CE222306 – PSoC 4 I2C Master

Objective

This example demonstrates the use of the PSoC® Creator Serial Communication Block (SCB) Component for PSoC 4 in I²C master mode. Two projects demonstrate the use of high-level and low-level functions to communicate with an I²C slave.

Overview

This code example consists of two projects:

- I2C_Master_High_Level – implements the I²C Master device to send commands to the I²C Slave device and read the status of the command execution: success or error. The RGB LED shows the result of the command execution: success – green; error – red. The API that is used transfers entire buffers to the slave device, eliminating the need for byte-wise communication.
- I2C_Master_Low_Level – Implements the same functionality as the High_Level project but uses lower-level firmware which performs writes and reads on a byte-by-byte basis. The firmware also keeps track of write and read completion.

Requirements

Tool: PSoC Creator™ 4.2

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 4 Parts

Related Hardware: CY8CKIT-042 PSoC 4 Pioneer Kit

Hardware Setup

This example uses the CY8CKIT-042 kit to demonstrate the I²C master device. However, other PSoC 4 kits with I²C support can be used. To get the most out of the example, it is recommended that a second PSoC 4 kit programmed with the EZI2C slave code example CE195362 is connected to the CY8CKIT-042.

If you use two kits, connect a ground pin (**GND**) and the I²C pins (**SCL** and **SDA**) together between the two kits. Table 1 below shows the I²C pins on each kit. Note that I²C bus lines are open drain and must be pulled up to V_{DD} using resistors as Figure 1 shows. Resistor values can be calculated from Chapter 7 of the NXP I²C bus specification, UM10204, available here.

Figure 1. I2C Pull-up Resistor Schematic

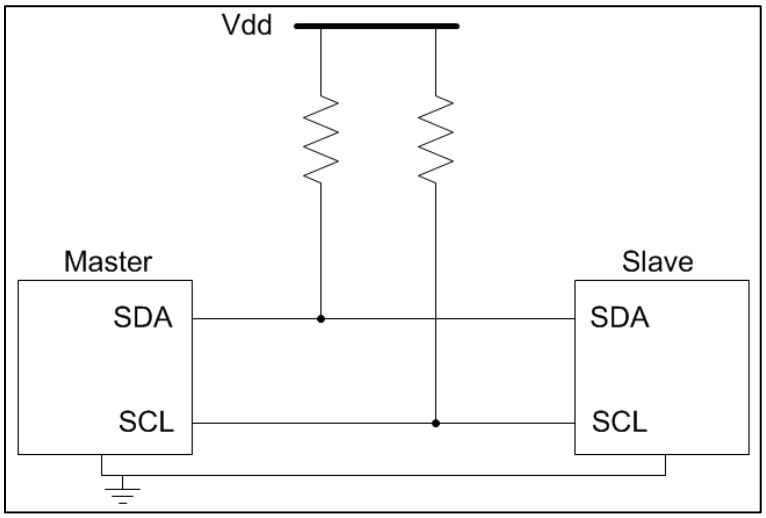


Table 1. Pin Assignments

Development Kit	\I2C:scl\	\I2C:sda\
CY8CKIT-041-40XX	P3[0]	P3[1]
CY8CKIT-041-41XX		
CY8CKIT-042	P4[0]	P4[1]
CY8CKIT-042-BLE	P3[5]	P3[4]
CY8CKIT-042-BLE-A		
CY8CKIT-044	P4[0]	P4[1]
CY8CKIT-046	P4[0]	P4[1]
CY8CKIT-048	P4[0]	P4[1]
CY8CKIT-149	P3[0]	P3[1]

Operation

1. Plug the CY8CKIT-042 kit into your computer's USB port.
2. Build and program either of the PSoC4 I2C Master projects into the CY8CKIT-042. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help.
3. If a second PSoC 4 Kit is available, program CE195362 into the second kit.
   a. Connect the two kits following the instructions in Hardware Setup.
   b. Press the reset button on each kit and observe the LED on the slave kit changing colors.
4. If a second PSoC 4 kit is not available, use an oscilloscope to observe the data and clock on the I^2C bus lines. Connect oscilloscope probes to each of the I^2C pins and ground each probe to a pin labeled GND.

Design and Implementation

In each project, CY8CKIT-042 acts as an I^2C master device sending commands to a PSoC 4 device programmed as an I^2C slave using code example CE195362.

The application uses two I^2C buffers to communicate between the master and slave device. The write buffer carries the commands to be written and the status buffer contains the status of the previous transaction. In each buffer, the first and last bytes contain specific values which are used to check for correct packet format. The remaining positions are reserved for commands and status values.

The command packet is organized as shown below.

Start of Packet (SOP)	Red Compare Value	Green Compare Value	Blue Compare Value	End of Packet (EOP)
(0x01)	(0x00 – 0xFF)	(0x00 – 0xFF)	(0x00 – 0xFF)	(0x17)

After a command packet is written to the slave device, the slave updates the contents of the status buffer indicating either success or failure of the previous write operation. The status packet is organized as shown below.

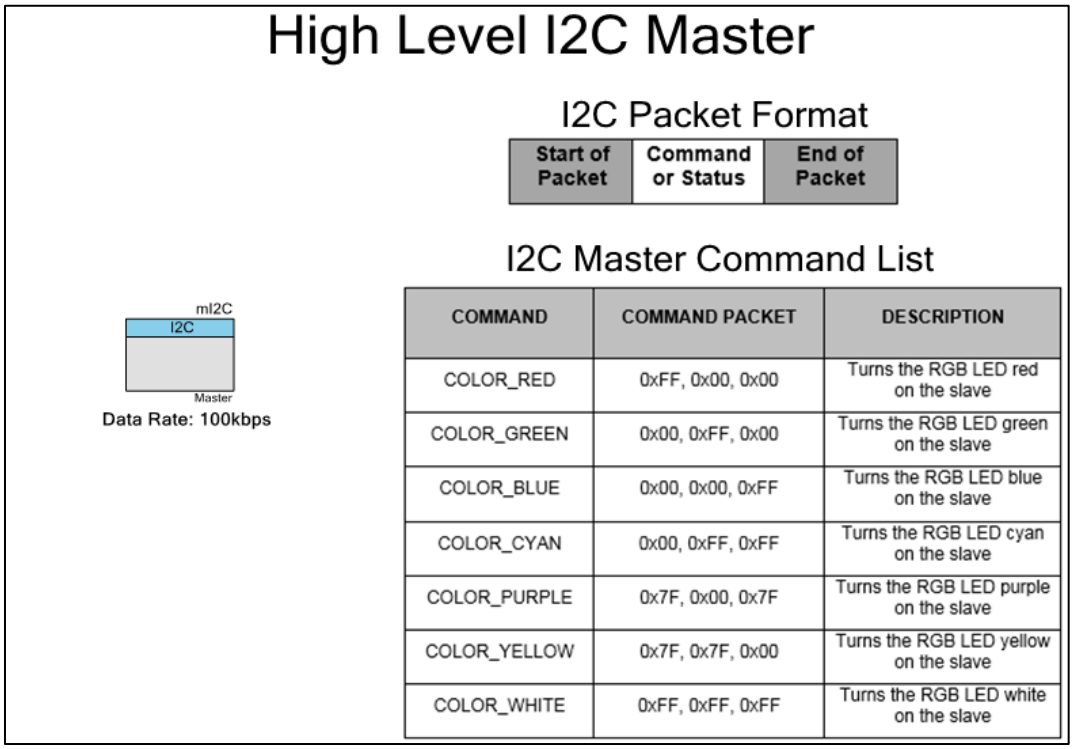
SOP	Status	EOP
(0x01)	(0x00 or 0xFF)	(0x17)

The firmware for both projects begins by initializing buffers for reads and writes. Firmware then writes the red compare value to the write buffer and transfers the buffer to the slave. After a successful write, the master reads status data from the slave into the read buffer. If this read is successful, the firmware waits for a half-second before cycling to the next LED color. The process repeats indefinitely.

I2C_Master_High_Level Design

The I2C_Master_High_Level project uses the SCB I2C Component in master mode with a data rate of 100 kbps. It uses high-level functions from the SCB API to write the command buffer and read the status from the slave. The high-level functions handle start and stop conditions and abstract away from the byte-wise transactions that lower-level functions use. The schematic in Figure 2 shows the available commands.

Figure 2. Schematic for the I2C_Master_High_Level



Note: The Schematic for the I2C_Low_Level project is the same.

I2C_Master_Low_Level Design

The SCB Component is configured as an I^2C master with a data rate of 100 kbps. The firmware uses low-level functions that send start and stop conditions before and after each write or read operation. Firmware transmits the write buffer by iterating through each byte in the buffer. Firmware reads the read buffer in a similar manner. The low-level API may be preferred if the user needs a higher level of control in the timing of transactions.

Components and Settings

Table 2 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 2. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
I2C (SCB mode)	mI2C	Enables I^2C communication as master	**Mode**: Master

For information on the hardware resources used by a Component, see the Component datasheet.

Reusing This Example

This example is designed for use with the CY8CKIT-042 pioneer kit. To port the design to a different PSoC 4 device and/or kit, change the target device using the Device Selector (**Project** > **Device Selector**) and update the pin assignments in the Design Wide Resources Pins settings as needed.

For this example, ensure that the I^2C pins **SCL** and **SDA** as well as a **GND** pin are connected properly between the master and slave devices. Pull the I^2C bus lines up to V_{DD} using resistors as shown in Figure 1. Resistor values can be calculated from Chapter 7 of the NXP I^2C bus specification, UM10204, available here.

In some cases, a resource used by a code example (for example, the CapSense hardware) 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 what a particular device supports.

Related Documents

Application Notes	
AN86526 – PSoC 4 and PSoC Analog Coprocessor I2C Bootloader	Describes an I^2C-based bootloader and demonstrates how to create an I^2C-based bootloadable project.
AN86439 – PSoC 4 – Using GPIO Pins	Describes how to use PSoC 4 GPIO pins with various use case examples
Code Examples	
CE195362 – PSoC 4 EZI2C Slave with Serial Communication Block (SCB)	Demonstrates the basic usage of the EZI2C Slave Component with the Serial Communication Block
CE224599 – I2C Slave using a Serial Communication Block (SCB) with PSoC 4	Demonstrates the basic operation of the I2C slave (SCB mode) Component.
PSoC Creator Component Datasheets	
Serial Communication Block (SCB)	Supports serial communication usage
Device Documentation	
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals
Development Kit Documentation	
PSoC 4 Kits	

Document History

Document Title: CE222306 – PSoC 4 I2C Master

Document Number: 002-22306

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5985995	MYKZTMP1	01/12/2018	New code example
*A	6469732	BFMC	02/07/2019	Converted to Master Only Updated I^2C command format Updated pin table Moved SCB to top of component list

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 Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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