PSoC 4 I2C Master

About this document

Scope and purpose

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.

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

Table of contents

1 Overview

This code example consists of two projects:

- **I2C_Master_High_Level** – implements the I^{2}C Master device to send commands to the I^{2}C 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.

2 Hardware setup

This example uses the CY8CKIT-042 kit to demonstrate the I^{2}C master device. However, other PSoC 4 kits with I^{2}C 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 I^{2}C pins (**SCL** and **SDA**) together between the two kits. The table below shows the I^{2}C pins on each kit. Note that I^{2}C bus lines are open drain and must be pulled up to V_{DD} using resistors as shown. Resistor values can be calculated from Chapter 7 of the NXP I^{2}C bus specification, UM10204, available **here**.

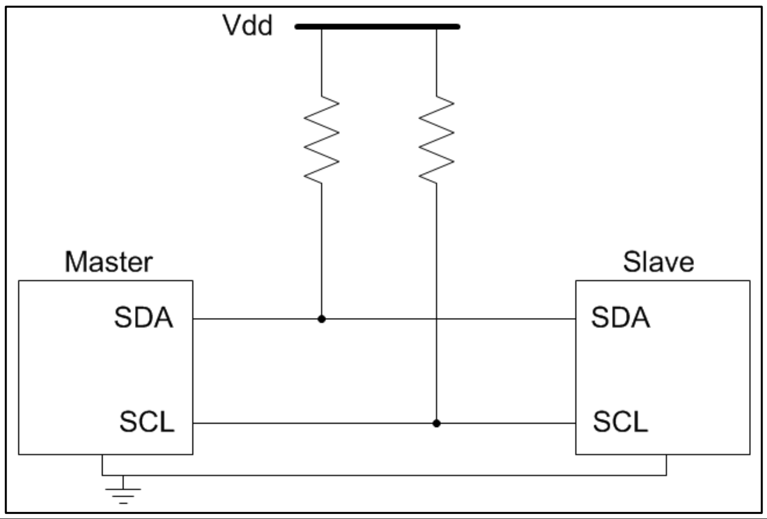


Figure 1 I2C Pull-up resistor schematic

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]

3 Operation

1. Plug the CY8CKIT-042 kit into your computer's USB port.
2. Build and program either of the PSoC 4 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^{2}C bus lines. Connect oscilloscope probes to each of the I^{2}C pins and ground each probe to a pin labeled GND.

4 Design and implementation

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

The application uses two I^{2}C 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)

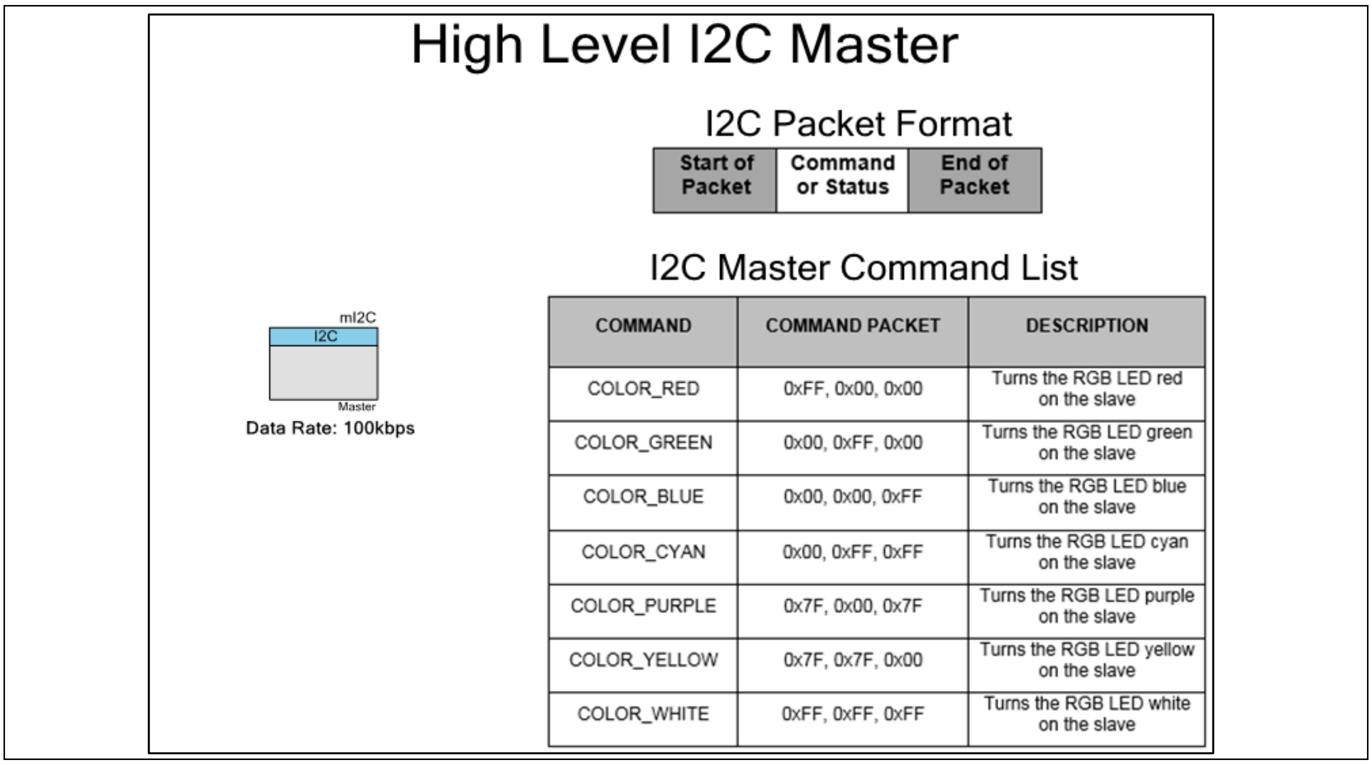
After a command packet is written to the slave device, the slave updates the contents of the status buffer indicating either success or failure of the previous write operation. The status packet is organized as shown below.

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 compare values of each LED 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.

4.1 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 shows the available commands.



High Level I2C Master

I2C Packet Format

Start of Packet	Command or Status	End of Packet

I2C Master Command List

COMMAND	COMMAND PACKET	DESCRIPTION
COLOR_RED	0xFF, 0x00, 0x00	Turns the RGB LED red on the slave
COLOR_GREEN	0x00, 0xFF, 0x00	Turns the RGB LED green on the slave
COLOR_BLUE	0x00, 0x00, 0xFF	Turns the RGB LED blue on the slave
COLOR_CYAN	0x00, 0xFF, 0xFF	Turns the RGB LED cyan on the slave
COLOR_PURPLE	0x7F, 0x00, 0x7F	Turns the RGB LED purple on the slave
COLOR_YELLOW	0x7F, 0x7F, 0x00	Turns the RGB LED yellow on the slave
COLOR_WHITE	0xFF, 0xFF, 0xFF	Turns the RGB LED white on the slave

Figure 2 Schematic for I2C_Master_High_Level

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

4.2 I2C_Master_Low_Level design

The SCB Component is configured as an I^{2}C 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.

4.3 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^{2}C communication as master	**Mode**: Master

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

5 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^{2}C pins **SCL** and **SDA** as well as a **GND** pin are connected properly between the master and slave devices. Pull the I^{2}C 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^{2}C 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

[1] **AN86526** – PSoC 4 I2C Bootloader: Describes an I2C-based bootloader and demonstrates how to create an I2C-based bootloadable project.

[2] **AN86439** – PSoC 4 – Using GPIO Pins: Describes how to use PSoC 4 GPIO pins with various use case examples

Code examples

[3] **CE195362** – PSoC 4 EZI2C Slave with Serial Communication Block (SCB): Demonstrates the basic usage of the EZI2C Slave Component with the Serial Communication Block

[4] **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

[5] **Serial Communication Block (SCB)**: Supports serial communication usage

Device documentation

[6] **PSoC 4 Datasheets**

[7] **PSoC 4 Technical Reference Manuals**

Development kit documentation

[8] **PSoC 4 Kits**

Revision history

Revision	Date of release	Description of change
**	2018-01-12	New code example
*A	2019-02-07	Converted to Master Only Updated I^{2}C command format Updated pin table Moved SCB to top of component list
*B	2021-03-08	Updated hyperlinks for Related Documents Updated Design and Implementation for High Level project

Edition 2021-03-08

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-22306 Rev. *B