Cypress's serial peripheral interface (SPI) nvSRAM is a high-performance nonvolatile serial memory that offers zero cycle delay write operation and infinite SRAM write endurance. The SPI nvSRAM is a slave SPI device and requires an SPI master controller to access nvSRAM in a system. This application note provides a few key design considerations and firmware tips to guide the users designing with SPI nvSRAM.

Contents

1 Introduction ................................................................. 1 4.1 Addressing in SPI nvSRAM ................................... 8
2 SPI nvSRAM Configurations ........................................... 2 5 nvSRAM Operations .................................................. 8
   2.1 Input Pin Configuration ...................................... 4 5.1 Status Register Operation ..................................... 8
   2.2 RTC Pin Configuration .................................... 5 5.2 SRAM Write/Read Operations in nvSRAM ...... 10
3 SPI Operating Modes .................................................. 5 6 Summary .......................................................... 12
4 SPI nvSRAM Opcodes .................................................. 6

1 Introduction

Cypress nvSRAM integrates a fast SRAM cell and a nonvolatile cell into a single nvSRAM cell. In the normal mode of operation, all reads and writes happen directly from and to the SRAM portion of the nvSRAM. This provides faster write and read access compared to any existing nonvolatile memory technology, such as EEPROM, flash, FRAM, MRAMs, and battery backed SRAMs. In the event of system power loss, data from the SRAM is transferred to its nonvolatile cell automatically using energy stored in a small capacitor connected to the device V_Cap pin. During the next power-on cycle, data from the nonvolatile cell is recalled automatically into the SRAM array and presented to the user. A capacitor connected to the V_Cap pin of nvSRAM is charged during the normal operation.

The nvSRAM specifies one million endurance cycles for its nonvolatile cells. The nvSRAM endurance cycle is consumed only when the data transfer takes place from an SRAM cell to its nonvolatile cell during a STORE operation. The nonvolatile STORE in the nvSRAM is initiated either automatically, when the device power drops below a predefined threshold level (V_SWITCH), or on demand through an opcode or a hardware (HSB) pin. Note that the SRAM cell provides infinite endurance for write and read operations; therefore, the nvSRAM does not consume any endurance cycle during normal operation. The nonvolatile STORE takes place only when a system power failure is detected and it is required to move data safely into the nonvolatile cells. This implies that the endurance cycle of an nvSRAM equates to the total number of system power failures or system shutdown events, which is unlikely to reach one million cycles in any real-time application.

The SPI nvSRAMs offer high-speed, low-power serial nvSRAMs in industry-standard 8-pin SOIC and 16-pin SOIC packages. The nvSRAM allows writing hundreds of bytes in tens of microseconds as against EEPROM and flash memories, which require tens of milliseconds to do the same. There are many data logging applications, which require instant saving of runtime critical information in the event of power loss. This critical information includes controller runtime execution states or scratch pad data, parameter settings, and other environment variables measured by controllers.

This application note elaborates the SPI nvSRAM connections and functionalities applicable to all standard SPI master controllers. The hardware recommendations made through this application note are not meant as requirements; however, their adoption leads to a more robust overall design. To explain the SPI nvSRAM behavior at the system level, a few opcodes are explained with the help of timing diagrams and PSoC 1-based pseudo codes.

This application note covers the following topics.

Document No. 001-64574 Rev. "I"
Designing with Serial Peripheral Interface (SPI) nvSRAM

- SPI nvSRAM connections
- SPI operating modes
- nvSRAM operations

The code example CE204087 - Interfacing SPI nvRAM with PSoC® 3/5 provides an overview of interfacing SPI nvSRAM with Cypress PSoC 3/5 with the help of an example project.

2 SPI nvSRAM Configurations

Cypress supports SPI nvSRAM in different configurations and package options, as shown in the following table.

<table>
<thead>
<tr>
<th>nvSRAM Part No.</th>
<th>Operating Voltage (Typ)</th>
<th>Package</th>
<th>WP Pin</th>
<th>VCAP Pin / AutoStore</th>
<th>HSB Pin / HW Store</th>
<th>RTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CY14CXXXQ1A</td>
<td>2.5 V</td>
<td>8-pin SOIC</td>
<td>Yes</td>
<td>No/No</td>
<td>No/No</td>
<td>no RTC</td>
</tr>
<tr>
<td>CY14BXXXQ1A</td>
<td>3.0 V</td>
<td>8-pin SOIC</td>
<td>No</td>
<td>Yes/Yes</td>
<td>No/No</td>
<td>no RTC</td>
</tr>
<tr>
<td>CY14EXXXQ2A</td>
<td>2.5 V</td>
<td>16-pin SOIC</td>
<td>Yes</td>
<td>Yes/Yes</td>
<td>Yes</td>
<td>no RTC</td>
</tr>
<tr>
<td>CY14EXXXQ2A</td>
<td>5.0 V</td>
<td>16-pin SOIC</td>
<td>Yes</td>
<td>Yes/Yes</td>
<td>Yes</td>
<td>RTC</td>
</tr>
<tr>
<td>CY14EXXXPA</td>
<td>3.0 V</td>
<td>16-pin SOIC</td>
<td>Yes</td>
<td>Yes/Yes</td>
<td>RTC</td>
<td></td>
</tr>
<tr>
<td>CY14EXXXPA</td>
<td>5.0 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The “XXX” in Table 1 represents the space for providing density options in the nvSRAM part number. XXX = 064 is 64 Kbit; XXX=256 is 256 Kbit; XXX=512 is 512 Kbit; XXX=101 is 1 Mbit; and XXX=102 is 2 Mbit nvSRAM density.

The connection between an SPI host controller and the nvSRAM device varies depending upon the nvSRAM device configuration and package option. Figure 2 to Figure 4 show the detailed schematic connections of SPI SRAMs available in different configuration and package options. The hardware connections between an SPI host controller and the SPI nvSRAM remains identical across all densities for a particular configuration and package option.

A typical system-level configuration of the SPI nvSRAM device is illustrated in Figure 1. For microcontrollers that have no dedicated SPI bus, a general-purpose I/O port may be used for the connection.
Figure 1. Typical SPI nvSRAM Connection

Figure 2. 8-pin SPI nvSRAM Interface (No V\text{CAP}) with Controller

Figure 3. 8-pin SPI nvSRAM Interface (with V\text{CAP}) with Controller

These are optional connections. These pins can be configured to a default state if not used in the application.
2.1 Input Pin Configuration

SPI nvSRAMs have many control input pins, which should be properly biased to a fixed logic state (HIGH or LOW) for proper device operation. If a pin is left floating, it can assume an intermediate level causing a high stand-by current or it can float to logic level LOW or HIGH. The direction in which the signal goes depends upon a number of factors, such as noise in the system, capacitive coupling, and leakage. Because of this, the level seen by the input 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.

**HOLD Pin**: The SPI nvSRAM features a HOLD pin, which is an active low input and allows the user to suspend the clock midstream to pause an ongoing communication. If this pin floats LOW, the device no longer reacts to any clock pulse received, communication is disrupted, and data potentially lost or corrupted. If not used, this pin should have a 10-kΩ pull-up resistor to avoid an undesired event due to noise during these conditions.

**CS Pin**: The microcontroller should always drive the Chip Select pin (CS) during normal operation. It has the potential to float during microcontroller powering-down or powering-up. This pin should also have a 10-kΩ pull-up resistor to avoid undesired commands due to noise during these conditions.

**WP Pin**: The Write Protect (WP) is an active low input signal used to protect writing into the main memory and the Status Register by pulling this pin low externally. The WPEN bit in the status register determines the functionality of the WP pin. If the WPEN bit is set to ‘1’, it enables the WP pin control; if it is set to ‘0’, then WP pin is disabled. This pin should have a 10-kΩ pull-up resistor to avoid an undesired event due to noise during these conditions.
**HSB Pin:** The HSB pin is a bidirectional pin on the nvSRAM. As an output, it provides nvSRAM’s ready/busy status during the nonvolatile STORE operation. When nvSRAM is in ready status it allows the host controller to access all functionalities of the device. When nvSRAM is in busy status all commands except the read status register are inhibited and device sets the RDY bit in the status register, which can be retrieved through the read status register command. As an input pin, the HSB pin is used to initiate hardware STORE externally by the controller. This pin can be left floating if not connected to any GPIO. An internal weak pull-up is provided to keep HSB HIGH during normal operation. If HSB is being controlled externally by a controller GPIO, then an external 10-kΩ pull-up resistor should be used to avoid an undesired triggering due to noise on this line.

**V<sub>CAP</sub>:** A capacitor connected on V<sub>CAP</sub> supplies power to the nvSRAM during power loss to store data from the SRAM to nonvolatile elements. During normal operation, the device draws current from V<sub>CC</sub> to charge capacitor. This stored charge is used by the chip to perform a single STORE operation. If the voltage on the V<sub>CC</sub> pin drops below V<sub>S</sub>SWITCH, the part automatically disconnects the V<sub>CAP</sub> pin from V<sub>CC</sub>. A STORE operation is initiated with power provided by the V<sub>CAP</sub> capacitor.

Always connect an appropriate value capacitor on the V<sub>CAP</sub> pin for a successful AutoStore operation. The capacitor value selected should fall within the range prescribed in the device datasheet. An improper selection of capacitor may lead to malfunctioning of the device. See application note, Storage Capacitor Options for Cypress nvSRAM – AN43593, for more details on capacitor selection guidelines for nvSRAM products.

### 2.2 RTC Pin Configuration

The following pins are specific to the RTC feature. If the RTC function is not used, these pins can be left floating on the board.

**INT Pin:** This is an output pin in the RTC parts. The INT output is multiplexed to bring out different functionalities in the RTC nvSRAM device. The INT pin can be configured to bring out any of the alarm status, watchdog timer status, calibration clock output, and square wave output depending upon the RTC register setting and their priority defined in the nvSRAM. The INT pin is a configurable driver output by setting H/L bit in Interrupt Status/Control register. When H/L bit is set to ‘1’, the INT pin is active HIGH and the driver mode is push pull. When H/L bit is set to ‘0’, the INT pin is active low open drain output and thus requires an external pull-up resistor to drive to a logic HIGH state. Therefore, the INT pin must be pulled up to V<sub>CC</sub> by using an external 10-kΩ pull-up resistor while using the INT in active low mode.

**V<sub>RTCbat</sub> and V<sub>RTCcap</sub> Pins:** These pins are used to provide the backup power supply to the RTC circuitry of the nvSRAM device to keep the RTC clock running when the system power supply (V<sub>CC</sub>) is down. The V<sub>RTCbat</sub> and V<sub>RTCcap</sub> pin should either connect to a non-rechargeable battery on the V<sub>RTCbat</sub> or a super capacitor on the V<sub>RTCcap</sub> pin. If not used, these pins should be left floating.

**Note** The V<sub>RTCcap</sub> pin cannot be shorted to the V<sub>SS</sub> directly because this pin is used to charge the super capacitor connected to it during the normal operation. Hence, connecting the V<sub>RTCcap</sub> pin directly to the ground (V<sub>SS</sub>) may draw excessive current from the nvSRAM.

For nvSRAM RTC design guidelines and best practices, see the application note, Non Volatile Static Random Access Memory (nvSRAM) Real Time Clock (RTC) Design Guidelines and Best Practices – AN61546.

### 3 SPI Operating Modes

The SPI nvSRAMs support SPI Mode 0 (CPOL=0, CPHA=0) and SPI Mode 3 (CPOL=1, CPHA=1) operation, which depends on the clock polarity (CPOL) and clock phase (CPHA) set by the SPI master at the beginning of SPI communication. Table 2 summarizes all SPI modes with respect to the SPI clocking and data driving on MOSI and MISO lines by the SPI master and slave, respectively. The SPI mode in the nvSRAM is automatically configured according to the master controller’s SPI mode.

<table>
<thead>
<tr>
<th></th>
<th>Mode 0 (CPOL=0; CPHA=0)</th>
<th>Mode 1 (CPOL=0; CPHA=1)</th>
<th>Mode 2 (CPOL=1; CPHA=0)</th>
<th>Mode 3 (CPOL=1; CPHA=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPI Clock (SCK) Start Logic Level</strong></td>
<td>LOW</td>
<td>LOW</td>
<td>HIGH</td>
<td>HIGH</td>
</tr>
<tr>
<td><strong>Data Latched-In by the nvSRAM on MOSI</strong></td>
<td>SCK Rising Edge (↑)</td>
<td>SCK Falling Edge (↓)</td>
<td>SCK Falling Edge (↓)</td>
<td>SCK Rising Edge (↑)</td>
</tr>
</tbody>
</table>

---

**www.cypress.com**  
Document No. 001-64574 Rev. *I*  
5
4 SPI nvSRAM Opcodes

All SPI opcodes, addresses, and data are considered 8-bit data transfers; therefore, all internal operations are byte-wide in nature. All transactions occur with CS low. Address, control, 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 nvSRAM supports industry standard opcodes for all read and write operation. It also supports special opcodes for nvSRAM specific NV operations and high-speed (104 MHz) SPI access. A unique opcode is assigned for each specific operation in the SPI nvSRAM. A list of SPI nvSRAM instructions with their respective opcodes are defined in Table 3.

Table 3. SPI nvSRAM Opcodes

<table>
<thead>
<tr>
<th>Instruction Category</th>
<th>Instruction Name</th>
<th>Opcode</th>
<th>CY14B101P/Qx</th>
<th>CY14C101PA/QxA, CY14B101PA/QxA, CY14E101PA/QxA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Opcode Support</td>
<td>SPI Frequency</td>
</tr>
<tr>
<td>Status Register Control Instructions</td>
<td>WREN</td>
<td>06H (0000 0110)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>WRDI</td>
<td>04H (0000 0100)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>RDSR</td>
<td>05H (0000 0101)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>FAST_RDSR</td>
<td>09H (0000 1001)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>WRSR</td>
<td>01H (0000 0001)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td>SRAM Read and Write Instructions</td>
<td>READ</td>
<td>03H (0000 0011)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>FAST_READ</td>
<td>0BH (0000 0111)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>WRITE</td>
<td>02H (0000 0010)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td>RTC Instructions [Note 1]</td>
<td>WRTC</td>
<td>12H (0000 0010)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>RDRTC</td>
<td>13H (0000 0011)</td>
<td>√</td>
<td>Up to 25 MHz</td>
</tr>
<tr>
<td></td>
<td>FAST_RDRTC</td>
<td>1DH (0001 1101)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>NV Instructions [Note 2]</td>
<td>STORE</td>
<td>3CH (0111 1100)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>RECALL</td>
<td>60H (0110 0000)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>ASENB</td>
<td>59H (0101 1001)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td></td>
<td>ASDISB</td>
<td>19H (0001 1001)</td>
<td>√</td>
<td>Up to 40 MHz</td>
</tr>
<tr>
<td>Sleep</td>
<td>SLEEP</td>
<td>99H (1011 1001)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>Serial Number</td>
<td>WRSN</td>
<td>C2H (1100 0010)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>RDSN</td>
<td>C3H (1100 0011)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>FAST_RDSN</td>
<td>C9H (1100 1001)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>Device ID Read</td>
<td>RDID</td>
<td>9FH (1001 1111)</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>FAST_RDID</td>
<td>99H (1001 1001)</td>
<td>X</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note 1 RTC instructions are specific to RTC nvSRAM devices (CY14C101P/PA, CY14B101P/PA, and CY14E101P/PA). These commands are not applicable for non-RTC devices.

Note 2 These commands are specific to the nvSRAM parts for the execution of NV operations.
Table 4 explains each opcode with associated data bytes required for its proper operation.

### Table 4. SPI nvSRAM Data Flow Format

<table>
<thead>
<tr>
<th>Instruction Name</th>
<th>Opcode</th>
<th>Master Transmits on SI</th>
<th>nvSRAM Transmits on SO</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>WREN</td>
<td>06H</td>
<td>06H</td>
<td>-</td>
<td>This command sets the WEN bit in the status register.</td>
</tr>
<tr>
<td>WRDI</td>
<td>04H</td>
<td>04H</td>
<td>-</td>
<td>Clear WEN bit (if set) in the status register</td>
</tr>
<tr>
<td>RDSR</td>
<td>05H</td>
<td>05H</td>
<td>StatusReg_Data</td>
<td>Read status register contents</td>
</tr>
<tr>
<td>FAST_RDSR</td>
<td>09H</td>
<td>09H, Dummy_Byte</td>
<td>StatusReg_Data</td>
<td></td>
</tr>
<tr>
<td>WSRD</td>
<td>01H</td>
<td>01H, StatusReg_Data</td>
<td>-</td>
<td>WEN bit must be set prior to write into Status Reg. WEN is cleared when $CS$ goes HIGH.</td>
</tr>
<tr>
<td>READ [Note 3]</td>
<td>03H</td>
<td>03H, Add1, Add2, Add3</td>
<td>Data1, Data2, Data3,..., DataN</td>
<td>Read data length 1 to N. N can be any integer value. nvSRAM’s internal address counter automatically increments by one. When nvSRAM count reaches its maximum addressing limit, it rolls over to the starting address and continues reading data from there. Read exits when $CS$ goes HIGH.</td>
</tr>
<tr>
<td>FAST_READ [Note 3]</td>
<td>0BH</td>
<td>0BH, Add1, Add2, Add3, Dummy_Byte</td>
<td>Data1, Data2, Data3,..., DataN</td>
<td>WEN bit must be set prior to write into nvSRAM memory. Write data length 1 to N, where N can be any integer value. nvSRAM’s internal address counter automatically increments by one. When nvSRAM count reaches its maximum addressing limit, it rolls over to the starting address and continues writing data from there by overlapping previously written data. The firmware must take care of data overwriting due to memory counter roll over during bulk write operation. Write exits when $CS$ goes HIGH.</td>
</tr>
<tr>
<td>WRITE [Note 3]</td>
<td>02H</td>
<td>02H, Add1, Add2, Add3, Data1, Data2, Data3,..., DataN</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>WRTC [Note 4]</td>
<td>12H</td>
<td>12H, Addr Data</td>
<td>-</td>
<td>‘W’ bit must be set to ‘1’ in the RTC flag register and WEN bit must be set to ‘1’ in the status register. WEN is cleared when $CS$ goes HIGH.</td>
</tr>
<tr>
<td>RDRRTC [Note 4]</td>
<td>13H</td>
<td>13H, Addr</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>FAST_RDRRTC [Note 4]</td>
<td>1DH</td>
<td>1DH, Addr, Dummy_Byte</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>STORE</td>
<td>3CH</td>
<td>3CH</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>RECALL</td>
<td>60H</td>
<td>60H</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ASENB</td>
<td>59H</td>
<td>59H</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ASDISB</td>
<td>19H</td>
<td>19H</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SLEEP</td>
<td>B9H</td>
<td>B9H</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>WRSN</td>
<td>C2H</td>
<td>C2H, Data1, Data2, ..., Data8</td>
<td>-</td>
<td>WEN bit must be set prior to initiating Sleep command. When $CS$ goes HIGH, the device registers Sleep command. WEN is cleared when $CS$ goes HIGH.</td>
</tr>
<tr>
<td>RDSN</td>
<td>C3H</td>
<td>C3H</td>
<td>Data1, Data2, Data3, ..., Data8</td>
<td>Read 8 bytes serial number, WEN is cleared when $CS$ goes HIGH.</td>
</tr>
<tr>
<td>FAST_RDSN</td>
<td>C9H</td>
<td>C9H, Dummy_Byte</td>
<td>Data1, Data2, Data3, ..., Data8</td>
<td>Read 8 bytes serial number</td>
</tr>
<tr>
<td>RDID</td>
<td>9FH</td>
<td>9FH</td>
<td>Data1, Data2, Data3, Data4</td>
<td>Device ID 4 bytes</td>
</tr>
<tr>
<td>FAST_RDID</td>
<td>99H</td>
<td>99H, Dummy_Byte</td>
<td>Data1, Data2, Data3, Data4</td>
<td>Device ID 4 bytes</td>
</tr>
</tbody>
</table>
Note 3 1 Mb and higher density SPI nvSRAM use 3-byte address; lower densities nvSRAMs (512 Kbit or less) use 2-byte address.

Note 4 RTC instructions are specific to RTC nvSRAM devices (CY14C101PA, CY14B101PA, and CY14E101PA). These commands are not applicable for non-RTC devices.

4.1 Addressing in SPI nvSRAM

An SPI host controller communicates with the SPI nvSRAM on byte-by-byte basis and always transmits the most significant bit in the first clock cycle and the least significant bit in the eighth clock cycle during a byte transmission. This holds good for all SPI communication including command, address, and data bytes. Similarly, when an SPI nvSRAM transmits data byte during a read operation, it always transmits the most significant bit first and the least significant bit in the last during data byte transmission. Figure 5 shows an example of address bits being transmitted over the SPI MOSI (master out slave in) line while transmitting three address bytes by the SPI master.

Figure 5. Address Bit Transmission in SPI nvSRAM

Figure 6 represents the addressing scheme for different nvSRAM densities. A0 is the least significant bit (LS Bit) in an address.

Figure 6. SPI nvSRAM Opcode and Addressing

Unused bits of the most significant address byte (MSB) are don’t care bits and nvSRAM ignores the status of these bits. However, it is good practice to set unused address bits to ‘0’ in the firmware. This approach makes it easy to upgrade the firmware while moving to a higher density device in the same socket.

5 nvSRAM Operations

This section describes nvSRAM operations with the help of a timing diagram and PSoC 1 specific pseudo codes. All functions starting with the prefix ‘SPIM_SPIM_’ are PSoC1 specific functions. Their representations and implementations are subject to change for controllers used as SPI master.

This section does not cover all opcode defined for the SPI nvSRAM. See the device datasheet for a detailed description on each opcode.

5.1 Status Register Operation

Write Status Register: To write into the Status Register, you need to send the status register write opcode (WRSR) command followed by a data byte to be written.

- Set the WEN bit by sending WREN opcode.

- 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 7 shows a timing diagram for writing into the Status Register.
Designing with Serial Peripheral Interface (SPI) nvSRAM

Figure 7. Writing into Status Register

Figure 8. Reading from Status Register

```c
/* ****************** PSoC1 Based Pseudo Code for Status Register Write ******************/  
#define CS_HI Port0_0(1)  
#define CS_LO Port0_0(0)  

void WRSR(BYTE data1)  // User Define Function  
{  
  BYTE WREN=0x06;  
  BYTE OPCODEWRSR=0x01;  
  
  CS_LO;  
  while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));  
  SPIM_SendTxData(WREN);  // This will set WEN='1'  
  CS_HI;  
  
  CS_LO;  
  while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));  
  SPIM_SendTxData(OPCODEWRSR);  // Send OPCODE for Status Register write  
  
  while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));  
  SPIM_SendTxData(data1);  // Send the data to be written into Status Register  
  CS_HI;  
}  

/* ****************** PSoC1 Based Pseudo Code for Status Register Read ******************/  
BYTE RDSR ()  // User Define Function  
{  
  BYTE OPCODERDSR=0x05;  
}  
```

**Read Status Register:** To read the Status Register contents, you need to send the read status register opcode (RDSR), after which the nvSRAM starts sending the Status Register contents on SO line. The SPI nvSRAM must remain selected by pulling the chip select pin to LOW and SPI clock should be available to read the Status Register contents followed by RDSR command. **Figure 8** shows a timing diagram for reading from the Status Register.
BYTE data;

CS_LO;
while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
SPIM_SendTxData(OFCODERDSR); //Send instruction

while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
SPIM_SendTxData(0x01); //Dummy write to generate CLK and read data

while(!(SPIM_bReadStatus() & SPIM_SPIM_RX_BUFFER_FULL));
data = SPIM_bReadRxData(); //Read Byte from Status Register
CS_HI;

return(data);

5.2 SRAM Write/Read Operations in nvSRAM

SRAM Write: To write into the SRAM array of SPI nvSRAM, the controller must initiate the write command in the following manner.

- Send the WREN opcode to set the write enable latch (WEN) bit
- Send the WRITE opcode
- Send the most significant address byte
- Send the intermediate address byte (in 3 bytes addressing)
- Send the lower address byte
- Send data byte/bytes

Any write command to the nvSRAM should be preceded with a Write Enable (WREN) instruction. If the device is not write enabled (WEN = ‘0’), it ignores the write instructions and returns to the standby state when CS is brought to the HIGH state. A new CS falling edge is required to re-initiate SPI serial communication.

After completion of a any write instruction (WRSR, WRITE, or WRTC) or nvSRAM special instruction (STORE, RECALL, ASEN, ASDISB) instruction, the WEN bit of the Status Register is cleared to ‘0’ on the rising edge of CS at the end of the write cycle. This provides protection from any inadvertent writes.

Also, note that reading the Status Register (RDSR opcode) between the WREN and any write instructions does not clear the WEN bit. Some users read the Status Register immediately following the WREN to confirm that the WEN bit is set prior to a write operation. Figure 9 shows a timing diagram for writing into the SRAM memory.

Figure 9. Writing into SRAM

/* PSoC1 Based Pseudo Code for nvSRAM write in burst mode. By sending tot_cnt =1, user can write only 1 byte at a given address location*/

void nvSRAMBURSTWRITE(BYTE addr1, BYTE addr2, BYTE addr3, DWORD tot_cnt, BYTE*data) // User Define Function
{  BYTE WREN=0x06;
/* PSoC1 Based Pseudo Code for nvSRAM write in burst mode. By sending tot_cnt =1, user can write only 1 byte at a given address location*/

void nvSRAMBURSTWRITE(BYTE addr1, BYTE addr2, BYTE addr3, DWORD tot_cnt, BYTE*data) // User Define Function
{  BYTE WREN=0x06;
BYTE OPCODEWRITE=0x02;
DWORD count=0;

CS_LO;
while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
SPIM_SendTxData(WREN); //Set WEN='1' prior to write
CS_HI;

CS_LO;
while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
SPIM_SendTxData(OPCODEWRITE); //Send OPCODE for Write into main memory
while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
SPIM_SendTxData(addr1); //Send MS Byte
while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
SPIM_SendTxData(addr2); //Send Intermediate Address Byte
while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
SPIM_SendTxData(addr3); //Send LS Byte of Address

for(count=0; count< tot_cnt; count++)
{
    while(!(SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY));
    SPIM_SendTxData(data[count+4]); //Byte written into main memory
    CS_HI;
}

SRAM Read: To read from the SRAM array of SPI nvSRAM, the controller must send opcode and address in the following format.

- Send the READ opcode
- Send the most significant address byte
- Send the intermediate address byte (in 3-byte addressing)
- Send the lower address byte

The nvSRAM sends data-out on SO line until the device remains selected by pulling chip select signal LOW and SPI clock is available.

To initiate SRAM read, the controller issues a READ opcode followed by read address bytes. After registering the read request and address, the nvSRAM sends data out on the SO pin. Subsequent data bytes can be accessed simply by keeping CS LOW while clocking-out data byte after data byte. This is called the burst mode read and the address increments automatically by the SPI nvSRAM device. When CS is de-asserted to HIGH, data output stops and SO goes to a high impedance (HI-Z) state. Figure 10 shows a timing diagram for reading from SRAM.

```c
void nvSRAMBURSTREAD(BYTE addr1, BYTE addr2, BYTE addr3, DWORD tot_cnt, BYTE * readDataArr)
{
    BYTE readdata;
    BYTE data;
    BYTE OPCODEREAD=0x03;
    DWORD count=0;
    //********************nvSRAM Read Burst Data***************************/
    //......
}```
6 Summary

Cypress SPI nvSRAMs support standard SPI access protocols similar to any other nonvolatile SPI memory products. This makes nvSRAM compatible to all SPI master controllers and reduces system development cycle time. All SPI opcodes, except a few which are specific to nvSRAMs, are matched with standard SPI memory product opcodes. This makes SPI nvSRAM a drop-in replacement for all other nonvolatile memory devices in the same functionality and form factor. This application note demonstrates how to configure SPI nvSRAM into an application with the help of schematics, timing diagrams, and example code.
Designing with Serial Peripheral Interface (SPI) nvSRAM

Document History

Document Title: AN64574 – Designing with Serial Peripheral Interface (SPI) nvSRAM

Document Number: 001-64574

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Orig. of Change</th>
<th>Submission Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>3073654</td>
<td>ZSK</td>
<td>10/29/2010</td>
<td>New application note.</td>
</tr>
<tr>
<td>*A</td>
<td>3127847</td>
<td>MEDU</td>
<td>01/04/2011</td>
<td>Added PSoC 3 Library component related information in all instances across the document.</td>
</tr>
<tr>
<td>*B</td>
<td>3337946</td>
<td>ZSK</td>
<td>08/05/2011</td>
<td>Updated SPI nvSRAM Configurations: Added Table 1 (with details of nvSRAM device configurations). Updated SPI Operating Modes: Provided additional information. Added Table 2. Updated SPI nvSRAM Opcodes: Updated Addressing in SPI nvSRAM: Provided more details. Added Figure 6.</td>
</tr>
<tr>
<td>*C</td>
<td>3349553</td>
<td>ZSK</td>
<td>08/19/2011</td>
<td>Minor ECN to include project files</td>
</tr>
<tr>
<td>*D</td>
<td>3508460</td>
<td>ZSK</td>
<td>02/09/2012</td>
<td>Minor fixes after internal Audits. No technical content update. Updated to new template.</td>
</tr>
<tr>
<td>*E</td>
<td>3743056</td>
<td>ZSK</td>
<td>09/13/2012</td>
<td>Updated PSoC 3 Library component (To include option for selecting two/three bytes addressing for SPI nvSRAM).</td>
</tr>
<tr>
<td>*F</td>
<td>4041221</td>
<td>ZSK</td>
<td>06/27/2013</td>
<td>No update to the application note contents. Changed the PSoC 3 Library component name from &quot;AN64574_NVSRAM_SPI&quot; to &quot;AN64574_NVSRAM_SPI&quot;. Added APIs to access the RTC registers in the PSoC 3 example project. Enhanced the PSoC 3 example project to add user select options for memory density, RTC/non RTC, and nvSRAM /FRAM. Updated to new template.</td>
</tr>
<tr>
<td>*G</td>
<td>4225903</td>
<td>ZSK</td>
<td>12/19/2013</td>
<td>Added Software revision as “PSoC Creator 3.0 or above, PSoC Designer 5.2 or above”. Completing Sunset Review.</td>
</tr>
<tr>
<td>*H</td>
<td>4591511</td>
<td>MEDU</td>
<td>12/09/2014</td>
<td>Added PSoC 4 Library component related information in all instances across the document. Updated attached associated project files.</td>
</tr>
<tr>
<td>*I</td>
<td>5670785</td>
<td>ZSK</td>
<td>03/24/2017</td>
<td>Removed PSoC 3/5 example project attached with this AN and replaced with a webpage link to an associated Code Example CE204087 - Interfacing SPI nvRAM with PSoC® 3/5. Removed all references to associated PSoC project setup. The project setup discussion is moved to code example project CE204087. Ported to new Cypress application note template.</td>
</tr>
</tbody>
</table>
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
- 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

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

© Cypress Semiconductor Corporation, 2010-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.