SCB_UnconfiguredComm Example Project

1.10

Features

- Illustrates dynamic run time configuration using Unconfigured mode of SCB component
- I^2C and UART communication using one pair of IO's

General Description

This example project demonstrates the capability of the SCB component to be reconfigured between multiple communication interfaces during run time. This is done using the Unconfigured mode of the SCB component. In this example, the component switches between the I2C and UART modes to execute SCB_UartComm and SCB_I2cCommSlave example projects using a shared pair of communication IO's.

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-041	CY8C4045AZI-S413 / CY8C4146AZI-S433
CY8CKIT-042	CY8C4245AXI-483
CY8CKIT-042-BLE	CY8C4247LQI-BL483
CY8CKIT-044	CY8C4247AZI-M485
CY8CKIT-046	CY8C4248BZI-L489
CY8CKIT-048	CY8C4445AZI-483

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

IMPORTANT: make sure that the **HFCLK** frequency is **24 or 12 MHz** after device is selected for correct code example operation.

Table 2. Pin Assignment

Pin Name	Development Kit					
	CY8CKIT-041	**CY8CKIT-042**	**CY8CKIT-042 BLE**	**CY8CKIT-044**	**CY8CKIT-046**	**CY8CKIT-048**
\Comm:uart_rx_i2c_scl_spi_mosi\	P3[0]	P3[0]	–	P4[0]	P4[0]	P4[0]
\Comm:uart_tx_i2c_sda_spi_miso\	P3[1]	P3[1]	–	P4[1]	P4[1]	P4[1]
\Comm:uart_rx_i2c_sda_spi_mosi\	–	–	P1[4]	–	–	
\Comm:uart_tx_i2c_scl_spi_miso\	–	–	P1[5]	–	–	
LED_RED	P3[4]	P1[6]	P2[6]	P0[6]	P5[2]	P1[4]
LED_GREEN	P2[6]	P0[2]	P3[6]	P2[6]	P5[3]	P2[6]
LED_BLUE	P3[6]	P0[3]	P3[7]	P6[5]	P5[4]	P1[6]
SW2	P0[7]	P0[7]	P2[7]	P0[7]	P0[7]	P0[3]

Note The project control files handle the pins placement automatically according to a selected PSoC.

The external connection is required to join appropriate UART and I^{2}C lines. The connection is summarized in table below:

Table 3. External Connections To Join UART and I^{2}C Lines

Pin Name	Development Kit			
	CY8CKIT-042 / CY8CKIT-044 / CY8CKIT-046	**CY8CKIT-042-BLE**	**CY8CKIT-041**	**CY8CKIT-048**
\Comm:uart_rx_i2c_scl_spi_mosi\	J8.9 (P5LP12_7)	–	J8.13 (P5LP12_6)	J20.13 (P5LP12_6)
\Comm:uart_tx_i2c_sda_spi_miso\	J8.10 (P5LP12_6)	–	J8.14 (P5LP12_7)	J20.14 (P5LP12_7)
\Comm:uart_rx_i2c_sda_spi_mosi\	–	J8.11 (P5LP12_1)	–	–
\Comm:uart_tx_i2c_scl_spi_miso\	–	J8.13 (P5LP12_0)	–	–

Note The I^{2}C signals are multiplexed differently for PSoC 4100 BLE/PSoC 4200 BLE devices than for other devices. That's why pins names are different in the tables above.

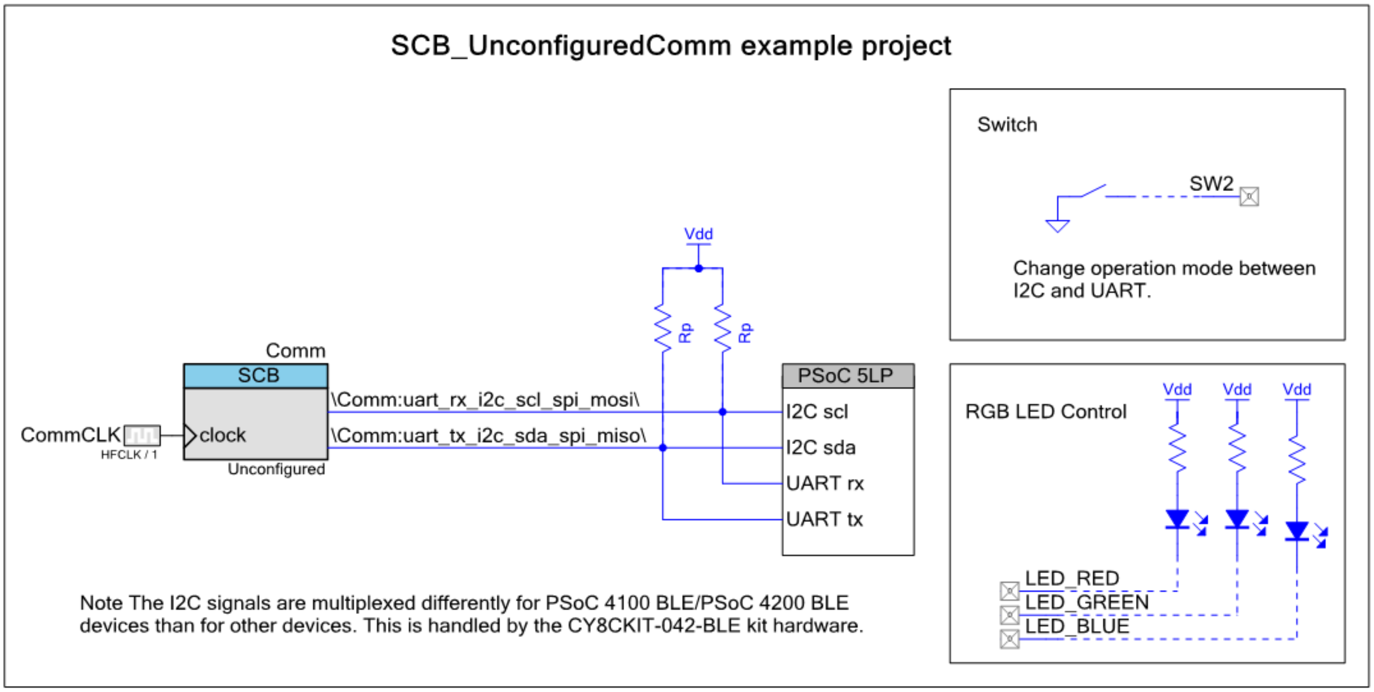
Project Configuration

The example project consists of the SCB component in the Unconfigured mode and pins components. The Unconfigured mode implies that the mode of the component will be configured during run time, allowing a single SCB component to support multiple communication interfaces. This example project switches between the I2C and UART modes. Figure 1 is the design schematic.

The blue annotation components represent RGB LED installed on the kit. Three pin components are used to control the LED color, using the fixed connections already provided on the kit. A single switch is used to allow a user to switch between the communication interfaces.

The kit provides connections between the I^2C or UART interfaces (PSoC 4) to the PSoC 5LP which can act as an I^2C master or USB-UART converter. The pull-up resistors on the I^2C bus are installed on the kit as well. The Bridge Control Panel software is provided to control the I^2C master operation and any serial terminal software (for example HyperTerminal or Putty) can be used to setup serial communication.

Figure 1. Example Project Design Schematic



The SCB component operates in the Unconfigured mode. Before enabling the component operation it must be configured to any of the following supported modes: I2C, EZI2C, SPI or UART.

Figure 2. SCB Unconfigured Mode Component Configuration (Configuration tab)

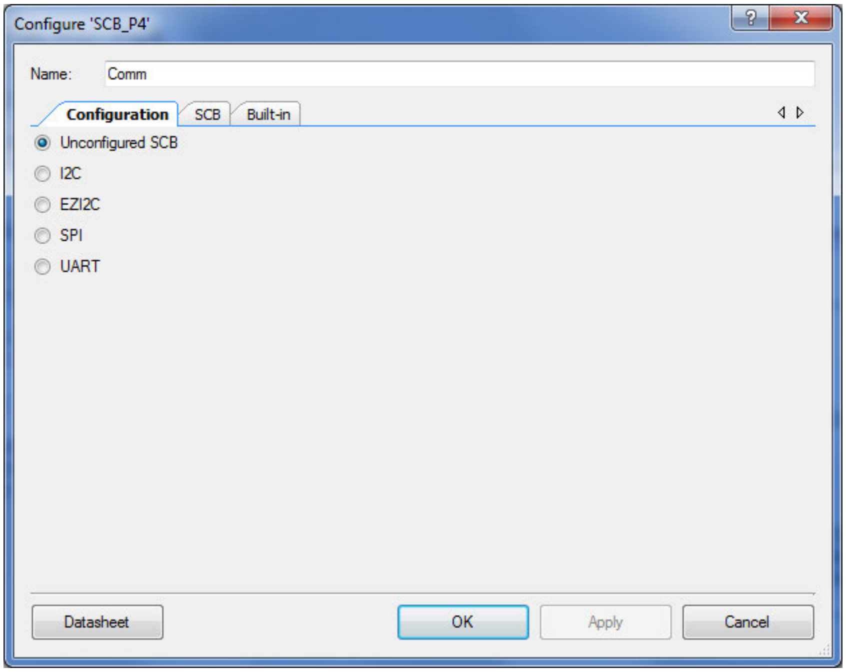
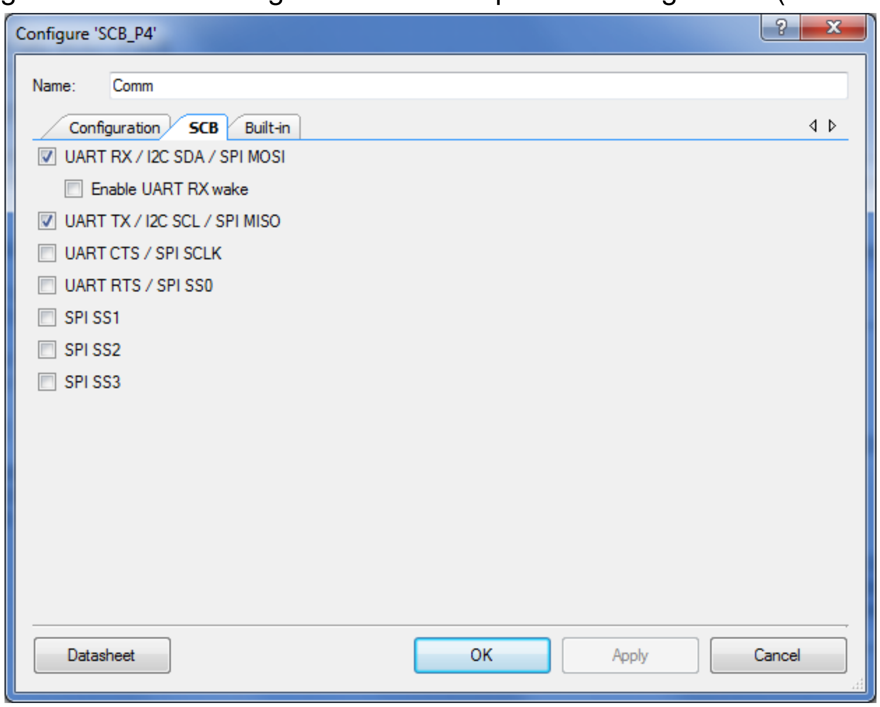


Figure 3. SCB Unconfigured Mode Component Configuration (SCB tab)



Project Description

In this project, the SCB component operation mode is controlled during run time so that Unconfigured mode can be selected. In this mode the component provides APIs to configure the component operation to support any of the following communication interfaces: I2C, EZI2C, SPI or UART. The pointers to the configuration structures are passed to the component API to allow run time configuration of the component. The fields of these structures are directly mapped to the component GUI controls. The user is responsible for allocating and initializing the configuration structures for the modes required for a given application. In addition the clock component frequency has to be set. The component clock frequency, along with the component oversampling parameter (if it applicable for the operation mode) defines the communication speed.

main.c contains a declaration and initialization of the UART and I2C configuration structures. The clock dividers are calculated to provide the desired component clock frequency for the project in UART and I2C modes.

The project defaults to the UART communication mode after the device is first programmed. The project remains in the UART mode until a switch push event is detected. This event is used to stop the UART interface and reconfigure SCB to operate in I2C mode. After reconfiguration is complete the I^2C slave interface is active. Another press of the switch will return the project to the UART mode.

The UART and I^2C operation interfaces in this project mimic the operation provided in the SCB_UartComm and SCB_I2cCommSlave example projects. For additional information on these projects, refer to their example project datasheets.

Expected Results

Project Setup:

1. Complete the external connections described in Table 3 to create a shared bus between I^2C and UART.
2. Build example project and program into the device.
3. After the device has been programmed, the SCB_UartComm example project executes. To change the example project to SCB_I2cCommSlave, push SW2. Additional switch presses toggle between the UART and I2C modes of operation.

Note While the I^2C interface is active, the terminal software may receive unrecognized characters. This is the expected behavior because I^2C and UART share the same connection.

SCB_UartComm example project execution details:

1. Connect a USB Mini B cable to the appropriate header of the kit. The kit enumerates as a **KitProg/KitProg2 USB-UART** and is available under Device Manager, Ports (COM & LPT). A communication port is assigned to KitProg/KitProg2 USB-UART. Note the **COM*X*** port number.

2. Run the available serial terminal software.
3. Select **COM*X*** (where *X* – is the communication port noted in the Device Manager that is assigned to **KitProg/KitProg2 USB-UART**).
4. Configure the serial terminal connection with the following parameters:
    a. Baud rate – 115200
    b. Data bits – 8
    c. Parity – None
    d. Stop bits –1
    e. Flow control – None
5. Build and program the SCB_UartComm example project into the device.
6. Observe the text displayed in the serial terminal output window:

   ****
   This is SCB_UartComm datasheet example project
   If you are able to read this text the terminal connection is configured
   correctly. Start transmitting the characters to see an echo in the terminal.

7. Every received character is sent back (echoed) to the terminal software.
8. When SW2 is pressed, the UART mode is exited with the following message:

   ****
   SCB_UartComm example ends its operation. The mode is changed to I2C and datasheet
   example project SCB_I2cCommSlave starts operation.
   Run Bridge Control Panel to communicate with I2C slave.
   ****

SCB_I2cCommSlave example project execution details:

Run the Bridge Control Panel software which is shipped with PSoC Creator. It controls the I^2C master implemented on the PSoC 5LP available on the kit. Follow the steps below to setup communication between the master and slave:

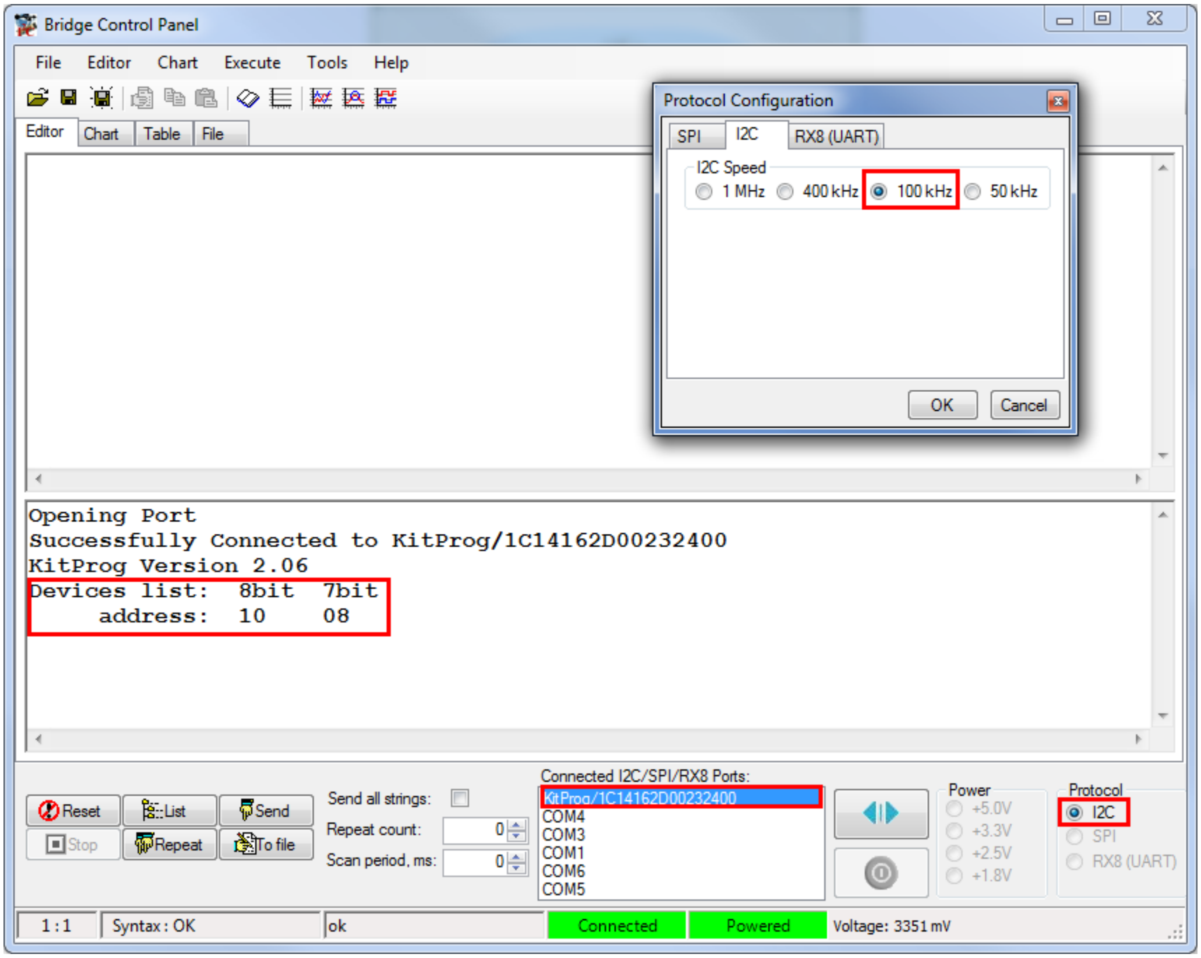
1. Select the KitProg device into the list of the Connected Ports.
2. Make sure that the selected Protocol is I^2C.
3. Go to Tools->Protocol Configuration and select I2C Speed 100 kHz.
4. Press the List button 

to make sure that the I^2C slave device with address 0x08 (7-bits) is available for communication[1].

---

1 Other I^2C devices can be connected to the I^2C bus. The addresses of these devices are shown after list operation completion. Refer to the development kit documentation for more information about other I^2C devices available on the kit.

Figure 4. Bridge Control Panel I2C Master Setup



5. To load master commands for communication with the I^{2}C slave use the Open icon . Navigate to BCP_Master_I2cCmd.iic file attached to the example project 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. For example, set the cursor to a line with a command in the Edit window and press Enter. RGB LED should change its color accordingly to the executed command.
   b. Repeated command execution. Select (highlight) a number of commands and press the Repeat button. The RGB LED should change its color according to the executed commands. Bridge Control Panel will repeatedly send a set of highlighted commands.

   Delays are added between commands to make the LED color changes noticeable.

Figure 5. Repeat Command Execution Result