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.

AN50989

USB To SPI Bridge using PSoC1

Author: Jaya Kathuria
Associated Project: Yes
Associated Part Family: CY8C24x94
Software Version: PSoC Designer 5.4
Related Application Notes: AN45022, AN49943

If you have a question, or need help with this application note, contact the author at xkj@cypress.com.

This application note discusses how to implement a USB to SPI bridge using PSoC1.The discussion includes a brief introduction to the PSoC USB subsystem, the USBFS user module, USB device descriptors and the PSoC SPI user module. It also covers the USB to SPI conversion process with the help of a flow chart. In addition, a GUI based PC application and an SPI master and SPI slave demonstration project are included in this application note. These two projects serve as example systems and allow a complete demonstration of the bridge. The GUI works only on systems with Windows XP.

Contents

Introduction 1
Features 1
PSoC USB Subsystem 2
USBFS User Module 2
PSoC SPI User Modules 3
USB to SPI Bridge 3
Example Project 3
Device Configuration 3
  USB Device Descriptors 3
Bridge Firmware 4
Setup and Demonstration 5
Summary 6
About the Author 7
Appendix 8
Document History 10
Worldwide Sales and Design Support 11

Introduction

The USB has taken over as the preferred method of communication between embedded systems and personal computers. SPI is a serial communication protocol widely used for chip-to-chip communication within embedded systems. This USB to SPI Bridge solution turns a PC into a comprehensive SPI bus master, allowing the PC to control hardware devices that use the SPI protocol.

This application note consists of the bridge firmware and hardware, SPI slave firmware and hardware, and a PC application. The bridge itself allows any full speed USB host to serve as an SPI bus master. The included slave project and PC application allow the demonstration of the protocol bridge solution between a USB equipped windows PC and an SPI equipped PSoC demonstration board. The PC application provides a GUI that enables data to be sent and received over the SPI interface.

In addition, this application note explains why choosing PSoC® for this application enables the bridge to become easy to use, highly flexible, and inexpensive.

Features

The key features of this solution are:

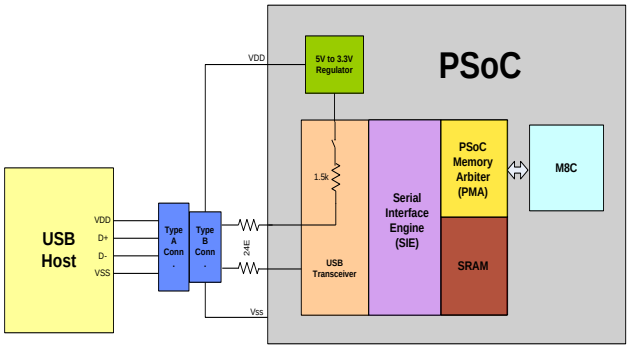
- Full Speed USB interface with no external hardware
- Support for both interrupt and control USB transfer types
- USB setup wizard for easy and accurate device descriptor configuration

- Support for runtime descriptor set selection
- SPI operation at speeds up to 4 Mbps
- Support for SPI clocking modes 0, 1, 2 and 3
- Support for selection of SPI clocking modes in firmware
- Slave select lines to allow connection of multiple slaves
- Operation with either 5.5 volt or 3.3 volt power supplies
- Programmable GPIOs
- Additional digital and analog resources that can be configured as PWMs, counters, timers, CRC generators, pseudo random number generators, I2C, IrDA ADCs, DACs, multi-pole filters, gain stages, and more
- Single chip solution

PSoC USB Subsystem

The CY8C24x94 family of PSoC device includes on-chip support for full-speed USB. The hardware provided includes a USB transceiver, a voltage regulator, a Serial Interface Engine (SIE), the PSoC Memory Arbiter (PMA), and 256 bytes of dedicated SRAM for endpoint data.

Figure 1. PSoC USB Subsystem



- **USB Transceiver**. The USB transceiver allows the PSoC to interface directly with a full-speed USB bus using dedicated D+ and D- pins. An internal 1.5k pull up resistor is provided on the D+ pin to signal the host that a Full Speed USB peripheral is attached. This pull up resistor can be enabled or disabled by firmware.
- **Voltage Regulator**. An on-chip voltage regulator allows generating 3.3 V USB signals, when the PSoC device is powered by USB bus power or by an external 5 V source. The firmware must disable this regulator if the PSoC device is powered by an external 3.3 V source, because the USB signals are generated directly from the power supply in this case. This is done by clearing bit 0 of the USB_CR1 register.
- **Serial Interface Engine (SIE).** The serial interface engine supports five USB endpoints: one control endpoint and four data endpoints. In addition, the SIE greatly simplifies the interface between the USB and the PSoC M8C core by automatically handling various tasks. These tasks are:
  - ☐ Translating encoded data received through USB
  - ☐ Formatting data to be transmitted over USB
  - ☐ CRC generation and verification
  - ☐ Address checking
  - ☐ USB ACK, NAK, and Stall generation
  - ☐ Token type identification
  - ☐ Start-of-Frame (SOF) identification
  - ☐ Transferring data to and from the USB SRAM
- **PSoC Memory Arbiter (PMA).** The PMA serves as the interface between the dedicated USB SRAM and the two systems that must access it: the PSoC M8C core and the SIE.
- **SRAM.** The PSoC contains 256 bytes of dedicated USB SRAM. This memory serves as a communication channel between the M8C core and the SIE. Transfers to the SRAM are managed by the PMA.

USBFS User Module

PSoC Designer™ provides the USBFS user module. This user module provides firmware support for USB operations enabling system development with minimal USB knowledge. The features of this user module are:

- **Data Flow Management**. Routines included in the user module manage the data flow in and out of the data endpoints.
- **Easy Operation of USB.** The user module includes an extensive API for managing the connection to the USB and all low level USB operations. This enables the user to focus on the specific data transfer requirements of the system being developed.
- **Descriptor Creation Wizard.** The USBFS user module includes a wizard that guides the user through creating all device descriptors required by the USB standard.

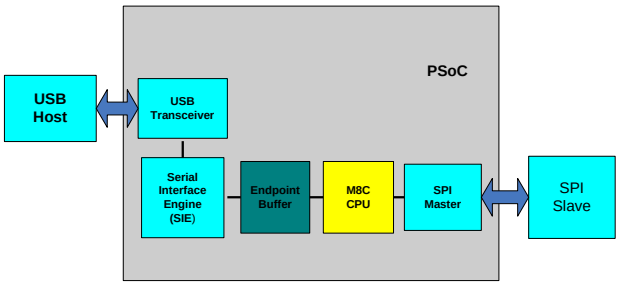
A complete description of the USB hardware and all associated registers is available in the PSoC Programmable System-on-Chip Technical Reference Manual. For detailed description of the USBFS user module API, refer to the USBFS user module data sheet that is included with PSoC Designer.

PSoC SPI User Modules

The PSoC device can act as a SPI master using the SPIM user module, or a SPI slave using the SPIS user module. SPI clock modes 0-3 are supported. The data word size is 8 bits. System interrupts can be generated when the transmit buffer is empty or when a SPI transaction is complete. For more information on using SPI in PSoC, refer the SPIS and SPIM user module data sheets that are included in PSoC Designer.

USB to SPI Bridge

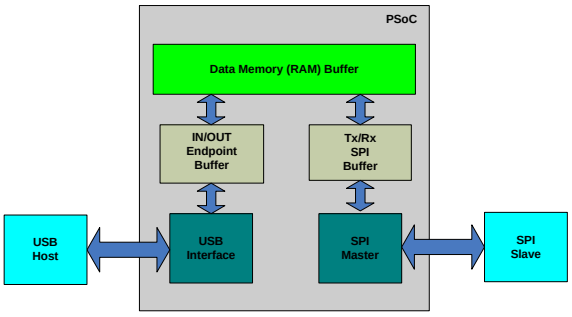
Figure 2. USB to SPI Bridge Block Diagram



A block diagram of the USB to SPI bridge is shown in Figure 2. Because USB and SPI use very different protocols and data rates, it is not possible to simply translate and move data from one bus to the other. Additional memory buffers must be used to implement protocol differences.

A diagram of the bridge data flow is shown in Figure 3. The USBFS user module handles reception of data from the USB in 64 byte blocks, and places this data into the dedicated USB SRAM. Application firmware moves this data to a memory buffer located in the PSoC device's main RAM. This data is then moved to the SPI Tx and Rx registers one byte at a time. The data is then transmitted over SPI through the SPIM user module. Data received from SPI during this transaction is finally sent back to the USBFS user module to be transmitted over USB.

Figure 3. USB to SPI Data Transfer through RAM Buffer inside PSoC



Example Project

The example project has three components. The main bridge is contained in the folder named "USB to SPI Master". The folder named "PC Software" is the PC application that provides a GUI for you to send and receive data from a PC. The folder named "SPI Slave" implements a SPI slave in another PSoC device to facilitate testing the bridge.

To use the PC application, the user must input 64 bytes of hex data into the textbox and press the button labeled OUTEP. This causes the data to be sent out the PC USB port, through the bridge, and to a connected SPI slave. Next, 64 bytes of data read from the SPI slave is displayed in the textbox on the PC.

Device Configuration

The user module placement within PSoC is shown in Figure 10 on page 9. Note that the USBFS user module and the LCD user module do not consume any PSoC blocks. This application only requires one digital block. The remaining blocks are available for other uses.

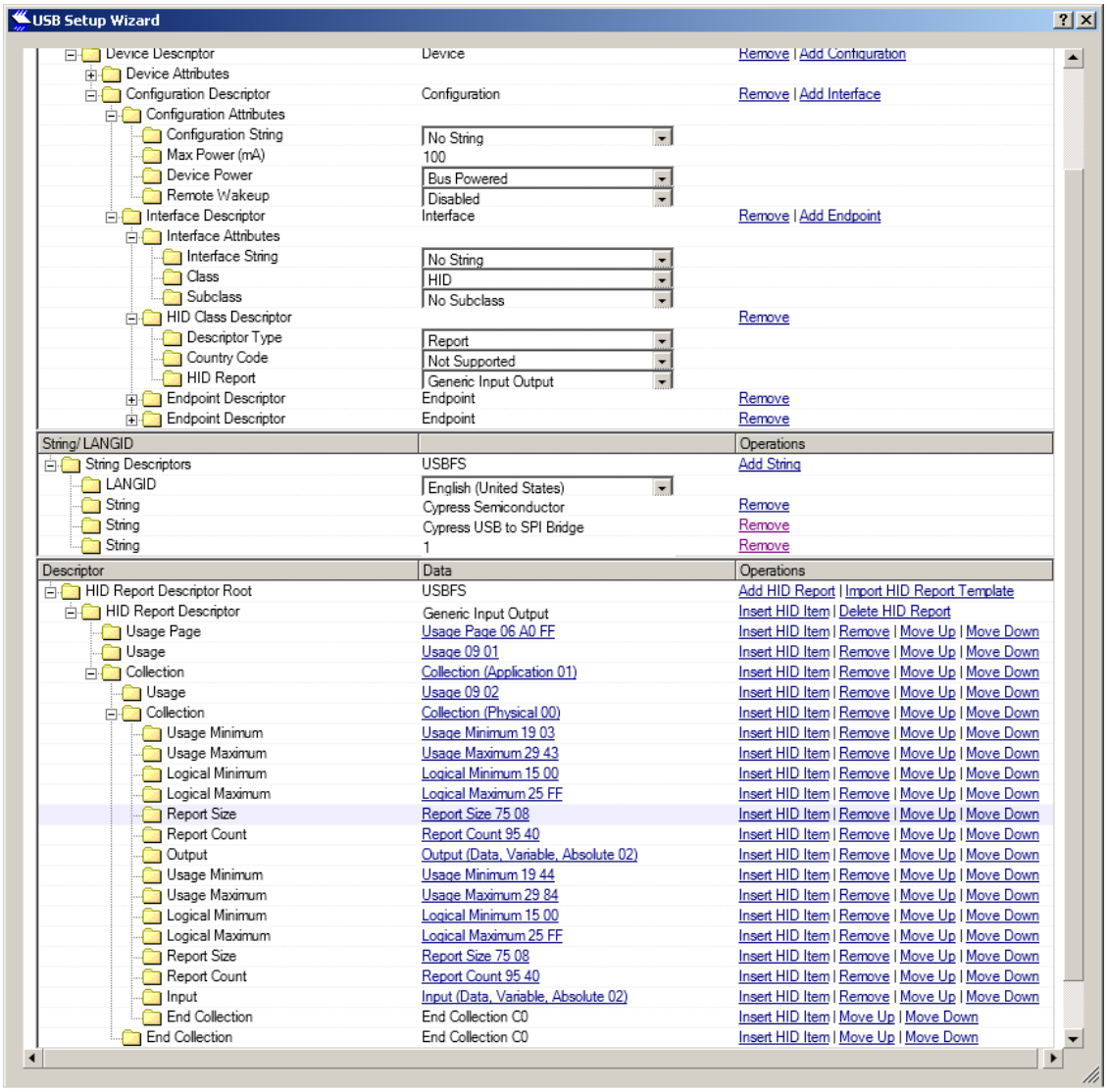
USB Device Descriptors

The PSoC device is configured as a generic HID device with one Out Interrupt endpoint and one In Interrupt endpoint, each having a data buffer size of 64 bytes. On every scanning interval (as specified by the endpoint configuration) the host checks for any In or Out packet from or to the HID device and performs an In or Out operation. When an Out data packet is sent to the device using the GUI, an Out endpoint interrupt is raised and the PSoC buffer gets filled. The In Endpoint is loaded with data received from the SPI Slave and is enabled in PSoC. This In Endpoint data is read by the PC when the user performs a data read operation from the GUI. The Usage page for the HID is defined as vendor specific.

The In and Out reports are defined in the HID Report descriptors; the logical minimum and logical maximum are fixed to 0 and 255 respectively. An In report and an Out report, each having a data size of 64 bytes, are declared in the HID descriptor.

The arrangement of the USB descriptors can be viewed and modified with the help of USB Setup Wizard. This wizard is opened by right clicking the USBFS User module under the Workspace Explorer window.

Figure 4. USB Setup Wizard



Bridge Firmware

Figure 5 shows a flowchart of the application C code contained in “SPI_USB_Master” Code execution is discussed in the following steps:

1. At the beginning of the main() function, global interrupts are enabled and the USBFS and SPIM user modules are started.
2. Firmware waits for the device to enumerate, and then enables endpoint 1 for incoming transfers (i.e. from SPI Master to PC) and endpoint 2 for outgoing transfers (i.e. from PC to SPI Master).

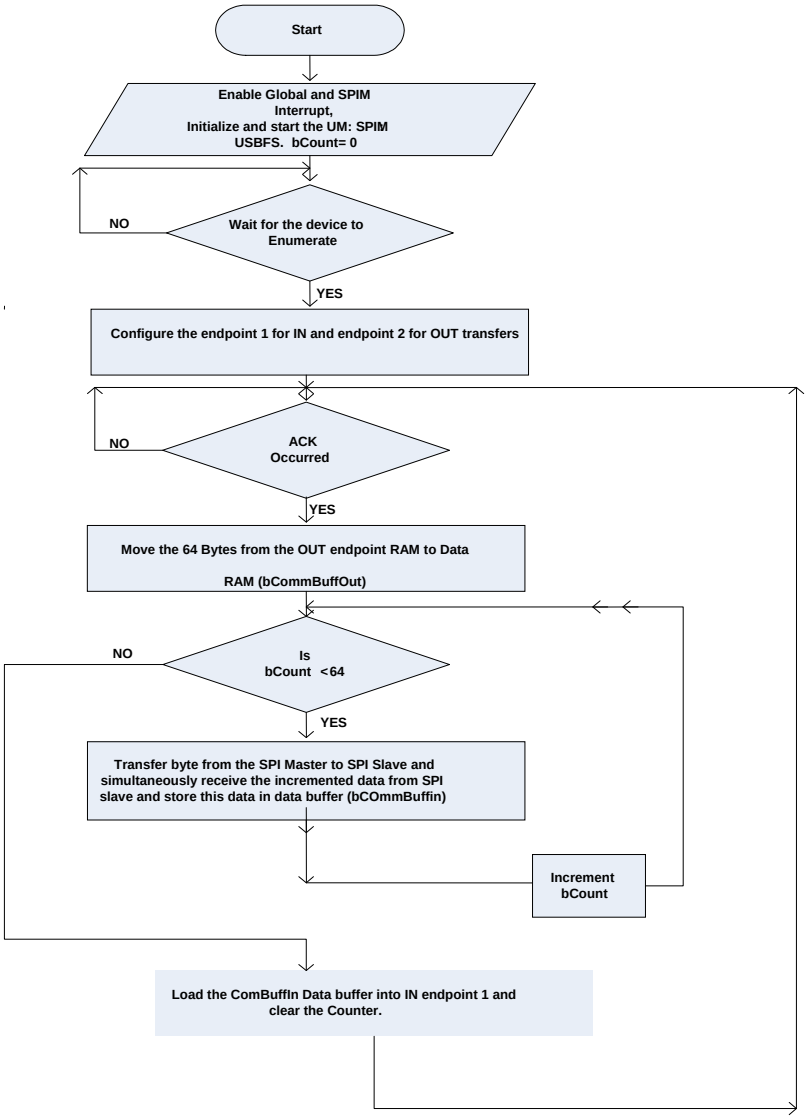
   Note: Direction of data transfer in USB is always referenced with respect to the USB host which is the PC in this case.
3. Send the data through the PC software by writing the hex data to be sent to the SPI master in the textbox, and then press the OUTEP button. The software calls the function named Transfer_Data, where the actual communication takes place.
4. Execution then enters an infinite loop that alternates between resetting the watchdog timer and calling function EP2_Handler().
5. EP2_Handler checks for data in the USB Endpoint buffers. If there is data, this data is copied to the Data

Memory Buffer in Ram, and function TransferData() is called.

6. Function TransferData() sets the Slave Select line to low to start a SPI transmission. Data is then transmitted from the Data Memory Buffer over SPI one byte at a time, and simultaneously received from SPI and placed in the Data Memory Buffer. The LCD function LCD_Delay50uTime is used to create a small delay between bytes to allow the slave time to respond. The exact delay required here is application specific. When this operation is complete, the received SPI data in the Data Memory Buffer is transmitted over to the IN_EP buffer.

.

Figure 5. Application Code Flowchart



Setup and Demonstration

Two PSoC devices and a PC are required for a complete demonstration of this example. The recommended PSoC devices are a CY8C24094 chip on a CY3214 USB Eval board, and a CY8C29466 on a CY3210 Eval1 board. The PC must run Microsoft Windows XP or Vista, and have a free USB port.

The “SPI_USB_Master” project is built on a CY8C24894 to take advantage of the USB support in the CY8C24x94 series of parts. Any other chip in this family would work for

this application if the project was cloned to make use of it. The CY3214 USB Eval board makes an excellent platform for testing this firmware, because it provides on-chip debugger support, connectors, and access to all required pins.

The SPI Slave Project (SPIS) is built on a CY8C29466 device, but nearly any PSoC device would be suitable with few or no modifications. A CY3210 Eval1 board works well to provide access to required pins.

Figure 9 on page 8 shows a schematic of the two PSoC devices and a USB port. Do the connections as shown in this schematic. The connections between the SPI Master and SPI Slave are summarized in Table 1.

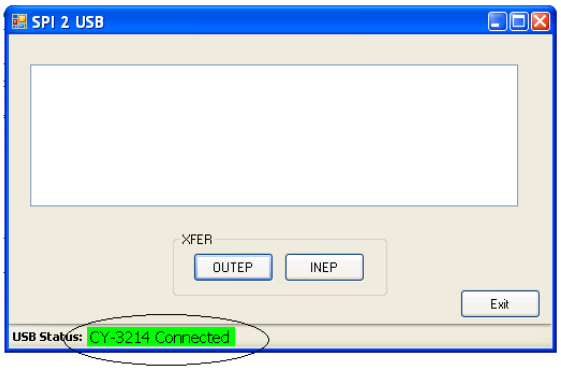
Table 1. Connections between Master and Slave

Pin	Master	Slave
SCLK	P0[0]	P0[4]
Slave Select	P0[2]	P0[6]
MISO	P0[3]	P0[7]
MOSI	P0[1]	P0[5]
GND	GND	GND

Program the PSoC CY8C24894 with the project named “SPI_USB_Master”, and the CY8C29466 with the project SPIS. Power up both the kits. Make sure that both kits operate at the same voltage levels (5 V)

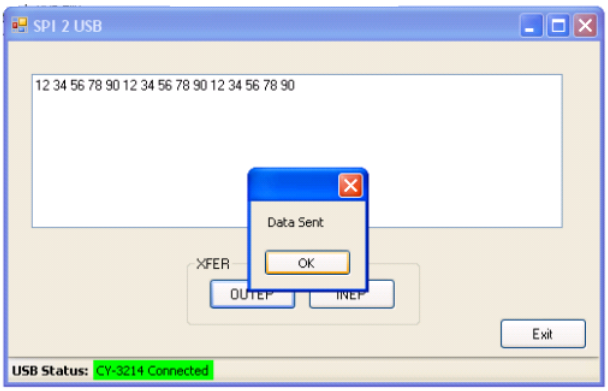
Run USB_TO_SPI.exe on the host PC. The display must be similar to Figure 6. Connect the CY8C24894 to one of the PC USB ports. After the device enumerates, the USB Status bar in the GUI must indicate that a CY-3214 is connected, as shown in the circled portion of Figure 6.

Figure 6. PC GUI



Type the data to be sent over the bridge in the textbox (maximum 64 bytes at a time), and press OUTEP button. The data is sent through the bridge to the SPI slave. The result on the PC screen must be similar to Figure 7. If fewer than 64 bytes are sent, the transmissions are padded with zeros to bring the size to exactly 64 bytes.

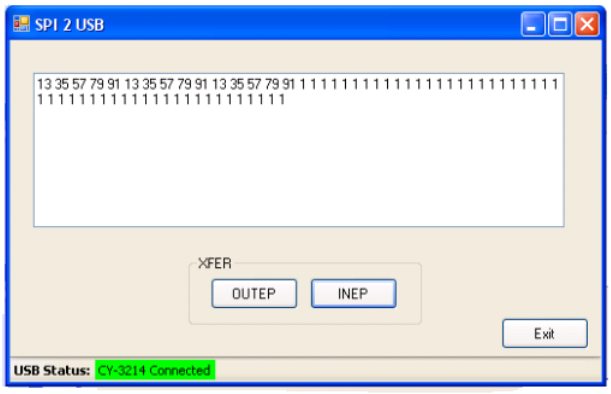
Figure 7. Data Transmission



The GUI combines two characters in the text box into a single hexadecimal value and sends it over the USB. So, when using the GUI, every number should be written in hexadecimal 2-digit format (for example, 01 for 1, 09 for 9 and so on).

The Slave receives the data, increments each byte received and transmits it to the Master. The data read from the SPI slave during transmission is displayed in the textbox by clicking on the INEP button. Figure 8 shows the results of clicking the INEP button.

Figure 8. Received Data


SPI 2 USB
13 35 57 79 91 13 35 57 79 91 13 35 57 79 91 1 1 1

About the Author

Name: Jaya Kathuria

Title: Application Engineer.

Background: Jaya Kathuria works as an Application Engineer in Cypress Semiconductor's Programmable Systems Division focused on PSoC Core applications.

Contact: xkj@cypress.com

Appendix

Figure 9. Schematic Diagram

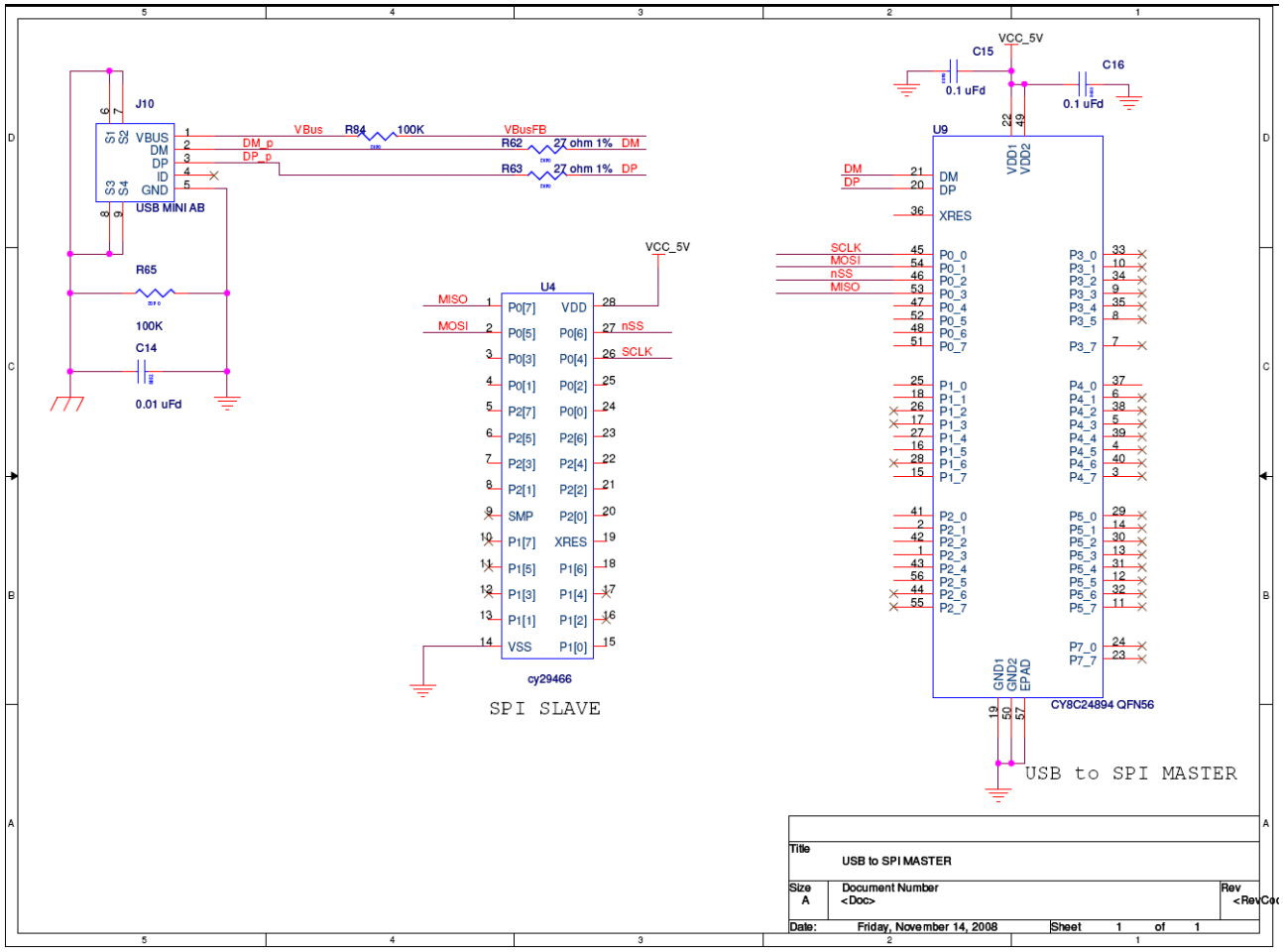
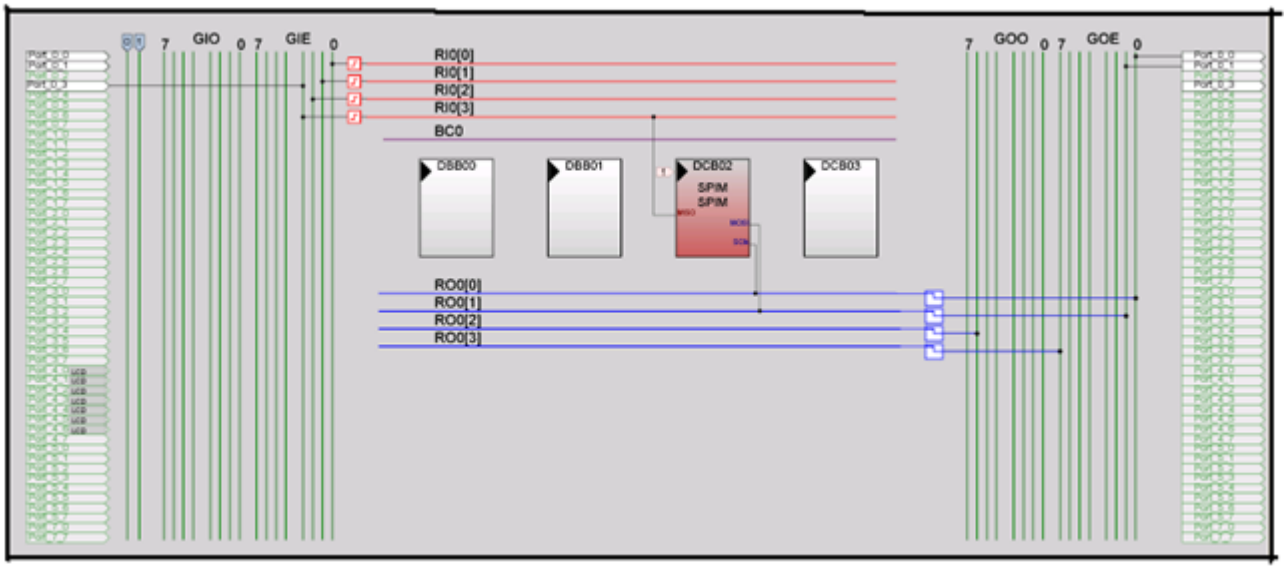


Figure 10. Resource Placement



Document History

Document Title: USB To SPI Bridge using PSoC1 - AN50989

Document Number: 001-50989

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2631861	XKJ/AESA	01/07/2009	New application note
*A	3147712	UDAY	01/19/2011	Updated PSoC USB Subsystem. Added USB Device Descriptors sub-section to the Device Configuration section. Updated Setup and Demonstration.
*B	3591735	DESH	04/19/2012	Title has been updated from USB To SPI BRIDGE USING PSOC(R) - AN50989 to USB To SPI BRIDGE USING PSOC1 - AN50989. Updated the abstract Corrected the Application Code flowchart. Changed the project name to "SPI_USB_Master" to be in sync with the attached project.
*C	4276632	KRIS	02/10/2014	Projects updated to PSoC Designer 5.4 Reference ANs added

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

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

PSoC is a registered trademark of Cypress Semiconductor Corp. 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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2009-2014. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.
Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
Use may be limited by and subject to the applicable Cypress software license agreement.