1011b
	FSTRD (2)	Fast Read Memory Data	0BH	0000 1011b
	WRITE	Write Memory Data	02H	0000 0010b
	WRITE	Write Memory Data	02H	0000 0010b
	WRITE (4Kb parts) (1)	Write Memory Data (to lower 256 Bytes)	02H	0000 0010b
		Write Memory Data (to upper 256 Bytes)	0AH	0000 1010b
Sleep Instruction	SLEEP	Enter Sleep Mode	B9H	1011 1001b
Device ID and Serial Number Instructions	RDID	Read Device ID	9FH	1001 1111b
	SNR	Read Serial Number	C3H	1100 0011b

Note: 1 The 4-Kb SPI F-RAM uses single-byte addressing for writes and reads. The MSB of the address bit is sent through the opcode byte in the fourth bit from LSB (bit 3). When bit 3 is set to '0', it addresses the lower 256 bytes of the 512-byte memory. When bit 3 set to '1', it addresses the upper 256 bytes. Therefore, read and write opcodes for the 4-Kb F-RAM change accordingly.

Note: 2 The SPI FSTRD is similar to the SPI READ command except that the FSTRD command requires an additional dummy address cycle. The FSTRD command enables the SPI F-RAM as a drop-in replacement for the SPI flash memories that support the FSTRD command. In SPI F-RAM, the FSTRD and READ commands work at the same frequency. Refer to the SPI F-RAM datasheet for more details on the FSTRD command.

Table 4. SPI F-RAM Data Flow

Instruction Name	Opcode	Master Transmits on SI	F-RAM Transmits on SO	Comments
WREN	06H	06H	–	This command sets the WEL bit in the Status Register. The WEL bit is set on the rising edge of the chip select ($\overline{CS}$) after this instruction.
WRDI	04H	04H	–	The WRDI command clears the WEL bit (if set) in the Status Register.
RDSR	05H	05H	StatusReg_Data	The RDSR command reads Status Register content.
WRSR	01H	01H , StatusReg_Data	–	The WEL bit must be set prior to write into the Status Register. WEL is cleared when $\overline{CS}$ goes HIGH.
READ	03H	03H , Add1, Add2, Add3	Data1, Data2, Data3,.,., DataN	Data length can be set from 1 to N for read operation, where N can be any integer value. Once the READ command starts executing, the F-RAM internal address automatically increments by 1, and the device is ready to transmit the next

Instruction Name	Opcode	Master Transmits on SI	F-RAM Transmits on SO	Comments
FSTRD	0BH	0BH , Add1, Add2, Add3, Dummy_Byte	Data1, Data2, Data3,.,., DataN	data byte from the incremented memory address. When the internal counter reaches its maximum readable address, it rolls over to the start address 00H and continues reading data from there onwards as long as the device is in Read mode and the serial (SPI) clock is available. Read exits when $\overline{CS}$ toggles HIGH. The FSTRD command requires an additional dummy address cycle. This starts driving SO with valid data after the dummy cycle.
WRITE	02H	02H , Add1, Add2, Add3, Data1, Data2, Data3.,.,., DataN	–	The WEL bit must be set prior to sending the WRITE opcode. Data length can be set from 1 to N for the write operation, where N can be any integer value. Once the WRITE command starts executing, the F-RAM internal address automatically increments by 1, and the device is ready to receive the next byte to write in the incremented memory location. When the internal counter reaches its maximum writable address, it rolls over to the start address 00H and continues writing data from there onwards by overwriting the previously written data. The data write continues as long as the device is in the Write mode and the serial (SPI) clock is available. The controller firmware should take care of data overwriting due to the memory counter rolling over during a write operation. The write exits and WEL is cleared when $\overline{CS}$ toggles HIGH.
SLEEP	B9H	B9H	–	When $\overline{CS}$ goes HIGH, the device enters Sleep mode, and the device consumes Sleep mode current (IZZ). The WEL bit must be set prior to initiating a SLEEP command. WEL is cleared when $\overline{CS}$ toggles HIGH.
RDID	9FH	9FH	Data1, Data2, Data3,., Data9	The RDID command reads the device ID (9 bytes).
SNR	C3H	C3H	Data1, Data2, Data3,., Data8	The SNR command reads the serial number (8 bytes).

Note: The 1-Mb and higher density SPI F-RAMs use 3-byte addressing; the lower density F-RAMs (512-Kb or less, down to 16 Kb) use 2-byte addressing. The 4-Kb F-RAM uses only 1-byte addressing.

6 Addressing in SPI F-RAM

The SPI host controller communicates with the SPI F-RAM on a byte-by-byte basis and always transmits the most-significant bit (MSb) in the first clock cycle and the least-significant bit (LSb) in the eighth clock cycle during a byte transmission. This is valid for all SPI communication including command, address, and data bytes.

Similarly, when an SPI F-RAM transmits a data byte during a read operation, it always transmits the most significant bit first and the least significant bit last during data byte transmission.

Figure 6 shows an example of address bits being transmitted over the SPI MOSI line while transmitting three address bytes by the SPI master.

The unused bits of the most-significant byte (MSB) are “don’t care” bits, which F-RAM ignores. However, it is a good practice to set unused address bits to ‘0’ in the firmware. This approach makes it easy to upgrade the firmware when moving to a higher density device in the same socket.

Figure 7 represents the addressing scheme for different F-RAM densities. A0 is the LSb in an address.

Figure 6. Address Bit Transmission in SPI F-RAM

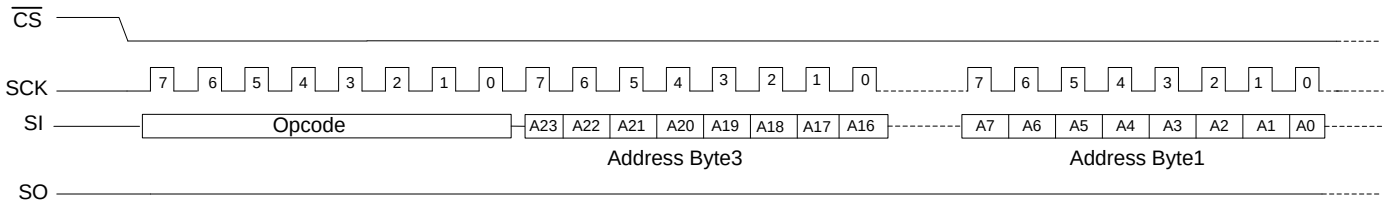


Figure 7. SPI F-RAM Opcode and Addressing

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4-Kbit [Note 3]	op	op	op	op	A	op	op	op	Not Applicable (1 Byte Addressing Only)																A7	A6	A5	A4	A3	A2	A1	A0
16-Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)								0	0	0	0	0	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
64-Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)								0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
128-Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)								0	0	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
256-Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)								0	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
512-Kbit	op	op	op	op	op	op	op	op	Not Applicable (2 Byte Addressing Only)								A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1-Mbit	op	op	op	op	op	op	op	op	0	0	0	0	0	0	0	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
2-Mbit	op	op	op	op	op	op	op	op	0	0	0	0	0	0	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
4-Mbit	op	op	op	op	op	op	op	op	0	0	0	0	0	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Note: The 4-Kb SPI F-RAM uses only 1-byte addressing to address its 512-byte array. Therefore, the most significant address bit A8 (the ninth bit) is sent in READ and WRITE opcodes. Bit 3 of the READ, WRITE opcode is used to transmit the MSb of the 4-Kb memory as discussed in Table 3.

6.1 Memory Density Upgrade

The SPI standard supports single-master multi-slave topology. This allows connecting more than one SPI slave device to the same SPI bus. To communicate with a SPI slave, the SPI master first selects the SPI slave by pulling its slave select ($\overline{CS}$) pin to LOW, followed by sending SPI commands. To support more than one SPI slave, the master should have one slave select control pin dedicated to each SPI slave. Figure 3 illustrates multiple SPI slaves connected to the same SPI master. This topology is used to control more than one SPI slave, for example, to expand system memory density by using multiple SPI memory devices.

7 SPI F-RAM Operations Example

This section describes the F-RAM operations with the help of timing diagrams and PSoC 4 specific pseudocode. All functions starting with the prefix NVRAM_SPI_1 are PSoC 4 specific functions. NVRAM_SPI_1 is the name of an instantiated SPI F-RAM component in the attached PSoC Creator example project.

This section covers only a few opcodes as an example to show the SPI F-RAM data flow during SPI communication between a SPI master and the SPI F-RAM. Refer to the SPI F-RAM datasheet for detailed descriptions of opcodes.

7.1 Status Register Operation

This section describes the F-RAM Status Register write and read operations with the help of a timing diagram and PSoC 4 code as an example.

7.1.1 Write Status Register

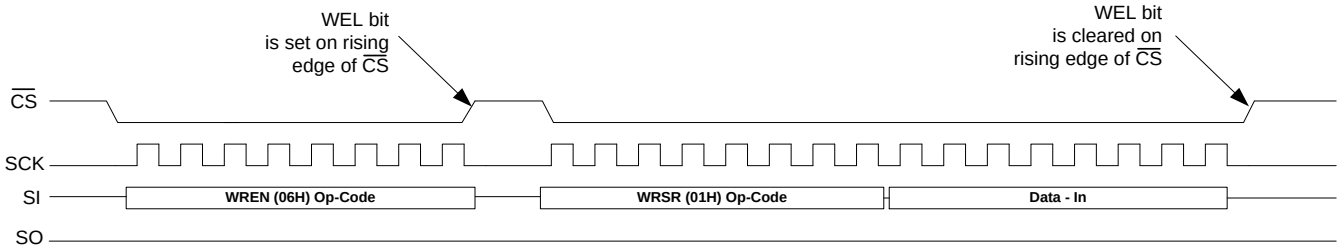
The WRSR command is used to set the user-writable bits in the Status Register. The other non-writable bits are either set by the device internally or they are reserved and return fixed values (‘0’ or ‘1’). Refer to the device datasheet for the Status Register details.

The following sequence is required to initiate a write in the F-RAM Status Register:

- Send the WREN opcode to set the Write Enable Latch (WEL) bit.

- Send the write status register opcode (WRSR) followed by a data byte to be written into the Status Register. Note that read-only bits in the Status Register are unaffected by WRSR operation. See the device datasheet for Status Register details.
- Figure 8 shows a timing diagram for writing into the Status Register.

Figure 8. Write into the Status Register (WRSR Opcode)




```
/*******************PSoC 4 pseudocode for Status Register write***************/

void NVRAM_SPI_1_Status_Reg_Write ( uint8 data_byte )

{

   NVRAM_SPI_1_CS_Reg_Write(0); // Enable the SPI slave by toggling chip select LOW

   NVRAM_SPI_1_SPIM_ClearTxBuffer(); //Clear SPI transmit buffer before sending command

   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_WREN); // Set the write enable (WEL) bit

                                             //prior to write


//Wait until SPI_DONE flag is cleared

   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)

                                        != NVRAM_SPI_1_SPIM_STS_SPI_DONE);

   NVRAM_SPI_1_CS_Reg_Write(1); //WEL is set high when CS is switched high

   NVRAM_SPI_1_CS_Reg_Write(0); //Re-enable the SPI slave

   NVRAM_SPI_1_SPIM_ClearTxBuffer();

   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_WRSR_CMD); //Send Write Status Register instruction

   NVRAM_SPI_1_SPIM_WriteTxData(data_byte); //Send data

//Wait until SPI_DONE flag is cleared

   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)

                                        != NVRAM_SPI_1_SPIM_STS_SPI_DONE);

      NVRAM_SPI_1_CS_Reg_Write(1); //Terminate the write operation by toggling

                                   //chip select HIGH

}
```

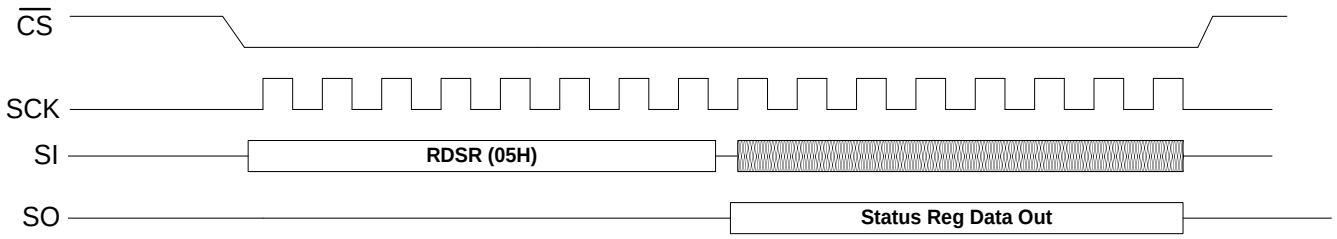
7.1.2 Read Status Register

The RDSR command reads the current value of the Status Register. For example, executing the RDSR command following the WREN command returns the Status Register value with the Write Enable Latch (WEL) bit set to '1'.

The following sequence is required to initiate a read from the F-RAM Status Register:

- Send the RDSR opcode.
- Read the Status Register value on the SO line.
- The host can terminate the Status Register command by toggling the chip select HIGH. Figure 9 shows a timing diagram for reading the SPI F-RAM Status Register.

Figure 9. Read from the Status Register (RDSR Opcode)




```
/************************** PSoC 4 pseudocode for Status Register read*****************/

uint8 NVRAM_SPI_1_Status_Reg_Read ( void )
{
   uint8 data_byte;

   NVRAM_SPI_1_CS_Reg_Write(0); //Enable the SPI slave by toggling chip select LOW
   NVRAM_SPI_1_SPIM_ClearTxBuffer(); //Clear SPI transmit buffer before sending command
   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_RDSR_CMD); //Send read status register command

//Wait until SPI_DONE flag is cleared
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                        != NVRAM_SPI_1_SPIM_STS_SPI_DONE);
   NVRAM_SPI_1_SPIM_ClearRxBuffer();
   NVRAM_SPI_1_SPIM_WriteTxData(0x00); //Dummy write for reading the
                                        //status register data byte

//Wait until SPI_DONE flag is cleared
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
```

```
                                        != NVRAM_SPI_1_SPIM_STS_SPI_DONE);


//Wait until there is data in the read buffer
   while(!NVRAM_SPI_1_SPIM_GetRxBufferSize());


   data_byte = NVRAM_SPI_1_SPIM_ReadRxData();
   NVRAM_SPI_1_CS_Reg_Write(1); //Terminate the read operation by toggling
                                //chip select HIGH
   return(data_byte);
}
```

7.2 F-RAM Write and Read Operations

This section describes the F-RAM write and read operations with the help of a timing diagram and PSoC 4 code as an example.

7.2.1 F-RAM Write Operation

The following sequence is required to initiate a write to the F-RAM:

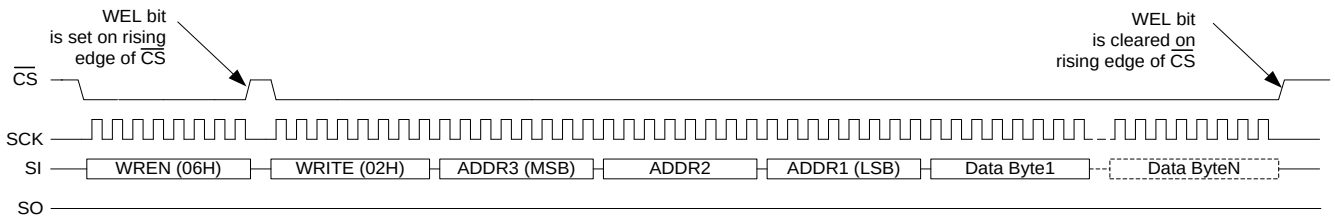
- Send the WREN opcode to set the Write Enable Latch (WEL) bit.
- Send the WRITE opcode.
- Send the most significant address byte.
- Send the intermediate address byte (in 3-byte addressing).
- Send the least significant address byte.
- Send data byte/bytes.

Any write command to the F-RAM should be preceded by a Write Enable (WREN) instruction. If the device is not write enabled (WEL = '0'), it ignores the write instructions. A new $\overline{CS}$ falling edge is required to reinitiate the SPI communication.

After the completion of a write instruction (WRSR or WRITE), the WEL bit of the Status Register is cleared to '0' on the rising edge of the chip select ($\overline{CS}$). This ensures that the SPI F-RAM has come out of the Write mode and thus prevents any inadvertent writes.

Note that reading the Status Register (RDSR opcode) does not clear the WEL bit. Some users read the Status Register immediately after executing the WREN to confirm that the WEL bit is set or not set prior to initiating a write operation. Figure 10 shows a timing diagram for writing into the F-RAM.

Figure 10. Write into the F-RAM (WRITE Opcode)



```
/* PSoC 4 pseudocode for 1 Mb (128kx8) F-RAM write in burst mode. By passing in
total_data_count =1, the user can write 1 byte at a given address location*/

void NVRAM_SPI_1_Write ( uint32 addr, uint8 *data_write_ptr, uint32 total_data_count )
{
   uint32 i;

   NVRAM_SPI_1_CS_Reg_Write(0);      //Enable the SPI slave by toggling chip select LOW
   NVRAM_SPI_1_SPIM_ClearTxBuffer(); //Clear SPI transmit buffer
   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_WREN);  //Set the write enable (WEN)
                                               //bit prior to write
//Wait till SPI_DONE flag is cleared
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                          != NVRAM_SPI_1_SPIM_STS_SPI_DONE);
   NVRAM_SPI_1_CS_Reg_Write(1);      //WEL is set here
   NVRAM_SPI_1_CS_Reg_Write(0);
   NVRAM_SPI_1_SPIM_ClearTxBuffer();
   NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_WRITE_CMD); //Send memory write command
   if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
    {
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16)); //Transmits most significant address
                                                      //byte (1 Mb and above)
    }
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8)); //Transmits intermediate address byte

     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr));   //Transmits least significant address

//Wait till SPI_DONE flag is cleared
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                           !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);

   for(i = 0; i < total_data_count; i++ )
    {
```

```
      NVRAM_SPI_1_SPIM_WriteTxData ((uint8) (data_write_ptr[i]));


//Wait until SPI_DONE flag is cleared
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                        != NVRAM_SPI_1_SPIM_STS_SPI_DONE);
    }
      NVRAM_SPI_1_CS_Reg_Write(1); //Terminate the write operation by toggling
                                   //chip select HIGH
}
```

7.2.2 F-RAM Read Operation

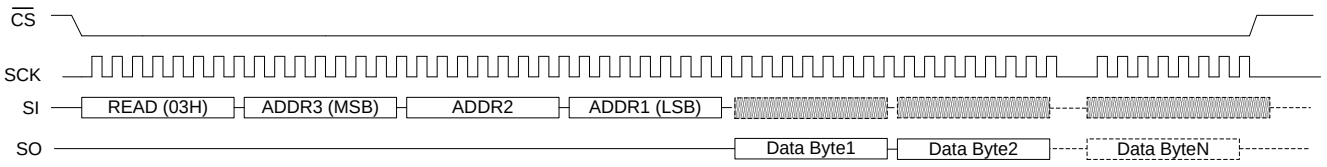
The following sequence is required to initiate a read from the F-RAM:

- Send the READ opcode.
- Send the most significant address byte first and the least significant address byte last during the address cycle.
- Once the SPI F-RAM device receives the READ command, it starts sending data bytes out on the SO line. The host controller can initiate either a single-byte read or a burst read (more than one byte).

The burst read allows reading successive memory location by initiating a single read command. In burst read, the F-RAM device increments the internal address counter automatically and continues sending data bytes out on the SO line. This continues as long as the chip select $\overline{CS}$ remains asserted LOW and the SPI clock is present. The burst read continues to cycle through the memory in a circular fashion.

When the chip select $\overline{CS}$ is de-asserted, the data output stops and SO goes to a high impedance (HI-Z) state. Figure 11 shows a timing diagram for reading from the F-RAM.

Figure 11. Read from the F-RAM (READ Opcode)




```
/* PSoC 4 pseudocode for 1 Mb (128kx8) F-RAM Read in burst mode. By passing in
total_data_count =1, user can read only 1 byte from a given address location*/

void NVRAM_SPI_1_Read ( uint32 addr, uint8 *data_read_ptr, uint32 total_data_count )

{

   uint32 i;


    NVRAM_SPI_1_CS_Reg_Write(0); // Enable the SPI slave by toggling chip select LOW

    NVRAM_SPI_1_SPIM_ClearTxBuffer();// Clear SPI transmit buffer

    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_READ_CMD); // Send memory read command
```

```
   if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
    {
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16)); //Transmits most significant address
                                                      //byte (1 Mb and above)
    }
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8)); //Transmits intermediate address byte
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr));    //Transmits least significant address

//Wait until SPI_DONE flag is cleared
  while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                        !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);
  for(i = 0; i < total_data_count; i++ )
   {
     NVRAM_SPI_1_SPIM_ClearRxBuffer();
     NVRAM_SPI_1_SPIM_WriteTxData((uint8) 0x00);//Dummy write to generate SCK for read

//Wait until SPI_DONE flag is cleared
  while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                        !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);

//Wait until there is data in the read buffer
   while(!NVRAM_SPI_1_SPIM_GetRxBufferSize());
   data_read_ptr[i] = NVRAM_SPI_1_SPIM_ReadRxData();

}

NVRAM_SPI_1_CS_Reg_Write(1); //Terminate the read operation by toggling chip select HIGH

}
```

7.2.3 F-RAM Fast Read Operation

The following sequence is required to initiate a fast read from the F-RAM:

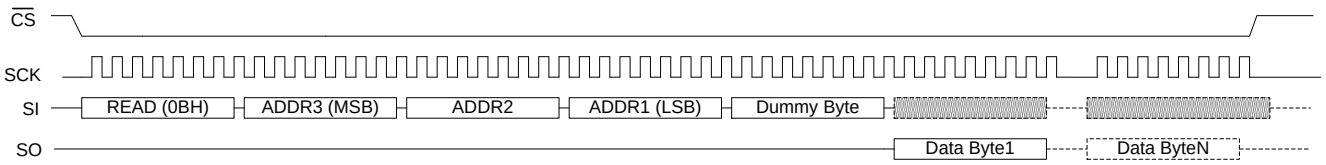
- Send the FSTRD opcode.
- Send the most significant address byte first and the least significant address byte last during the address cycle.
- Send a dummy address byte.
- Once the SPI F-RAM device receives the FSTRD command, it starts sending data bytes out on the SO line. The host controller can initiate either a single-byte read or a burst read (more than one byte).

The burst read allows reading successive memory locations by initiating a single read command. In burst read, the F-RAM device increments the internal address counter automatically and continues sending data bytes out on the SO line.

This continues as long as the chip select $\overline{CS}$ remains asserted LOW and the SPI clock is present. The burst read continues to cycle through the memory in a circular fashion.

When the chip select $\overline{CS}$ is de-asserted, the data output stops and SO goes to a high impedance (HI-Z) state. Figure 12 shows a timing diagram for reading from the F-RAM using the fast read (FSTRD) command.

Figure 12. Fast Read from the F-RAM (FSTRD Opcode)



```
/* PSoC 4 pseudocode for 1 Mb (128kx8) F-RAM fast read in burst mode. By passing in
total_data_count =1, the user can read only 1 byte from a given address location*/

void NVRAM_SPI_1_FastRead ( uint32 addr, uint8 *data_read_ptr, uint32 total_data_count )
{
   uint32 i;

    NVRAM_SPI_1_CS_Reg_Write(0); //Enable the SPI slave by toggling chip select LOW
    NVRAM_SPI_1_SPIM_ClearTxBuffer();//Clear SPI transmit buffer
    NVRAM_SPI_1_SPIM_WriteTxData(NVRAM_SRAM_READ_CMD); //Send memory read command

   if(NVRAM_SPI_1_spi_density >= SPI_1MBit)
    {
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>16)); //Transmits most significant address
                                                       //byte (1 Mb and above)
    }
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr>>8)); //Transmits intermediate address byte
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr));    //Transmits least significant address
     NVRAM_SPI_1_SPIM_WriteTxData((uint8)(addr));    //Transmits dummy address byte for
                                                     //fast read operation

//Wait until SPI_DONE flag is cleared
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                        !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);
   for(i = 0; i < total_data_count; i++ )
    {
     NVRAM_SPI_1_SPIM_ClearRxBuffer();
     NVRAM_SPI_1_SPIM_WriteTxData((uint8) 0x00); //dummy write

//Wait until SPI_DONE flag is cleared
   while((NVRAM_SPI_1_SPIM_ReadTxStatus() & NVRAM_SPI_1_SPIM_STS_SPI_DONE)
                                         !=NVRAM_SPI_1_SPIM_STS_SPI_DONE);
```

```
//Wait until there is data in the read buffer
   while(!NVRAM_SPI_1_SPIM_GetRxBufferSize());
    data_read_ptr[i] = NVRAM_SPI_1_SPIM_ReadRxData();


     }

NVRAM_SPI_1_CS_Reg_Write(1); //Terminate the read operation by toggling chip select HIGH

}
```

8 Summary

Cypress's SPI F-RAM supports industry-standard SPI access protocols similar to other nonvolatile SPI memory products, such as SPI EEPROM, flash, and MRAM. This makes F-RAM compatible with all standard SPI master controllers. The SPI F-RAM opcodes are matched with the standard SPI memory products, which makes SPI F-RAM an easy drop-in replacement. This application note demonstrated how to interface SPI F-RAM with Cypress's PSoC 4 controller using schematics, timing diagrams, and example code.

Appendix A. PSoC 4 Example Project

The project accompanying this application note is an example project that demonstrates the SPI F-RAM interface with a PSoC 4 device. You can download the project file from the application note link with the file name *AN89659.zip*. The programmability and flexibility of PSoC 4 enables you to add more functionality and modify the project according to a target application. To execute the example project on a hardware setup, you should have a PSoC 4 connected to the SPI F-RAM.

A.1 Example Project Pin Assignments

Table 5 shows the connection details in the attached example project. The following software and hardware components are used for the example:

- PSoC Creator 4.0 Service Pack 1.
- Hardware kit – CY8CKIT-042.
- PSoC 4 part number CY8C4245AXI-483 is selected to build the project.
- V_{DD} supply to PSoC 4 controller via USB.

Table 5. PSoC 4 Port Configuration in Example Project

F-RAM Signal Name	PSoC (Master) Signal Name	PSoC 4 I/O Assignment	Signal Direction
$\overline{CS}$	CS	P2[0]	PSoC 4 Output
SI	MOSI	P2[3]	PSoC 4 Output
SO	MISO	P1[4]	PSoC 4 Input
SCK	SCK	P2[2]	PSoC 4 Output
$\overline{HOLD}$	HOLD	P2[1]	PSoC 4 Output
$\overline{WP}$	WP	P2[5]	PSoC 4 Output

The example project executes the following:

1. Write 4 bytes of data into memory and read back 4 bytes of written data. Data is stored in an internal register.
2. Write 1 byte of data into the Status Register and read it back. The read data byte is stored in an internal register.

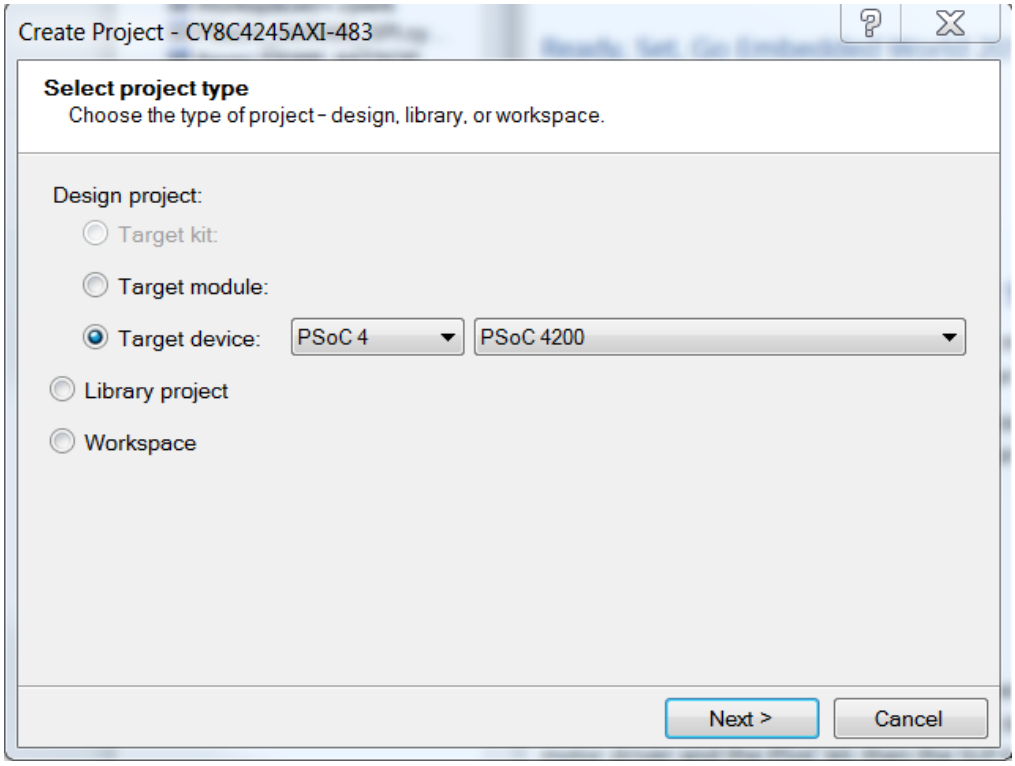
Note: Every time reading a byte from the SPI F-RAM over the SPI bus, a dummy byte write is required. PSoC 4 SPI master generates the SPI clock (during writes and reads) only when a write on the MOSI is executed, even though the write data byte is ignored on the SI line while reading from the SPI slave. You can refer to READ opcode example code below Figure 12.

A.2 Integrating the SPI F-RAM Component into a Project

This section demonstrates integrating the F-RAM component into a new PSoC 4 project. The archived *AN89659.zip* file creates a folder name, “PSoC4_NVRAM_SPI,” when unarchived. This folder contains both the example project and the NVRAM_SPI component. The following steps explain how to integrate and use the NVRAM_SPI component in a new PSoC 4 design.

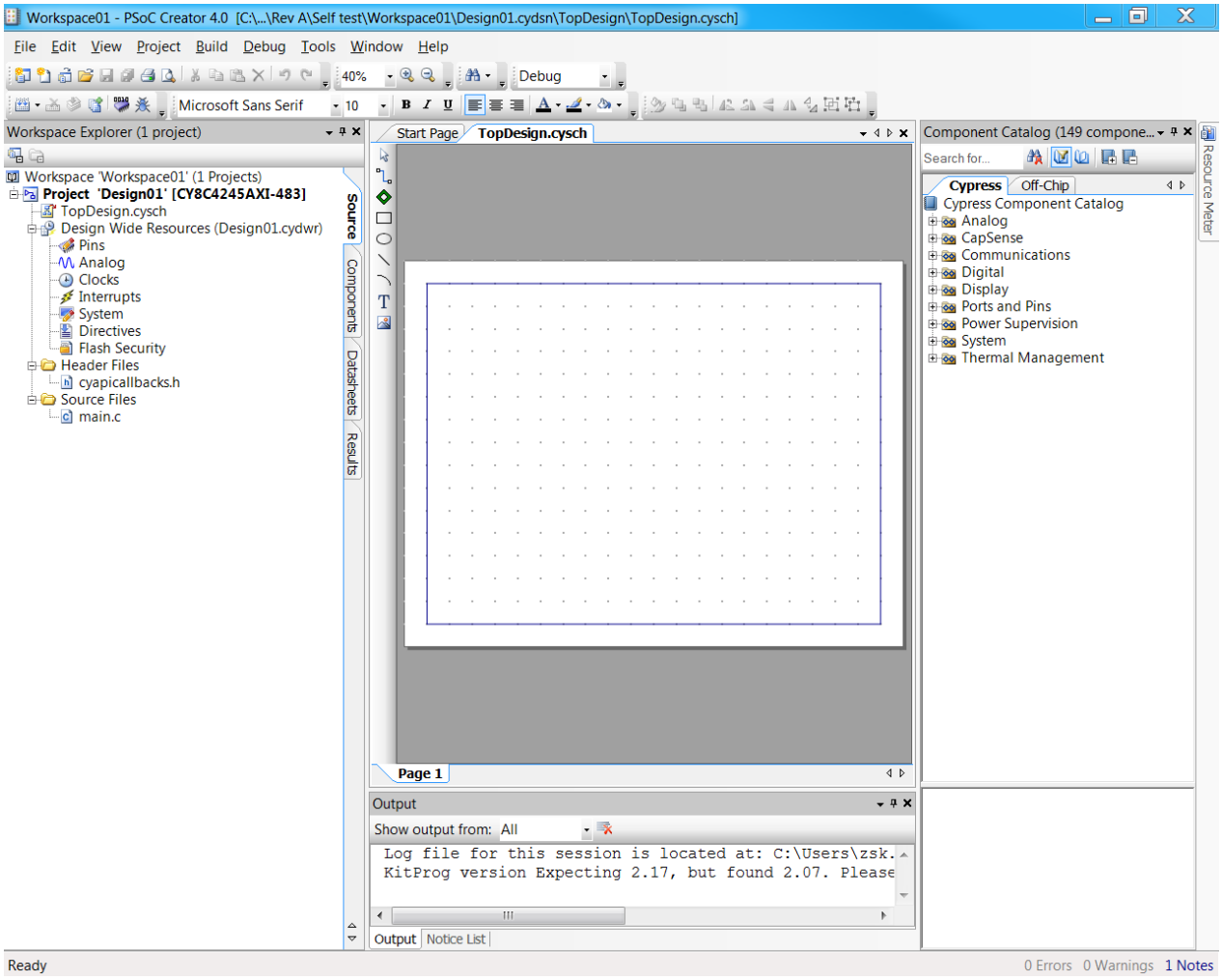
1. Open PSoC Creator and choose **File** > **New** > **Project** to create a new project. Select the Target device as “PSoC 4” from the drop-down menu and then select the target PSoC 4 device series (“PSoC 4200” in this example) as shown in Figure 13. Chose **Next** and select “Empty schematic”.

Figure 13. Select Project Type and Target Device



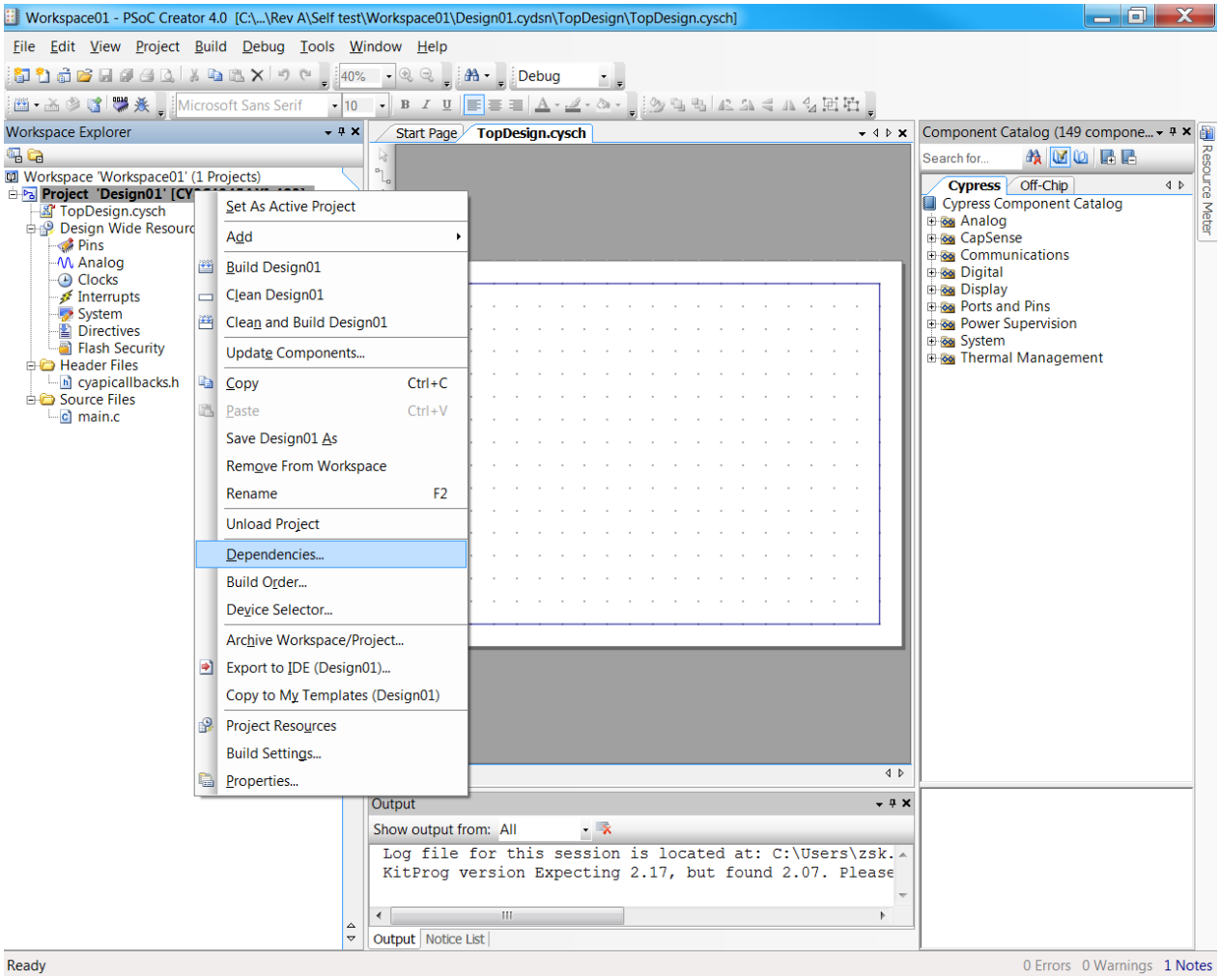
2. Enter a name for the project in the Name field. A new project, “Design01” (workspace), is created as shown in Figure 14.

Figure 14. Create Project “Design01”



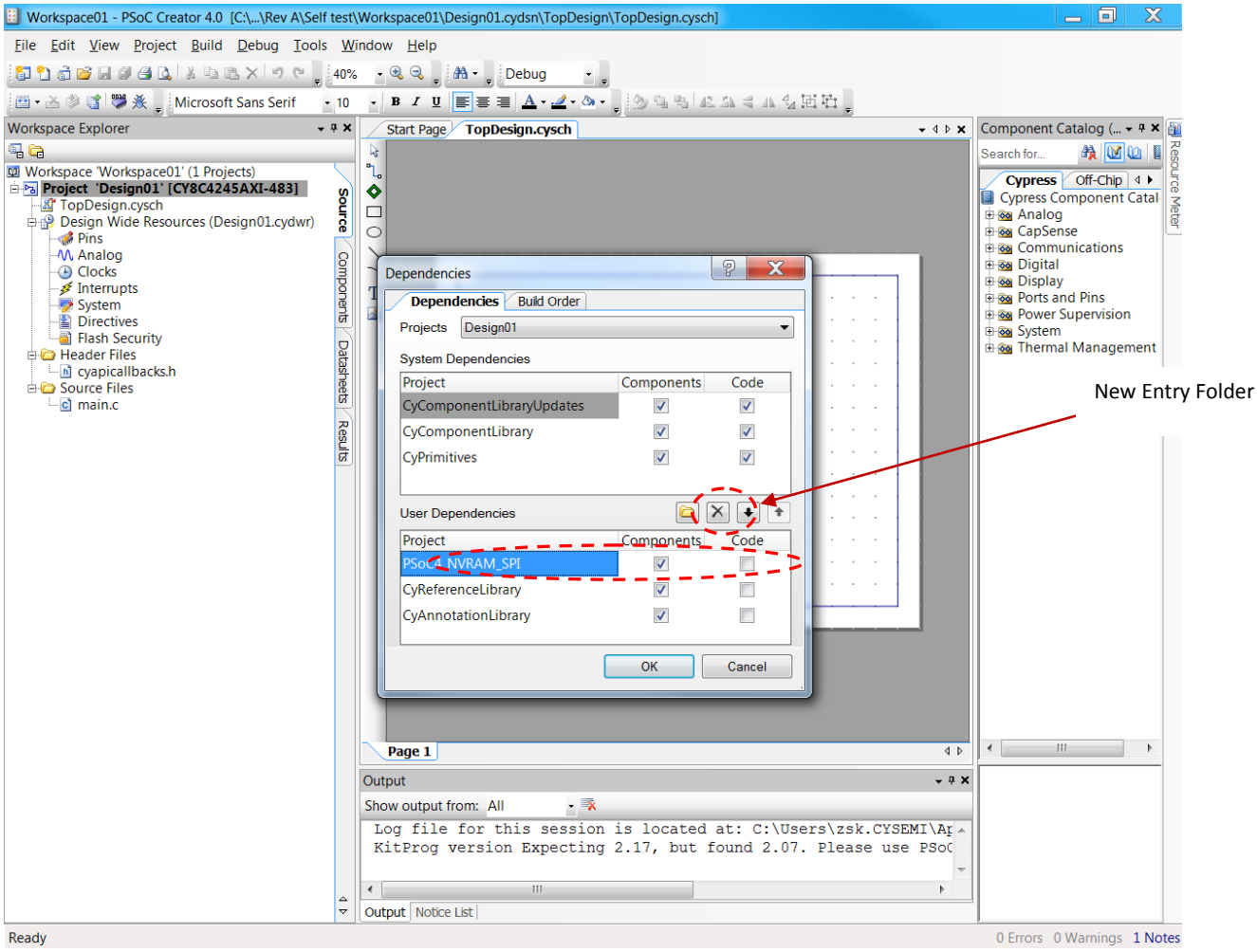
3. Right-click on the project and select the **Dependencies** tab in the **Workspace Explorer.** Bring the NVRAM_SPI component into your design as shown in Figure 15.

Figure 15. Open Dependencies



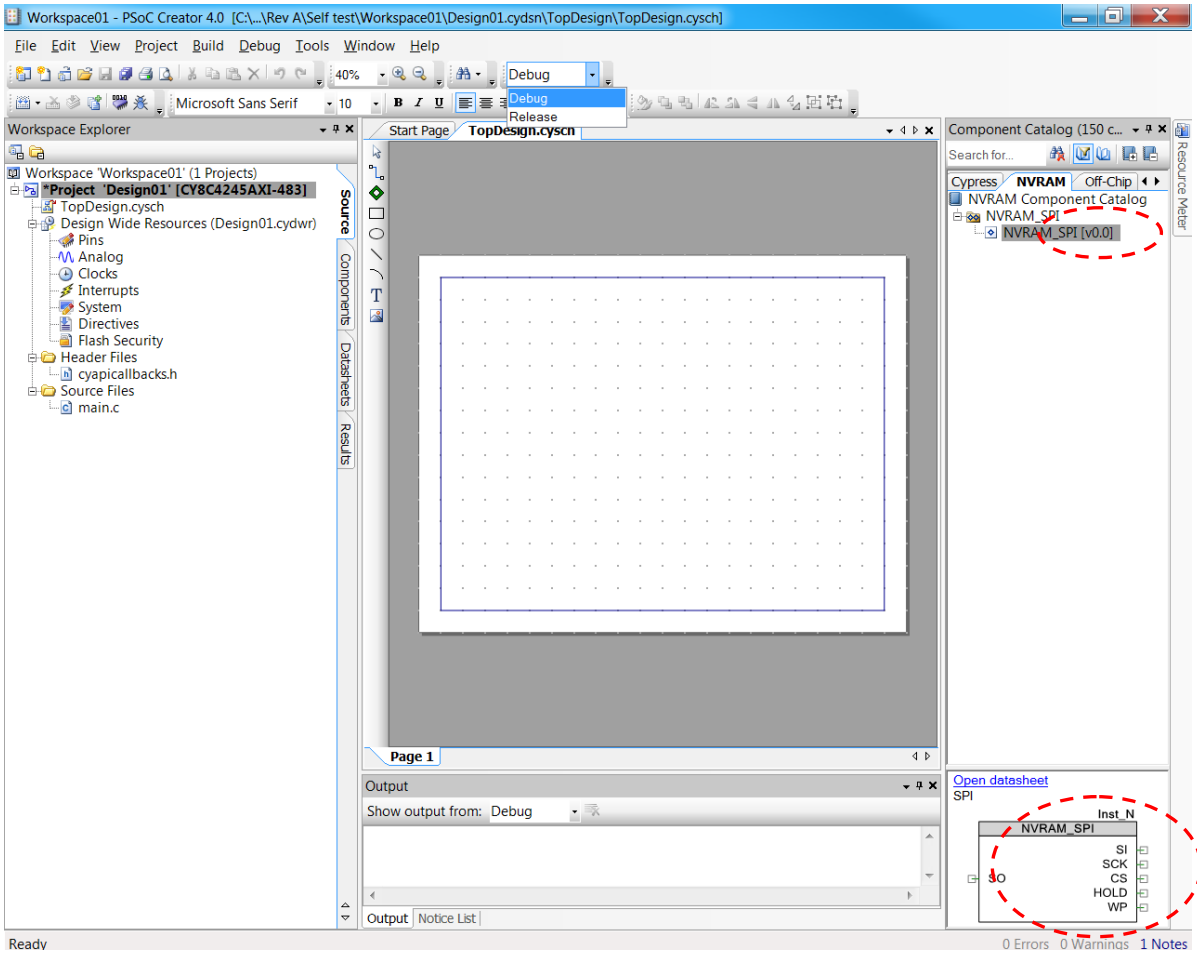
4. Click on **New Entry** (User Dependencies) and select *PSoC4_NVRAM_SPI.cyprj* from the *PSoC4_NVRAM_SPI.cydsn* folder, as shown in Figure 16.

Figure 16. Add Dependencies



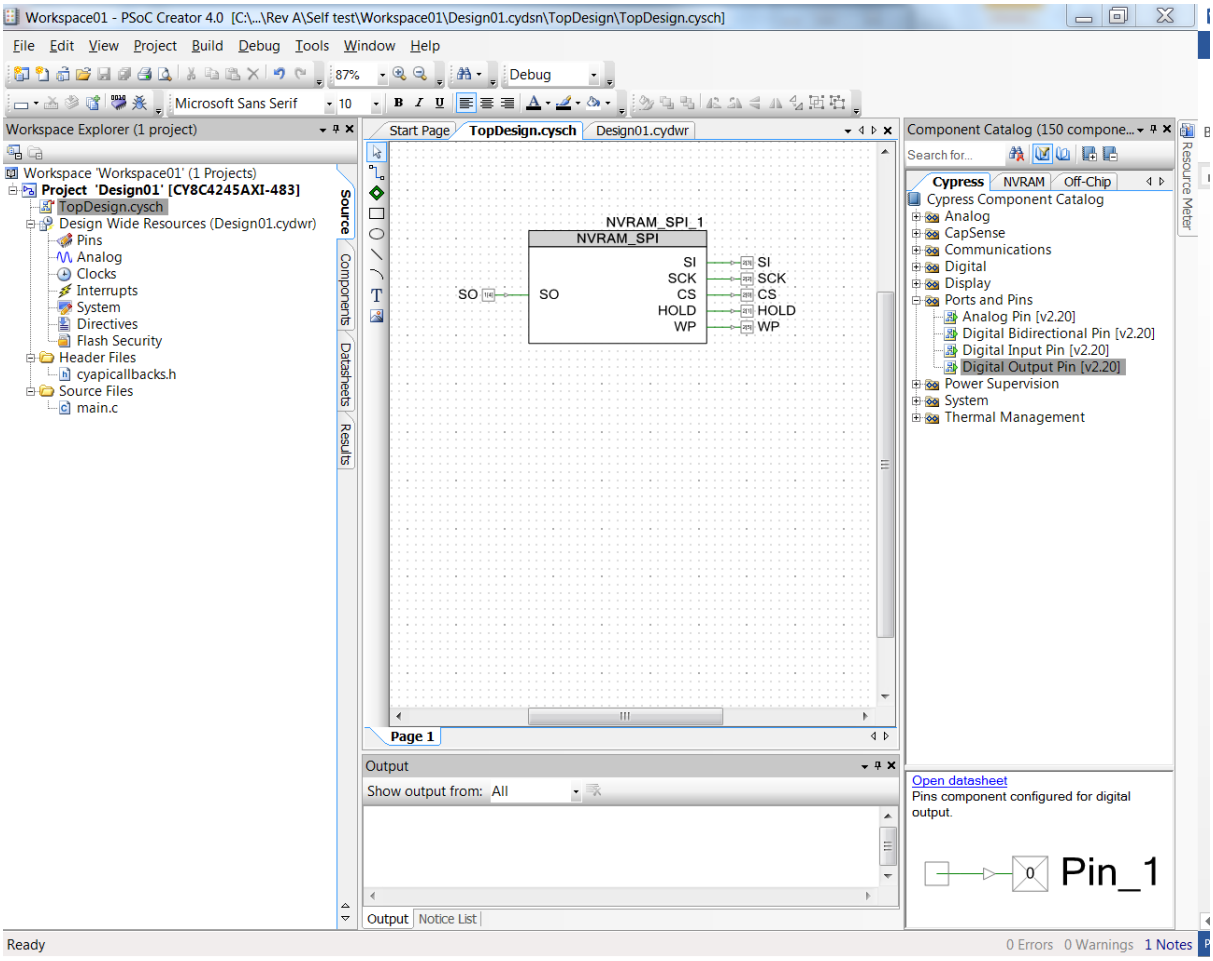
5. The NVRAM_SPI component appears under the **NVRAM** tab and under NVRAM_Component Catalog**,** as shown in Figure 17.

Figure 17. NVRAM_SPI Component in Catalog



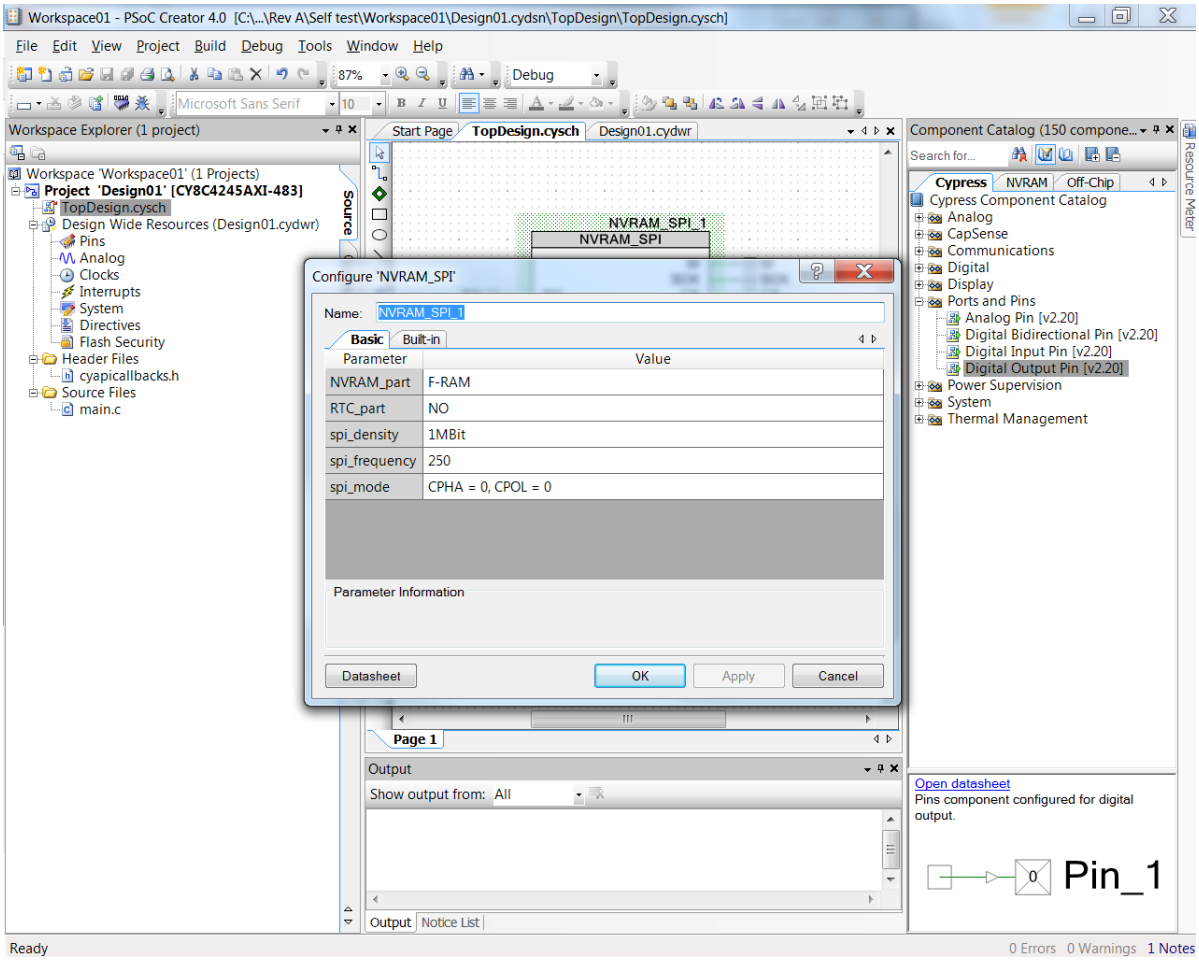
6. Drag and drop the NVRAM_SPI component onto the TopDesign.cysch and assign digital I/Os. Connect the component input/output pins to the appropriate port pins of PSoC 4. Table 5 identifies the pin assignments in the example project.

Figure 18. Creating Design Schematic Using NVRAM_SPI Component



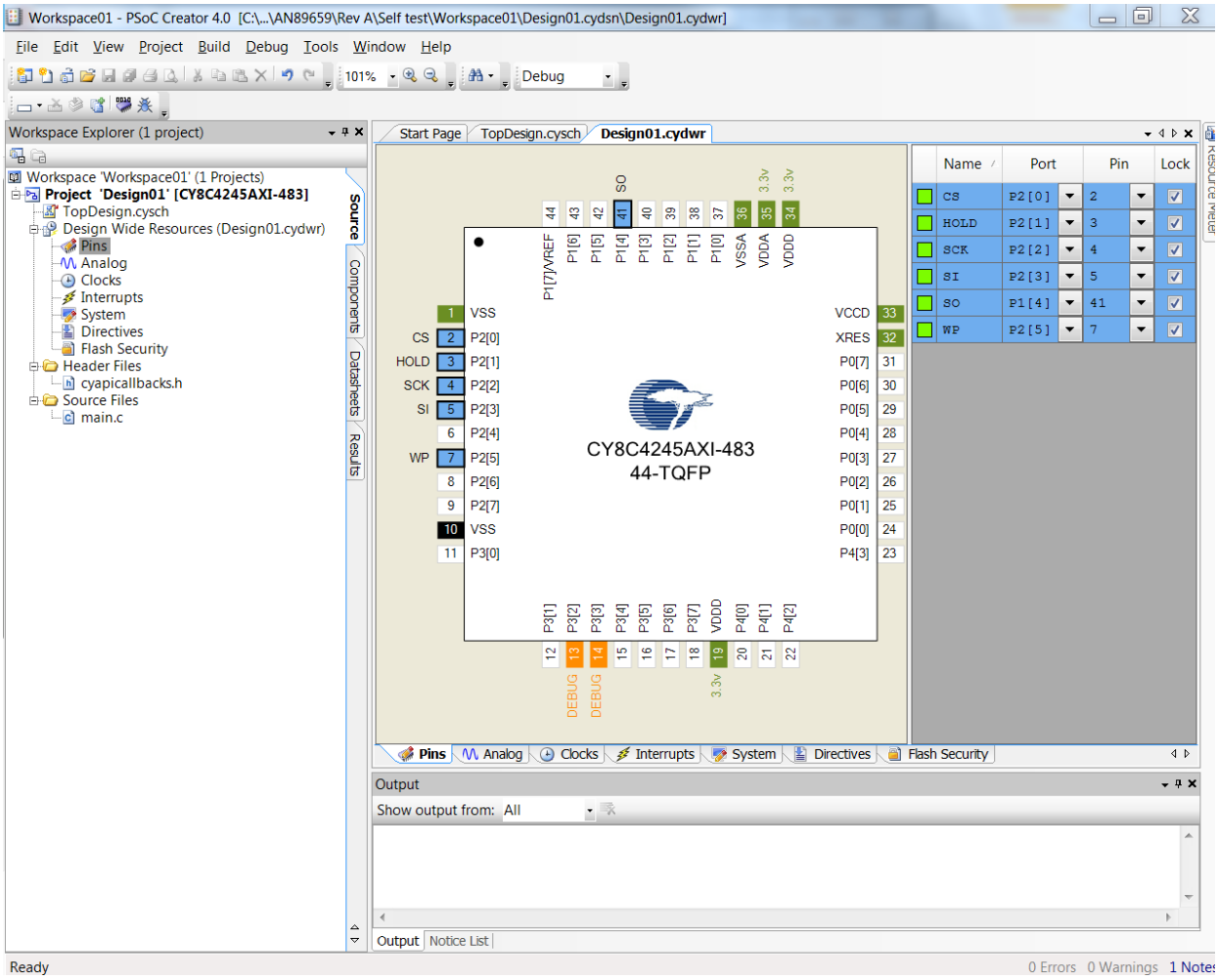
7. Double-click the NVRAM_SPI component and configure the component parameters as follows. Select the NVRAM_Part as F-RAM, the RTC as NO, and the SPI density per the F-RAM density used in the application. Then set the SPI frequency and mode (Mode 0 or Mode 3) per the application requirements as shown in Figure 19.

Figure 19. Configure NVRAM_SPI Component Parameters



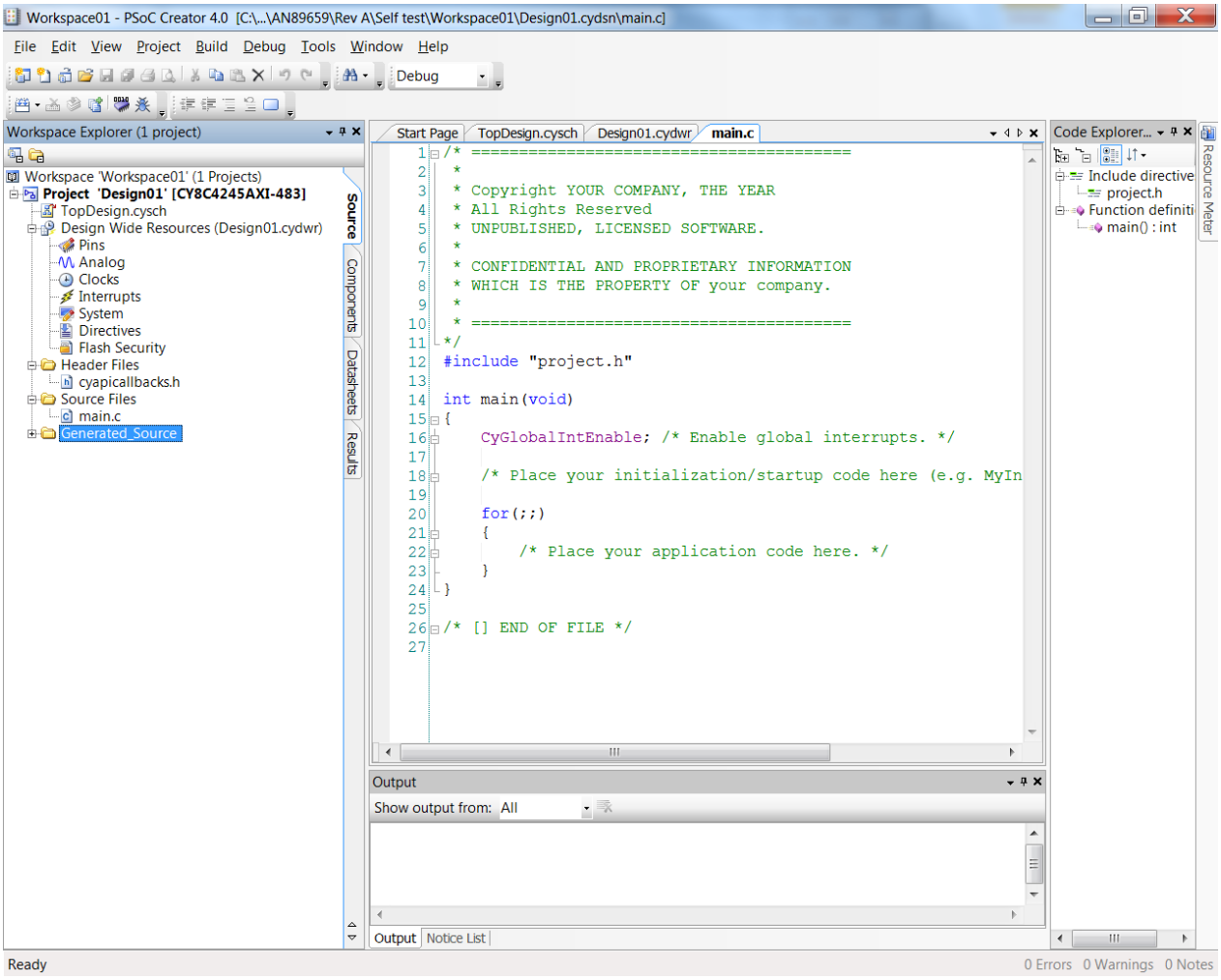
8. Assign appropriate input/output pins per your design, as shown in Figure 20, and build the project.

Figure 20. Assign PSoC 4 Pins



9. Develop your code in the *main.c* file. You can call APIs directly into your program and execute F-RAM functionality.

Figure 21. Project main.c File



For more details on PSoC 4 and PSoC Creator, refer to AN79953 – Getting Started with PSoC® 4 and the associated links.

Document History

Document Title: Interfacing SPI F-RAM with PSoC® 4 – AN89659

Document Number: 001-89659

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4222006	ZSK	12/16/2013	New Specification.
*A	5665375	ZSK	05/09/2017	1. New AN template 2. Included a short description on SPI F-RAM custom component on PSoC 4 in Section 4.2 SPI Master Implementation Using SCB 3. Updated Table 3. SPI F-RAM Opcodes to add clarification on 4-Kb WRITE/ READ opcodes 4. Added a note in section A.1 Example Project Pin Assignments on SPI clocking while read operation 5. Recompiled the attached PSoC 4 example project with PSoC Creator 4.0 Service Pack 1 6. Added Figure 13 7. Updated Figure 14 to Figure 21 per PSoC Creator 4.0 Service Pack 1 screen

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709