CE219174 - Interfacing SPI nvRAM with PSoC® 4

Overview

The objective of this code example is to interface Cypress's SPI nonvolatile RAM (nvRAM) devices with a PSoC 4 controller. The nvRAM consists of both F-RAM™ and nvSRAM products. The code example has a User Component nvRAM that is configurable as a different density F-RAM/nvSRAM and frequency. The import of the User Component into a project and the usage of supported APIs are shown.

Requirements

Tool: PSoC Creator™ 4.1

Programming Language: C (GCC 5.4)

Associated Parts: PSoC 4200

Related Hardware: CY8CKIT-042

Design

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

Design Considerations

The SPI NVRAM User Component is designed with PSoC4. Serial memories can run up to 40 MHz but in this example project, it is limited by the PSoC controller.

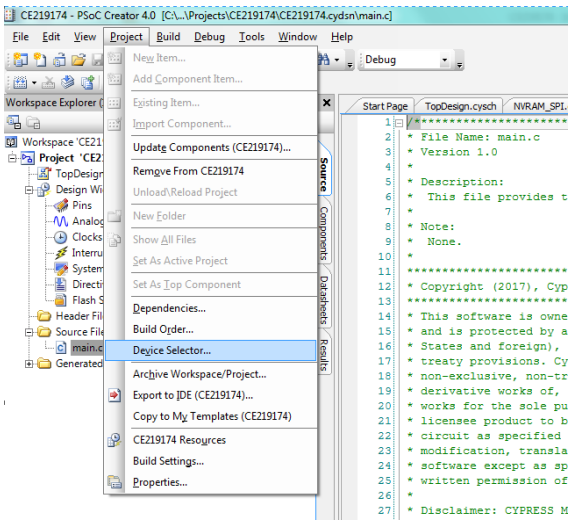
Hardware Setup

Hardware setup involves interfacing the F-RAM/nvSRAM board with the CY8CKIT-042 PSoC kit. The hardware setup is not limited to CY8CKIT-042 alone. Because the connection is made through external wires, any PSoC 4 kit along with any F-RAM/nvSRAM board can be used with proper pin assignments.

Software Setup

The code example is developed on PSoC Creator 4.1. It is applicable to PSoC 4 devices. 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. The library also has a version of the component without HOLD and WP pins.

Table 1. List of PSoC Creator Components

Component	Hardware Resources
Clock	Clock
SPI Mater	1 datapath cells, 13 macrocells, 3 status cells, 1 Control Cell, and 2 interrupts
Digital Input Pins	P2[5]
Digital Output Pins	P0[6], P2[4], P2[6], P3[4], P3[7]

Parameter Settings

Table 2. SPI NVRAM User Module Configuration

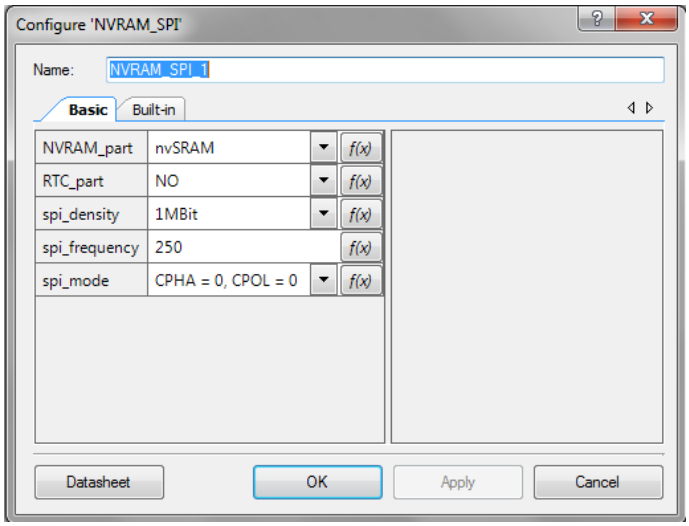


Table 3 lists the parameter settings of the Components used by the SPI NVRAM User Component.

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

Table 4. Pin Assignments

Name	Port	Pin	Lock
CS	P3[4]	15	☑
HOLD	P2[6]	8	☑
SCK	P0[6]	30	☑
SI	P2[4]	6	☑
SO	P2[5]	7	☑
WP	P3[7]	18	☑

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table 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 a given 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 5. 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 Write	nvSRAM and F-RAM
NVRAM_SPI_Read	SPI NVRAM Read	nvSRAM and F-RAM
NVRAM_SPI_FastRead	SPI NVRAM Fast 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 128-Kbit
NVRAM_SPI_WP_Set	Set WP Pin to HIGH	All nvSRAM and F-RAMs
NVRAM_SPI_WP_Reset	Set WP Pin to LOW	All nvSRAM and F-RAMs
NVRAM_SPI_HOLD_Set	Set HOLD Pin to HIGH	All nvSRAM and F-RAMs
NVRAM_SPI_HOLD_Reset	Set HOLD Pin to LOW	All nvSRAM and F-RAMs

`void NVRAM_SPI_Init (void)`

Description: Initialization routine for the NVRAM_SPI component. Initializes the SPI block, CS, and Hold pin.

Parameters: None

Return Value: None

Side Effects: None

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

Description: Writes 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: None

Side Effects: None

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

Description: Reads 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: None

Side Effects: None

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

Description: Writes 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: None

Side Effects: None

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

Description: Reads 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: None

Side Effects: None

`void NVRAM_SPI_ nvCommand (uint8 nvcmd)`

Description: Sends NVRAM Command. This is supported by nvSRAM only. It supports the nvSRAM special functions like STORE, RECALL, AUTOSTORE ENABLE, AUTOSTORE DISABLE and SLEEP.

Parameters: **`uint8 nvcmd:`** 8-bit NVRAM Command

Return Value: None

Side Effects: None

`void NVRAM_SPI_ Sleep (void)`

Description: Sends NVRAM Command for Sleep.

Parameters: None

Return Value: None

Side Effects: None

`uint8 NVRAM_SPI_ Status_Reg_Read (void)`

Description: NVRAM Status Register Read

Parameters: None

Return Value: **`uint8:`** 1-byte Status register data

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: None

Side Effects: None

`void 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: None

Side Effects: None

`void 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: None

Side Effects: None

`void 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: None

Side Effects: None

`void NVRAM_SPI_ WP_Set (void)`

Description: Set WP Pin to HIGH. This is supported by nvSRAM and FRAM.

Parameters: None

Return Value: None

Side Effects: None

`void NVRAM_SPI_ WP_Reset (void)`

Description: Set WP Pin to LOW. This is supported by nvSRAM and FRAM.

Parameters: None

Return Value: None

Side Effects: None

`void NVRAM_SPI_ HOLD_Set (void)`

Description: Set HOLD Pin to HIGH. This is supported by nvSRAM and FRAM.

Parameters: None

Return Value: None

Side Effects: None

`void NVRAM_SPI_ HOLD_Reset (void)`

Description: Set HOLD Pin to LOW. This is supported by nvSRAM and FRAM.

Parameters: None

Return Value: None

Side Effects: None

Operation

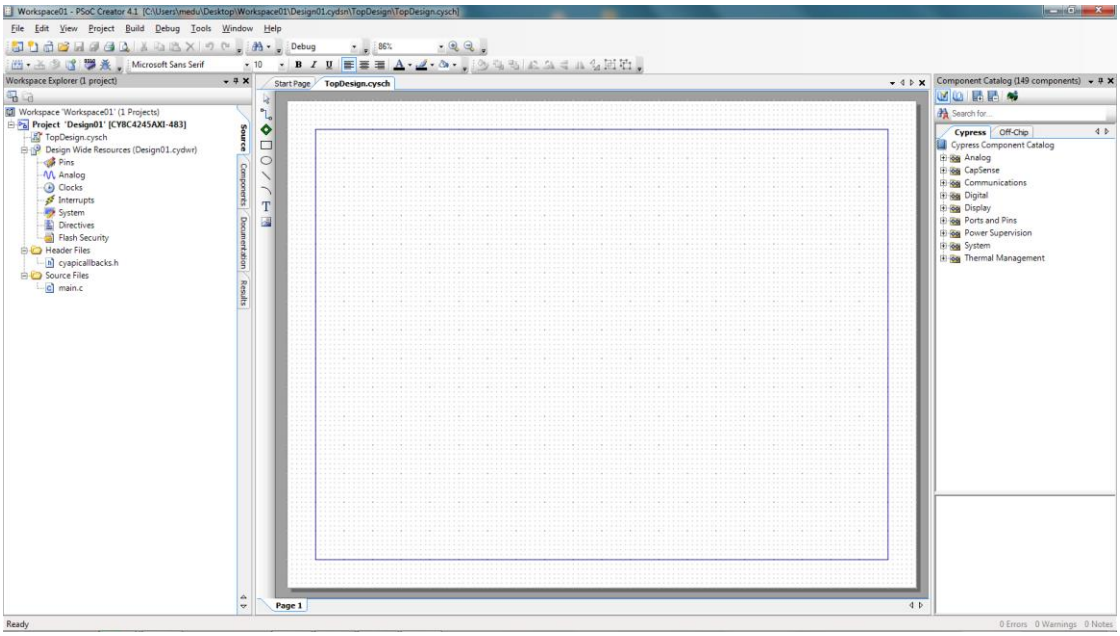
This section shows how to import the SPI NVRAM User Component into code example and the how to use the APIs of the User Component.

Setup

The NVRAM_SPI archive contains both the example project and the NVRAM_SPI Component. The following snapshots show how to use the NVRAM_SPI Component in your design.

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

Figure 2. Create project ‘Design01’



2. Select the **Dependencies** tab on the workspace explorer and bring the NVRAM_SPI Component into your design as shown below.

Figure 3. Right-click on the Project and Open Dependencies

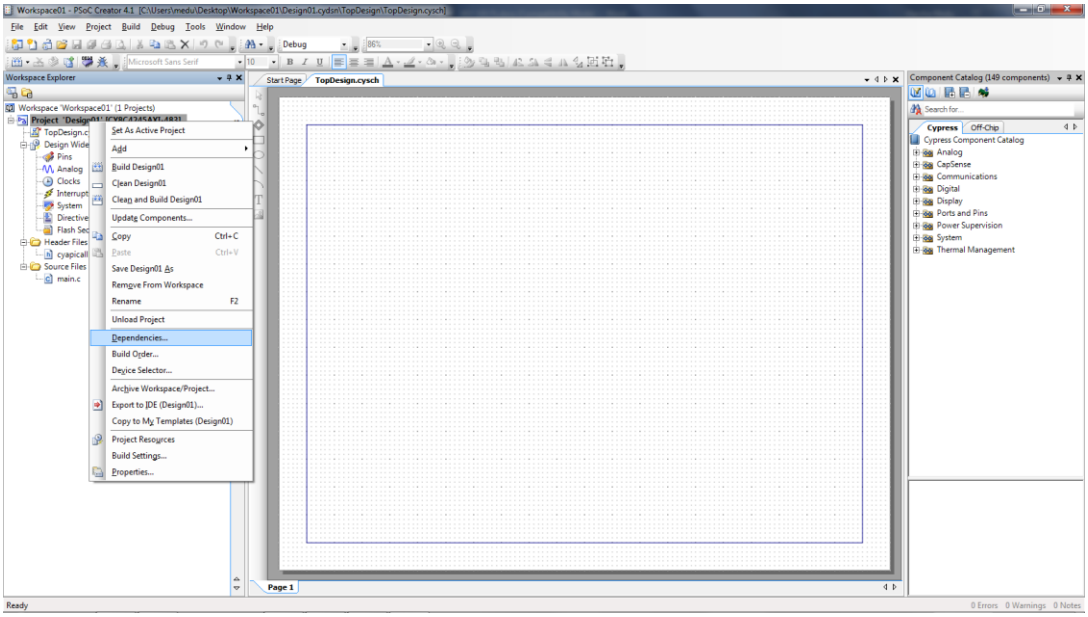


Figure 4. Click on New Entry (User Dependencies) and Select *NVRAM_Component_Lib.cyprj* from *NVRAM_Component_Lib.cylib* Folder

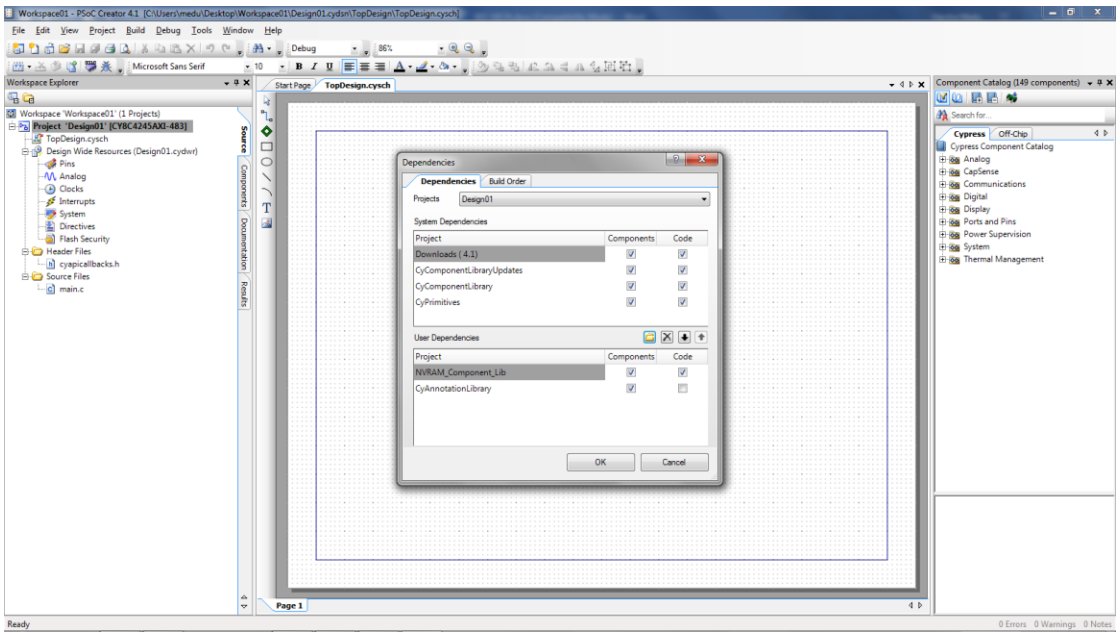
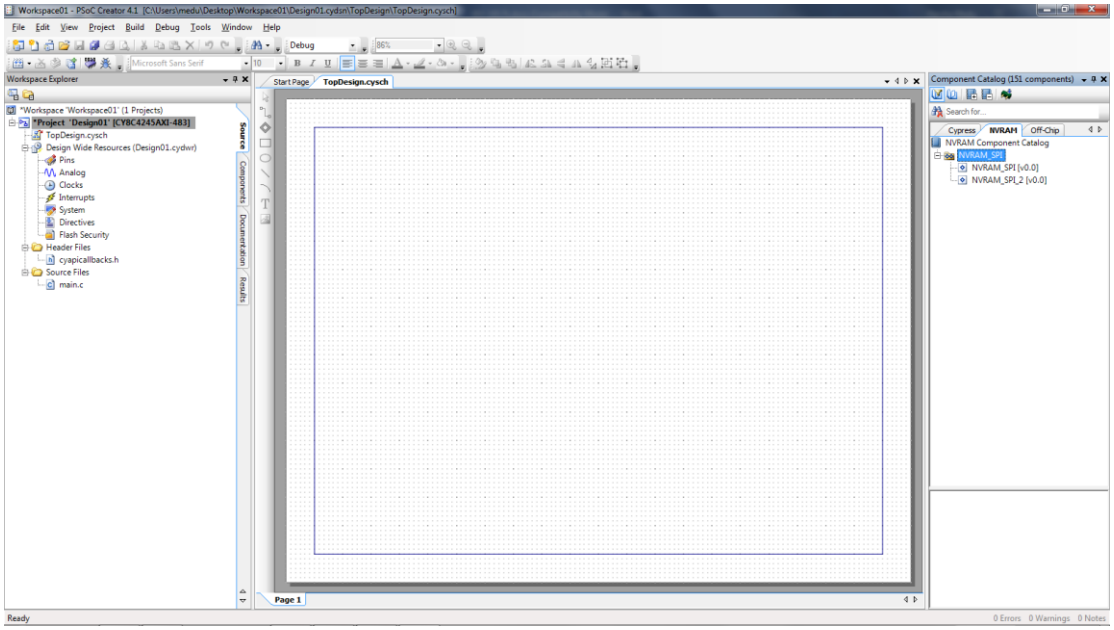
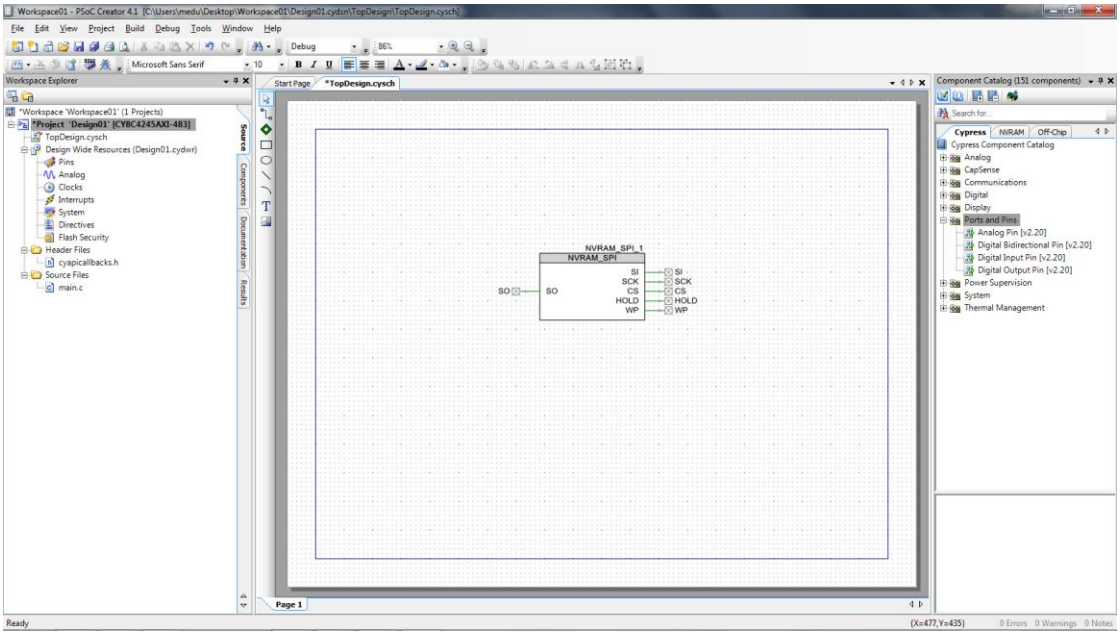


Figure 5. NVRAM_SPI Component Appears Under NVRAM/NVRAM_SPI in Component Catalog



3. Add the NVRAM_SPI Component to your design.

Figure 6. Drag and Drop NVRAM_SPI Component onto *TopDesign.cysch* and Assign Digital I/Os



4. NVRAM_SPI Configuration

Right-click the NVRAM_SPI_1 Component in *TopDesign.cysch* and select **Configure**. Set the `NVRAM_part`, `RTC_part`, `spi_density`, `spi_frequency`, and `spi_mode` parameters per your design. In this project, 1-MBit non-RTC nvSRAM part, 250-kHz SPI frequency, and mode 0 are selected.

Figure 7. User Component Configuration

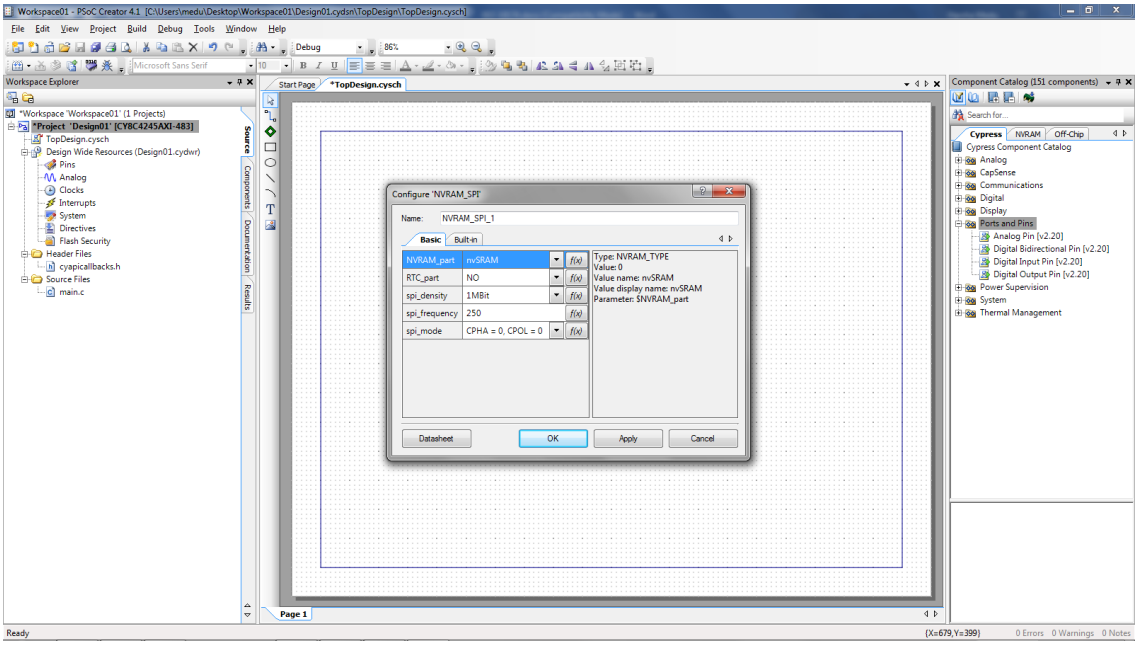
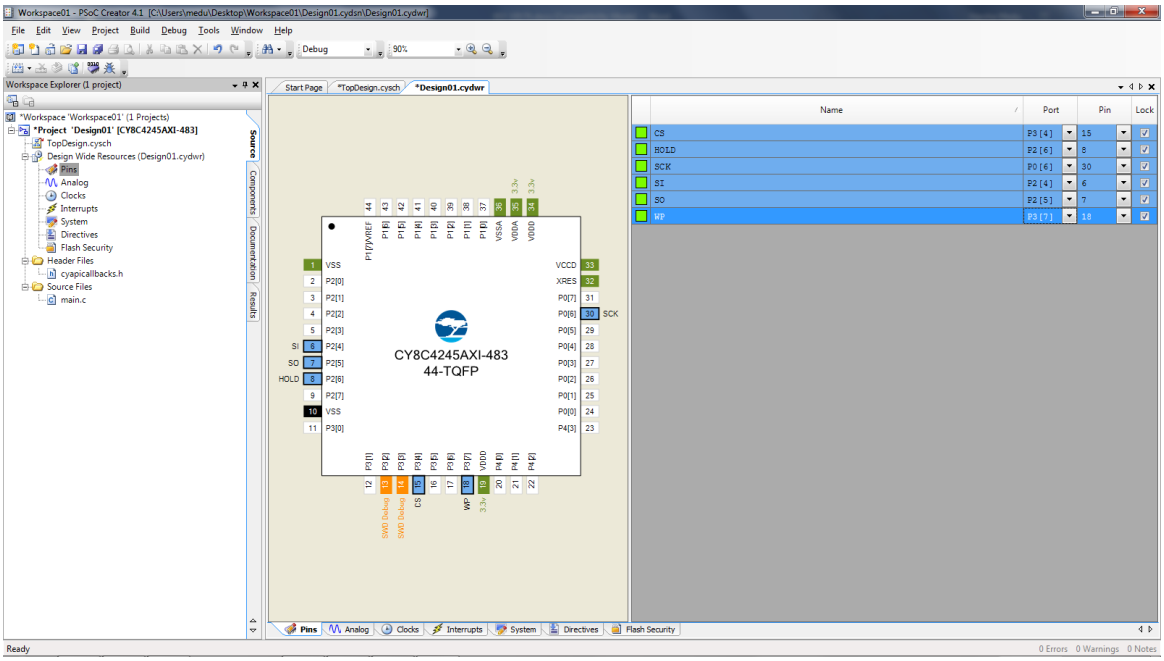


Figure 8. Assign Appropriate Input/Output Pins per Design and Build the Project



Exporting to Eclipse IDE

This section explains the steps necessary to build the exported project successfully on Eclipse IDE.

After exporting the project to Eclipse IDE, open it on Eclipse and exclude custom component sources from build by “Right mouse click on the custom components in the eclipse -> Resource Configurations -> Exclude from the build -> Select All -> OK”. Refer to PSoC Creator Help for exporting PSoC projects to Eclipse IDE.

Figure 9. Exclude custom components (NVRAM_SPI and NVRAM_SPI_2) in eclipse

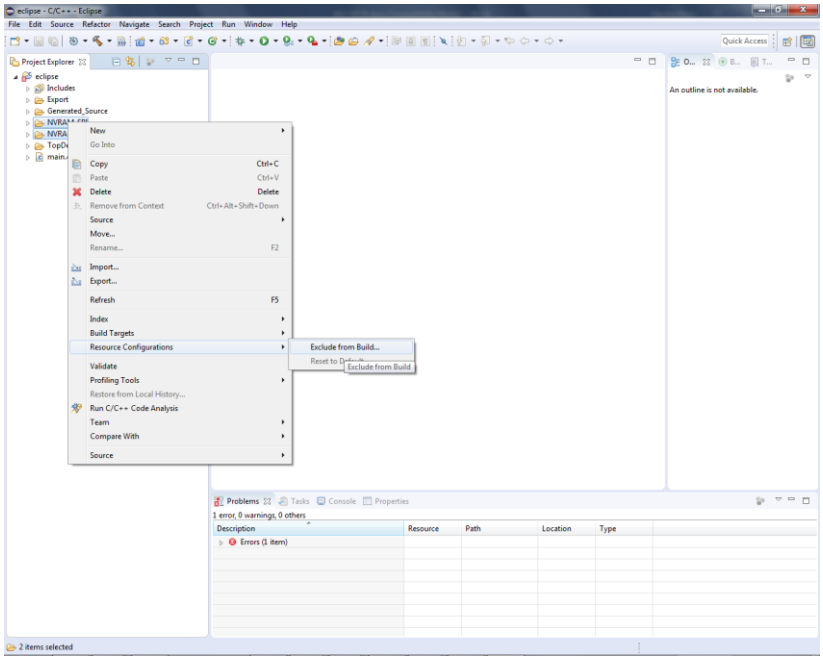


Figure 10. Exclude custom components for all builds

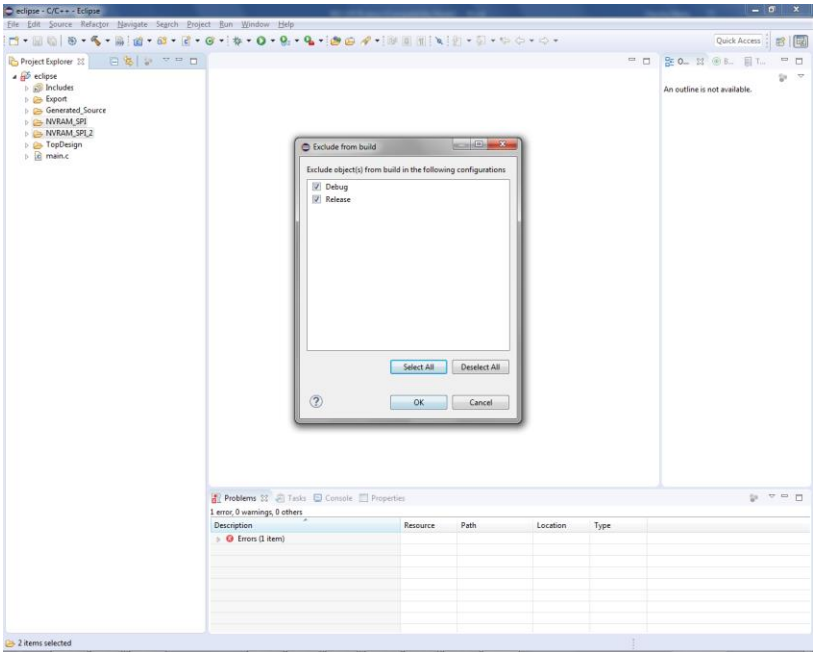


Figure 11. custom components (NVRAM_SPI and NVRAM_SPI_2) excluded in eclipse

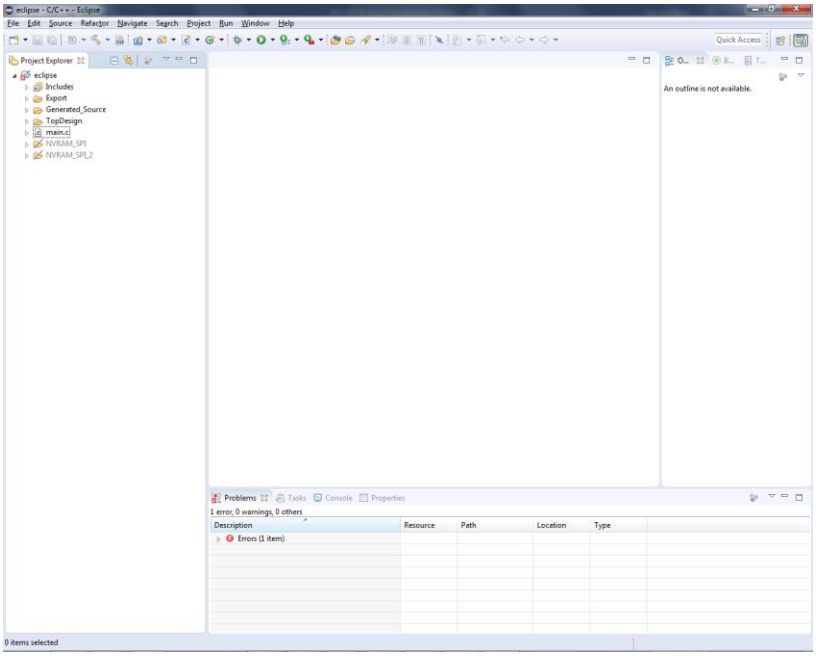
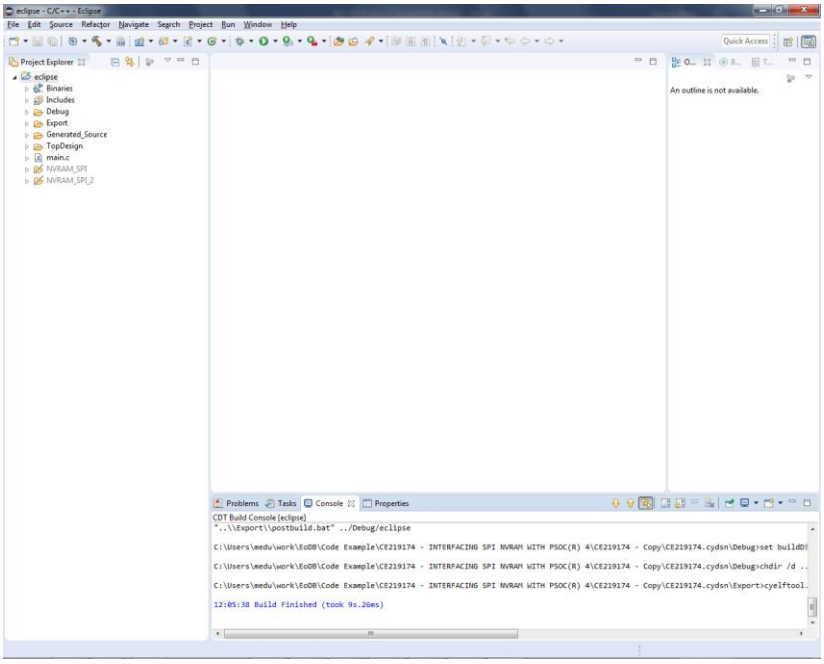


Figure 12. Build Project in Eclipse



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. The Component instance is NVRAM_SPI_1. The example project can be opened by selecting CE219174 workspace in CE219174 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
address = 0x123456;
length = 4;
NVRAM_SPI_1_Write(address, data_bytes, length);
```

```
// API NVRAM_SPI_1_Read reads 4 bytes of data from NVRAM location 0x012345 to
array data_bytes
address = 0x123456;
length = 4;
NVRAM_SPI_1_Read(address, data_bytes, length);
```

```
// API NVRAM_SPI_1_Status_Reg_Write writes 0x08 to status register
status_reg = 0x08;
NVRAM_SPI_1_Status_Reg_Write(status_reg);
```

```
// 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_Device_ID_Read reads 4 (nvSRAM)/9(F-RAM) bytes of device id to
array device_id
NVRAM_SPI_1_Device_ID_Read (device_id);
```

```
// 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_RTC_Read reads 1 byte from RTC register 0 to array rtc_data
NVRAM_SPI_1_RTC_Read (nvSRAM_RTC_REG_FLAGS, 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 (nvSRAM_RTC_REG_FLAGS, 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 (nvSRAM_RTC_REG_FLAGS, rtc_read_data, 1);

// Set Write Bit
rtc_write_data[0] = (rtc_read_data[0] | 0x02);
NVRAM_SPI_1_RTC_Write (nvSRAM_RTC_REG_FLAGS, rtc_write_data, 1);
// Write 8 bytes of data from RTC SECONDS register (0x09)
Length = 8;
NVRAM_SPI_1_RTC_Write (nvSRAM_RTC_REG_SECOND, &rtc_write_data[9], length);
```

```
// 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 (nvSRAM_RTC_REG_FLAGS, rtc_read_data, 1);
// Set Read Bit
rtc_write_data[0] = (rtc_read_data[0] | 0x01);
NVRAM_SPI_1_RTC_Write (nvSRAM_RTC_REG_FLAGS, rtc_write_data, 1);
// Read 8 bytes of data starting from RTC SECONDS register (0x09)
length = 8;
NVRAM_SPI_1_RTC_Read (nvSRAM_RTC_REG_SECOND, & rtc_read_data[9], length);
```

```
// 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);
```

Related Documents

Table 6. Related Documents

Document	Title	Comment
AN89659	Interfacing SPI F-RAM with PSoC® 4	This application note shows how to interface Serial Peripheral Interface (SPI) F-RAM with Cypress's PSoC® 4 (Programmable System-on-Chip) device with the help of example circuits, timing diagrams, and pseudocode.
AN64574	Designing with Serial Peripheral Interface (SPI) nvSRAM	This application note provides a few key design considerations and firmware tips to guide the users designing with SPI nvSRAM.
AN304	SPI Guide for F-RAM™	This application note provides the functional description, timing, and example code for SPI F-RAMs.
AN408	A Design Guide to SPI F-RAM™ Processor Companion - FM33256B	This application note 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.
AN52433	Advantages of Serial Peripheral Interface (SPI) nvSRAM over SPI EEPROM in Metering Applications	This application note 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.
- **Development Kits:**
  - CY15FRAMKIT-001 is an F-RAM development board compatible with Cypress's CY8CKIT-042, CY8CKIT-042-BLE and CY8CKIT-040 kits and Arduino UNO R3 board.

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:

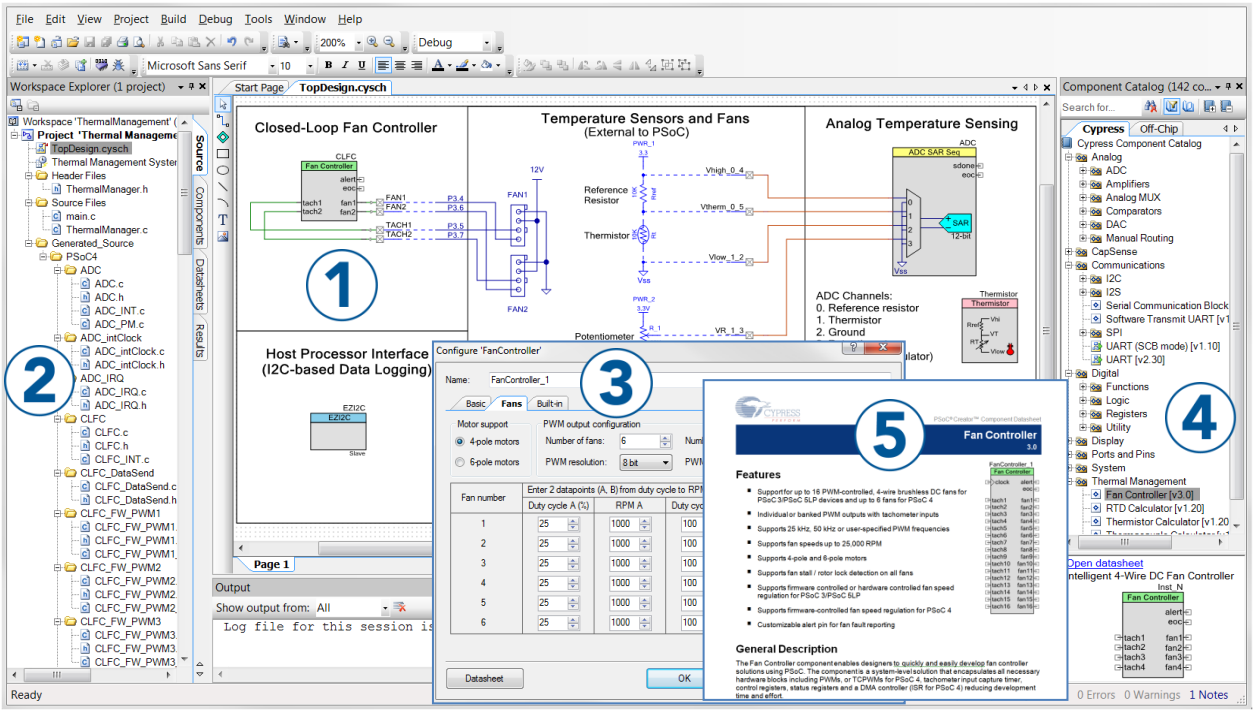
- **Overview:** PSoC Portfolio, PSoC Roadmap
- **Product Selectors:** PSoC 1, PSoC 3, PSoC 4, or PSoC 5LP. In addition, PSoC Creator includes a device selection tool.
- **Datasheets**: Describe and provide electrical specifications for the PSoC 4 device families.
- **CapSense Design Guide:** Learn how to design capacitive touch-sensing applications with the PSoC 4 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 4 device family.
- **Development Kits:**
  - CY8CKIT-042 is a common development platform for PSoC 4 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, and PSoC 5LP. See Figure 13 – with PSoC Creator, you can:

1. Drag and drop Components to build your hardware system design in the main design workspace
2. Code your application firmware with the PSoC hardware
3. Configure Components using configuration tools
4. Explore the library of 100+ Components
5. Review Component datasheets

Figure 13. PSoC Creator Features



Document History

Document Title: CE219174 - Interfacing SPI nvRAM with PSoC® 4

Document Number: 002-19174

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5780724	MEDU	07/26/2017	Initial release

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

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
198 Champion Court
San Jose, CA 95134-1709

© 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.