

SCB_I2cCommSlave Example Project

1.20

Features

- Communication between I²C master and slave
- Simple packet protocol with command and status byte

General Description

This example project demonstrates the basic operation of the I²C slave (SCB mode) component. The I²C slave accepts a packet with a command from the I²C master to control the RGB LED color. The I²C slave updates its buffer with a status packet in response to the accepted command.

The I²C slave in this project is designed to communicate with an I²C master in either of the following configurations:

- The I²C master already present on the development kit. The master can be controlled using the provided Bridge Control Panel software.
- The SCB_I2cCommMaster Example project, programmed onto another device to serve as the master.

Development Kit Configuration

This example project is designed to run on the CY8CKIT-042 kit from Cypress Semiconductor. A description of the kit, along with more example programs and ordering information, can be found at <http://www.cypress.com/go/cy8ckit-042>.

The project requires configuration settings changes to run on other kits from Cypress Semiconductor. Table 1 is the list of the supported kits. To switch from CY8CKIT-042 to any other kit, change the project's device with the help of Device Selector called from the project's context menu.

Table 1. Development Kits vs Parts

Development Kit	Device
CY8CKIT-040	CY8C4014LQI-422
CY8CKIT-042	CY8C4245AXI-483
CY8CKIT-042-BLE	CY8C4247LQI-BL483
CY8CKIT-044	CY8C4247AZI-M485

The pin assignments for the supported kits are in Table 2.

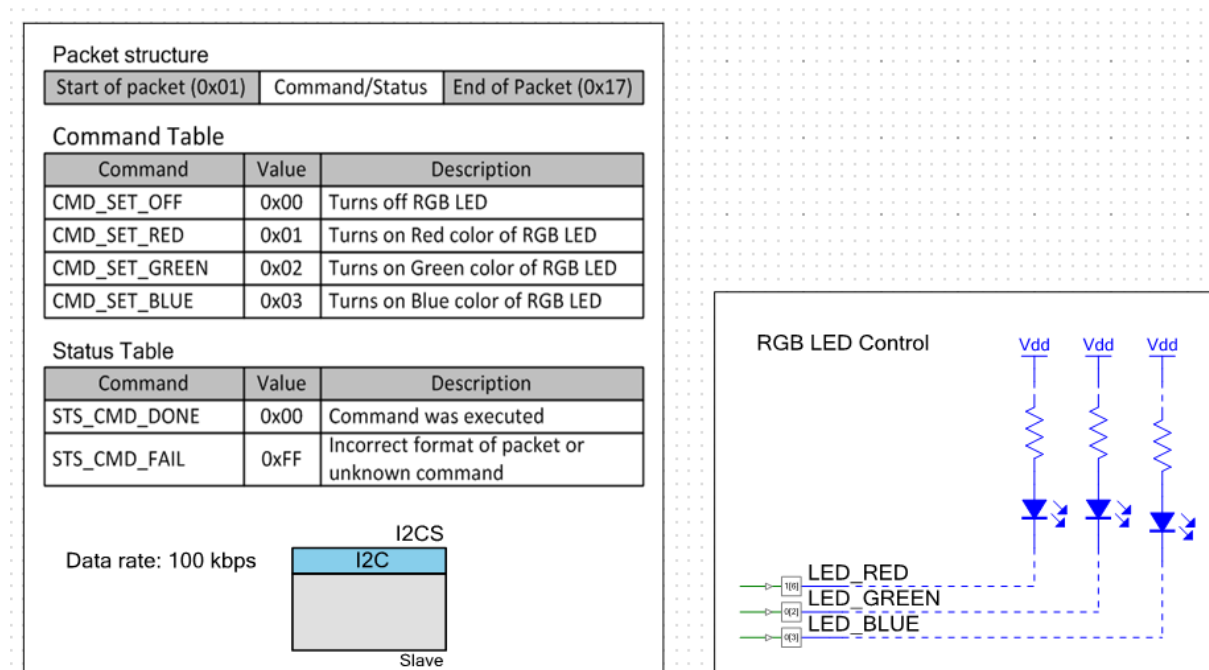
Table 2. Pin Assignment

Pin Name	Development Kit			
	CY8CKIT-040	CY8CKIT-042	CY8CKIT-042 BLE	CY8CKIT-044
\I2CS:scl\	P1[2]	P3[0]	P3[5]	P4[0]
\I2CS:sda\	P1[3]	P3[1]	P3[4]	P4[1]
LED_BLUE	P0[2]	P0[3]	P3[7]	P6[5]
LED_GREEN	P1[1]	P0[2]	P3[6]	P2[6]
LED_RED	P3[2]	P1[6]	P2[6]	P0[6]

Project Configuration

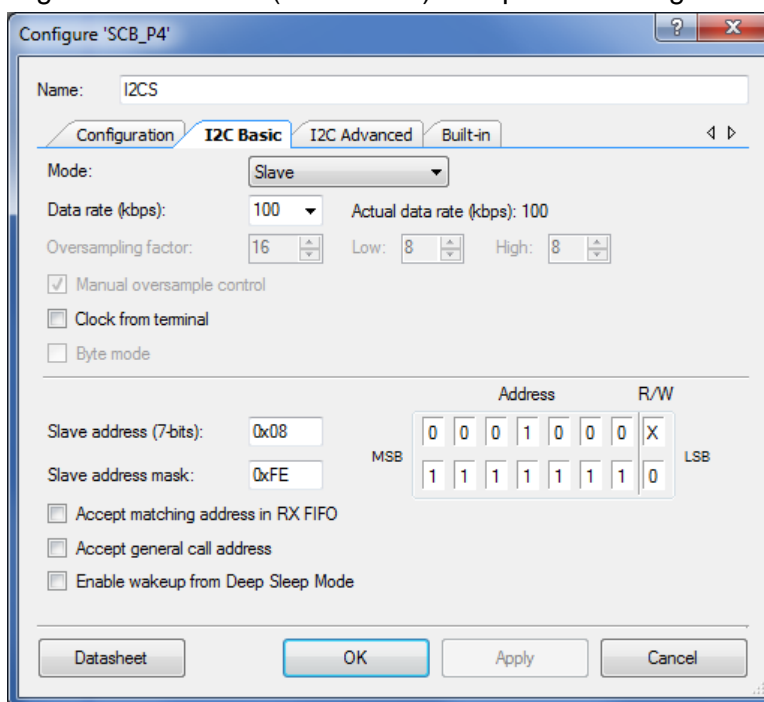
The example project consists of the I²C slave (SCB mode) and pin components. The design schematic is shown in Figure 1. The blue annotation components are used to represent the RGB LED installed on the kit. The three pin components are used to control the LED color. The kit provides connection between the I²C slave (PSoC 4) and I²C master (PSoC 5LP) as well as a pull-up resistor required for the I²C bus operation. The Bridge Control Panel software is provided to control the I²C master.

Figure 1. Example Project Design Schematic



The I²C slave is configured to operate with the data rate of 100 kbps and responds to address 0x08 (7-bits). The component configuration window is shown below.

Figure 2. I2C Slave (SCB mode) Component Configuration



Project Description

In the main firmware routine, the I²C slave read and write buffers are configured when a component is started. The write buffer is exposed to the master to write a packet with a command and reads back the packet with a status. Interrupts are enabled to the CPU core as required by the I2C slave component for operation.

The I²C slave waits for communication from the I²C master. The main loop polls I2CS_I2CSlaveStatus() API continuously for a write or read completion event.

When a write completion event is reported, the write buffer content is checked against the valid packet. The packet structure and table with commands are shown below. The basic checks of the packet structure are done: the length of the packet, the start and end of the packet byte. If packet considered as valid the command is retrieved and passed to be executed. The result of a command execution is a change of the LED color. Initially the LED is turned off. The status is updated with a successful command execution or failure in the case when a command is unknown or the packet is considered as invalid. The table with return statuses is shown below. The packet with a status is exposed to the master in the slave read buffer.

When a read completion event is reported, the slave just exposes the read buffer to the master again. The master may not read the packet with a status at all and then it just sends a next packet with a command.

Packet structure

Start of packet (0x01)	Command/Status	End of Packet (0x17)
------------------------	----------------	----------------------

Table 3. Command constants

Command	Value	Description
CMD_SET_OFF	0	Turns off RGB LED
CMD_SET_RED	1	Turns on Red color of RGB LED
CMD_SET_GREEN	2	Turns on Green color of RGB LED
CMD_SET_BLUE	3	Turns on Blue color of RGB LED

Table 4. Status constants

Status	Value	Description
STS_CMD_DONE	0x00	Command was executed
STS_CMD_FAIL	0xFF	Incorrect format of packet or unknown command

The packets with a command and status are converted into the following I²C master transfers. The packet with a command has a write direction set in the address byte and the packet with a status has a read direction set appropriately.

Packet with command

S	ADDR = 0x08	W	A	SOP = 0x01	A	Command	A	EOP = 0x17	A	P
---	-------------	---	---	------------	---	---------	---	------------	---	---

Packet with status


S	ADDR = 0x08	R	A	SOP = 0x01	A	Status	A	EOP = 0x17	\bar{A}	P
---	-------------	---	---	------------	---	--------	---	------------	-----------	---

☐ - Master drives the bus ☒ - Slave drives the bus

Expected Results

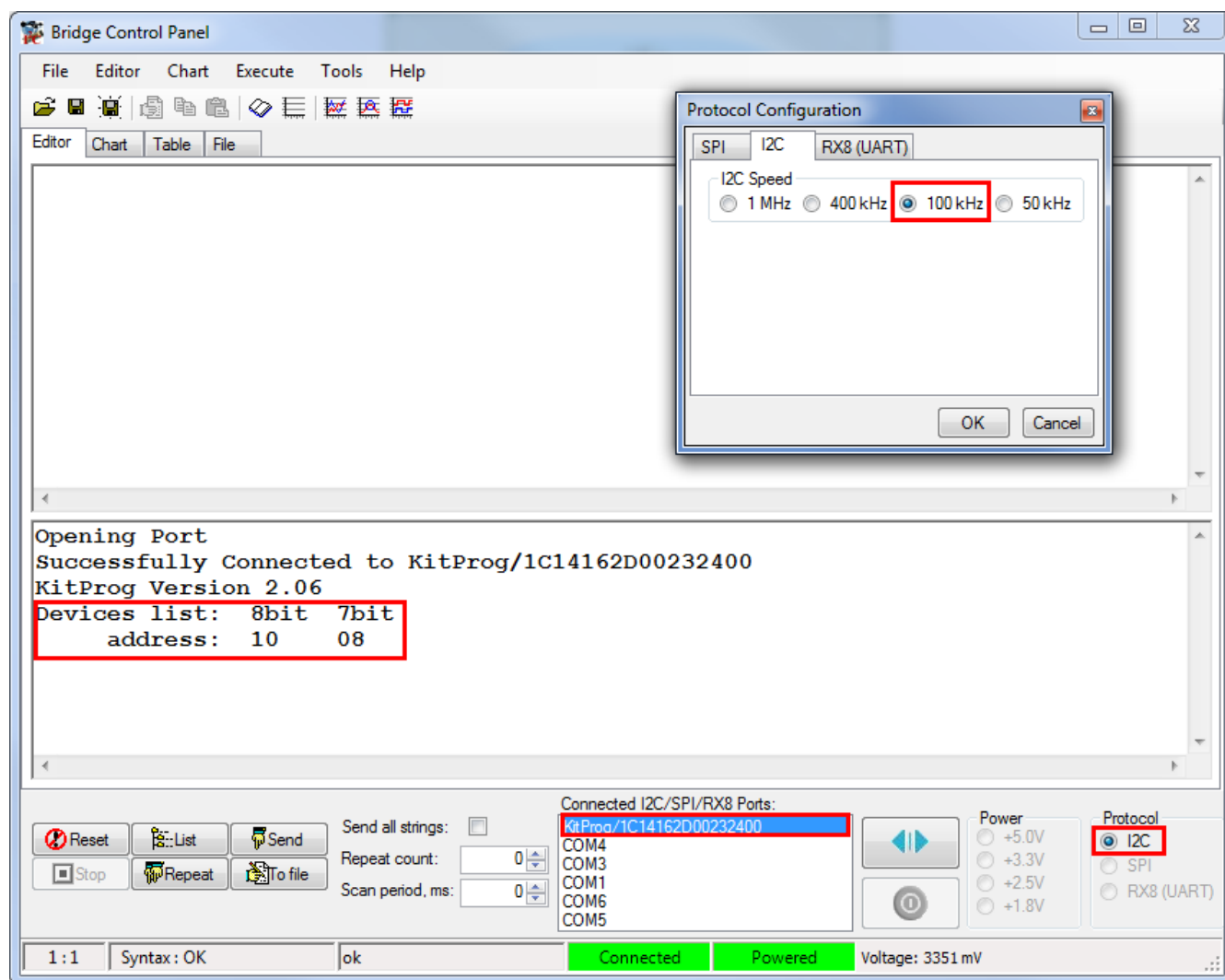
Build example project and program into the device.


Run the Bridge Control Panel software which is shipped with the PSoC Creator. It is used to control the I²C master implemented on the PSoC 5LP which is available on the kit. Follow the steps below to setup communication between the master and slave:

1. Select the KitProg device into the list of the Connected Ports.
2. Make sure that the selected Protocol is I²C.
3. Go to Tools->Protocol Configuration and select I2C Speed 100kHz.
4. Press the List button  to make sure that the I²C slave device with address 0x08 (7-bits) is available for communication¹.

¹ Other I²C devices can be connected to the I²C bus. The addresses of these devices are shown after list operation completion. Refer to the development kit documentation for more information about other I²C devices available on the kit.

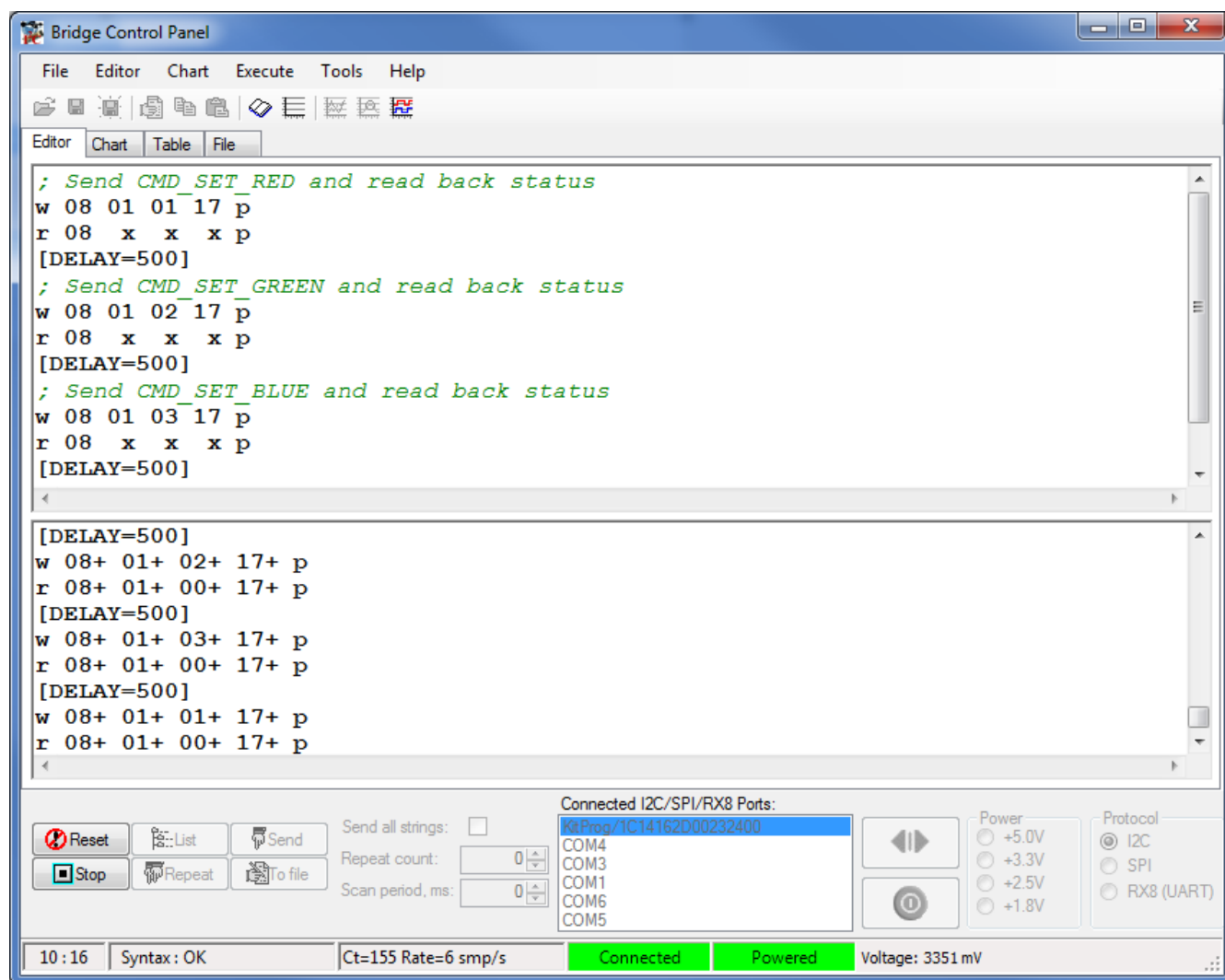
Figure 3. Bridge Control Panel I2C Master Setup



5. To load master commands for communication with the I²C slave use Open icon  . Navigate to BCP_Master_I2cCmd.iic file which is attached to the workspace and open it. The commands should appear in the Edit window.
6. There are two options of a master transfer execution.
 - A single command execution: set the cursor to the line with the command into the Edit window and press Enter. The RGB LED should change its color accordingly to the executed command.
 - Repeat the command execution: select a number of commands and press the Repeat button. The RGB LED should change its color accordingly to the executed commands.

The delays are added between commands to notice a LED color change.

Figure 4. Repeat Command Execution Result



© Cypress Semiconductor Corporation, 2015. 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.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any 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.