CE95314 – PSoC® 3, PSoC 4, and PSoC 5LP EZI2C

Objective

These code examples demonstrate the usage of the EZI2C slave and I^2C master Components in PSoC 3, PSoC 4, and PSoC 5LP.

Overview

These code examples show how two I^2C Components – EZI2C slave and I^2C master – communicate with each other. Normally, these Components would be on separate devices, but for this example project, they are on the same PSoC chip. An off-chip connection is made between them.

There are two examples:

- For PSoC 3 and PSoC 5LP, running on a kit with two buttons and a character LCD, such as the Cypress CY8CKIT-030 and CY8CKIT-050 kits.
- For PSoC 4200, running on the Cypress CY8CKIT-042 kit, which has one button and an RGB LED.

Each I2C Component maintains its own data buffer. Note that an EZI2C buffer can be defined such that only the first N bytes are writeable by the master and the remaining bytes are read-only. This functionality is demonstrated in this example.

Requirements

Tool: PSoC Creator™ 4.1

Programming Language: C: GCC 5.4-1026-q2-update or MDK/armcc for PSoC 4200 and PSoC 5LP; DP 8051 Keil 9.5.1 for PSoC 3

Associated Parts: All PSoC 3, PSoC 4200, and PSoC 5LP parts

Related Hardware: CY8CKIT-030, CY8CKIT-042, CY8CKIT-050

Design

Figure 1 shows the code example design for PSoC 3 and PSoC 5LP, and Figure 2 shows the example design for PSoC 4200.

Figure 1. EZI2C Code Example for PSoC 3 and PSoC 5LP

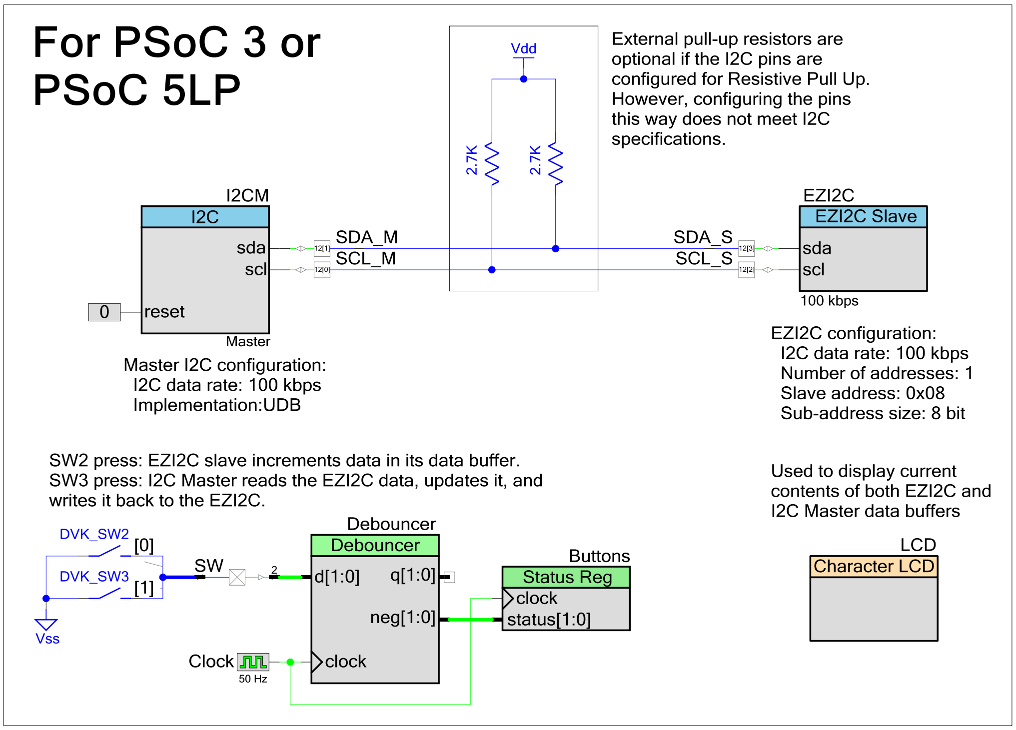
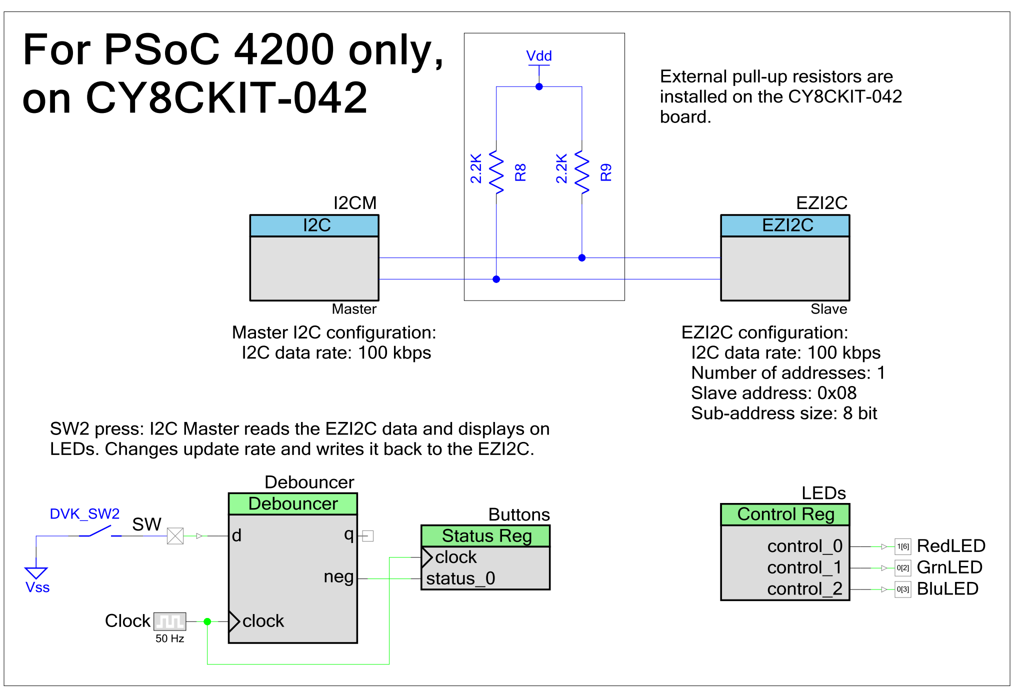


Figure 2. EZI2C Code Example for PSoC 4200 on the CY8CKIT-042



Both code examples feature the following:

- An I^{2}C master communicating with an EZI2C slave over an off-chip I^{2}C bus connection
  - For the PSoC 3 and PSoC 5LP design, the EZI2C uses one of the fixed I^{2}C blocks, and the master is configured in the universal digital blocks (UDBs)
  - For PSoC 4200, both Components use a PSoC 4 serial communication block (SCB), in I^{2}C or EZI2C mode
- A display Component (Character LCD or Pins driving LEDs) to show that the I^{2}C Components are communicating.
- The button press detect subsystem causes the CPU to read a '1' when a button transitions from not pressed to pressed, and a '0' at all other times. Pressing a button changes the data on the master and/or slave side.

Code Design

In both code examples, the main loop first executes the EZI2C slave side code, followed by the I^{2}C master code:

```
for(;;)
{
    /* Do slave side tasks, with the EZI2C Component */
    . . .
    /* Do master side tasks, with the I2C Master Component */
    . . .
}
```

The master and slave side each maintain their own data buffers. With EZI2C, the first N bytes of the buffer can be written by the master; the remaining bytes are read-only. The master side buffer includes an additional byte 'writeOffset' to indicate the location in the write area to start writing. All buffers are packed to ensure reading and writing the correct bytes.

PSoC 3 and PSoC 5LP Design

The PSoC 3 and PSoC 5LP example uses two buttons, which are available on the CY8CKIT-030 and CY8CKIT-050:

- If button SW2 is pressed, the EZI2C side updates its data buffer.
- If button SW3 is pressed, the I^{2}C master side does the following:
  - Reads the EZI2C buffer to its data buffer
  - Updates the write portion of its data buffer
  - Writes the write portion of its data buffer to the EZI2C

The Character LCD Component displays the contents of both data buffers.

PSoC 4200 Design

The PSoC 4200 example uses the single button, SW2, on the CY8CKIT-042 kit. The master and slave sides do the following:

- The EZI2C side runs a code-based counter; the reload value is in its data buffer. Each time the counter rolls over, a control byte for the LEDs is updated.
- The I^{2}C master side does the following:
  - Reads the EZI2C buffer to its data buffer
  - If SW2 is pressed:
    - Updates the counter reload value in the write portion of its data buffer
    - Writes the write portion of its data buffer to the EZI2C
  - Updates the LEDs based on the control byte in its buffer

The LEDs change color continually; pressing the button changes the update rate.

Design Considerations

- Off-chip connections between the I^{2}C master and slave pins form an I^{2}C bus. External I^{2}C bus pull-up resistors may need to be installed, depending on the kit that is used as well as the Pin Component configuration.
- These code examples can be modified to:
  - Run on other kits, such as the CY8CKIT-049 or the CY8CKIT-001.
  - Communicate between two or more kits.

- The PSoC Creator installation includes a program called Bridge Control Panel (BCP). BCP enables communications between your PC and PSoC target devices, over I2C. You can use this link to control the PSoC and read and display data from the PSoC. For more information, click the Help menu item in the BCP window.

Hardware Setup

For basic kit board setup, see the corresponding Kit Guide.

To form the off-chip I^{2}C bus, connect the master and slave SCL and SDA pins on the kit board:

- For the PSoC 3 and PSoC 5LP example, connect P12[0] to P12[2], and P12[1] to P12[3]. This applies to all supported kits.
  - To avoid having to add external I^{2}C bus pull-up resistors, configure the Pin Components as Resistive Pull Up instead of the default Open Drain, Drives Low. This technique does not meet formal I^{2}C specifications but does work in most cases.
- For the PSoC 4200 example, using the CY8CKIT-042:
  - Connect P0[4] to P4[0], by wiring kit connector J4 pin 1 to J3 pin 10.
  - Connect P0[5] to P4[1], by wiring kit connector J4 pin 2 to J3 pin 9.
  - Note that on the CY8CKIT-042 board, P4[0] and P4[1] have I^{2}C bus pull-up resistors installed.

Software Setup

No special software setup is required. All supported compilers can be used with any optimization.

At the PSoC Creator project's default CPU clock speed (48 MHz for PSoC 3 and PSoC 5LP, 24 MHz for PSoC 4200), the CPU has enough cycles to support the examples.

Components

Table 1 and Table 2 list the PSoC Creator Components used in each of the examples, as well as the hardware resources used by each Component.

Table 1. List of PSoC Creator Components for PSoC 3 and PSoC 5LP Example

Component	Version	Hardware Resources
EZI2C Slave	2.0	PSoC 3 or PSoC 5LP fixed I^{2}C block, 1 interrupt
I2C Master (UDB)	3.50	~2 UDBs, 1 interrupt, 1 clock divider
Debouncer, 2 inputs	1.0	UDB (10 macrocells)
Clock	2.20	1 clock divider
Status Register, 2 input	1.90	UDB (1 status register)
Character LCD	2.20	7 pins
Pin	2.20	4 pins for the two I^{2}C Components, 2 pins for the buttons, 7 pins for the Character LCD

Table 2. List of PSoC Creator Components for PSoC 4200 Example

Component	Version	Hardware Resources
EZI2C Slave (SCB mode)	3.20	PSoC 4200 SCB, 2 pins, 1 interrupt, 1 clock divider
I2C (SCB mode)	3.20	PSoC 4200 SCB, 2 pins, 1 interrupt, 1 clock divider
Debouncer, 1 input	1.0	UDB (5 macrocells)
Clock	2.20	1 clock divider
Status Register, 1 input	1.90	UDB (1 status register)
Control Register, 3 outputs	1.80	UDB (1 control register)

Component	Version	Hardware Resources
Pin	2.20	4 pins for the two I²C Components, 1 pin for the button, 3 pins for the RGB LED

Parameter Settings

Table 3 and Table 4 list the parameter settings for each of the PSoC Creator Components used in each of the examples. Only the parameters that vary from the default values are listed.

Table 3. List of PSoC Creator Component Parameter Settings for PSoC 3 and PSoC 5LP Example

Component	Non-default Parameter Settings
EZI2C Slave	None
I2C Master (UDB)	UDB Clock Source = Internal Clock
Debouncer, 2 inputs	Signal width (bits) = 2, only Negative edge is checked
Clock	Frequency = 50 Hz
Status Register, 2 input	Inputs = 2, Display as bus is checked, Mode = Sticky for all bits
Character LCD	None
Pin	I²C Component pins: Drive mode = Resistive Pull Up Button pins: Number of Pins = 2, Drive mode = Resistive Pull Up

Table 4. List of PSoC Creator Component Parameter Settings for PSoC 4200 Example

Component	Non-default Parameter Settings
EZI2C Slave (SCB mode)	None
I2C (SCB mode)	Mode = Master
Debouncer, 1 input	Only Negative edge is checked
Clock	Frequency = 50 Hz
Status Register, 1 input	Inputs = 1, Mode = Sticky for all bits
Control Register, 3 outputs	Outputs = 3
Pin	Button pin: Drive mode = Resistive Pull Up

Design-Wide Resources

Figure 3 and Figure 4 show the pin assignments for each of the examples. No other design-wide resources need to be changed from their default setting.

Figure 3. Pin Assignments for PSoC 3 and PSoC 5LP Example

Alias	Name	Port	Pin	Lock
	\LCD:LCDPort[6:0]\	P2[6:0]	95..99,1..2	☑
	SCL_M	P12[0] I2C1:SCL	53	☑
	SCL_S	P12[2]	67	☑
	SDA_M	P12[1] I2C1:SDA	54	☑
	SDA_S	P12[3]	68	☑
	SW[0]	P6[1]	90	☑
	SW[1]	P15[5]	94	☑

Figure 4. Pin Assignments for PSoC 4200 Example

Alias	Name	Port	Pin	Lock
	\EZI2C:scl\	P4[0] SCB0:I2C:SCL, SCB0:SPI:MOSI SCB0:UART:RX	20	☑
	\EZI2C:sda\	P4[1] SCB0:I2C:SDA, SCB0:SPI:MISO SCB0:UART:TX	21	☑
	\I2CM:scl\	P0[4] SCB1:I2C:SCL, SCB1:SPI:MOSI SCB1:UART:RX	28	☑
	\I2CM:sda\	P0[5] SCB1:I2C:SDA, SCB1:SPI:MISO SCB1:UART:TX	29	☑
	BluLED	P0[3]	27	☑
	GrnLED	P0[2] SCB0:SPI:SS3	26	☑
	RedLED	P1[6]	43	☑
	SW	P0[7] SCB1:SPI:SS0, WAKEUP	31	☑

Operation

Build and install the code examples in the corresponding kits. For more information on building a project and device programming, see PSoC Creator Help.

Test the code example by doing the following:

For the PSoC 3 and PSoC 5LP example:

- Reset the PSoC; press kit button SW1. Observe the character LCD.
  - Confirm that the top row displays "EZ: 00 00 00 00". That is, the EZI2C buffer is all zeros.
  - Confirm that the bottom row is blank, indicating that the I^2C master has not yet read the EZI2C buffer.
- Press kit button SW2.
  - Confirm that the bytes in the EZI2C buffer are incremented, by different values, on each button press.

- Press kit button SW3.
  - Confirm that the bottom row displays “MST: ” followed by the contents of the EZI2C buffer. Confirm also that the first two bytes are decremented by different values. That is, the I²C master has read the EZI2C buffer and decremented the read/write bytes of its buffer.
  - Confirm that the top row displays the first two bytes in the EZI2C buffer as the same as those in the master buffer. This indicates a successful write of the EZI2C data by the master.

For the PSoC 4200 example:

- Reset the PSoC; press kit button SW1.
- Confirm that the RGB LED changes color at a high rate. This indicates a successful read of the EZI2C data by the master. The EZI2C data has LED control bits that are continually changed by the EZI2C side code.
- Press kit button SW2. Confirm that the RGB LEDs change color at a different rate. This indicates a successful write of the EZI2C data by the master.

Related Documents

Table 5 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 5. Related Documents

Application Notes		
AN60317	PSoC 3 and PSoC 5LP I²C Bootloader	Shows how to build an I²C-based bootloader for PSoC 3 and PSoC 5LP
AN86526	PSoC 4 I²C Bootloader	Shows how to build an I²C-based bootloader for PSoC 4 family devices
AN50987	Getting Started with I²C in PSoC 1	Discusses the I²C protocol, and how PSoC 1 devices handle I²C communications
AN74875	Designing with Cypress Serial I2C nvSRAM	Provides design guidelines and example circuits for the Cypress I²C nvSRAM device

Code Examples	
DelSig_I2CM	Provides an 8-channel multiplexed Delta Sigma ADC with sequencing logic. The analog inputs to the ADC are converted to digital sequentially and then made available through an I²C Master interface.
DelSig_I2CS	Provides an 8-channel multiplexed Delta Sigma ADC with sequencing logic. The analog inputs to the ADC are converted to digital sequentially and then made available through an I²C Slave interface.
I2C_LCD_Example	Demonstrates the functionality of the I²C LCD Component
SCB_EzI2cCommSlave	Demonstrates the basic operation of the EZI2C Slave (SCB mode) Component
SCB_I2cCommMaster	Demonstrates the basic operation of the I²C Master (SCB mode) Component
SCB_I2cCommSlave	Demonstrates the basic operation of the I²C Slave (SCB mode) Component

Knowledge Base Articles	
I2C pins in PSoC 3 and PSoC 5	Per PSoC 3 and PSoC 5 pinouts in the datasheet, there are only two sets of I²C pins. Are these the only pins which can be used for I²C or is there a way to use some other pins for I²C?
Assigning I2C SDA and SCL pins to any GPIO in PSoC 3 and PSoC 5LP	When I try to route the I²C SCL and SDA pins to any GPIO, I get the following error: IO "I2C_SCL(0)" cannot be placed into "PX[x]" because the pin does not support the features required by the IO. (App=cydsfit) What is the reason for this error and how can this be fixed?
Wiring a bus to I2C in PSoC Creator	How can I connect my I2C Component to a digital pin through a bus?
Multiple Slave Addresses with EZI2C	Can I have three slave address using an EZI2C Slave Component?
EZI2C does not work with address greater than 63	Why does the EZI2C User Module not work when the I²C address is greater than 63?

MiniProg3 connections for bootloading over I2C	How should I connect the MiniProg3 to a DVK board, to bootload over I^2C?
BootLdrI2C - In Master mode	Is it possible to configure the I^2C bootloader in Master mode and read the firmware from an external source?
Clock Stretching and I2C speed	How does the I^2C clock speed affect the duration of clock stretching introduced by the I2C slave?
Series resistors on I2C lines	Why are resistors of 330 ohm required on I^2C lines?
PSoC Creator Component Datasheets	
EZI2C Slave	Implements an I^2C register-based slave device
I2C Master/Multi-Master/Slave	Supports I^2C Slave, Master, and Multi-Master configurations
PSoC 4 Serial Communication Block (SCB)	Supports a PSoC 4 multifunction hardware block that implements I^2C, SPI, UART, and EZI2C communications
Debouncer	Takes an input signal from a bouncing switch contact and generates a clean output for digital circuits
Control Register	Allows firmware to generate output digital signals
Status Register	Allows firmware to read digital signals
Character LCD (CharLCD)	Contains a set of library routines that enable simple use of one, two, or four-line LCD modules that follow the Hitachi 44780 standard 4-bit interface
Clock	Creates local clocks, and allows connection to system and design-wide clocks
Pins	Controls interface with physical I/O port pins
External Library	Provides a way to include components external to the PSoC device – resistors, capacitors, transistors, inductors, switches, etc. – on a PSoC Creator schematic.
Device Documentation	
PSoC 3 Datasheets	PSoC 3 Technical Reference Manuals
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals
PSoC 5LP Datasheets	PSoC 5LP Technical Reference Manuals
Development Kit (DVK) Documentation	
PSoC 3 and PSoC 5LP Kits	
PSoC 4 Kits	

Document History

Document Title: CE95314 – PSoC® 3, PSoC 4, and PSoC 5LP EZI2C

Document Number: 001-95314

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4622360	MKEA	01/15/2015	New code example
*A	5081848	TDU	01/13/2016	Minor Grammatical Fixes
*B	5789464	MKEA	06/28/2017	Updated project and document for PSoC Creator 4.1. Miscellaneous edits.

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

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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