Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE204087 – Interfacing SPI nvRAM with PSoC(R) 3 and PSoC 5LP

Objective

This code example demonstrates the interface of SPI nvRAM device (F-RAM and nvSRAM) with PSoC®3 and PSoC 5LP controller.

Overview

The code example provides a nvRAM User Component, designed specifically for SPI F-RAM/nvSRAM. The User Component is configurable for different density F-RAM/nvSRAM and different frequency. The user component is imported into any PSoC 3 and PSoC 5LP based design and usage of supported APIs is shown.

Requirements

Tool: PSoC Creator™ 3.3

Programming Language: C (GCC 4.9)

Associated Parts: All PSoC 3 and PSoC 5LP parts

Related Hardware: CY8CKIT-001

Design

The code example implements the SPI nvRAM User Component with APIs to access SPI F-RAM/nvSRAM. The APIs include F-RAM/nvSRAM read/writes, status register read/writes, device id read, and RTC APIs.

Design Considerations

The SPI NVRAM User Component is designed with PSoC 3 and PSoC 5LP. Serial memories can run up to 40 MHz but in this example project, it is limited by the PSoC controller.

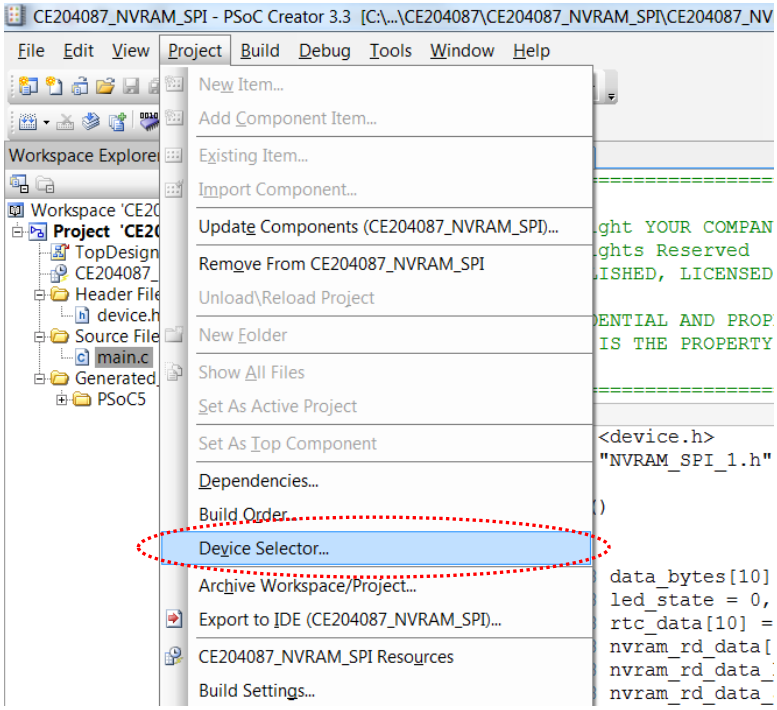
Hardware Setup

Hardware setup involves interfacing F-RAM/nvSRAM board with generic PSoC kit CY8CKIT-001. The hardware setup is not limited to CY8CKIT-001 kit. Since the connection is made through external wires, any PSoC 3 and PSoC 5LP kit along with any F-RAM/nvSRAM board can be used with proper pin assignments.

Software Setup

There are no specific software required for this code example. The code example is applicable to both PSoC 3 and PSoC 5LP. The selection of the PSoC device can be made in PSoC Creator under **Project** > **Device Selector…**.

Figure 1. PSoC Device Selection



Components

The code example uses the SPI nvRAM User Component. Table 1 lists the PSoC Creator Components and hardware resources by the SPI nvRAM User Component.

Table 1. List of PSoC Creator Components

Component	Hardware Resources
Clock	Clock
SPI Mater	1 Data path cells, 13 Macrocells, 3 Status Cells, 1 Control Cell and 2 interrupts
Digital Input Pins	P5[3]
Digital Output Pins	P5[0], P5[1], P5[2], P5[5]

Parameter Settings

Table 2. SPI nvRAM User Module Configuration

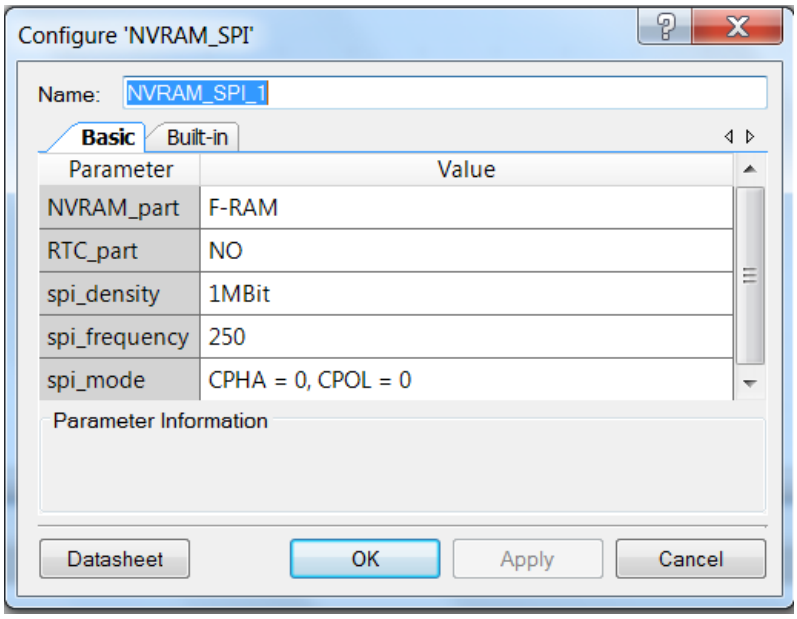


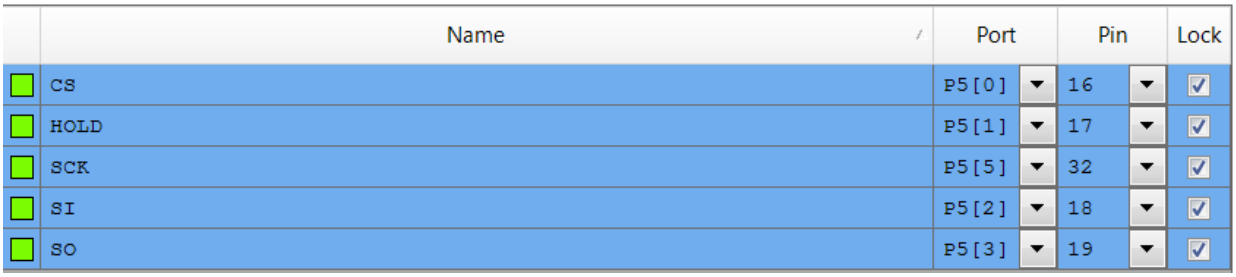
Table 3 lists the parameter settings of the components used by the SPI nvRAM User Component.

Table 3. Parameter Settings

Component	Non-default Parameter Settings
Clock	Frequency: 500 kHz
SPI Mater	Use default setting Mode: CPHA = 0, CPOL = 0 Data Lines: MOSI + MISO Data Bits: 8 Shift Direction: MSB First Bit rate: ½ Input clock frequency Clock Selection: External Clock Rx Buffer Size (8-bit words): 4 Tx Buffer Size (8-bit words): 4
Digital Input Pins	Checked boxes: Digital Output, Hardware Connection Drive Mode: Strong Drive
Digital Output Pins	Checked boxes: Digital Output, Hardware Connection Drive Mode: High Impedance Digital

Design-Wide Resources

Figure 2. Pin Assignments



Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. Table 4 lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name “NVRAM_SPI_1” to the first instance of a component in each design. You can rename the instance 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. For readability, the instance name used in the following table is “NVRAM_SPI”.

Table 4. List of APIs for NVSRAM_SPI User Component

Function	Description	Applicability (nvSRAM/F-RAM)
NVRAM_SPI_Init	Initialization routine for NVRAM_SPI component	nvSRAM and F-RAM
NVRAM_SPI_Write	SPI NVRAM SRAM Write	nvSRAM and F-RAM
NVRAM_SPI_Read	SPI NVRAM SRAM Read	nvSRAM and F-RAM
NVRAM_SPI_RTC_Write	SPI NVRAM RTC Write	nvSRAM and F-RAM Processor companion
NVRAM_SPI_RTC_Read	SPI NVRAM RTC Read	nvSRAM and F-RAM Processor companion
NVRAM_SPI_nvCommand	Send NVRAM Command	nvSRAM only
NVRAM_SPI_Sleep	NVRAM Sleep mode entry	nvSRAM and F-RAM
NVRAM_SPI_Status_Reg_Read	NVRAM Status Register Read	nvSRAM and F-RAM
NVRAM_SPI_Status_Reg_Write	NVRAM Status Register Write	nvSRAM and F-RAM
NVRAM_SPI_Serial_No_Write	NVRAM Serial Number Write	All nvSRAM and F-RAM processor companions
NVRAM_SPI_Serial_No_Read	NVRAM Serial Number Read	All nvSRAM and F-RAM 1-MBit and F-RAM Processor companions
NVRAM_SPI_Device_ID_Read	NVRAM Device ID Read	All nvSRAM and F-RAMs above 128Kbit

void NVRAM_SPI_Init (void)

Description: Initialization routine for NVRAM_SPI Component. Initializes the SPI block, CS and Hold pins.

Parameters: None

Return Value: None

Side Effects: None

uint8 NVRAM_SPI_Write (uint32 addr, uint8 *data_write_ptr, uint32 total_data_count)

Description: Write total_data_count number of data into NVRAM.

Parameters: uint32 addr: 32 bit NVRAM address for write.

uint8 *data_write_ptr: Pointer to an array of data bytes to be written.

uint32 total_data_count: Number of data bytes to be written.

Return Value: uint8: Returns the SPI Communication Status

0: SPI Communication Error

1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_Read (uint32 addr, uint8 *data_read_ptr, uint32 total_data_count)

Description: Read total_data_count number of data from NVRAM.

Parameters: uint32 addr: 32 bit NVRAM address for read.

uint8 *data_read_ptr: Pointer to an array for storing data bytes.

uint32 total_data_count: Number of data bytes to be read.

Return Value: uint8: Returns the SPI Communication Status

0: SPI Communication Error

1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_RTC_Write (uint8 addr, uint8 *data_write_ptr, uint8 total_data_count)

Description: Write total_data_count number of data into NVRAM RTC / F-RAM Processor Companion registers.

Parameters: uint8 addr: 8 bit NVRAM RTC / F-RAM Proc companion register address for write.

uint8 *data_write_ptr: Pointer to an array of RTC data bytes to be written.

uint8 total_data_count: Number of RTC data bytes to be written.

Return Value: uint8: Returns the SPI Communication Status

0: SPI Communication Error

1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_RTC_Read (uint8 addr, uint8 *data_read_ptr, uint8 total_data_count)

Description: Read total_data_count number of data from nvSRAM RTC / F-RAM Processor Companion registers.

Parameters: uint8 addr: 8 bit nvSRAM RTC / F-RAM Proc companion register address for read.

uint8 *data_read_ptr: Pointer to an array for storing RTC data bytes.

uint8 total_data_count: Number of RTC data bytes to be read.

Return Value: uint8: Returns the SPI Communication Status

0: SPI Communication Error

1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_ nvCommand (uint8 nvcmd)

Description: Send NVRAM Command. This is supported by nvSRAM only.

Parameters: uint8 nvcmd: 8 bit NVRAM Command

Return Value: uint8: Returns the SPI Communication Status

0: SPI Communication Error

1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_ Sleep (void)

Description: Send NVRAM Command for Sleep.

Parameters: None

Return Value: uint8: Returns the SPI Communication Status
0: SPI Communication Error
1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_ Status_Reg_Read (uint8 *data_byte)

Description: NVRAM Status Register Read

Parameters: uint8 *data_byte: Pointer to status register data to be written

Return Value: uint8: Returns the SPI Communication Status

0: SPI Communication Error

1: SPI Communication Success

Side Effects: None

void NVRAM_SPI_ Status_Reg_Write (uint8 data_byte)

Description: NVRAM Status Register Write

Parameters: uint8 data_byte: 1 byte Status register data to be written

Return Value: uint8: Returns the SPI Communication Status

0: SPI Communication Error

1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_ Serial_No_Write (uint8 *data_ptr)

Description: NVRAM Serial Number Write. This is supported by nvSRAM and F-RAM Processor companions only.

Parameters: uint8 *data_ptr: Pointer to an array of serial number data to be written.

Return Value: uint8: Returns the SPI Communication Status
0: SPI Communication Error
1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_ Serial_No_Read (uint8 *data_ptr)

Description: NVRAM Serial Number Read. This is supported by nvSRAM, 1-MBit F-RAM and F-RAM processor companions only.

Parameters: uint8 *data_ptr: Pointer to an array for storing serial number data.

Return Value: uint8: Returns the SPI Communication Status
0: SPI Communication Error
1: SPI Communication Success

Side Effects: None

uint8 NVRAM_SPI_ Device_ID_Read (uint8 *data_ptr)

Description: NVRAM Device ID Read. This is supported by nvSRAM and F-RAM above 128Kbit.

Parameters: uint8 *data_ptr: Pointer to an array for storing device id.

Return Value: uint8: Returns the SPI Communication Status
0: SPI Communication Error
1: SPI Communication Success

Side Effects: None

Operation

This section shows how to import the SPI NVRAM User Component into the code example and how to use the APIs of the user component.

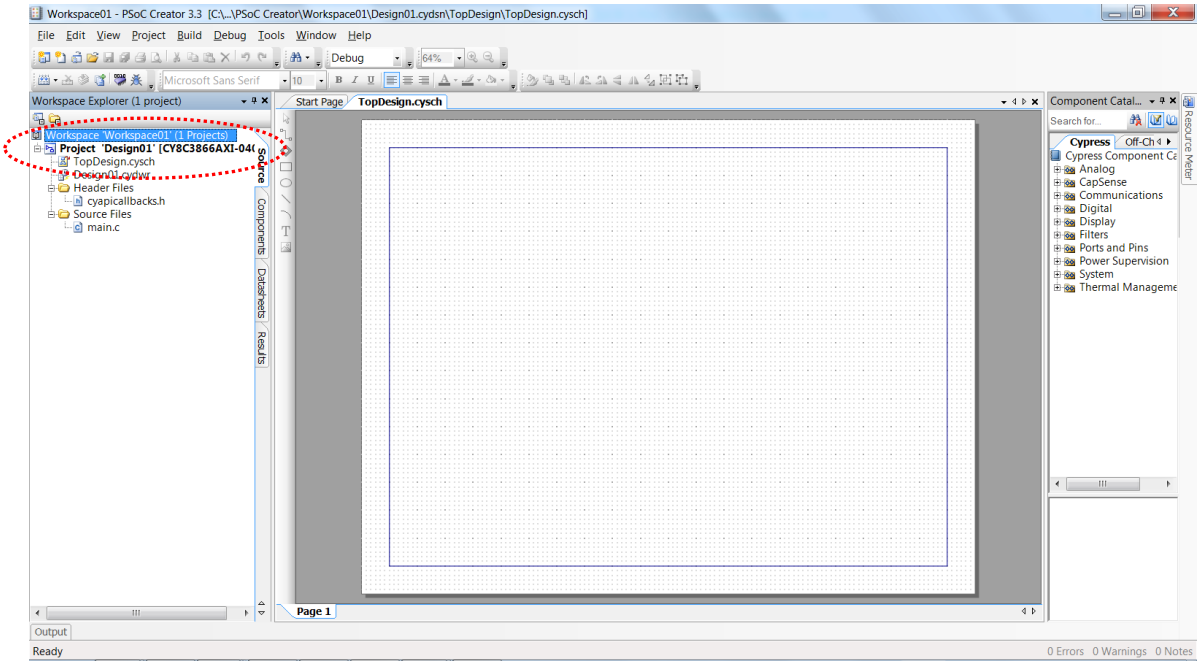
Setup

NVRAM_SPI archive contains both the example project and NVRAM_SPI Component.

Follow these steps to use the NVRAM_SPI Component in your design:

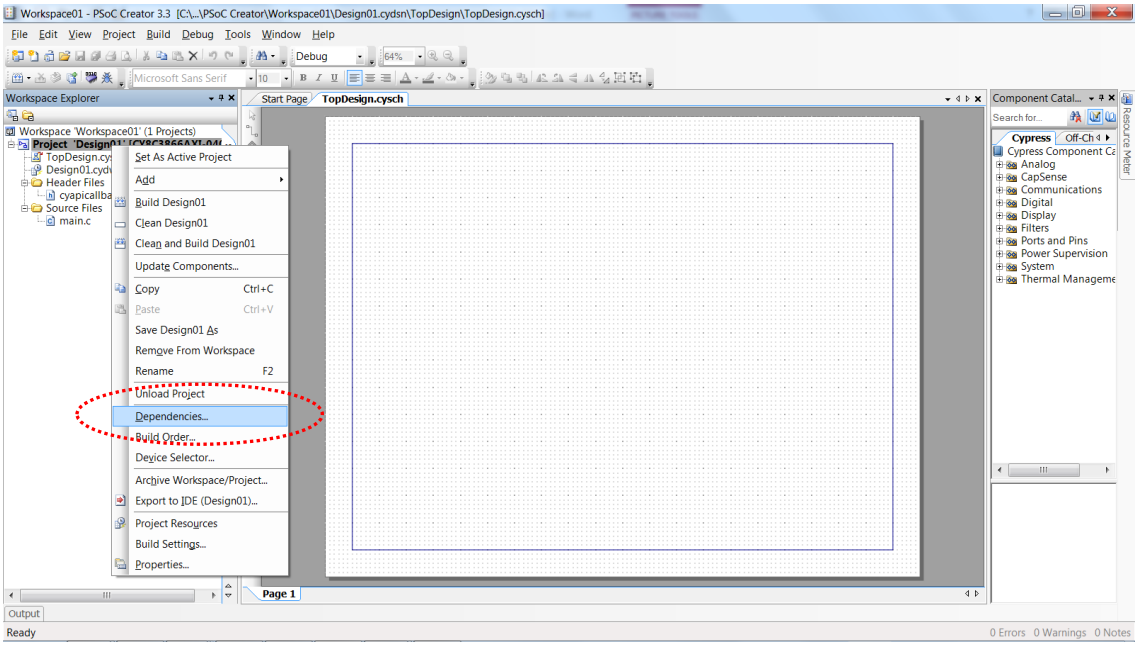
1. Open PSoC Creator and open your design (workspace) as shown in Figure 3. Create new project ‘Design01’.

Figure 3. Create project ‘Design01’



2. Right-click the project and click **Dependencies**. In the **Dependencies** tab on the workspace explorer bring the NVRAM_SPI Component into your design, as shown in Figure 4.

Figure 4. Open Dependencies Tab



3. Click **New Entry (User Dependencies)** and then select *CE204087.cyprj* from the *CE204087.cydsn* folder (see Figure 5). NVRAM_SPI Component appears under NVRAM/NVRAM_SPI in component Catalog (see Figure 6).

Figure 5. Importing User Component

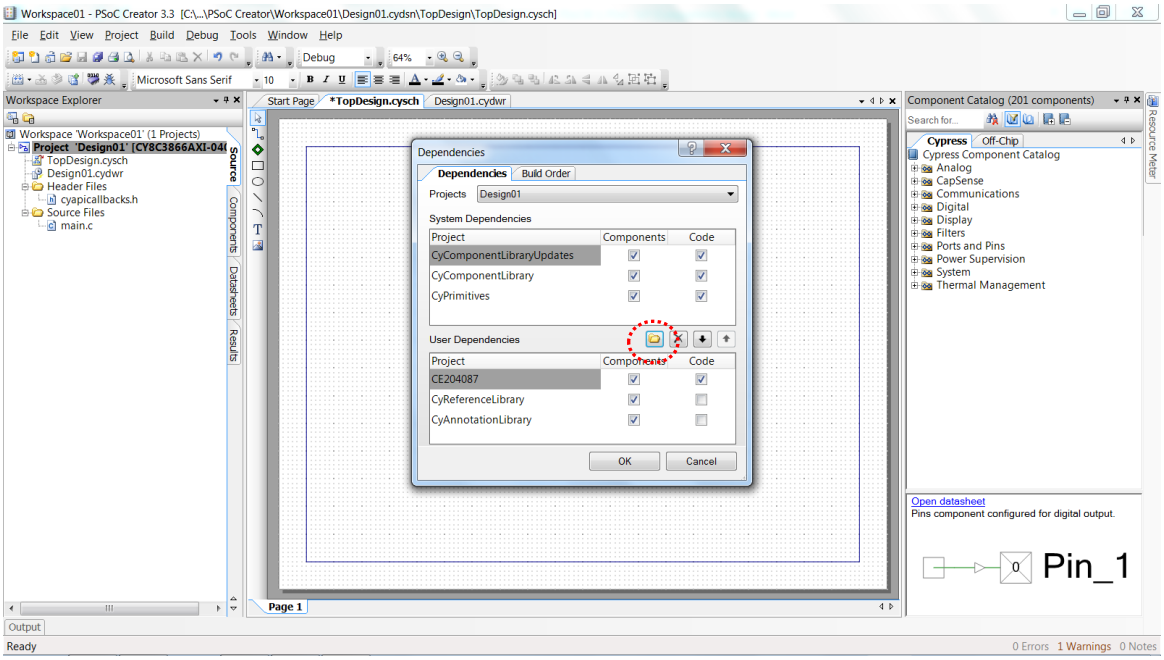
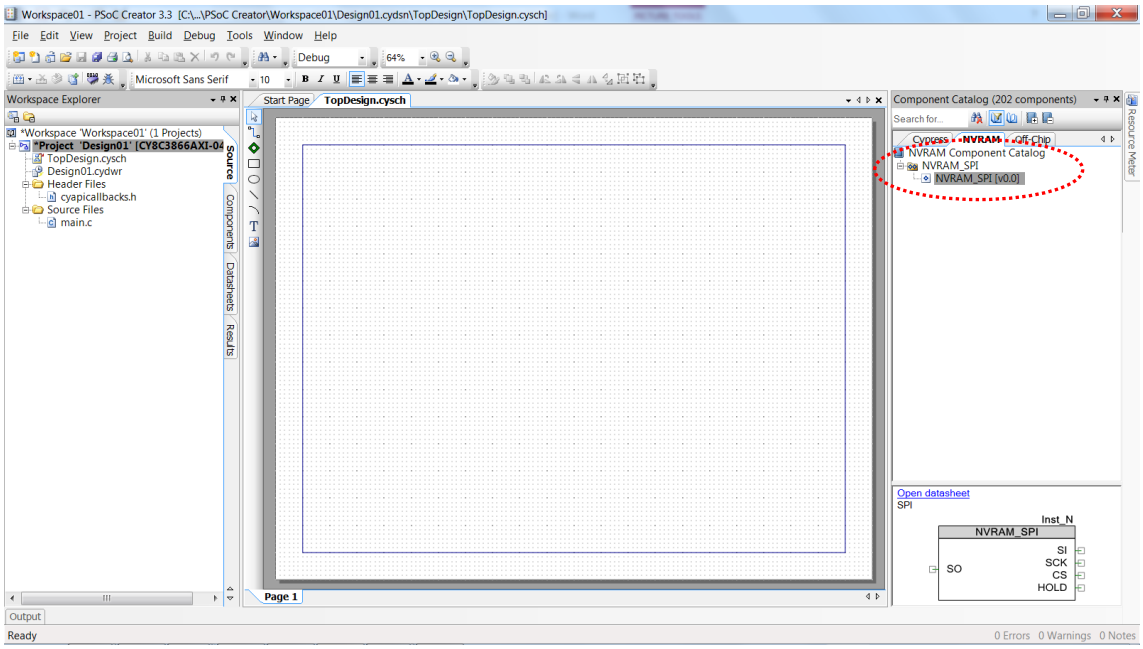
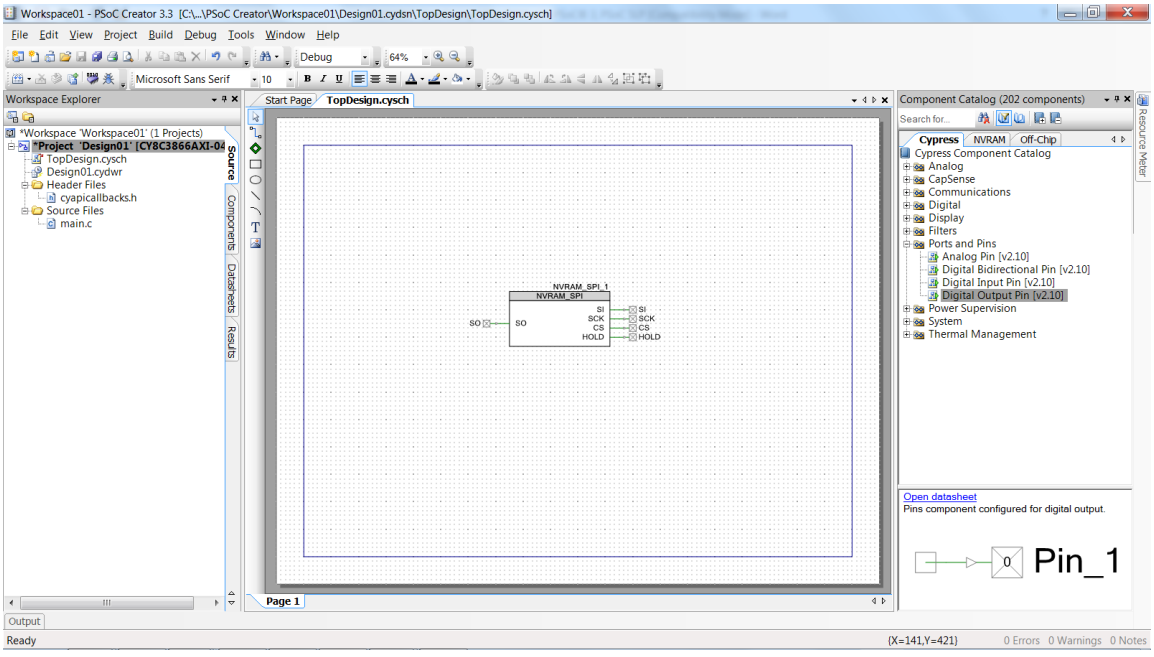


Figure 6. NVRAM_SPI Component under Component Catalog



4. To add NVRAM_SPI component to your design, drag and drop the NVRAM_SPI Component onto *TopDesign.cysch* and assign Digital IOs from **Ports and Pins** component, as shown in Figure 7.

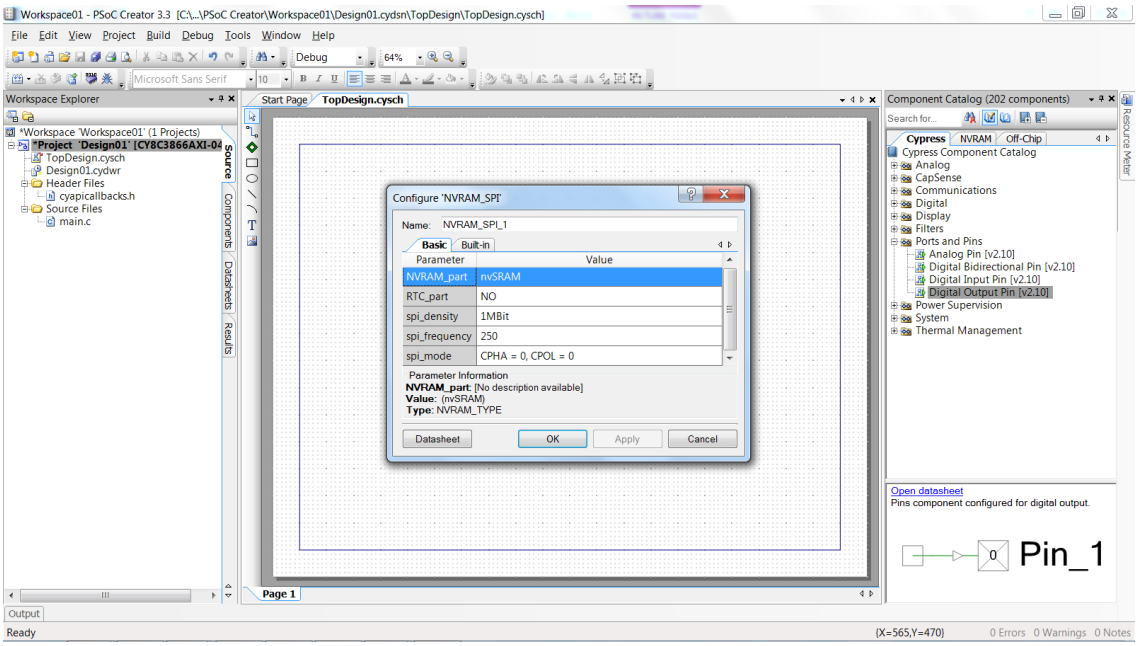
Figure 7. NVRAM_SPI Component Usage



5. NVRAM_SPI Configuration:

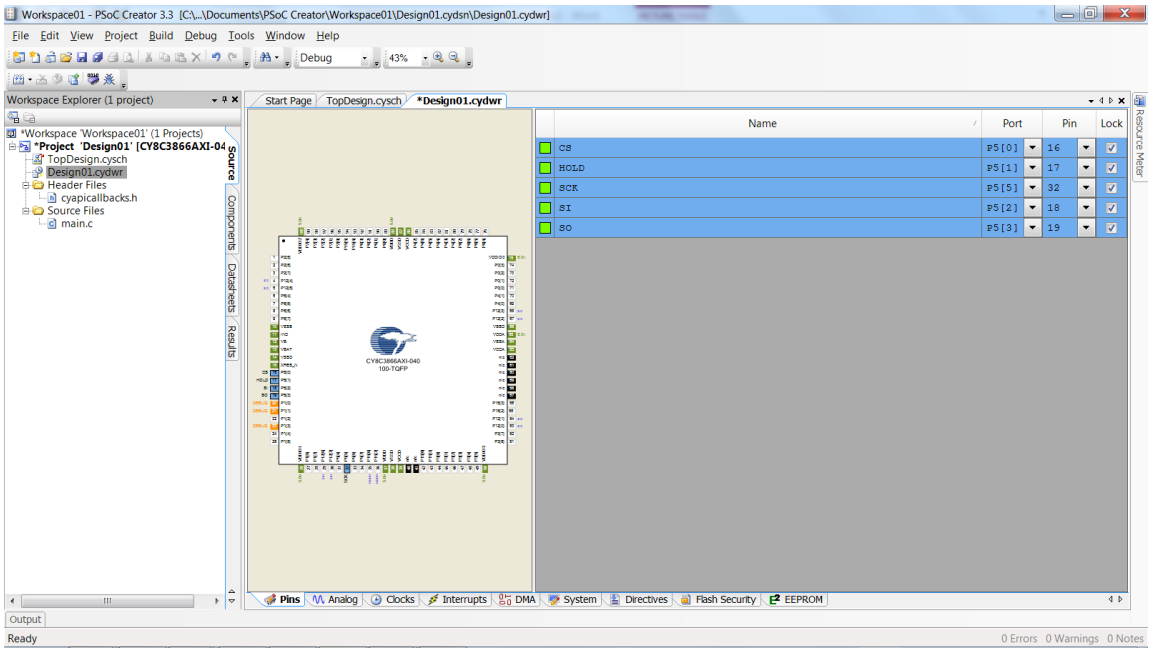
   a. Right-click the NVRAM_SPI_1 Component in *TopDesign.cysch* and select **Configure**. Set the parameters **NVRAM_part**, **RTC_part**, **spi_density**, **spi_frequency**, and **spi_mode** as per your design. This project uses 1-MBit non-RTC nvSRAM part, 250 KHz SPI frequency and mode 0.

Figure 8. User Component Configuration



   b. Assign appropriate input/output pins as per your design and build the project.

Figure 9. Pin Assignment



Code Example

This section provides the sample codes to access the SPI nvRAM user component APIs. The complete code can be found in the *main.c* file of the code example. Component instance is NVRAM_SPI_1. You can open the example project by selecting CE204087 workspace in the CE204087 folder.

```
// API NVRAM_SPI_1_Init initializes the SPI nvRAM user component

NVRAM_SPI_1_Init();
```

```
// API NVRAM_SPI_1_Write writes 4 bytes from array data_bytes to NVRAM location
// 0x012345

NVRAM_SPI_1_Write(0x012345, data_bytes, 4);
```

```
// API NVRAM_SPI_1_Read reads 4 bytes of data from NVRAM location 0x012345 to
array data_bytes

NVRAM_SPI_1_Read(0x012345, data_bytes, 4);
```

```
// API NVRAM_SPI_1_Status_Reg_Write writes 0x08 to status register

NVRAM_SPI_1_Status_Reg_Write(0x08);
```

```
// API NVRAM_SPI_1_Status_Reg_Read reads data from status register to status_reg

status_reg = NVRAM_SPI_1_Status_Reg_Read();
```

```
// API NVRAM_SPI_1_Serial_No_Write writes 8 bytes from array data_bytes to serial
number register

NVRAM_SPI_1_Serial_No_Write (data_bytes);
```

```
// API NVRAM_SPI_1_Serial_No_Read reads 8 bytes of serial number to array
serial_number

NVRAM_SPI_1_Serial_No_Read (serial_number);
```

```
// API NVRAM_SPI_1_Device_ID_Read reads 4 bytes of device id to array device_id

NVRAM_SPI_1_Device_ID_Read (device_id);
```

```
// API NVRAM_SPI_1_RTC_Read reads 1 byte from RTC register 0 to array rtc_data

NVRAM_SPI_1_RTC_Read (RTC_FLAG_REG, rtc_read_data, 1);
```

```
// API NVRAM_SPI_1_RTC_Write writes 1 byte from array rtc_write_data to RTC
register 0

NVRAM_SPI_1_RTC_Write (0, rtc_write_data, 1);
```

```
// Following code example gives the sequence to write to an RTC register.

uint8 rtc_write_data[16];
uint8 rtc_read_data[16];

// Read RTC Flags Register
NVRAM_SPI_1_RTC_Read (0x00, rtc_read_data, 1);

// Set Write Bit
rtc_write_data[0] = (rtc_read_data[0] | 0x02);
NVRAM_SPI_1_RTC_Write (0x00, rtc_write_data, 1);

// Write 8 bytes of data from RTC SECONDS register (0x09)
NVRAM_SPI_1_RTC_Write (0x09, &rtc_write_data[9], 8);

// Clear Write Bit
rtc_write_data[0] = (rtc_read_data[0] & 0xFD);
NVRAM_SPI_1_RTC_Write (0x00, rtc_write_data, 1);
```

```
// Following code example gives the sequence to read from an RTC register.

uint8 rtc_write_data[16];
uint8 rtc_read_data[16];

// Read RTC Flags Register
NVRAM_SPI_1_RTC_Read (0x00, rtc_read_data, 1);

// Set Read Bit
rtc_write_data[0] = (rtc_read_data[0] | 0x01);
NVRAM_SPI_1_RTC_Write (0x00, rtc_write_data, 1);

// Read 8 bytes of data starting from RTC SECONDS register (0x09)
NVRAM_SPI_1_RTC_Read (0x09, & rtc_read_data[9], 8);

// Clear Read Bit
rtc_write_data[0] = (rtc_read_data[0] & 0xFE);
NVRAM_SPI_1_RTC_Write (0x00, rtc_write_data, 1);
```

```
// API NVRAM_SPI_1_nvCommand is specific to nvSRAM. It sends soft sequence
command to nvSRAM. Following line sends STORE command (nvSRAM_STORE_CMD=0x3C) to
nvSRAM.

NVRAM_SPI_1_nvCommand(nvSRAM_STORE_CMD);
```

```
// Steps to disable autostore

// Autostore disable soft sequence
NVRAM_SPI_1_nvCommand(nvSRAM_ASDISB_CMD);

// Wait for 1 ms for processing autostore disable
CyDelay(1);

// Store the autostore disable to NV
NVRAM_SPI_1_nvCommand(nvSRAM_STORE_CMD);

// Wait for 8ms
CyDelay(8);
```

Related Documents

Application Notes	
AN89659 - Interfacing SPI F-RAM with PSoC® 4	Shows how to interface Serial Peripheral Interface (SPI) F-RAM with Cypress' PSoC 4 device with the help of example circuits, timing diagrams, and pseudocode.
AN64574 - Designing with Serial Peripheral Interface (SPI) nvSRAM	Provides a few key design considerations and firmware tips to guide the users designing with SPI nvSRAM.
AN304 - SPI Guide for F-RAM™	Provides the functional description, timing, and example code for SPI F-RAMs.
AN408 - A Design Guide to SPI F-RAM™ Processor Companion - FM33256B	Provides an overview and design guidelines for the SPI F-RAM real-time clock (RTC) Processor Companion part - FM33256B. This document also includes a typical application, an example code, and a PSoC 3-based user module.
AN302 - F-RAM™ SPI Read and Write Internal Operation and Data Protection	Discusses the importance of keeping CS high during power transitions and suggests a circuit to accomplish this. It also describes the internal operation of Cypress' high-speed SPI F-RAM devices during memory read and write operations.
AN52433 - Advantages of Serial Peripheral Interface (SPI) nvSRAM over SPI EEPROM in Metering Applications	Describes the benefits of using SPI nvSRAM in metering applications and is aimed at designers and architects of the latest 'smart' electrical energy meters.

F-RAM/nvSRAM Resources

A range of information on F-RAM and nvSRAM can be found at www.cypress.com. The following is an abbreviated list:

- **Overview:** Nonvolatile RAM portfolio, Nonvolatile RAM Roadmap.
- **Product Selectors:** F-RAM / nvSRAM.
- **Datasheets**: Describe and provide electrical specifications for the F-RAM / nvSRAM devices.
- **Application Notes:** Cover a broad range of topics, from basic to advanced level. Some of the application notes include code examples.

PSoC Resources

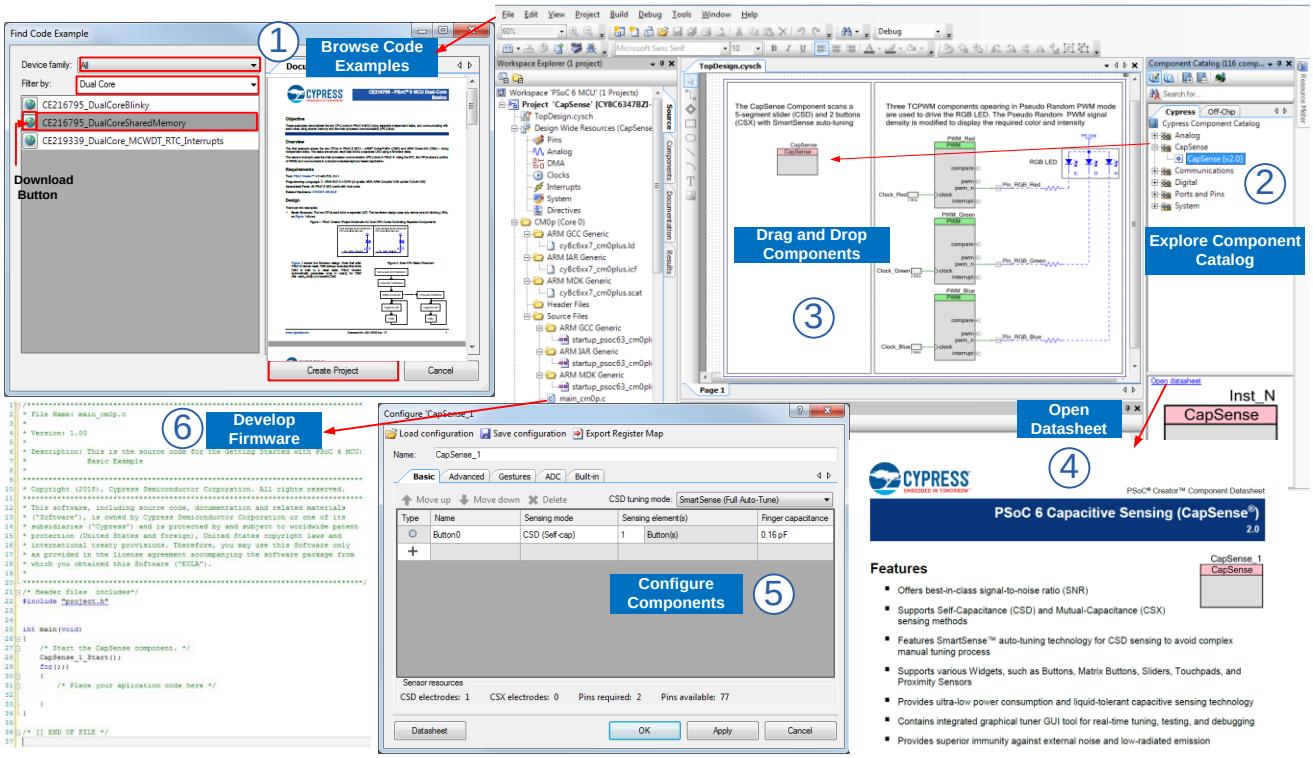
Cypress provides a wealth of data at www.cypress.com to help you to select the right PSoC device for your design, and quickly and effectively integrate the device into your design. For a comprehensive list of resources, see KBA86521, How to Design with PSoC 3, PSoC 4, and PSoC 5LP. The following is an abbreviated list for PSoC x:

- **Overview:** PSoC Portfolio, PSoC Roadmap
- **Product Selectors:** PSoC 1, PSoC 3, PSoC 4, or PSoC 5LP. In addition, PSoC Creator includes a device selection tool.
- **Datasheets**: Describe and provide electrical specifications for the PSoC 3 and PSoC 5LP device families.
- **CapSense Design Guide:** Learn how to design capacitive touch-sensing applications with the PSoC 3 and PSoC 5LP family of devices.
- **Application Notes and Code Examples:** Cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples.
- **Technical Reference Manuals (TRM):** Provide detailed descriptions of the architecture and registers in each PSoC 3 / PSoC 5LP device family.
- **Development Kits:**
  - CY8CKIT-001 is a common development platform for all PSoC family devices.
- The MiniProg3 device provides an interface for flash programming and debug.

PSoC Creator

PSoC Creator is a free Windows-based Integrated Design Environment (IDE). It enables concurrent hardware and firmware design of systems based on PSoC 3, PSoC 4, PSoC 5LP, and PSoC 6 MCU. With PSoC Creator, you can:

1. Browse the collection of code examples from the **File** > **Code Example** menu.
2. Explore the library of 100+ Components
3. Drag and drop Components to build your hardware system design in the main design workspace
4. Review Component datasheets
5. Configure Components using configuration tools
6. Codesign your application firmware with the PSoC hardware



Document History

Document Title: CE204087 – Interfacing SPI nvRAM with PSoC(R) 3 and PSoC 5LP

Document Number: 002-04087

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5060304	MEDU	12/22/2015	New spec
*A	6437755	MEDU	01/14/2019	Sunset Review Updated Template

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 | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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