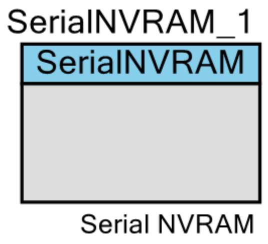
PSoC 4 Serial NVRAM (SerialNVRAM_P4)

1.0



Features

- Serial NVRAMs (nvSRAMs and F-RAMs) Read/Write
- nvSRAM Real Time Clock (RTC)
- F-RAM Processor Companion Read/Write [1]
- nvSRAM / F-RAM Special Commands [1]
  - SLEEP
  - Product ID Read
  - Serial Number Read/Write
- nvSRAM Special Nonvolatile Commands (STORE, RECALL, ASENB, ASDISB)
- SPI and I^{2}C Component configuration for accessing serial NVRAMs

General Description

The SerialNVRAM_P4 Component combined with a standard PSoC 4 SPI or I^{2}C Master Component can be used to configure the PSoC 4 device as the SPI Master or the I^{2}C Master to interface with serial NVRAMs. High-level APIs are provided for different F-RAM/nvSRAM operations to significantly reduce the time to integrate an F-RAM/nvSRAM into an application.

Note For the PSoC 4000 family, the SerialNVRAM_P4 Component supports the I^{2}C Master only, because there is no SPI interface.

When to Use a SerialNVRAM_P4 Component

Use this Component with the Cypress Serial F-RAM or Serial nvSRAM part and a PSoC 4 device. PSoC Creator generates high-level APIs for different F-RAM or nvSRAM features.

[1] These features are not supported by all NVRAM parts. Refer to the respective Serial NVRAM device datasheet for more details.

Definitions

- **SPI**: Serial Peripheral Interface
- **I^{2}C**: Inter-Integrated Circuit
- **F-RAM**: Ferro-electric Random Access Memory – A non-volatile memory cell, which retains data into a Ferro cell when the power goes off.
- **nvSRAM**: Non-volatile Random Access Memory – A non-volatile memory cell, which saves data into a SONOS (Silicon-Oxide-Nitride-Oxide-Silicon) cell when the power goes off. When a device is active, all Write and Read accesses happen from SRAM cells.

Quick Start

The following steps show how to configure the SerialNVRAM Component for interfacing the NVRAM device with an SPI interface on the CY15FRAMKIT-001 Serial F-RAM^{TM} development kit:

1. Drag a SerialNVRAM_P4 Component from the Component Catalog Cypress/System/External Memory Interface folder onto your schematic. The placed instance takes the name SerialNVRAM_1.
2. Double-click to open the SerialNVRAM_1 Configure dialog; enter SPI_1 for the **Master Instance** parameter.
3. Drag a SPI (SCB mode) Component from the Component Catalog Cypress/Communications/SPI folder onto your schematic. The placed instance takes the name SPI_1.
4. Double-click to open the SPI_1 Configure dialog.
   - Change the **Mode** parameter to Master.
   - Set the **Number of SS** to 0 when the SPI is not planned for other purposes.
   - Configure the Component as SPI operating mode: either SPI Mode 0 (by initializing with CPOL =0, CPHA =0) or SPI Mode 3 (by initializing with CPOL =1, CPHA =1).

     Note The SPI operating frequency is determined by the target PSoC device selected for your design. Refer to the respective PSoC product datasheet from http://www.cypress.com/ to configure the SPI frequency.
5. Drag a Digital Output Pin from the Component Catalog Cypress/Ports and Pins folder onto your schematic. The placed instance takes the name Pin_1. This is the status pin which connects to LED on the board.
6. Double-click to open the Pin_1 Configure dialog; remove the check mark for HW connection.

7. Open the Pin Editor and select pins for the CY15FRAMKIT-001 kit:
   - SerialNVRAM_1:CS0 – P3[4]
   - SPI_1:miso_m – P3[1]
   - SPI_1:mosi_m – P3[0]
   - SPI_1:sclk_m – P0[6]
   - Pin_1 - P0[3]
8. Build the project to verify the correctness of your design and generate configuration data for the SerialNVRAM_1 instance.
9. In the *main.c* file, initialize the peripheral and start the application:

```
#include "project.h"
/* Example F-RAM Data */
#define EXAMPLE_DATA_BYTE   (0xa5)

int main(void)
{
    uint8 status;
    uint8 data_byte;
    uint8 fram_rd_byte;
    CyGlobalIntEnable; /* Enable the global interrupts. */
    data_byte = EXAMPLE_DATA_BYTE;
    /* FRAM Component initialization */
    SerialNVRAM_1_Start();
    /* Write 1 byte to FRAM address 0x2000 */
    status = SerialNVRAM_1_MemoryWrite(0u, 0x2000, &data_byte, 1u);
    /* Read 1 byte from FRAM address 0x2000 */
    status = SerialNVRAM_1_MemoryRead(0u, 0x2000, &fram_rd_byte, 1u);
    /* If PASS, turn off the LED */
    if((fram_rd_byte == EXAMPLE_DATA_BYTE) &&
       (status == SerialNVRAM_1_SUCCESS))
    {
        Pin_1_Write(1u);
    }

    for(;;)
    {
        /* Place your application code here. */
    }
}
```

10. Build and program the device.

Input/Output Connections

This section describes the various input and output connections for the SerialNVRAM Component. An asterisk (*) in the following list indicates that it may not be shown in the Pins Editor for the conditions listed in the description of that particular I/O.

Name	I/O Type	Description
CS_0 *	Digital Output	Chip Select (CS) for the SPI NVRAM device. This active LOW input activates the device. This pin is available for SPI interfaces only.
CS_1 *	Digital Output	CS for the SPI NVRAM device. This active LOW input activates the device. This pin is available for SPI interfaces only when the SPI CS is greater than 1 or two SPI NVRAM slaves share the same SPI bus.
CS_2 *	Digital Output	CS for the SPI NVRAM device. This active LOW input activates the device. This pin is available for SPI interfaces only when the SPI CS is greater than 2 or three SPI NVRAM slaves share the same SPI bus.
CS_3 *	Digital Output	CS for the SPI NVRAM device. This active LOW input activates the device. This pin is available for SPI interfaces only when the SPI CS is greater than 3 or four SPI NVRAM slaves share the same SPI bus.
HOLD *	Digital Output	HOLD Pin for the SPI NVRAM device. The HOLD pin is used when the host CPU must interrupt a memory operation for another task. When HOLD is LOW, the current operation is suspended. The device ignores any transition on SCK or CS. This pin is available for SPI interfaces only when the Enable Hold parameter is set in the customizer.
WP *	Digital Output	Write Protect. The Active LOW pin SPI NVRAM and Active HIGH pin I^{2}C NVRAM. In SPI NVRAM, active LOW WP prevents the Write operation to the Status Register. In I^{2}C NVRAM, active HIGH WP prevents the Write operation to the memory array. This pin is available when the Enable Write Protect parameter is set in the customizer.

The Serial NVSRAM Component is a wrapper around an SPI or I^{2}C SCB Master Component. You must place an SPI or I^{2}C SCB Component in your design.

The SPI Master Component has miso_m, mosi_m, sclk_m, and ss0_m pins. The ss0_m pin is not required for SerialNVRAM Component operation, because it has its own CS_0 - CS_3 pins. The ss0_m pin can be disabled by setting the **Number of SS** parameter to 0 in the SPI Master Component Configure dialog.

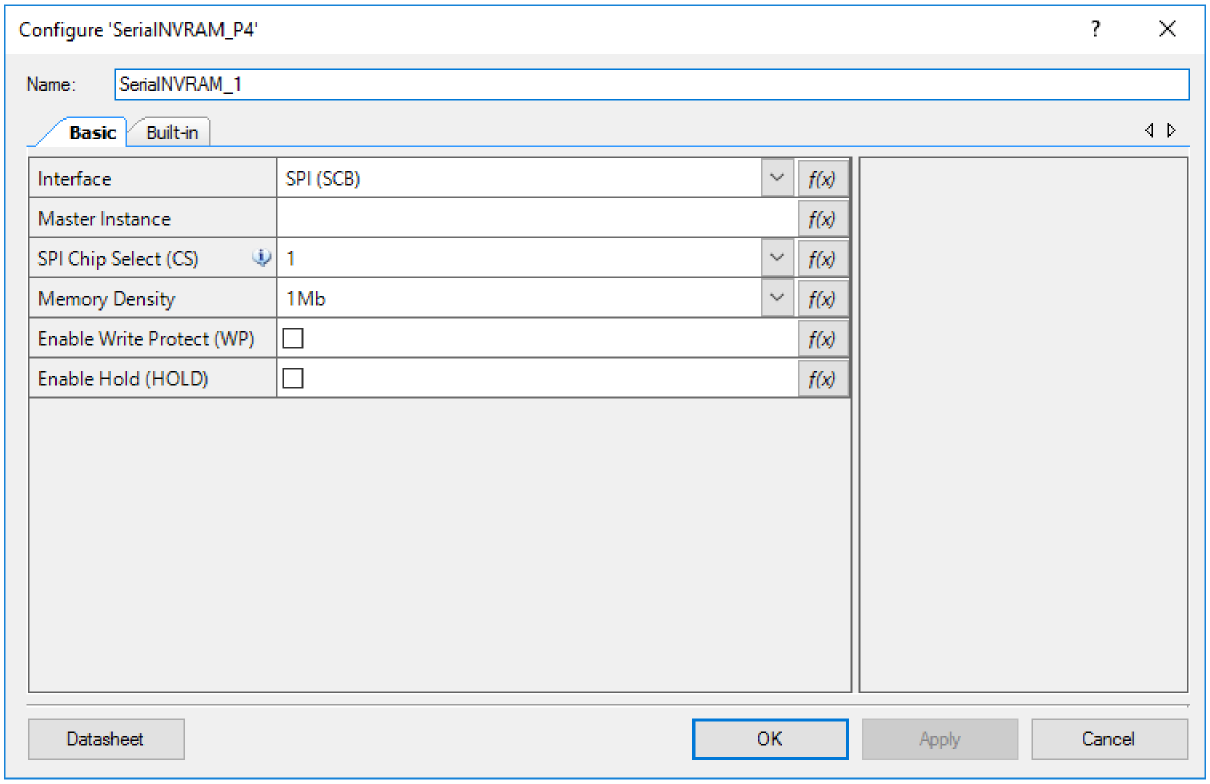
The I^{2}C Master Component has scl and sda pins.

Component Parameters

The SerialNVRAM Component Configure dialog allows editing configuration parameters for the Component instance.

Basic Tab

This tab contains the Component parameters used in the general peripheral initialization settings.



Name: SerialNVRAM_1

Interface	SPI (SCB)
Master Instance	
SPI Chip Select (CS)	1
Memory Density	1Mb
Enable Write Protect (WP)	☐
Enable Hold (HOLD)	☐

Parameter Name	Description
Interface	This parameter selects the Serial NVRAM interface: either I^{2}C (SCB) or SPI (SCB). Based on the selection, PSoC Creator uses the PSoC 4 SCB resource to configure the selected interface. Default is SPI (SCB).
Master Instance	This parameter is set to the instance name of the SPI / I^{2}C Master Component name. An SPI or I^{2}C Master must reside in the schematic for this Component to operate. It uses this name to access the SPI or I^{2}C Master APIs. The NVRAM must also be connected to the SPI / I^{2}C bus controlled by the SPI / I^{2}C Master Component. Allowed characters are A-Z, 0-9, _. The instance name cannot start or end with an underscore. Default is SPI_1 Master.

Parameter Name	Description
SPI Chip Select (CS)	This option is available to select the number of the SPI NVRAM slave connected to the bus. Default is 1 Chip Select. The range is 1-4. This parameter is not visible when the interface selected is I^{2}C (SCB) **Note** The SPI Master Component has its own SS pins, which are not used in the Serial NVRAM Component. These SS pins can be disabled if they are not used for other purposes.
Density	The current NVRAM portfolio supports the following density: • SPI F-RAM - 4Kb to 4Mb • SPI nvSRAM - 64Kb to 1Mb • I^{2}C F-RAM - 4Kb to 1Mb • I^{2}C nvSRAM - 64Kb to 1Mb. The default density – 1Mb. Available values: 4Kb, 16Kb, 64Kb, 128Kb, 256Kb, 512Kb, 1Mb, 2Mb, 4Mb, 8Mb.
Enable Write Protect	Enables the Write Protect pin. The default level is HIGH for SPI and LOW for I^{2}C. Default is the Write protect disabled.
Enable Hold	Enables the HOLD pin. Default is the HOLD control disabled.

Application Programming Interface

Application Programming Interface (API) routines allow configuring the Component usage software.

By default, PSoC Creator assigns the instance name SerialNVRAM_1 to the first instance of a Component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol.

Component Functions

Functions are generated during the build process and are all prefixed with the name of the Component instance.

void SerialNVRAM_1_Init (void)

The initialization routine for the NVRAM Component. Initializes the SPI/I^{2}C block, CS, WP, and Hold configurations.

void SerialNVRAM_1_Start (void)

The start function, initializes the SerialNVRAM with the default values.

void SerialNVRAM_1_Stop (void)

Disables the Serial NVRAM Component but does not stop the Master Component.

SerialNVRAM_1_status SerialNVRAM_1_GetStatus (uint8 *nvramId*)

Returns the external device ready/busy status after the initiated transfer is complete.

Note:

The function always returns SerialNVRAM_1_SUCCESS for F-RAM devices.

Parameters:

nvramId	CS for SPI, the slave address for I²C. The nvramId parameter for I²C devices have the following I²C slave address: 0 1 0 1 0 A2 A1 A0. The three LSB bits are configurable slave address inputs A2, A1, A0. Some I²C NVRAMs use three LSB bits as memory page select to address higher density than addressable through dedicated address byte (s) (one or two bytes) sent after the I²C command byte. For example 4 Kb and 16 Kb I²C F-RAMs use A0 and A2, A1,A0 bits, respectively, as page select bits followed by 1-byte memory address. Similarly, 1 Mb devices use A0 as page select address bit followed by 2-byte memory address. The I²C NVRAM devices internally ignore the page select bit setting and always send an ACK during the command cycle. However, page select bits are internally used to set the memory page address for the subsequent access. For example: the CY14C101I device has Pin A2 connected to the high level and Pin A1 connected to the ground. This corresponds to valid nvramId = 0xA4 or 0xA5 with lower or upper page address set.

Returns:

status:

SerialNVRAM_1_SUCCESS - data is sent/received and the Component is ready to send/receive new data

SerialNVRAM_1_DEVICE_BUSY - the Component is sending/receiving previous data or the device error.

SerialNVRAM_1_status SerialNVRAM_1_MemoryWrite (uint8 *nvramId*, uint32 *addr*, const uint8 **dataWritePtr*, uint32 *totalDataCount*)

Writes the totalDataCount number of data into NVRAM.

Parameters:

nvramId	CS for SPI, the slave address for I²C. For I²C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
addr	The 1/2/3-byte NVRAM address for Write.
**dataWritePtr*	The pointer to an array of data bytes to be written.
totalDataCount	The number of data bytes to be written.

Returns:

Error status:

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_MemoryRead (uint8 *nvramId*, uint32 *addr*, uint8 **dataReadPtr*, uint32 *totalDataCount*)

Reads the totalDataCount number of data from NVRAM.

Parameters:

nvramId	CS for SPI, the slave address for I^{2}C. For I^{2}C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
addr	The 1/2/3-byte NVRAM address for Write.
**dataReadPtr*	The pointer to an array for storing data bytes.
totalDataCount	The number of data bytes to be read.

Returns:

Error status:

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_CurrentMemoryRead (uint8 *nvramId*, uint8 **dataReadPtr*, uint32 *totalDataCount*)

Reads totalDataCount number of the data current address of I^{2}C NVRAM.

Parameters:

nvramId	The slave address for I^{2}C. For I^{2}C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
**dataReadPtr*	The pointer to an array for storing data bytes.
totalDataCount	The number of data bytes to be read.

Returns:

error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_COMMUNICATION_ERROR - A Communication error.

SerialNVRAM_1_status SerialNVRAM_1_MemoryFastReadOpcode (uint8 *nvramId*, uint32 *addr*, uint8 **dataReadPtr*, uint32 *totalDataCount*)

Reads the totalDataCount number of data from SPI NVRAM using the Fast Read command.

Parameters:

nvramId	CS for SPI
addr	The 1/2/3-byte NVRAM address for Write.
**dataReadPtr*	The pointer to an array for storing data bytes.
totalDataCount	The number of data bytes to be read.

Returns:

Error status:

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_StatusRegWrite (uint8 *nvramId*, uint8 *dataByte*)

Writes the NVRAM Status register.

Parameters:

nvramId	CS for SPI.
dataByte	The 1-byte status register data to be written.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_StatusRegRead (uint8 *nvramId*, uint8 **dataByte*)

Reads the NVRAM Status register.

Parameters:

nvramId	CS for SPI
**dataByte*	The pointer to the holding status register value.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_MemCtrlRegWrite (uint8 *nvramId*, uint8 *dataByte*)

Writes to NVRAM Memory Control Register.

Parameters:

nvramId	The slave address for I^2C. For I^2C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
dataByte	The 1-byte Memory Control register data to be written.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_MemCtrlRegRead (uint8 *nvramId*, uint8 **dataByte*)

Reads the NVRAM Memory Control Register content.

Parameters:

nvramId	The slave address for I²C. For I²C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
**dataByte*	The pointer for holding the Control register value.

Returns:

error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_COMMUNICATION_ERROR - A communication error.

SerialNVRAM_1_status SerialNVRAM_1_Sleep (uint8 *nvramId*)

Sends the NVRAM command for sleep.

Parameters:

nvramId	CS for SPI, the slave address for I²C. For I²C devices see the detailed format in SerialNVRAM_1_GetStatus() description. The nvramId parameter for the FM24VN10 families has the following I²C slave address format: 0 1 0 1 0 A2 A1 A16. Bits A2 and A1 are the slave address inputs. Bit A16 is the page select bit and it can have any value for this function. Four FM24VN10 devices can be connected to I²C bus with the following nvramId parameter values: 0x50 - Both A2 and A1 pins are connected to the ground. 0x52 - Pin A2 connected to the ground and Pin A1 connected to the high level. 0x54 - Pin A2 connected to the high level and Pin A1 connected to the ground. 0x56 – Both A2 and A1 pins are connected to the high level.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_NvCommand (uint8 *nvramId*, uint8 *nvcmd*)

Sends the NVRAM command. This is supported by nvSRAM only.

Parameters:

nvramId	CS for SPI, the slave address for I²C. For I²C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
nvcmd	The 8-bit NVRAM command.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_SerialNoWrite (uint8 *nvramId*, const uint8 **dataPtr*)

Writes the NVRAM Serial number.

Parameters:

nvramId	CS for SPI, the slave address for I²C. For I²C devices see the detailed format in SerialNVRAM_1_GetStatus() description.

Note:

This function is applicable for devices with a serial number in the Control registers map. The SerialNVRAM_1_RtcRegWrite()/SerialNVRAM_1_WriteProcessorCompanion() functions are used for devices with a serial number in the RTC / processor companion register map. For more detail, refer to the device datasheet.

Parameters:

**dataPtr*	The pointer to an array of the serial number data to be written.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_SerialNoRead (uint8 *nvramId*, uint8 **dataPtr*)

Reads the NVRAM Serial number.

Parameters:

nvramId	CS for SPI, the slave address for I²C For I²C devices see the detailed format in SerialNVRAM_1_GetStatus() description. The nvramId parameter for the FM24VN10 families has the following I²C slave address format: 0 1 0 1 0 A2 A1 A16. Bits A2 and A1 are the slave address inputs. Bit A16 is the page select bit and it can have any value for this function. Four FM24VN10 devices can be connected to I²C bus with the following nvramId parameter values: 0x50 - Both A2 and A1 pins are connected to the ground. 0x52 - Pin A2 connected to the ground and Pin A1 connected to the high level. 0x54 - Pin A2 connected to the high level and Pin A1 connected to the ground. 0x56 – Both A2 and A1 pins are connected to the high level.

Note:

This function is applicable for devices with a serial number in the Control registers map. The SerialNVRAM_1_RtcRegRead()/SerialNVRAM_1_ReadProcessorCompanion() functions are used for devices with a serial number in the RTC / processor companion register map. For more detail, refer to the device datasheet.

Parameters:

**dataPtr*	The pointer to an array for storing serial number data.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_DevIdRead (uint8 *nvramId*, uint8 **dataPtr*, uint8 *iDLength*)

Reads the NVRAM device ID. This is supported by nvSRAM and F-RAM above 128Kbit. A pre-defined ID length covers the following varying device's IDs across different product families:

- CY14B101PA - The DevID size is 4 bytes
- FM25VN10 - The DevID size is 9 bytes
- CY15B104Q - The DevID size is 9 bytes
- CY14B101I - The DevID size is 4 bytes
- FM24VN10 - The DevID size is 3 bytes
- FM31L278 - The DevID size is 8 bytes

Parameters:

nvramId	CS for SPI, the slave address for I^{2}C. For I^{2}C devices see the detailed format in SerialNVRAM_1_GetStatus() description. The nvramId parameter for the FM24VN10 families has the following I^{2}C slave address format: 0 1 0 1 0 A2 A1 A16. Bits A2 and A1 are the slave address inputs. Bit A16 is the page select bit and it can have any value for this function. Four FM24VN10 devices can be connected to I^{2}C bus with the following nvramId parameter values: 0x50 - Both A2 and A1 pins are connected to the ground. 0x52 - Pin A2 connected to the ground and Pin A1 connected to the high level. 0x54 - Pin A2 connected to the high level and Pin A1 connected to the ground. 0x56 – Both A2 and A1 pins are connected to the high level.
**dataPtr*	The pointer to an array for storing the device ID.
IDLength	The four predefines for the ID length (3ByteDeviceID, 4ByteDeviceID, 8ByteDeviceID, 9ByteDeviceID).

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_RtcRegWrite (uint8 *nvramId*, uint32 *addr*, const uint8 **dataWritePtr*, uint32 *totalDataCount*)

Writes the totalDataCount number of data into NVRAM RTC / F-RAM Processor Companion registers.

Note:

To write the RTC register, this function should set the W bit of the Flags register (SerialNVRAM_1_RTC_FLAGS) and then write the RTC register.

Parameters:

nvramId	CS for SPI, the slave address for I^{2}C. For I^{2}C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
addr	The 8-bit NVRAM RTC / F-RAM Proc companion register address for Write.
**dataWritePtr*	The pointer to an array of RTC data bytes to be written.
totalDataCount	The number of RTC data bytes to be written.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_RtcRegRead (uint8 *nvramId*, uint32 *addr*, uint8 **dataReadPtr*, uint32 *totalDataCount*)

Reads the totalDataCount number of data from nvSRAM RTC / F-RAM Processor Companion registers.

Note:

To read the data from the RTC register of the companion device, this function should set the R bit of the Flags register (SerialNVRAM_1_RTC_FLAGS) and then read the RTC register.

Parameters:

nvramId	CS for SPI, the slave address for I^{2}C For I^{2}C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
addr	The 8-bit nvSRAM RTC / F-RAM Proc companion register address for read.
**dataReadPtr*	The pointer to an array for storing RTC data bytes.
totalDataCount	The number of RTC data bytes to be read.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_CurrentRtcRegRead (uint8 *nvramId*, uint8 **dataReadPtr*, uint32 *totalDataCount*)

Reads the current totalDataCount number of data from nvSRAM RTC / F-RAM Processor Companion registers.

Parameters:

nvramId	The slave address for I^{2}C. For I^{2}C devices see the detailed format in SerialNVRAM_1_GetStatus() description.
**dataReadPtr*	The pointer to an array for storing RTC data bytes.
totalDataCount	The number of RTC data bytes to be read.

Returns:

error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_COMMUNICATION_ERROR - A communication error.

void SerialNVRAM_1_SetWp (uint8 *value*)

Sets the WP pin to HIGH or LOW.

Parameters:

value	HIGH (1), LOW (0).

Note:

The Active LOW level for SPI NVRAM and Active HIGH level for I²C NVRAM prevent write operation to the memory and the Status register. Refer to the device datasheet for the choices about the software and hardware write protection.

void SerialNVRAM_1_SetHold (uint8 *value*)

Sets the HOLD pin to HIGH or LOW.

Parameters:

value	HIGH (1), LOW (0).

SerialNVRAM_1_status SerialNVRAM_1_WriteProcessorCompanion (uint8 *nvramId*, uint32 *addr*, const uint8 **dataPtr*, uint32 *totalDataCount*)

Writes the SPI F-RAM processor companion register. This function is applicable for the devices which support WRPC and RDPC. Refer to the device datasheet for details.

Parameters:

nvramId	CS for SPI.
addr	The 8-bit F-RAM Processor companion register address for Write.
**dataPtr*	The pointer to an array of data to be written.
totalDataCount	The number of data bytes to be written.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

SerialNVRAM_1_status SerialNVRAM_1_ReadProcessorCompanion (uint8 *nvramId*, uint32 *addr*, const uint8 **dataReadPtr*, uint32 *totalDataCount*)

Reads the SPI F-RAM processor companion register. This function is applicable for the devices which support WRPC and RDPC. Refer to the device datasheet for details.

Parameters:

nvramId	CS for SPI.
addr	The 8-bit F-RAM Processor companion register address for Read.
**dataReadPtr*	The pointer to an array for storing data bytes.
totalDataCount	The number of data bytes to be written.

Returns:

Error status

SerialNVRAM_1_SUCCESS - No errors.

SerialNVRAM_1_DEVICE_ERROR - The wrong device or a device error.

SerialNVRAM_1_TIMEOUT_ERROR - The device does not respond or SerialNVRAM_1_SPI_COM_TIMEOUT elapsed.

Global Variables

The SerialNVRAM Component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the Component (e.g., *SerialNVRAM_1.c*). Each variable is also prefixed with the instance name of the Component.

uint8 SerialNVRAM_1_initVar

SerialNVRAM_1_initVar indicates whether the SerialNVRAM_1 Component has been initialized. The variable is initialized to 0 and set to 1 the first time SerialNVRAM_1_Start() is called. This allows the Component to restart without reinitialization after the first call to the SerialNVRAM_1_Start() routine.

If re-initialization of the Component is required, then the SerialNVRAM_1_Init() function can be called before the SerialNVRAM_1_Start() function.

Enumeration Type

enum SerialNVRAM_1_status

The error codes

Enumerator

SerialNVRAM_1_SUCCESS	Successful
SerialNVRAM_1_DEVICE_BUSY	The device is busy
SerialNVRAM_1_DEVICE_ERROR	A device error
SerialNVRAM_1_TIMEOUT_ERROR	Operation has timed out
SerialNVRAM_1_COMMUNICATION_ERROR	A communication error

Code Examples and Application Notes

This section lists the projects that demonstrate the Component usage.

Code Examples

PSoC Creator provides access to code examples in the Code Example dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or the **File** menu. **Filter Options** in the dialog are used to narrow the list of projects available to select.

Refer to the "Code Example" topic in the PSoC Creator Help for more information.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

Application Notes

Cypress provides a number of application notes describing how PSoC can be integrated into your design. You can access the Cypress Application Notes search web page at www.cypress.com/appnotes. The Application Notes that use this Component:

- AN64574 - Designing with Serial Peripheral Interface (SPI) nvSRAM

- AN68174 - Migrating from Serial Peripheral Interface (SPI) EEPROM to SPI nvSRAM

API Memory Usage

The Component memory usage varies significantly depending on the compiler, device, number of APIs used and Component configuration. The following table provides the memory usage for all APIs available in the given Component configuration.

The measurements have been done with an associated compiler configured in Release mode with optimization set for Size. For a specific design, the map file generated by the compiler can be analyzed to determine the memory usage.

PSoC 4 (GCC)

Configuration	PSoC 4100/PSoC 4200/ PSoC 4100 BLE/PSoC 4200 BLE	
	Flash Bytes	SRAM Bytes
Default SPI (SCB)	1118	1
Default I^{2}C (SCB)	1092	1

Note Memory usage does not include the SCB Master Component's used memory.

Functional Description

Serial NVRAMs are available with various features/attributes, such as an interface, density, access speed (frequency of operation), as well as RTC/non-RTC. The Serial NVRAM Component allows setting these attributes for a target NVRAM. Some of these attributes can change the command protocol, such as addressing 1-Byte vs. 2-Byte vs. 3-Byte based on the selected density.

The Serial NVRAM Component is targeted at SPI and I^{2}C NVRAMs. It greatly simplifies PSoC designs by allowing you to drag and drop the Serial NVRAM Component from the Component Catalog onto the target schematic. You can call those ready-to-use APIs directly into their firmware program instead of writing your own APIs.

Block Diagram and Configuration

The Serial NVSRAM Component is a wrapper around an SPI or I^{2}C SCB Master Component and has NVRAM-specific additional controls, such as multiple Chip Select (CS), Write Protect (WP), and HOLD (HOLD). The Serial NVSRAM Component enables communication with several NVRAMs. Also, you can share the same SPI or I^{2}C Component for other communication purposes. All SPI or I^{2}C devices can share the same SPI or I^{2}C SCB Master Component.

- For SPI NVRAMs, the default configuration for WP and HOLD are HIGH
- For I^{2}C NVRAMs, the default configuration for WP is LOW. HOLD is N/A.

Figure 1. Interaction of SPI Serial NVRAM and SPI Master Components

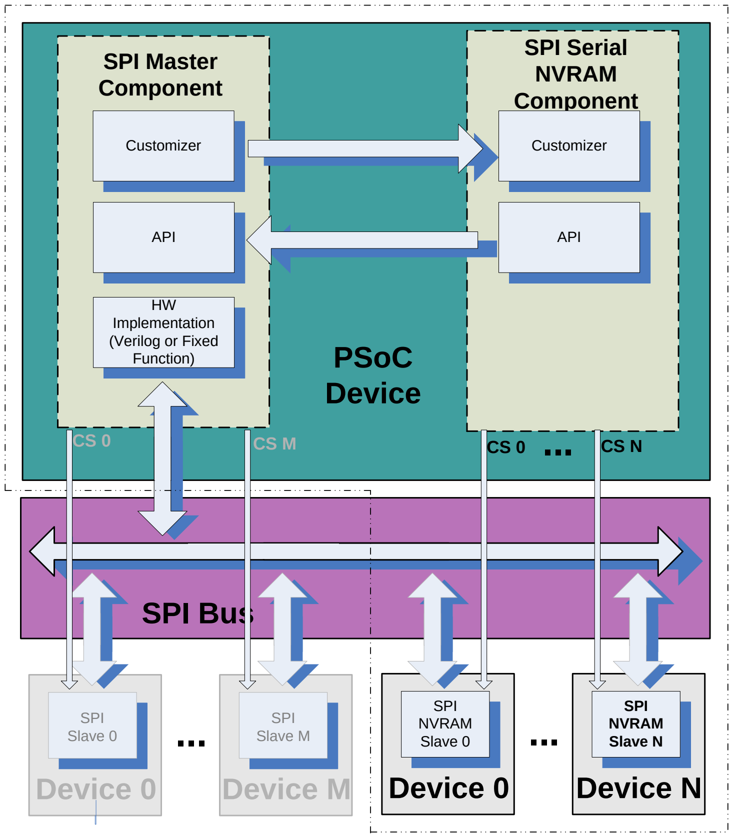


Figure 2. Interaction of I²C Serial NVRAM and I²C Master Components

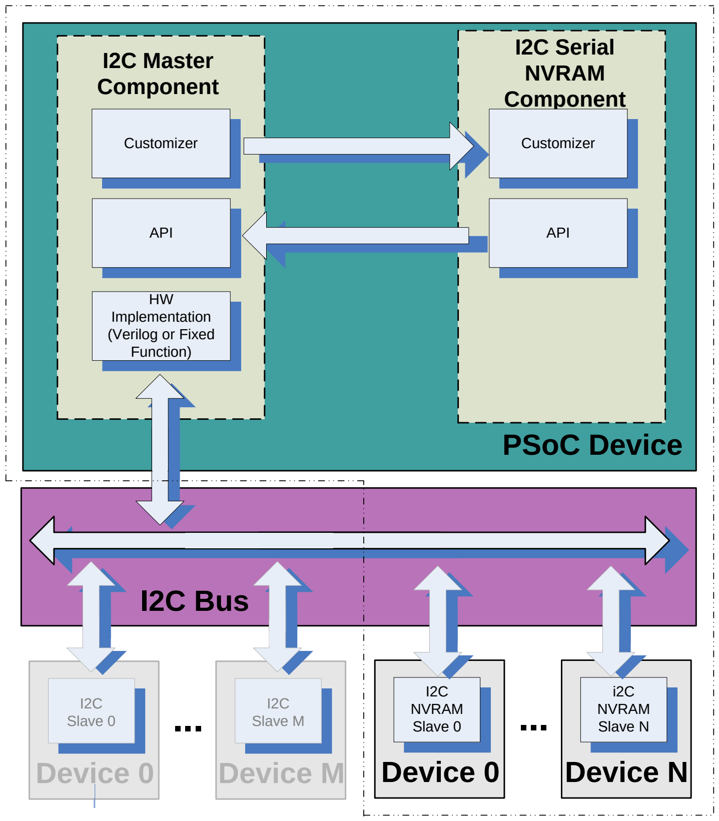


Figure 3 through Figure 6 show various use cases for interfacing Serial NVRAM to a PSoC 4 device.

Figure 3. SPI NVRAM Interface to PSoC 4 – With HOLD and WP Control

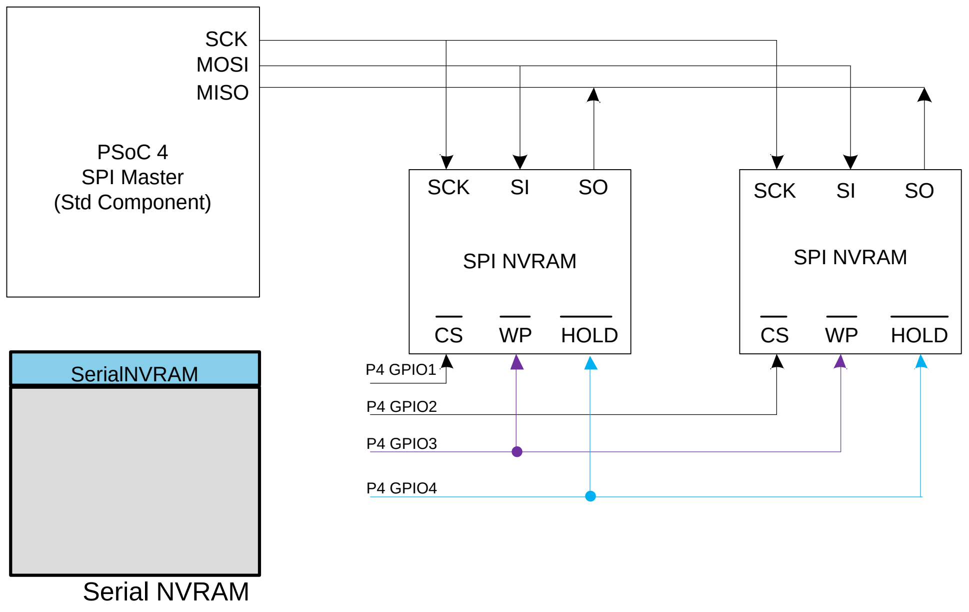


Figure 4. SPI NVRAM Interface to PSoC 4 – Without HOLD and WP Control

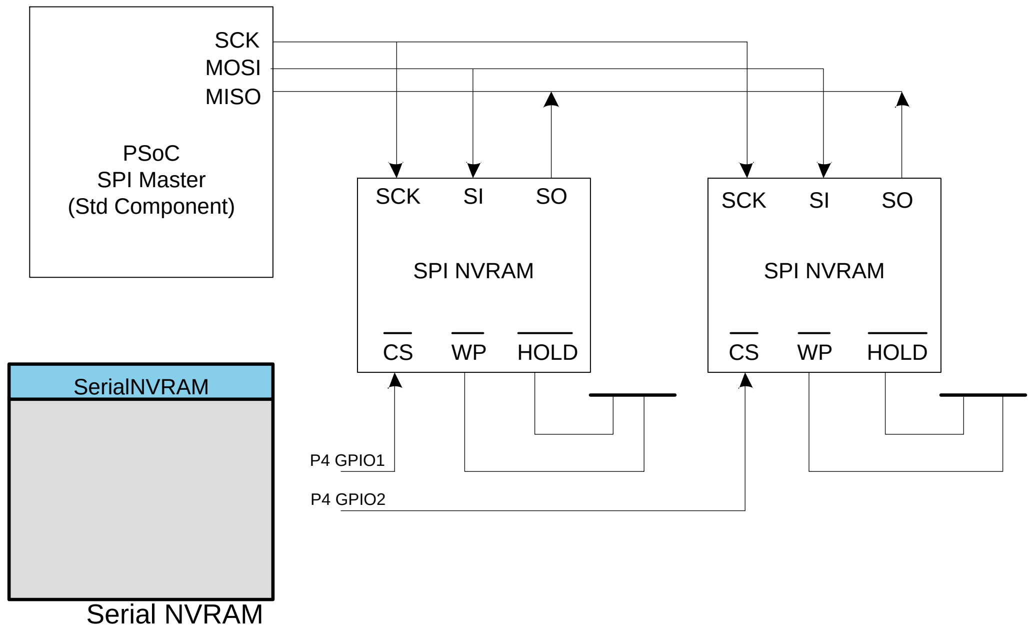


Figure 5 shows a typical use case for interfacing I^{2}C NVRAM to I^{2}C Master PSoC.

Figure 5. I^{2}C NVRAM Interface to PSoC 4 – With WP Control

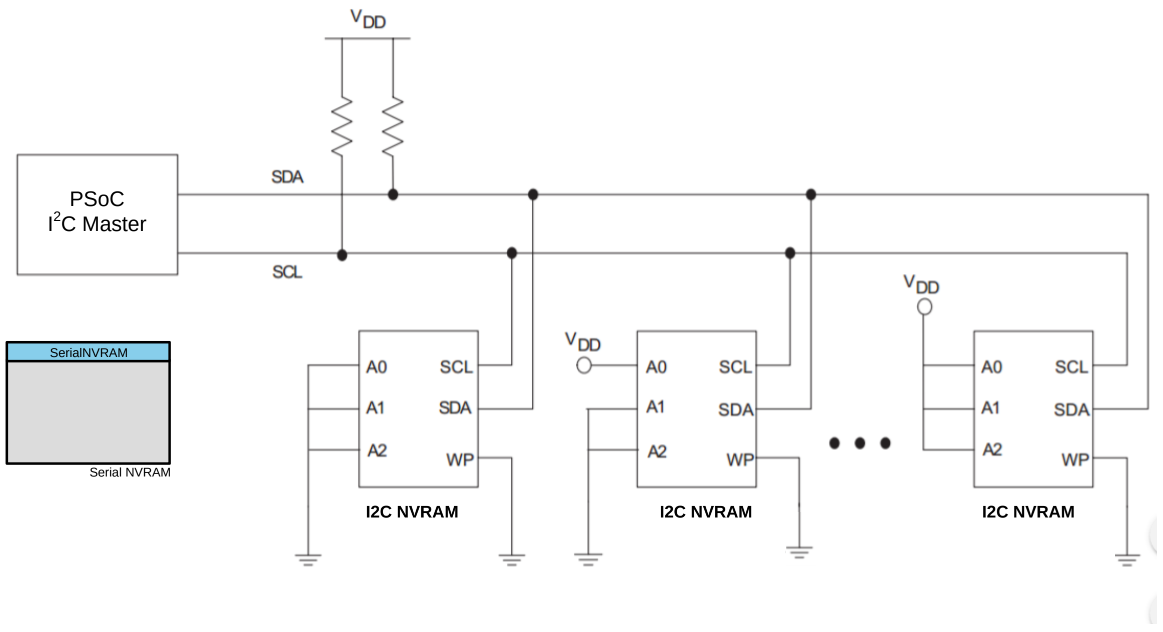
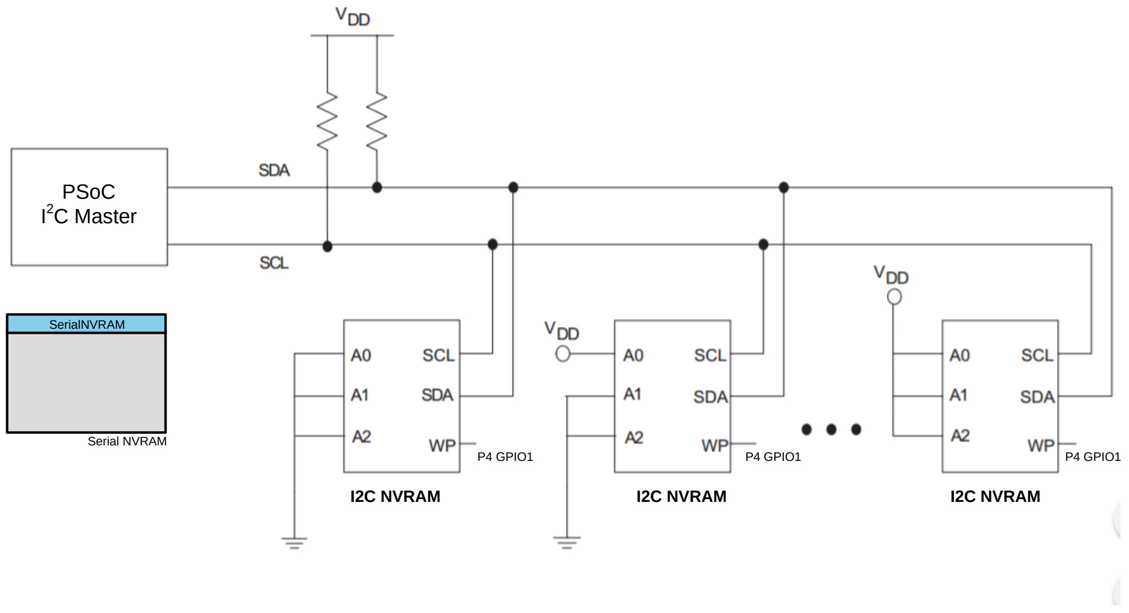


Figure 6. I^{2}C NVRAM Interface to PSoC 4 – Without WP Control



The Serial NVRAM Component supports the I^{2}C slave address A[2:0] configuration through GPIOs. You must configure these hardware pins as appropriate in multi-slave configuration.

Placement

The SerialNVRAM Component is placed as a Fixed Function block. All placement information is provided to the API through the *cyfitter.h* file. SerialNVRAM pins placement information is available in the Pin Editor. Refer to the device datasheet Pinout section for the pins functions and placement information.

Generally, debug pins have an alternate function to be SerialNVRAM pins. To use these pins for the SerialNVRAM functionality, disable the debug capability in the System Editor. Set the **Debug Select** option to GPIO to allow the tool to use an alternate function.

Note The debugger will not be functional after choosing this option.

This is specifically important for **PSoC 4000** devices that have a limited number of pins. The debug pins for these devices are P3[0] (SWD_IO) and P3[1] (SWD_CLK). The alternate function of these pins is SerialNVRAM functionality SCB[0] (SDA) and SCB[0] (SCL). For other devices, refer to the appropriate device datasheet to determine alternate functions of the debug pins.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations:

- project deviations – applicable for all PSoC Creator Components
- specific deviations – applicable only for this Component

This section provides information on the Component-specific deviations. The project deviations are described in the *MISRA Compliance* section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The Component does not have MISRA violations.

Registers

See the *Serial Communications Block (SCB) Registers* section in the chip Technical Reference Manual (TRM) for more information about the registers.

Resources

The SerialNVRAM Component uses the Serial Communications Block (SCB) peripheral block.

DC and AC Electrical Characteristics

Refer to the SPI and I^{2}C tables in the DC and *AC Electrical Characteristics* section of the PSoC 4 Serial Communication Block (SCB) Component datasheet.

References

- nvSRAM (Nonvolatile SRAM)
- F-RAM (Nonvolatile Ferroelectric RAM)

Component Changes

This section lists the major changes in the Component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.0	Initial Version	

© Cypress Semiconductor Corporation, 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.