CE221119 – PSoC 6 MCU SCB I2C Slave

Objective

This code example demonstrates the operation of the PSoC® 6 MCU Serial Control Block (SCB) in I²C Slave mode. Two applications show the use of APIs to receive data from an I²C Master using two methods: callback and polling. It also demonstrates how to control the intensity of an LED using a TCPWM resource.

Requirements

Tool: ModusToolbox™ IDE 1.0; Bridge Control Panel

Programming Language: C

Associated Parts: All PSoC 6 MCU parts

Related Hardware: PSoC 6 BLE Pioneer Kit, PSoC 6 WiFi-BT Pioneer Kit, PSoC 6 WiFi-BT Prototyping Kit

Overview

The SCB in I²C Slave mode accepts command packets to control the intensity of an LED. The I²C Slave updates its read buffer with a status packet in response to the accepted command. Two applications in this example demonstrate the use of callback and polling methods.

In this example, a host PC running Cypress' Bridge Control Panel (BCP) software is used as the I²C Master. LED control is implemented using a TCPWM resource (configured as PWM). The intensity of the LED is controlled by changing the duty cycle of the PWM signal.

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly.

Note: The PSoC 6 BLE Pioneer kit and the PSoC 6 WiFi-BT Pioneer kit ship with KitProg2. ModusToolbox only works with KitProg3. Before using this code example, make sure that the kit is upgraded to KitProg3. See ModusToolbox Help > ModusToolbox IDE Documentation > User Guide; section PSoC 6 MCU KitProg Firmware Loader. If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".

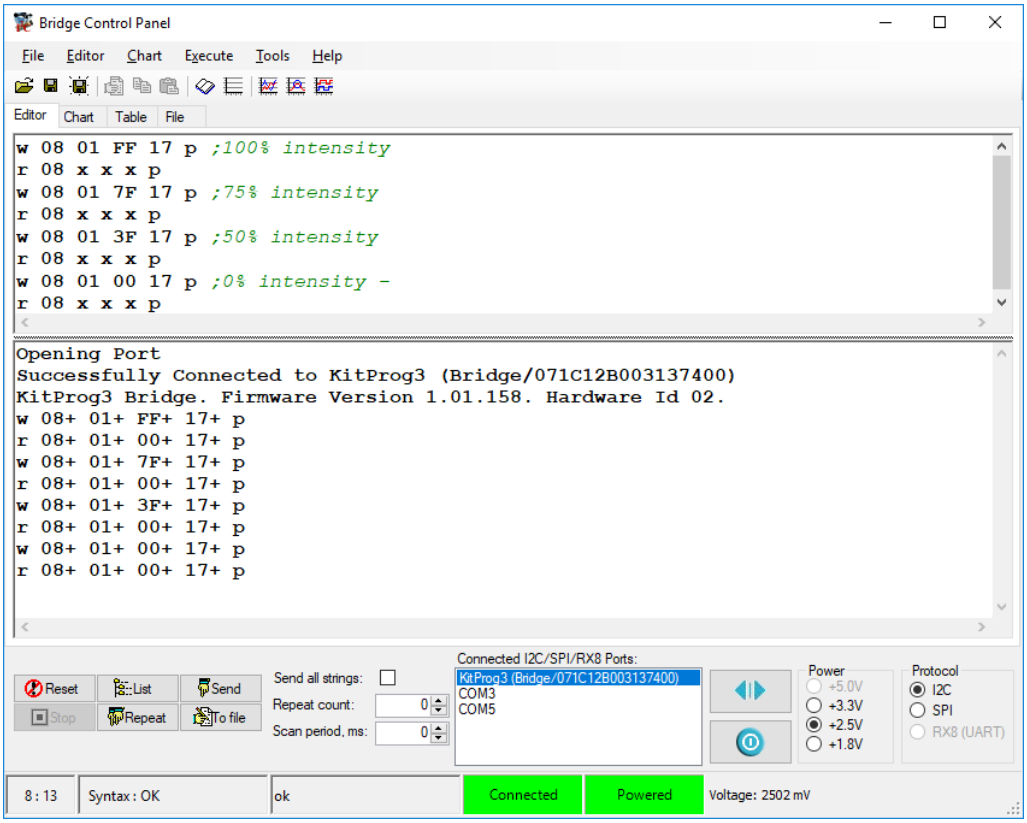
Software Setup

This section describes how to set up the BCP software for transmitting and receiving data over I²C.

The BCP software is installed automatically as part of the PSoC Programmer installation. Follow these steps to configure BCP:

1. Open the BCP from **Start** > **All Programs** > **Cypress** > **Bridge Control Panel <version>** > **Bridge Control Panel <version>**.
2. Select **KitProg3/<serial_number>** under **Connected I2C/SPI/RX8 Ports** (see Figure 1). Note that the PSoC 6 Kit must be connected to the USB port of your computer.

Figure 1. Bridge Control Panel



3. Select **Tools** > **Protocol Configuration**, navigate to the **I2C** tab, and set the **I2C speed** to '100 kHz'. Click **OK**.

BCP is now ready for transmitting and receiving data.

Operation

1. Connect the kit to your PC using the provided USB cable.
2. Import the code example into a new workspace. If you aren't familiar with this process, see KBA225201. Choose the *modus.mk* file according to the method you want to use; callback or polling.
3. Program the PSoC 6 MCU device. In the project explorer, select the **mainapp** project. In the Quick Panel, scroll to the **Launches** section and click the **Program (KitProg3)** configuration.
4. Configure the BCP software as described in the section Software Setup.
5. In the **Editor** tab of BCP, type the command to send LED intensity data and then click **Send**. Observe that the LED turns ON with the specified intensity.

   The command format that is used to write the data to the Slave if BCP is used as the I^2C Master is shown below. The symbol 'SOP' means 'start of packet' and 'EOP' means 'end of packet'.

Start for Write	Slave Address	SOP	LED TCPWM Compare Value	EOP	Stop
W	(0x08)	(0x01)	(0x00 to 0xFF)	(0x17)	p

   For example, sending the command '`w 08 01 FF 17 p`' will turn the LED ON with full intensity; sending the command '`w 08 01 00 17 p`' will turn the LED OFF.

6. Type the command '`r 08 x x x p`' to read the status of the write performed.

   The following is the command format to read the status form the Slave's read buffer. The symbol 'x' denotes one byte to read from the Slave's read buffer. In this example, three bytes are read from the Slave.

Start for Read	Slave Address	Read SOP (0x01)	Read Status (0x00 = Success) (0xFF =Fail)	Read EOP (0x17)	Stop
r	(0x08)	x	X	X	p

   After each command is sent, the status packet must be read from the read buffer of the Slave by sending the '`r 08 x x x p`' command. If the packet read is in the format '`r 08 01 00 17 p`', the status is set as 'success'; if the packet read is '`r 08 01 FF 17 p`', the status is set as 'fail' for the command sent by the I²C Master. See Figure 1 for more information.

Debugging

You can debug the example to step through the code. Use a **Program+Debug** configuration. If you are unfamiliar with how to start a debug session with ModusToolbox IDE, see KBA224621.

Design and Implementation

This code example implements an I²C Slave using an SCB resource. The I²C Slave is configured with a 3-byte write buffer, which can be accessed by the I²C Master to write commands. In addition, a 3-byte read buffer is configured to read the status of the Slave by the Master. The BCP software is used as the I²C Master.

The first byte in the write buffer contains the Start of Packet (SOP) value, the next byte contains the LED's TCPWM compare value, and the third byte in the write buffer is the End of Packet (EOP) value. The Slave updates the Master's read buffer with the status packet. The first byte of the status packet is SOP, the second byte contains the status where the value 0x00 means success and 0xFF means failure for the command data sent by the Master, and the third byte in the read buffer is EOP.

To control the intensity of the LED, a PWMs with a period value of 255 is used. The duty cycle of the PWM is controlled in firmware and specified by the I²C Master.

I²C Slave Using Callback Method

In the callback method, data write and read complete events from the Master are handled through interrupts. SCB I²C APIs are used to configure the SCB resource to act as an I²C Slave, and to configure its relevant interrupts to handle data write and read complete events by the Master.

I²C Slave Using Polling Method

In the polling method, the system checks the status of the I²C Slave continuously. SCB I²C APIs are used to configure the SCB resource to act as an I²C Slave and to know the status of the Slave. The Slave gets to know whether the Master has completed writing data to the write buffer or completed reading status from the read buffer. If the Master has completed writing data to the write buffer, the data written is checked for SOP and EOP; if it is correct, the LED's PWM compare value is updated with the data written by the Master.

Resources and Settings

Table 1 lists some of the ModusToolbox resources used in the example, and how they are used in the design. The *design.modus* file contains all the configuration settings. For example, for pin usage and configuration, open the **Pins** tab of the design file.

To see all the settings, review the *design.modus* file in the project.

Table 1. ModusToolbox Resources

Resource	Alias	Purpose	Non-default Settings
SCB3	KIT_I2C	Provide I2C communication between the master and slave	**Clock:** 8 bit Divider 0 clk **SCL**: P6[0] **SDA**: P6[1]
TCPWM[0] Counter 3	KIT_LED2_PWM	Generate PWM signal	**Clock:** 8 bit Divider 1 clk **Period:** 255u **Compare**: 0u
GPIO P13[7]	KIT_LED2	Drive the PWM signal to LED	**Drive Mode:** Strong Drive, Input buffer off **Digital Output**: TCPWM[0] 32-bit Counter 3 pwm_n

Reusing This Example

This example is configured for the supported kit(s). To port the design to a different PSoC 6 MCU device, right-click an application project and choose **Change Device**. If changing to a different kit, you may need to reassign pins.

Table 2. Device and Pin Mapping Across PSoC 6 MCU Kits

Kit Name	Device Used	LED	I2C_SCL	I2C_SDA
CY8CKIT-062-WiFi-BT	CY8C6247BZI-D54	P13[7]	P6[0]	P6[1]
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	P13[7]	P6[0]	P6[1]
CY8CPROTO-062-4343W	CY8C624ABZI-D44	P13[7]	P6[0]	P6[1]

In some cases, a resource used by a code example (for example, an IP block) is not supported on another device. In that case, the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on which resources a device supports.

Related Documents

Application Notes	
AN221774 – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first ModusToolbox application and PSoC Creator project
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first ModusToolbox application and PSoC Creator project
AN215656 – PSoC 6 MCU: Dual-CPU System Design	Describes the dual-CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-CPU design
Code Examples	
Visit the Cypress GitHub site for a comprehensive collection of code examples using ModusToolbox IDE.	
Device Documentation	
PSoC 6 MCU: PSoC 62 Datasheet	PSoC 6 MCU: PSoC 62 Architecture Technical Reference Manual (TRM)
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kits	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	
CY8CPROTO-062-4343W PSoC 6 WiFi BT Prototyping Kit	

Tool Documentation	
ModusToolbox IDE	The Cypress IDE for IoT designers

Cypress Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right device, and quickly and effectively integrate the device into your design.

For the PSoC 6 MCU devices, see KBA223067 in the Cypress community for a comprehensive list of PSoC 6 MCU resources.

Document History

Document Title: CE221119 – PSoC 6 MCU SCB I2C Slave

Document Number: 002-25531

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6369414	SNVN	11/21/2018	New code example

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 | Code Examples | Projects | Videos | Blogs | Training| Components

Technical Support

cypress.com/support

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