KitProg User Guide

Doc. # 001-96359 Rev. *I

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Copyrights

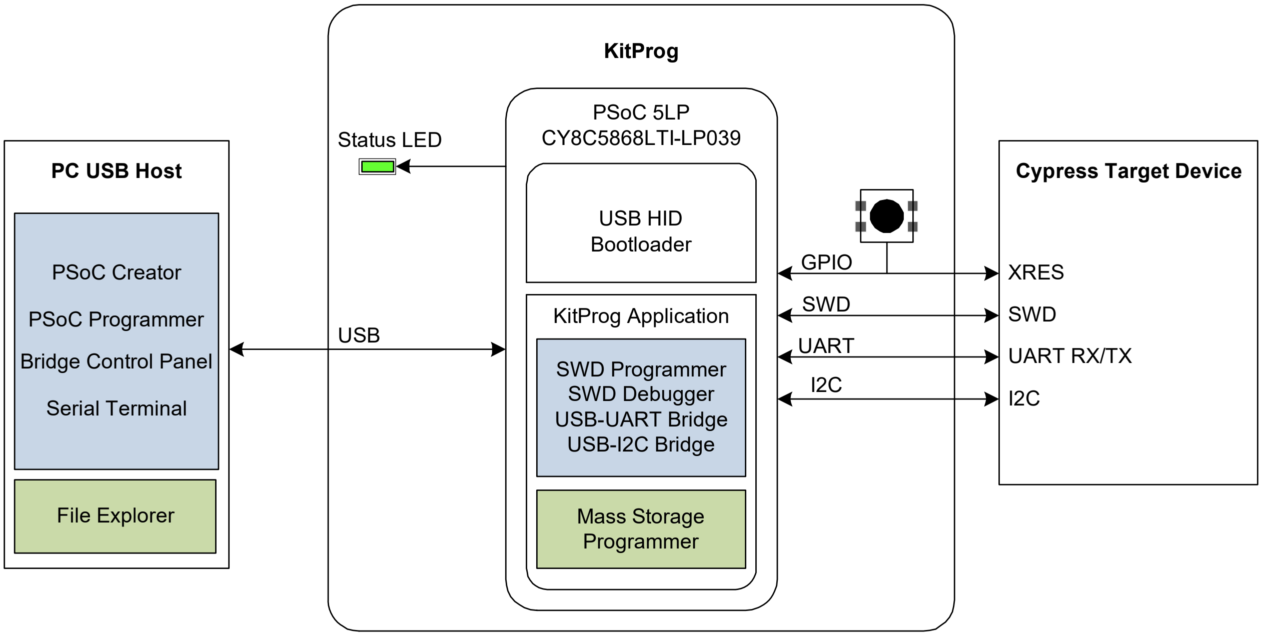
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Contents

1. Introduction

The KitProg is an onboard programmer/debugger with USB-I2C and USB-UART bridge functionality. The KitProg is integrated onto most PSoC development kits. This user guide provides comprehensive information on how to use the KitProg functionalities with PSoC development kits. Figure 1-1 shows the KitProg Ecosystem. The Cypress PSoC 5LP device is used to implement the KitProg functionality.

Figure 1-1. KitProg Ecosystem



2. KitProg Ecosystem

Table 2-1 lists the development kits that use the KitProg. Table 2-2 lists the prerequisite Cypress software needed to use the KitProg.

Table 2-1. Development Kits Supported by KitProg

Functionality	Download Link/Remarks
CY8CKIT-042 PSoC 4 Pioneer Kit	PSoC 4200
CY8CKIT-040 PSoC 4000 Pioneer Kit	PSoC 4000
CY3280-MBR3 CapSense Evaluation Kit	CapSense MBR3
CY8CKIT-042-BLE Bluetooth Low Energy (BLE) Pioneer Kit	PSoC 4200 BLE, PRoC BLE
CY8CKIT-044 PSoC 4 M-Series Pioneer Kit	PSoC 4200M
CY8CKIT-043 PSoC 4 M-Series Prototyping Kit	PSoC 4200M
CY8CKIT-046 PSoC 4 L-Series Pioneer Kit	PSoC 4200L
CY8CKIT-059 PSoC 5LP Prototyping Kit	PSoC 5LP

The CY3280-MBR3 CapSense Evaluation Kit features a fixed-function CapSense controller device; the KitProg on this kit is only used for the USB-I2C bridge functionality. Therefore, except the chapter 6. KitProg USB-I2C Bridge on page 27, other chapters of this UG are not applicable to the CY3280-MBR3 kit.

Table 2-2. Prerequisite Software for KitProg Operation

Functionality	Pre-requisite Software	Download Link/Remarks
Programmer	PSoC Programmer	www.cypress.com/psocprogrammer
Debugger	PSoC Creator	www.cypress.com/psoccreator
USB-I2C Bridge	Bridge Control Panel (BCP)	Installed along with PSoC Programmer
USB-UART Bridge	Terminal Emulator Program	Any terminal emulator program can be used such as HyperTerminal (available as part of Microsoft Windows XP installtaion) or PuTTY (available from www.putty.org)

The KitProg supports different speeds for communication interfaces. Table 2-3 summarizes the KitProg operating modes.

Table 2-3. KitProg Operating Modes

Functionality	Supported Speed	Units
Programmer	1.6	MHz
USB-I2C Bridge	50, 100, 400, 1000	kHz
USB-UART Bridge	1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200	Baud

This document assumes that you know the basics of how to use PSoC Creator™. If you are new to PSoC Creator, refer to the documentation in the PSoC Creator home page. You can also refer to the following application notes to get started with PSoC devices:

- Getting Started with PSoC® 4
- Getting Started with PSoC® 4 BLE
- Getting Started with PSoC® 5LP
- Getting Started with CapSense®

3. KitProg Programmer/Debugger

This section explains the method to use the KitProg programmer/debugger integrated onto the PSoC development kits. The KitProg supports the development kits listed in Table 2-1. This section uses the PSoC 4 M-Series Pioneer and PSoC 4 M-Series Prototyping Kit as examples.

3.1 Programming Using PSoC Creator

1. Connect the USB cable into the USB connector, J6, as shown in Figure 3-1. If you are connecting the kit to your PC for the first time, it enumerates as a USB composite device and installs the required driver software. See the 3.4 KitProg Driver Installation on page 11 section for more information.

Figure 3-1. Connect USB Cable to J6 (Pioneer Kits)

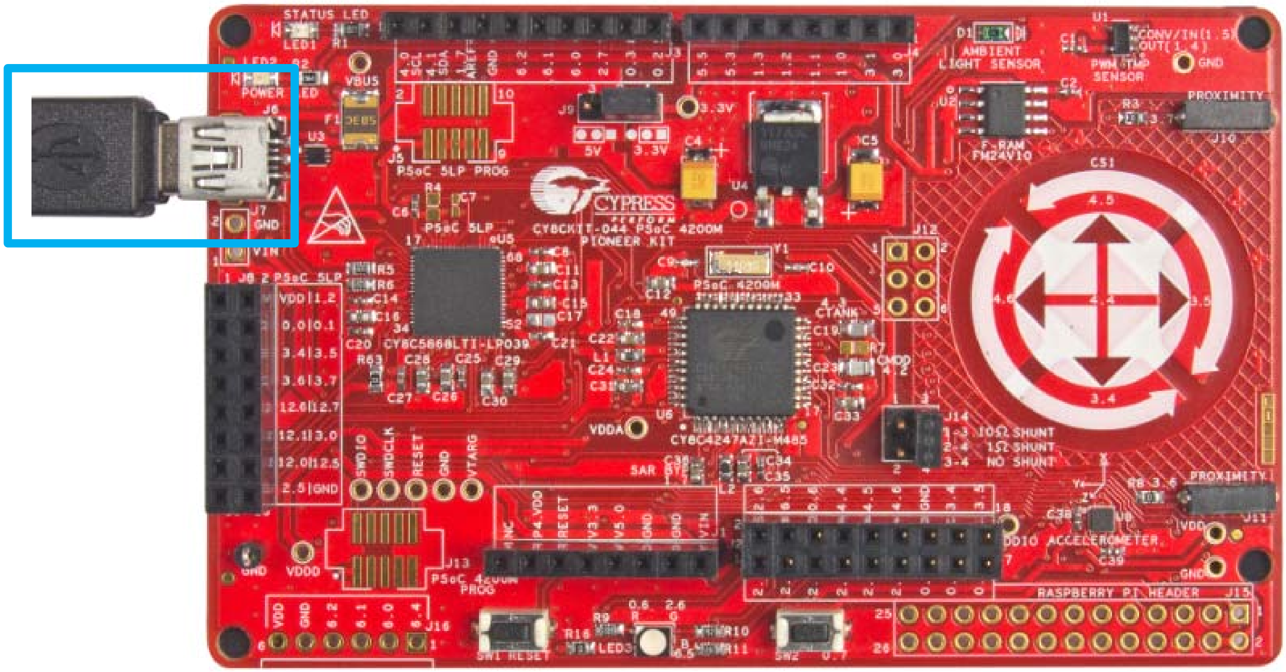
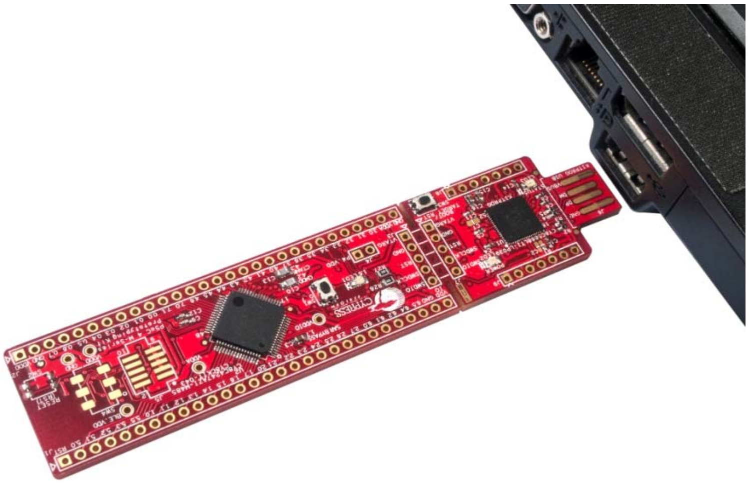
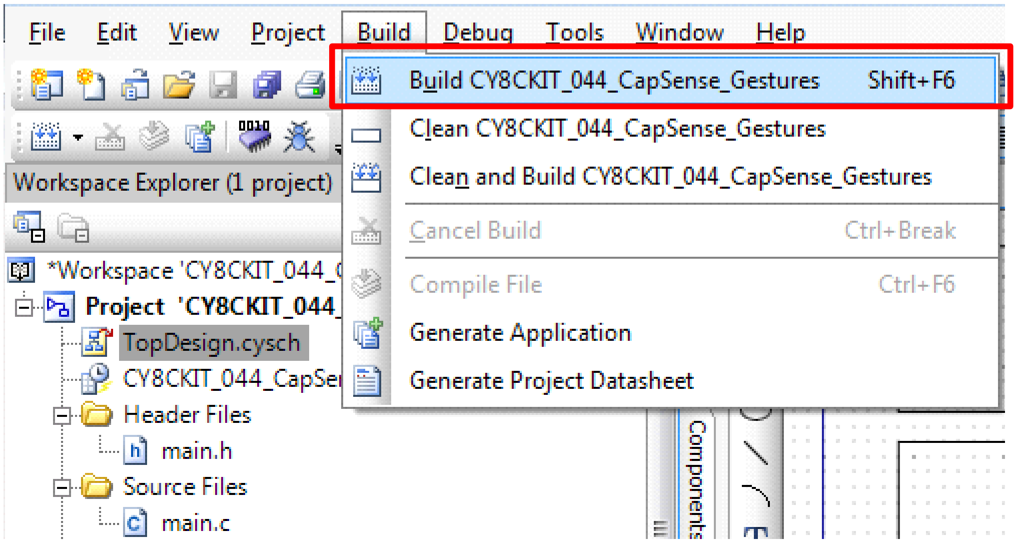


Figure 3-2. Connect USB Cable to J6 (Prototyping Kits)



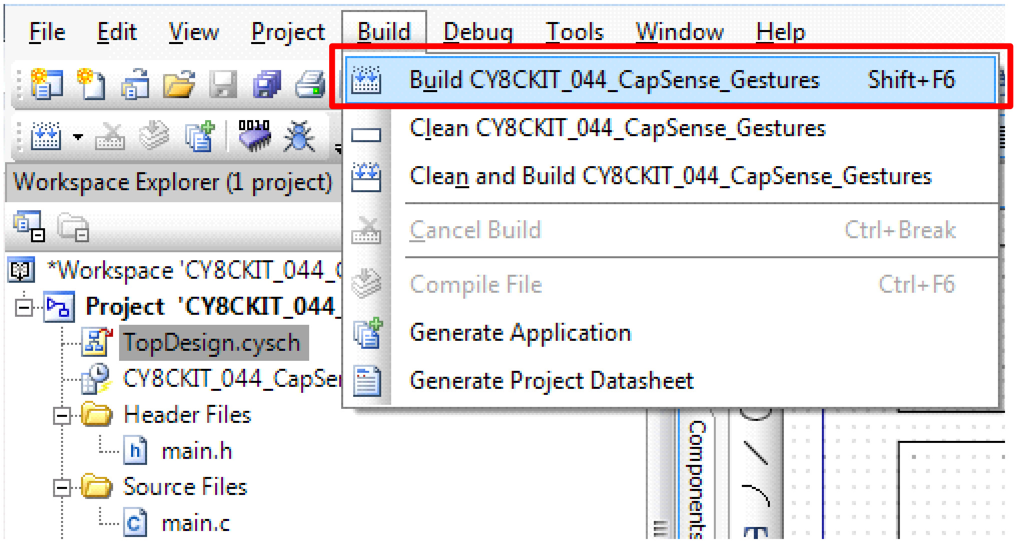
2. Launch PSoC Creator from **Start** > **All Programs** > **Cypress** > **PSoC Creator <version>** > **PSoC Creator <version>**.
3. Select **File** > **Open** > **Project/Workspace** in PSoC Creator and browse to the desired project.
4. Select **Build** > **Build Project** or press **[Shift] [F6]** to build the project, as shown in Figure 3-3.

Figure 3-3. Build an Example Project



5. If there are no errors during build, program the PSoC 4200M device on the kit by choosing **Debug** > **Program** or pressing **[Ctrl] [F5]**, as shown in Figure 3-4.

Figure 3-4. Programming Device from PSoC Creator

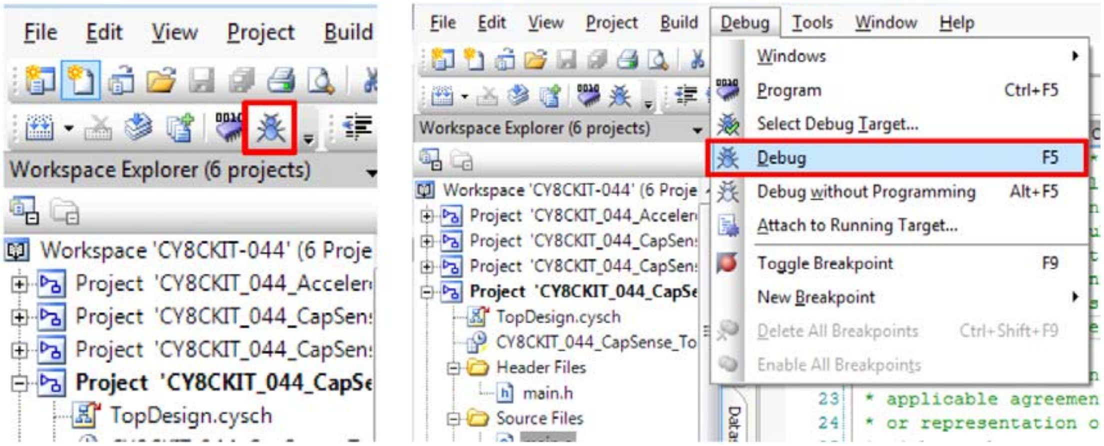


3.2 Debugging Using PSoC Creator

To debug the project using PSoC Creator, follow steps 1 to 4 from 3.1 Programming Using PSoC Creator on page 7. Then, follow these steps:

1. Click the **Debug** icon or press **[F5]**, as shown in Figure 3-5. Alternately, you can select **Debug** > **Debug**. This programs the device and starts the debugger.

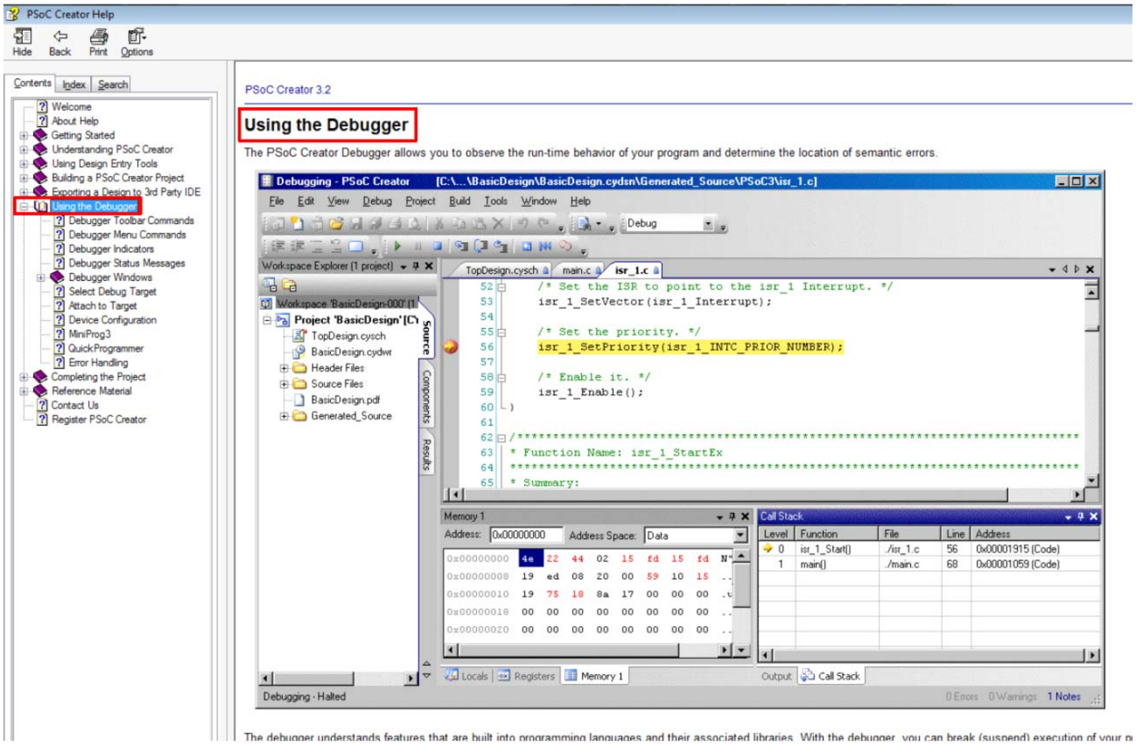
Figure 3-5. Debug Option in PSoC Creator



2. When PSoC Creator enters the Debug mode, use the buttons on the toolbar or keyboard shortcuts to debug your project.

   For more details on using the debug features, refer to the PSoC Creator Help. Select **Help** > **PSoC Creator Help Topics** in the PSoC Creator menu. In the PSoC Creator Help window, locate **Using the Debugger** section in the **Contents** tab, as shown in Figure 3-6.

Figure 3-6. Using the PSoC Creator Debugger



Using the Debugger

The PSoC Creator Debugger allows you to observe the run-time behavior of your program and determine the location of semantic errors.

```
    /* Set the ISR to point to the isr_1 Interrupt. */
    isr_1_SetVector(isr_1_Interrupt);

    /* Set the priority. */
    isr_1_SetPriority(isr_1_INTC_PRIOR_NUMBER);

    /* Enable it. */
    isr_1_Enable();
}

/*******************************************************************************
* Function Name: isr_1_StartEx
********************************************************************************
* Summary:
```

```
0x00000000  4e 22 44 02 15 fd 15 fd  N"
0x00000008  19 ed 08 20 00 59 10 15  ..
0x00000010  19 75 18 8a 17 00 00 00  .u
0x00000018  00 00 00 00 00 00 00 00  ..
0x00000020  00 00 00 00 00 00 00 00  ..
```

Level	Function	File	Line	Address
0	isr_1_Start()	./isr_1.c	56	0x00001915 (Code)
1	main()	./main.c	68	0x00001059 (Code)

The debugger understands features that are built into programming languages and their associated libraries. With the debugger, you can break (suspend) execution of your p

3.3 Programming Using PSoC Programmer

PSoC Programmer (3.22.2 or later) can be used to program existing *.hex* files into the kit. To do this, follow these steps.

1. Connect the kit to your PC and open PSoC Programmer from **Start** > **All Programs** > **Cypress** > **PSoC Programmer <version>** > **PSoC Programmer <version>**.
2. Click the **File Load** button at the top left corner of the window. Browse to the desired *.hex* file and click **Open**. For the PSoC 4 device, the *.hex* file is located at: `<Project Directory>\<Project Name.cydsn>\CortexM0\ <Compiler Name and Version>\<Debug> or <Release>\<Project Name.hex>`.
3. Click the **KitProg/<serial number>** in the **Port Selection** list to connect the kit to your computer.

Note: If the CY5670 CySmart USB Dongle (BLE Dongle) is used, the device will enumerate as **KitProg/BLE<serial number>**.

4. Click the **Program** button to start programming the kit with the selected file.

Note: If the *.hex* file does not match the selected device, then PSoC Programmer will display a device mismatch error and terminate programming. Ensure that you have selected the correct .hex file.

5. When the programming is completed successfully, indicated by a PASS message on the status bar, the kit is ready for use. Close PSoC Programmer.

3.4 KitProg Driver Installation

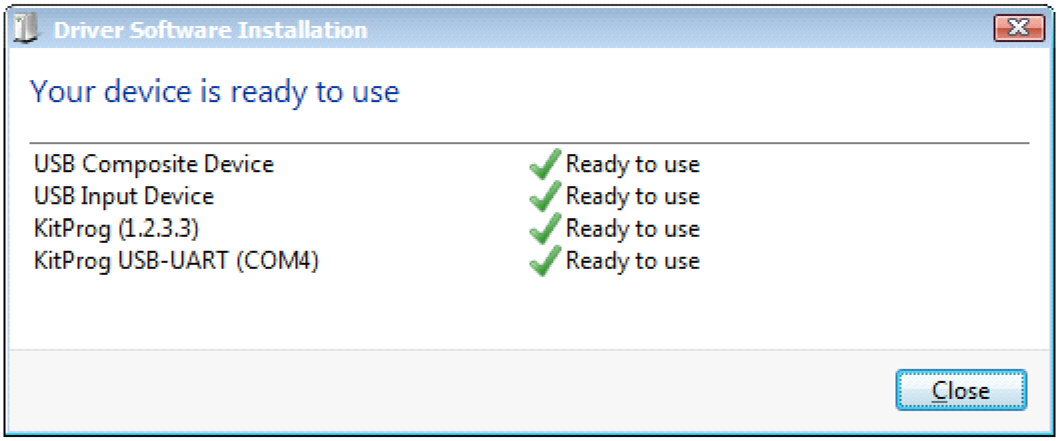
The kits are powered from a computer over the USB interface. It enumerates as a composite device, as shown in Table 3-1. The USB drivers required for enumeration are part of the kit installer and should be appropriately installed for its correct operation.

Table 3-1. Enumerated Interfaces

Function	Description
USB Composite Device	USB Composite device
USB Input Device	USB-I2C bridge, KitProg command interface
KitProg	Programmer and debugger
KitProg USB-UART	USB-UART bridge, which appears as the COM# port

Note: It is recommended to use KitProg with Windows 7 or later operating systems to ensure proper operation of KitProg USB-UART COM port. If Windows XP 32-bit is used, ensure that the SP3 is installed. The Windows XP 64-bit does not support UART debug function.

Figure 3-7. KitProg Driver Installation (appearance may differ depending on the Windows version)



Driver Software Installation

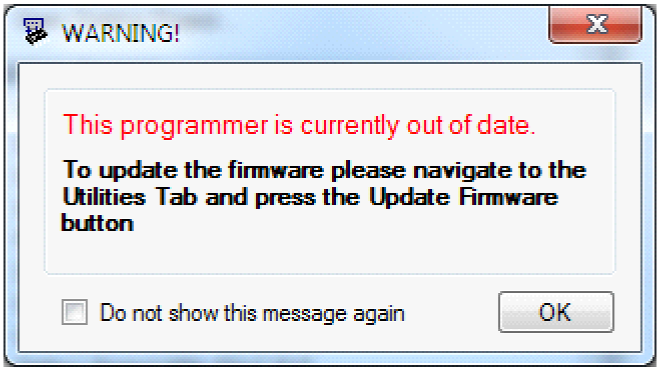
Your device is ready to use

USB Composite Device ✓ Ready to use
USB Input Device ✓ Ready to use
KitProg (1.2.3.3) ✓ Ready to use
KitProg USB-UART (COM4) ✓ Ready to use

3.5 Updating the KitProg Firmware

The KitProg firmware generally does not require any update. If an update is required, then PSoC Programmer will display a warning message when the kit is connected to it, as shown in Figure 3-8.

Figure 3-8. KitProg Firmware Update Warning



WARNING!

This programmer is currently out of date.

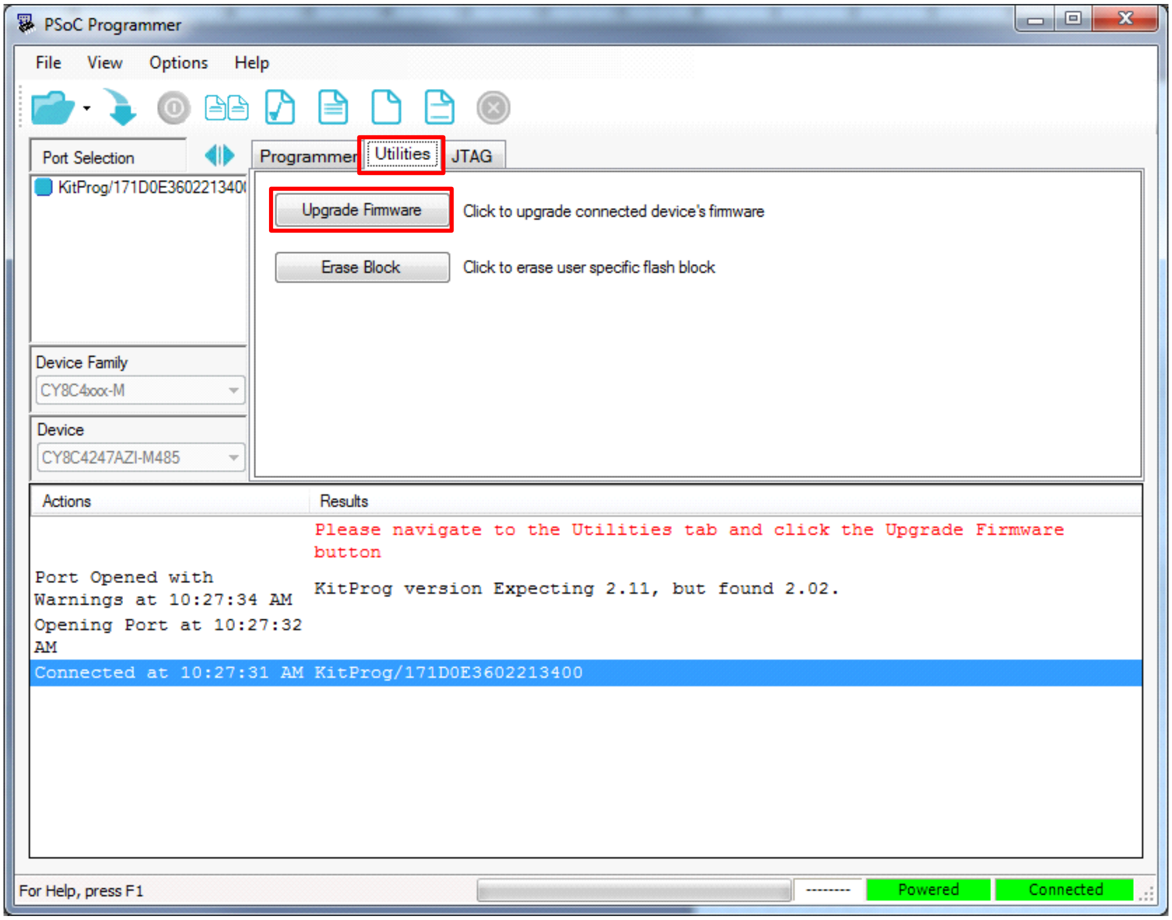
To update the firmware please navigate to the Utilities Tab and press the Update Firmware button

□ Do not show this message again

Click **OK** to close the window. On closing the warning window, the Actions and Results window displays: “Please navigate to the Utilities tab and click the Upgrade Firmware button”, as shown in Figure 3-9.

To update the KitProg, go to the **Utilities** tab on PSoC Programmer and click **Upgrade Firmware**, as shown in Figure 3-9.

Figure 3-9. Upgrade Firmware in PSoC Programmer



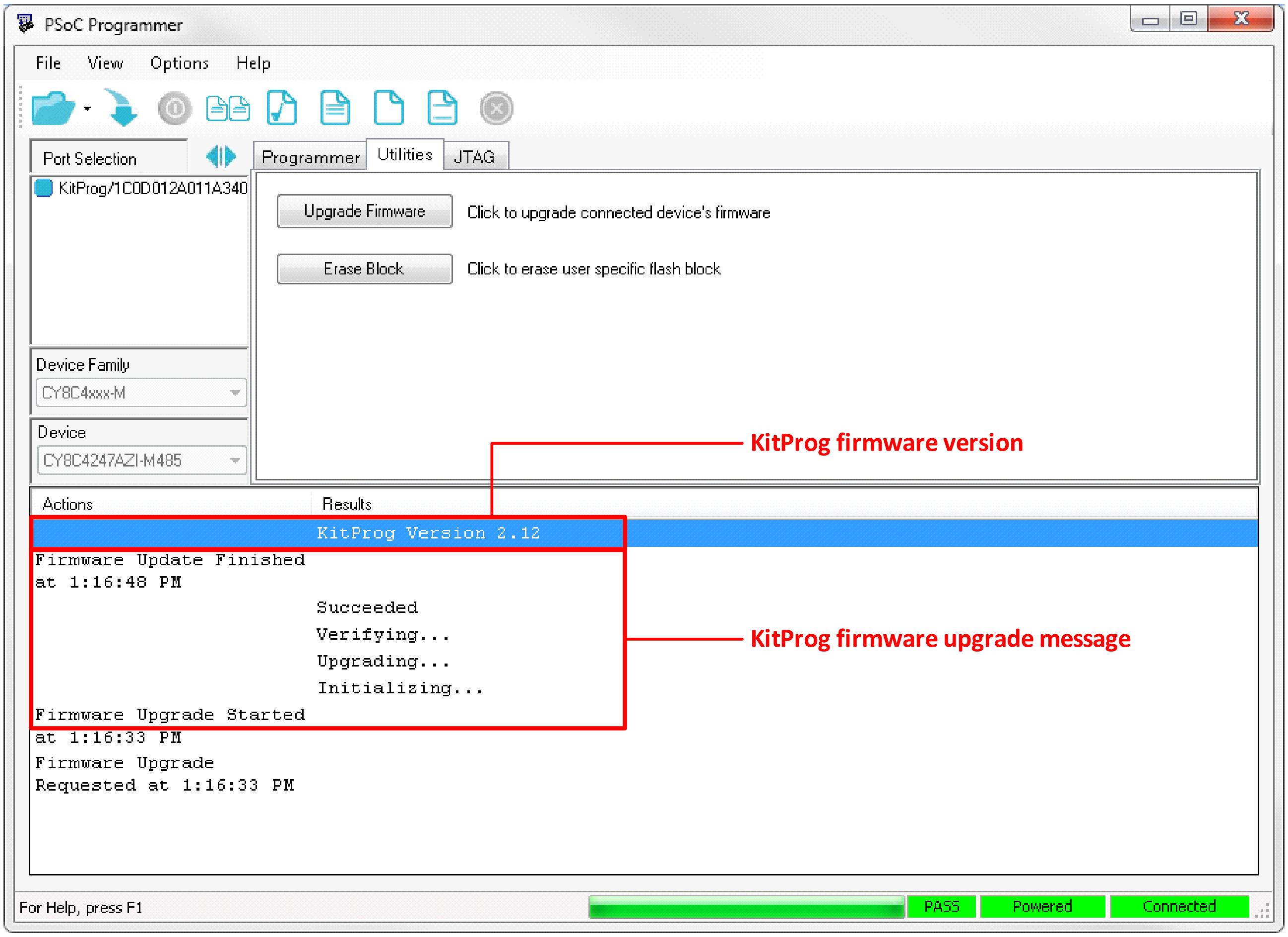
Device Family: CY8C4xxx-M

Device: CY8C4247AZI-M485

```
Actions                      Results
                             Please navigate to the Utilities tab and click the Upgrade Firmware
                             button
Port Opened with             KitProg version Expecting 2.11, but found 2.02.
Warnings at 10:27:34 AM
Opening Port at 10:27:32
AM
Connected at 10:27:31 AM KitProg/171D0E3602213400
```

On successful upgrade, the Actions and Results window displays the firmware update message with the KitProg version, as shown in Figure 3-10.

Figure 3-10. Firmware Updated in PSoC Programmer



```
                                KitProg Version 2.12
Firmware Update Finished
at 1:16:48 PM
                                Succeeded
                                Verifying...
                                Upgrading...
                                Initializing...
Firmware Upgrade Started
at 1:16:33 PM
Firmware Upgrade
Requested at 1:16:33 PM
```

4. KitProg Mass Storage Programmer

The KitProg can act as a USB Mass Storage Programmer. The KitProg Programmer and Debugger, KitProg USB-I2C Bridge, and KitProg USB-UART Bridge functionalities are not available in this configuration.

The kits listed in Table 2-1 except CY3280-MBR3 CapSense Evaluation Kit support the USB Mass Storage Programmer feature.

Visit www.cypress.com/psocprogrammer to download the latest version of the PSoC Programmer.

4.1 Enter or Exit the Mass Storage Programmer Mode

Follow these steps to enter or exit the Mass Storage Programmer mode of KitProg:

1. Connect the kit to the PC. Ensure that the Status LED is on and not blinking. Refer to the section 8.1 KitProg Status LED Indication on page 44 for details on the Status LED indications.
2. Press and hold the reset switch (**SW1**) of the kit for more than 5 seconds. The Status LED of the kit turns off when the KitProg changes configurations.
3. Release the reset switch on the kit after Status LED has turned off. The KitProg re-enumerates in the alternate configuration. For example, the kit enumerates as Mass Storage Programmer if the previous configuration is KitProg Programmer and Debugger.

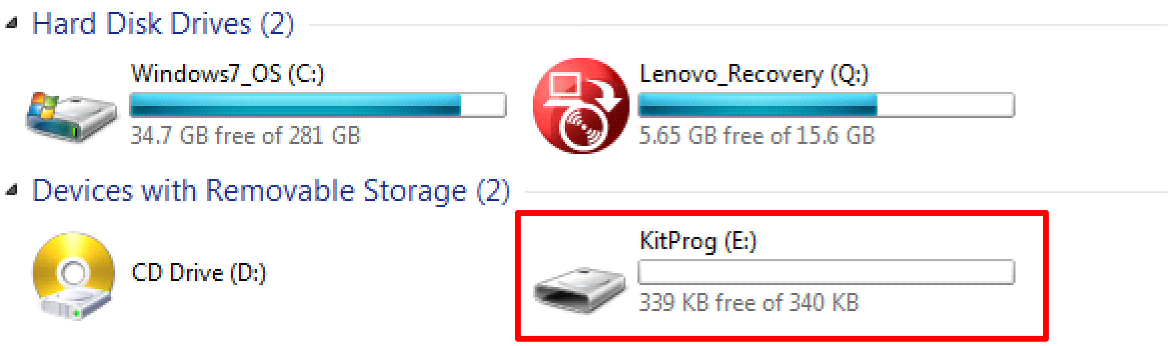
Note: The KitProg remains in the selected mode until the user changes the mode manually using the above steps.

4.2 Programming Using the Mass Storage Programmer

Follow these steps to program the target device using the Mass Storage Programmer:

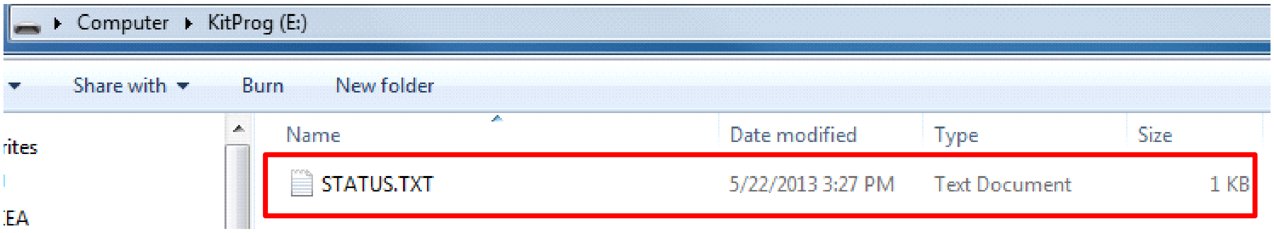
1. Enter the Mass Storage Programmer mode as explained in the section 4.1 Enter or Exit the Mass Storage Programmer Mode. The KitProg is visible as a removable disk drive in the file explorer of the PC, as shown in Figure 4-1.

Figure 4-1. KitProg Emulated as Mass Storage Device



2. Open the KitProg Drive to view the *STATUS.TXT* file, as shown in Figure 4-2. Note that the file extension *.TXT* is visible for the file, if it is enabled in your PC settings. The *STATUS.TXT* shows the current status of the Mass Storage Programmer.

Figure 4-2. *STATUS.TXT* in the KitProg Drive

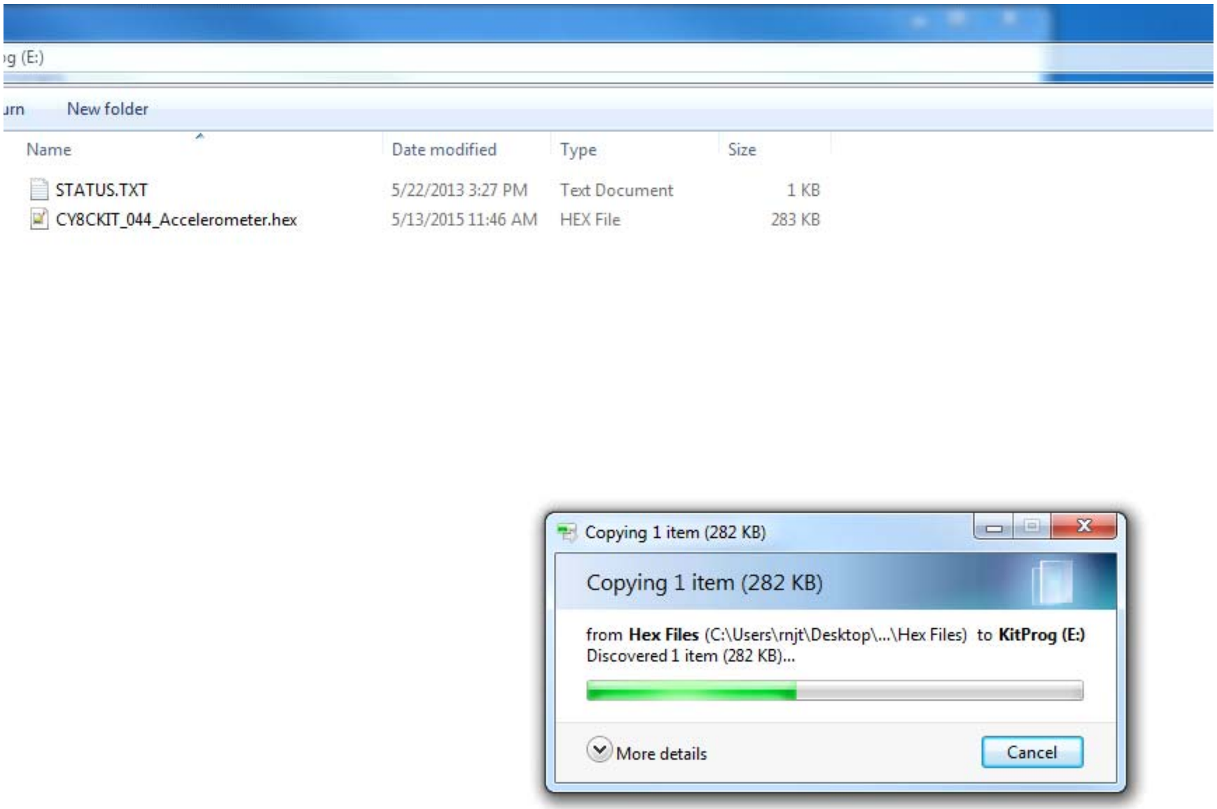


Name	Date modified	Type	Size
STATUS.TXT	5/22/2013 3:27 PM	Text Document	1 KB

3. Copy any PSoC 4200M device based project *.hex* file to the KitProg Drive to begin programming. Alternately, you can also drag and drop the *.hex* file on to the drive. The *.hex* file is available in the following path:

```
<Project Directory>\<Project Name.cydsn>\CortexM0\<Compiler Name and
Version>\<Debug> or <Release>\<Project Name.hex>
```

Figure 4-3. Copy the .hex File to KitProg Drive



Name	Date modified	Type	Size
STATUS.TXT	5/22/2013 3:27 PM	Text Document	1 KB
CY8CKIT_044_Accelerometer.hex	5/13/2015 11:46 AM	HEX File	283 KB

Copying 1 item (282 KB)

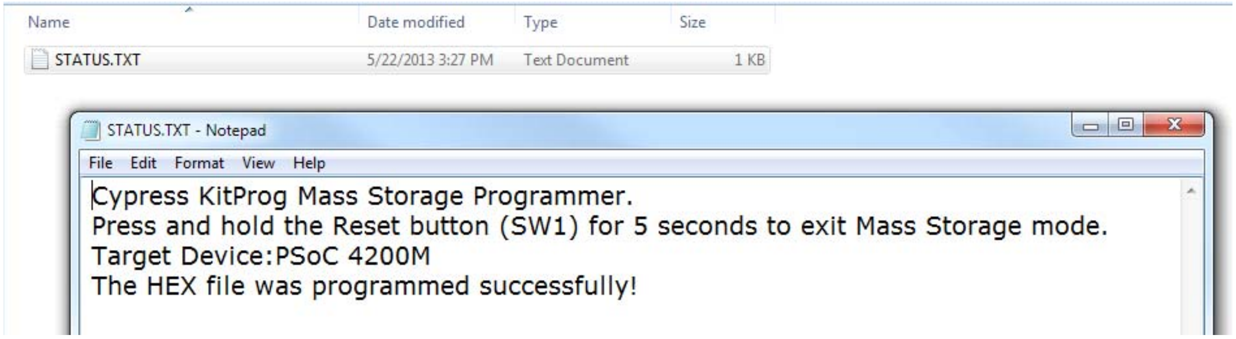
Copying 1 item (282 KB)

from **Hex Files** (C:\Users\rnjt\Desktop\...\Hex Files) to **KitProg (E:)**
Discovered 1 item (282 KB)...

4. The Status LED on the kit blinks during the programming operation. The Status LED continues to blink for 2 seconds after the programming operation and the KitProg Drive automatically removes the copied file from the drive. Press **F5** in the file explorer to refresh the contents of the drive. This will display only the *STATUS.TXT* file in the KitProg Drive.

5. Open the *STATUS.TXT* file to view the status of the programming operation, as shown in Figure 4-4.

Figure 4-4. Status Displayed in the KitProg Drive after Programming



Name	Date modified	Type	Size
STATUS.TXT	5/22/2013 3:27 PM	Text Document	1 KB

```
Cypress KitProg Mass Storage Programmer.
Press and hold the Reset button (SW1) for 5 seconds to exit Mass Storage mode.
Target Device:PSoC 4200M
The HEX file was programmed successfully!
```

4.3 Frequently Asked Questions on KitProg Mass Storage Programmer

1. What are the Cypress kits supported by the KitProg Mass Storage Programmer?

   The KitProg Mass Storage Programmer currently supports only CY8CKIT-044 PSoC 4 M-Series Pioneer Kit.

2. What are the operating systems supported by KitProg Mass Storage Programmer?

   The KitProg Mass Storage Programmer works on Microsoft Windows and Apple Mac Operating Systems. The KitProg Mass Storage Programmer is currently not supported on Linux Operating System.

3. Why are the contents of F-RAM removed and filled with random values after programming operation?

   The KitProg Mass Storage Programmer uses the on-board F-RAM of the CY8CKIT-044 PSoC 4 M-Series Pioneer Kit to store the contents of the copied .hex file for programming operation. This is the reason for removal of any stored data.

4. What happens if I copy an incorrect .hex file to the KitProg Drive?

   If you copy a PSoC 4200M *.hex* file with invalid data (incorrect Silicon ID, incorrect Checksum, and so on), the KitProg Mass Storage Programmer attempts a programming operation and generates an error indicating which step of the programming operation has failed in the *STATUS.TXT* file.

   If you copy a *.hex* file which corresponds to any other device, the KitProg Mass Storage Programmer does not attempt a programming operation and generates an error indicating that the copied file is not a valid *.hex* file in the *STATUS.TXT* file.

   If you copy any other file than specified above and file size does not exceed the KitProg Drive size, the file will be visible in the KitProg Drive until the KitProg Drive is removed from the PC. Note that the file is not actually copied to the KitProg Drive. Delete these files before attempting to program a new *.hex* file.

5. Why does my Operating System display the pop-up “Disk Not Ejected Properly” after every programming operation in KitProg Mass Storage Programmer mode?

   The KitProg Mass Storage Programmer temporarily ejects 2 seconds after the programming operation. This can also cause the file explorer window of the KitProg Drive to close after programming operation in some operating systems.

6. Is it possible to program an external PSoC other than PSoC 4200M using the KitProg Mass Storage Programmer?

   No. The KitProg Mass Storage Programmer supports only PSoC 4200M on the CY8CKIT-044 PSoC 4 M-Series Pioneer Kit.

7. Can I use *.hex* files generated by any other IDE other than PSoC Creator to program the PSoC 4200M using KitProg Mass Storage Programmer?

   Yes. You can also use the *.hex* file generated for PSoC 4200M by an external IDE such as Eclipse, IAR, Keil µVision that supports PSoC 4 devices, to program the PSoC 4200M using KitProg Mass Storage Programmer.

8. Why does the programming time for different files vary?

   The KitProg Mass Storage Programmer intelligently programs only the flash rows with non-zero data. Depending on the contents of your project, the programming time may take up to 20 seconds.

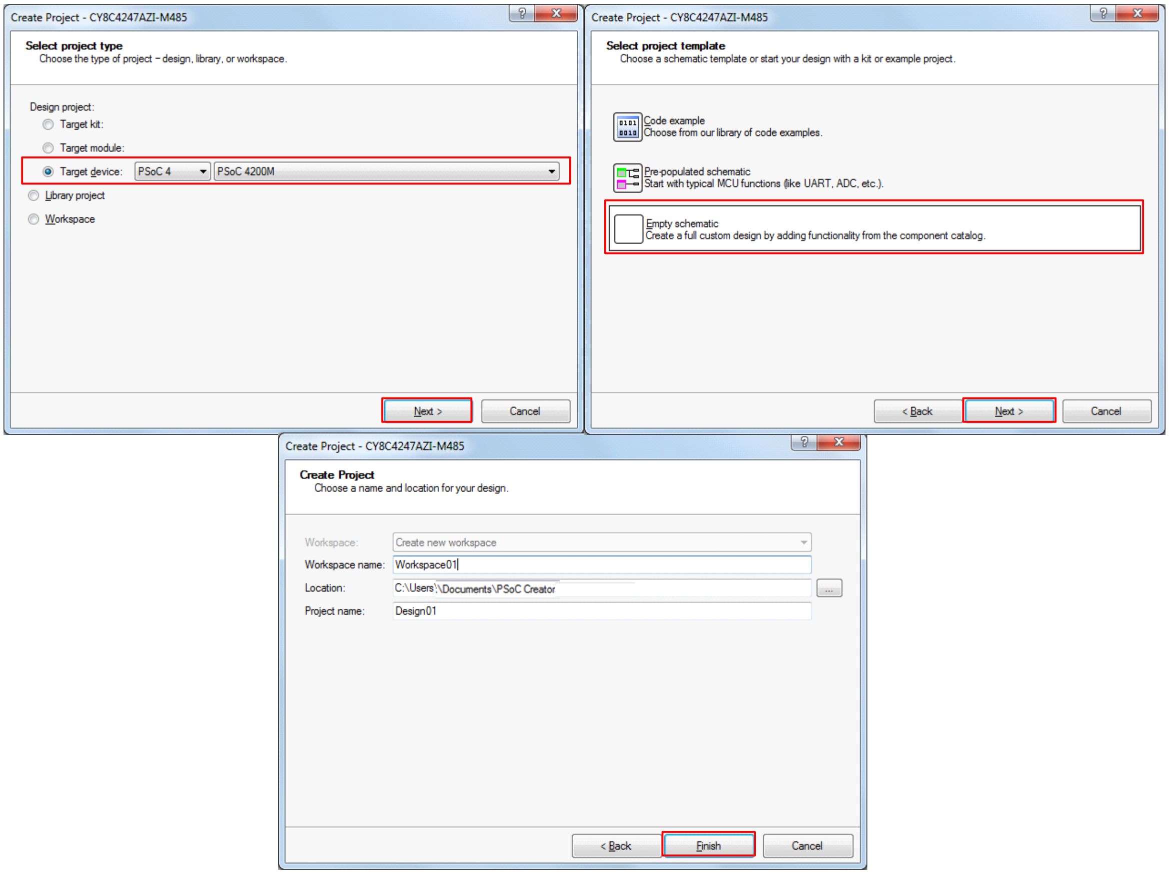
5. KitProg USB-UART Bridge

The KitProg can act as a USB-UART bridge. This feature of the KitProg is useful to send and receive data between the Cypress device on the kit and the PC. For example, in the PSoC 4 M-Series Pioneer Kit, the KitProg USB-UART can be used to print debug messages on COM terminal software running on the PC.

This section explains a method to create a PSoC 4 code example, which communicates with the COM terminal software using the KitProg USB-UART Bridge. This example uses Windows HyperTerminal as the COM terminal software. If you have a Windows operating system that does not have HyperTerminal, use alternate terminal software such as PuTTY.

1. Create a new PSoC 4 project in PSoC Creator, as shown in Figure 5-1. Select a specific location for your project and name the project as desired. You must select the appropriate device for this project depending on the kit as provided in Table 2-1. Ensure that the Project template option is set to Empty schematic. This example uses PSoC 4200M as the target device and PSoC 4 M-Series Pioneer Kit as the target board.

Figure 5-1. Create New Project in PSoC Creator



Create Project - CY8C4247AZI-M485

Select project type
Choose the type of project – design, library, or workspace.

Design project:
- ○ Target kit:
- ○ Target module:
- ● Target device: PSoC 4 | PSoC 4200M

○ Library project
○ Workspace

Create Project - CY8C4247AZI-M485

Select project template
Choose a schematic template or start your design with a kit or example project.

Code example
Choose from our library of code examples.

Pre-populated schematic
Start with typical MCU functions (like UART, ADC, etc.).

Empty schematic
Create a full custom design by adding functionality from the component catalog.

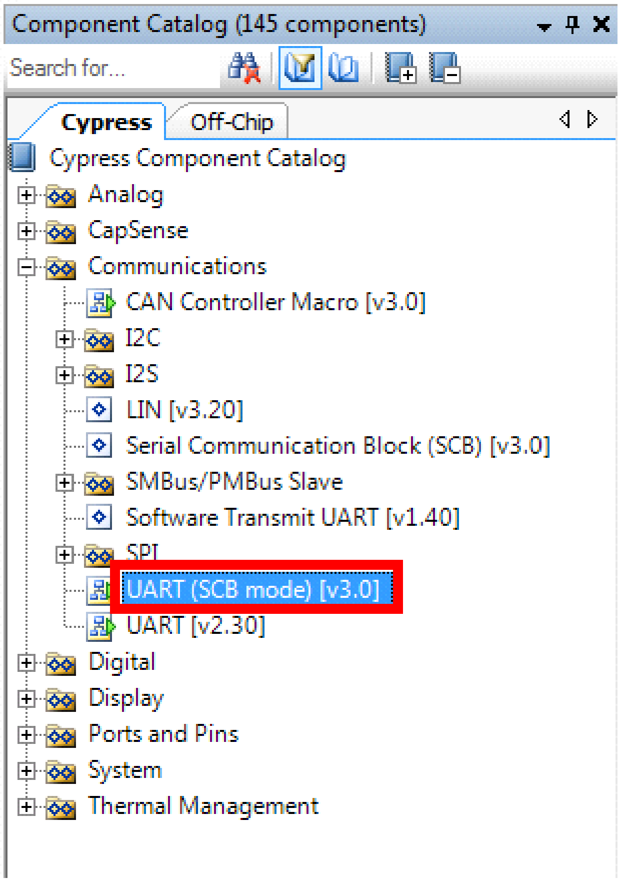
Create Project - CY8C4247AZI-M485

Create Project
Choose a name and location for your design.

Workspace: Create new workspace
Workspace name: Workspace01
Location: C:\Users\\Documents\PSoC Creator
Project name: Design01

2. Drag and drop a UART (SCB mode) Component from the Component Catalog (see Figure 5-2) to the TopDesign. The Component Catalog is located along the right side of the PSoC Creator window by default. To configure the UART, double-click or right-click the UART Component and select **Configure**, as shown in Figure 5-3.

Figure 5-2. UART Component in Component Catalog



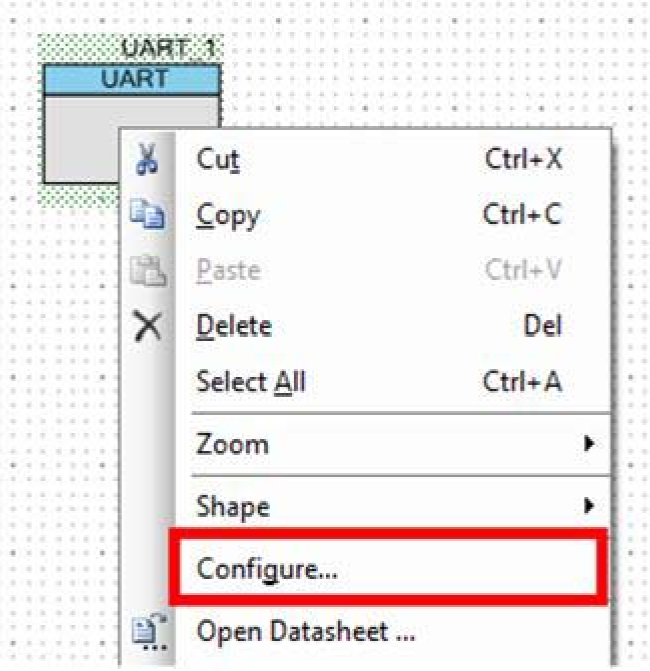
Component Catalog (145 components)

Search for...

Cypress | Off-Chip

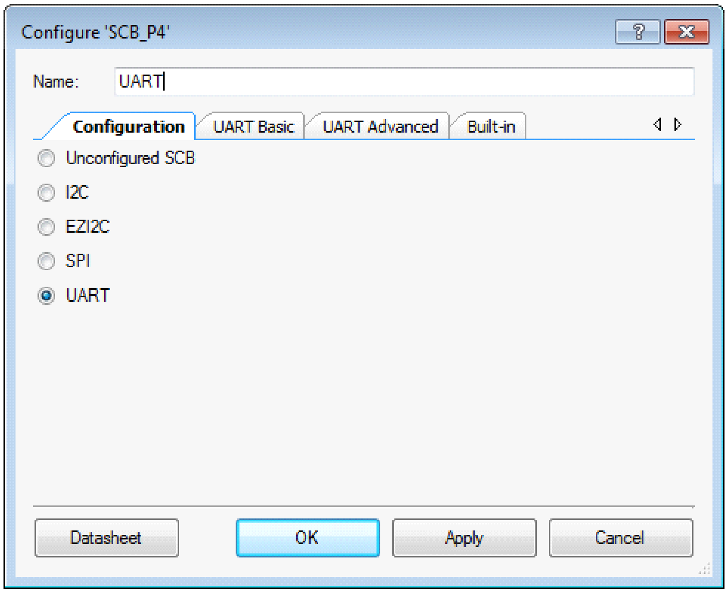
- Cypress Component Catalog
  - Analog
  - CapSense
  - Communications
    - CAN Controller Macro [v3.0]
    - I2C
    - I2S
    - LIN [v3.20]
    - Serial Communication Block (SCB) [v3.0]
    - SMBus/PMBus Slave
    - Software Transmit UART [v1.40]
    - SPI
    - UART (SCB mode) [v3.0]
    - UART [v2.30]
  - Digital
  - Display
  - Ports and Pins
  - System
  - Thermal Management

Figure 5-3. Open UART Configuration Window



3. Configure the UART Component as shown in Figure 5-4, Figure 5-5 and Figure 5-6, and then click **OK**.

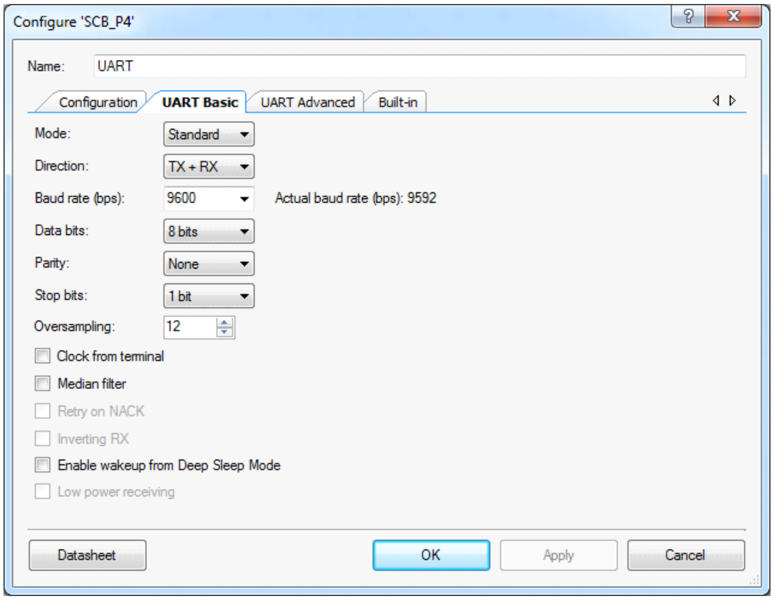
Figure 5-4. UART Configuration Tab Window



Name: UART

○ Unconfigured SCB
○ I2C
○ EZI2C
○ SPI
● UART

Figure 5-5. UART Basic Tab Window



Configure 'SCB_P4'

Name: UART

Configuration | **UART Basic** | UART Advanced | Built-in

Mode: Standard

Direction: TX + RX

Baud rate (bps): 9600 Actual baud rate (bps): 9592

Data bits: 8 bits

Parity: None

Stop bits: 1 bit

Oversampling: 12

☐ Clock from terminal

☐ Median filter

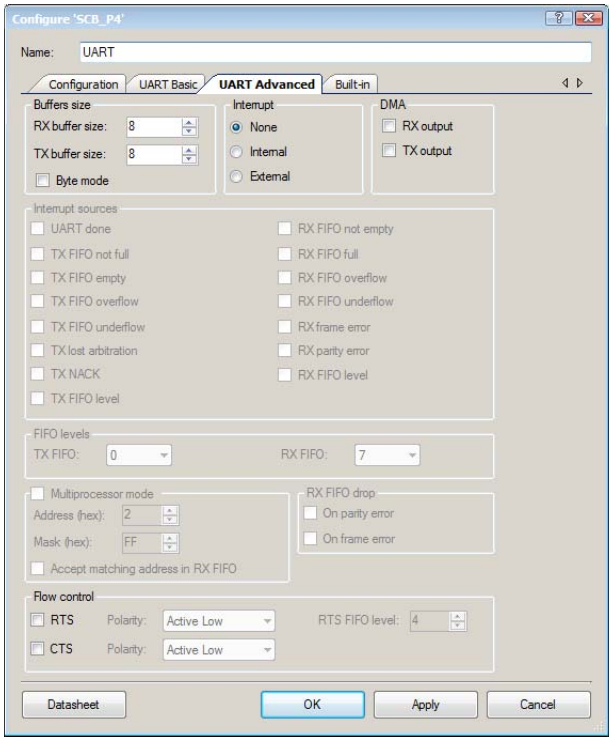
☐ Retry on NACK

☐ Inverting RX

☐ Enable wakeup from Deep Sleep Mode

☐ Low power receiving

Figure 5-6. UART Advanced Tab Configuration Window



Configure 'SCB_P4'

Name: UART

Configuration | UART Basic | **UART Advanced** | Built-in

Buffers size

RX buffer size: 8

TX buffer size: 8

☐ Byte mode

Interrupt

● None

○ Internal

○ External

DMA

☐ RX output

☐ TX output

Interrupt sources

☐ UART done

☐ TX FIFO not full

☐ TX FIFO empty

☐ TX FIFO overflow

☐ TX FIFO underflow

☐ TX lost arbitration

☐ TX NACK

☐ TX FIFO level

☐ RX FIFO not empty

☐ RX FIFO full

☐ RX FIFO overflow

☐ RX FIFO underflow

☐ RX frame error

☐ RX parity error

☐ RX FIFO level

FIFO levels

TX FIFO: 0 RX FIFO: 7

☐ Multiprocessor mode

Address (hex): 2

Mask (hex): FF

☐ Accept matching address in RX FIFO

RX FIFO drop

☐ On parity error

☐ On frame error

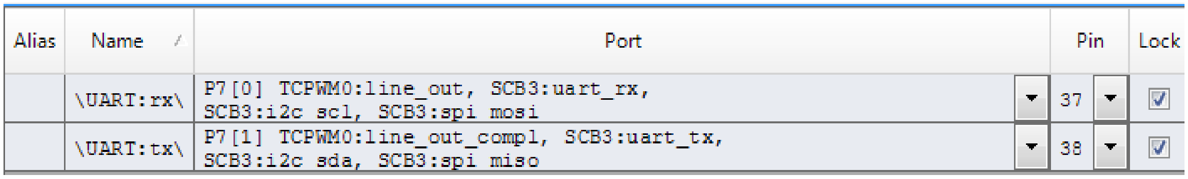
Flow control

☐ RTS Polarity: Active Low RTS FIFO level: 4

☐ CTS Polarity: Active Low

4. Select P7[0] for UART RX and P7[1] for UART TX in the **Pins** tab of *<Project_Name>.cydwr*, as shown in Figure 5-7. The file *<Project_Name>.cydwr* can be found in the Workspace Explorer window, which is located along the left side of the PSoC Creator window by default. Double-click on the file to open it. Note that these are the pins for the USB-UART interface on the PSoC 4 M-Series Pioneer kit. If you are using a different kit, refer to the respective Kit Guide for the appropriate pins.

Figure 5-7. Pin Selection



Alias	Name	Port	Pin	Lock
	\UART:rx\	P7[0] TCPWM0:line_out, SCB3:uart_rx, SCB3:i2c scl, SCB3:spi mosi ▾	37 ▾	☑
	\UART:tx\	P7[1] TCPWM0:line_out_compl, SCB3:uart_tx, SCB3:i2c sda, SCB3:spi miso ▾	38 ▾	☑

5. Place the following code in the *main.c* file. The code echoes any data received through UART.

   Note: The *main.c* file can be found on the Workspace Explorer window, which is located along the left side of the PSoC Creator window by default. Double-click on the file to open it.

```
#include <project.h>
int main()
{
         uint8 ch;
         /* Start SCB UART TX+RX operation */

         UART_Start();
         /* Transmit String through UART TX Line */

         UART_UartPutString("CY8CKIT-044 USB-UART");
         for(;;)
         {
              /* Get received character or zero if nothing has been
              received yet */
             ch = UART_UartGetChar();
             if(0u != ch)
             {
                   /* Send the data through UART. This functions is
                  blocking and waits until there is an entry into the TX
                  FIFO. */
                  UART_UartPutChar(ch);
             }
         }
}
```

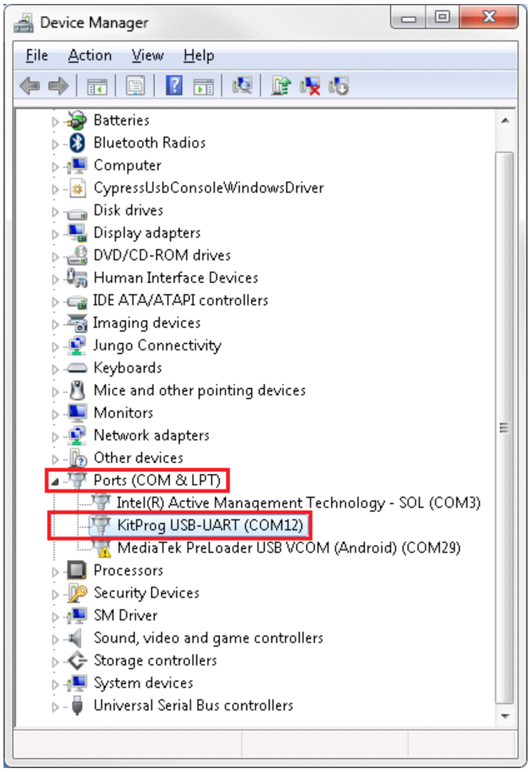
6. Build the project by choosing **Build** > **Build [Project Name]** or pressing **[Shift] [F6]**. After the project is built without errors and warnings, program the project (by choosing **Debug** > **Program**) to the PSoC 4200M using KitProg.

   Note: UART RX and UART TX can be routed to any digital pin on PSoC 4 by using the UDB implementation of the UART Component. In this case, we are using the SCB implementation of the UART, which routes the pins to one of the specific set of pins supported by the device. This will vary depending on the PSoC 4 device used.

To communicate with the PSoC 4200M device from the terminal software, follow this procedure:

1. Connect the USB Mini-B cable to J6. The kit enumerates as a **KitProg USB-UART**, and is available in the **Device Manager** under **Ports (COM & LPT)**. A communication port is assigned to the **KitProg USB-UART**, as shown in Figure 5-8.

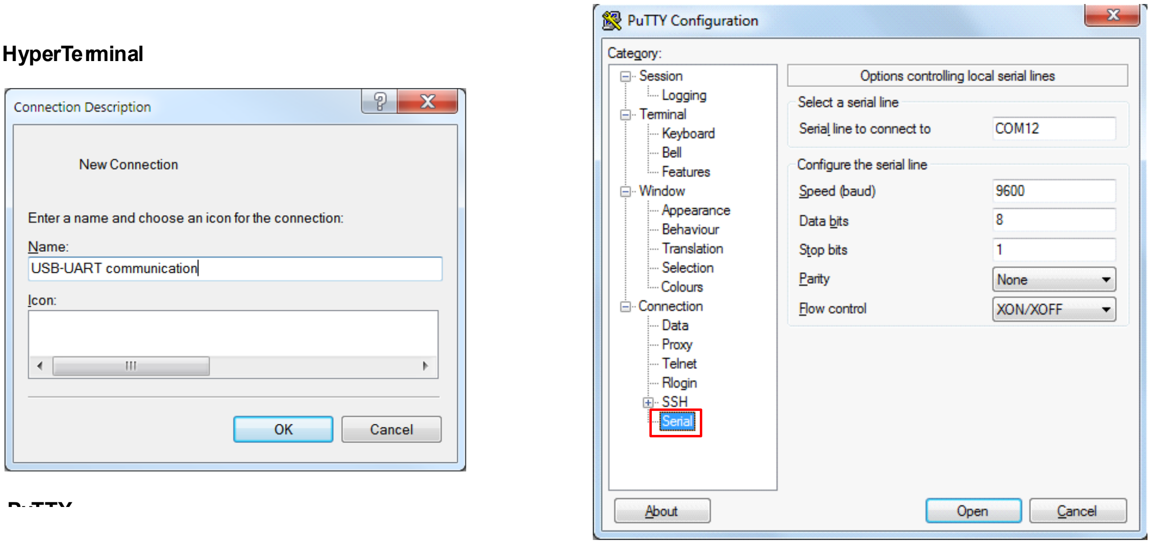
Figure 5-8. KitProg USB-UART in Device Manager



- Batteries
- Bluetooth Radios
- Computer
- CypressUsbConsoleWindowsDriver
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Imaging devices
- Jungo Connectivity
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Other devices
- Ports (COM & LPT)
  - Intel(R) Active Management Technology - SOL (COM3)
  - KitProg USB-UART (COM12)
  - MediaTek PreLoader USB VCOM (Android) (COM29)
- Processors
- Security Devices
- SM Driver
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers

2. Open HyperTerminal, choose **File** > **New Connection**, enter a name for the new connection, and then click **OK** as shown in Figure 5-9. For PuTTY, double-click the PuTTY application and select **Serial** under **Category**.

Figure 5-9. Open New Connection



New Connection

Enter a name and choose an icon for the connection:

Name: USB-UART communication

Icon:

Options controlling local serial lines

Select a serial line

Serial line to connect to: COM12

Configure the serial line

Speed (baud): 9600

Data bits: 8

Stop bits: 1

Parity: None

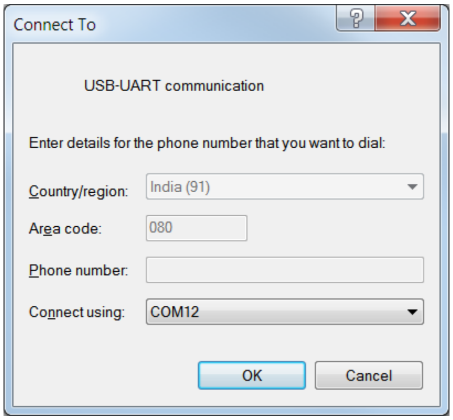
Flow control: XON/XOFF

3. A new window opens, where the communication port can be selected. In HyperTerminal, select COMx (the specific communication port that is assigned to the KitProg USB-UART) in **Connect using** and click **OK**, as shown in Figure 5-10.

   In PuTTY, enter the COMx in **Serial line to connect to**. This example uses COM12.

Figure 5-10. Select Communication Port

HyperTerminal



Connect To

USB-UART communication

Enter details for the phone number that you want to dial:

Country/region: India (91)

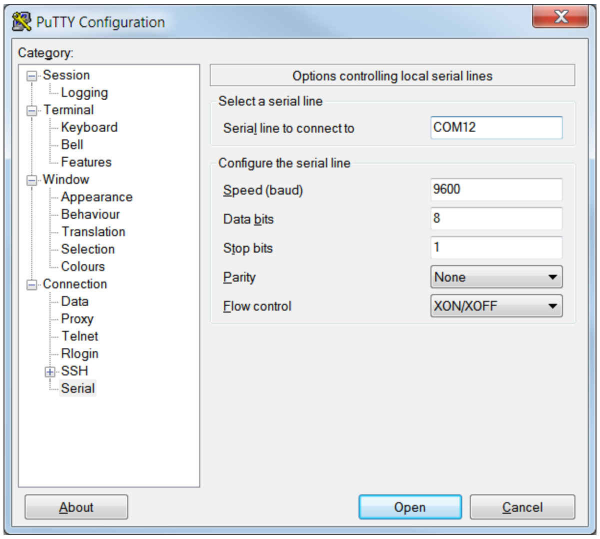
Area code: 080

Phone number:

Connect using: COM12

PuTTY

PuTTY



PuTTY Configuration

Options controlling local serial lines

Select a serial line

Serial line to connect to: COM12

Configure the serial line

Speed (baud): 9600

Data bits: 8

Stop bits: 1

Parity: None

Flow control: XON/XOFF

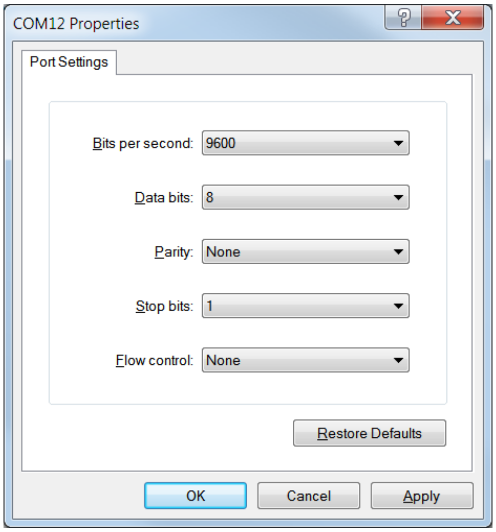
4. In HyperTerminal, select **Bits per second**, **Data bits**, **Parity**, **Stop bits**, and **Flow control** under **Port Settings** and click **OK** (see Figure 5-11). Ensure that the settings are identical to the UART settings configured for the PSoC 4200M device.

   In PuTTY, enter the **Speed (baud)**, **Data bits**, **Stop bits**, **Parity**, and **Flow control** under **Configure the serial line**.

Figure 5-11. Configure the Communication Port

PuTTY

HyperTerminal



COM12 Properties

Port Settings

Bits per second: 9600

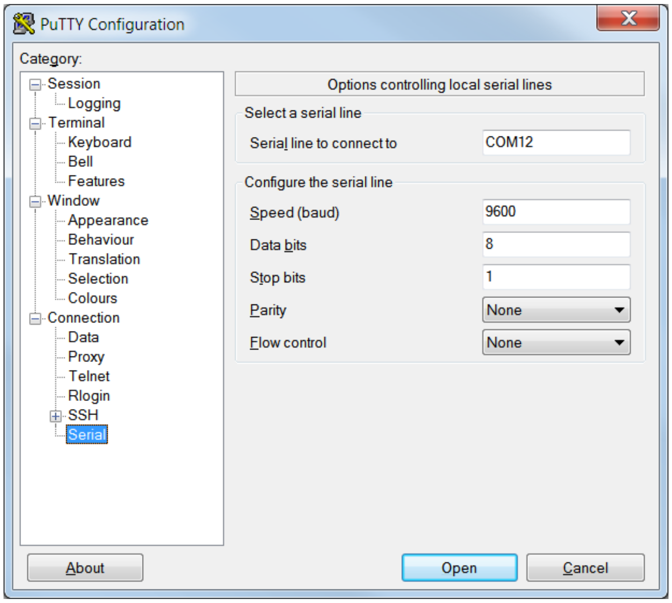
Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

PuTTY



PuTTY Configuration

Options controlling local serial lines

Select a serial line

Serial line to connect to: COM12

Configure the serial line

Speed (baud): 9600

Data bits: 8

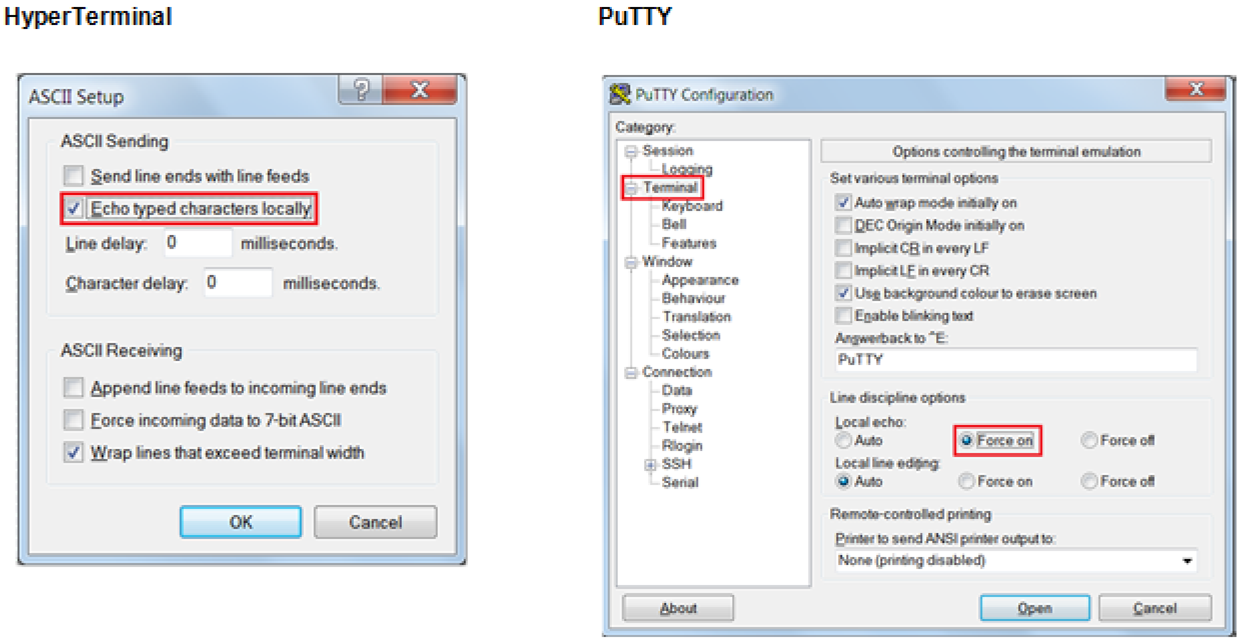
Stop bits: 1

Parity: None

Flow control: None

5. Enable **Echo typed characters** locally under **File** > **Properties** > **Settings** > **ASCII Setup** to display the typed characters on HyperTerminal, as shown in Figure 5-12. In PuTTY, select **Force on** under **Terminal** > **Line discipline options** to display the typed characters on PuTTY, as shown in Figure 5-12.

Figure 5-12. Enable Echo of Typed Characters in HyperTerminal and PuTTY



HyperTerminal

ASCII Setup

ASCII Sending

□ Send line ends with line feeds

☑ Echo typed characters locally

Line delay: 0 milliseconds.

Character delay: 0 milliseconds.

ASCII Receiving

□ Append line feeds to incoming line ends

□ Force incoming data to 7-bit ASCII

☑ Wrap lines that exceed terminal width

PuTTY

PuTTY Configuration

Options controlling the terminal emulation

Set various terminal options

☑ Auto wrap mode initially on

□ DEC Origin Mode initially on

□ Implicit CR in every LF

□ Implicit LF in every CR

☑ Use background colour to erase screen

□ Enable blinking text

Answerback to ^E: PuTTY

Line discipline options

Local echo: ○ Auto ● Force on ○ Force off

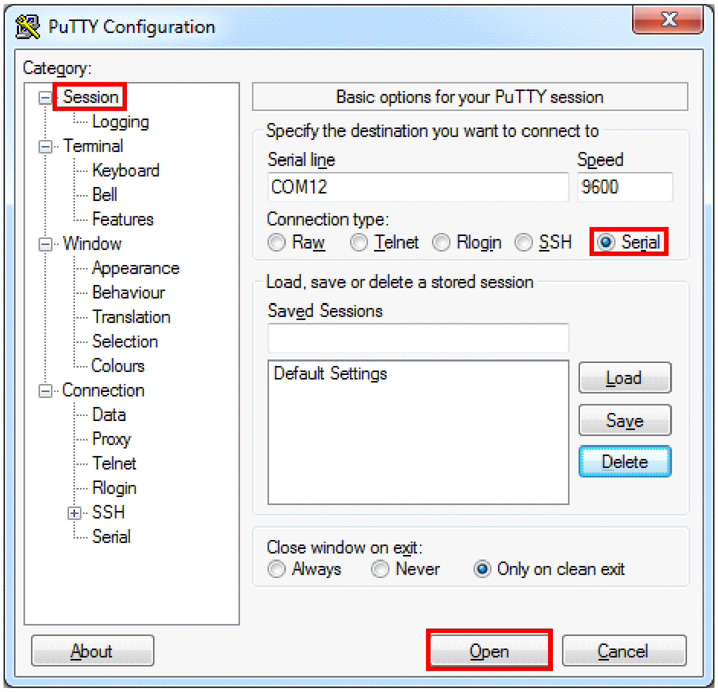
Local line editing: ● Auto ○ Force on ○ Force off

Remote-controlled printing

Printer to send ANSI printer output to: None (printing disabled)

6. In PuTTY, click **Session** and select **Serial** under **Connection type**. Serial line shows the communication port (COM12) and **Speed** shows the baud rate selected. Click **Open** to start the communication, as shown in Figure 5-13.

Figure 5-13. Opening Port in PuTTY



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM12

Speed: 9600

Connection type: ○ Raw ○ Telnet ○ Rlogin ○ SSH ● Serial

Load, save or delete a stored session

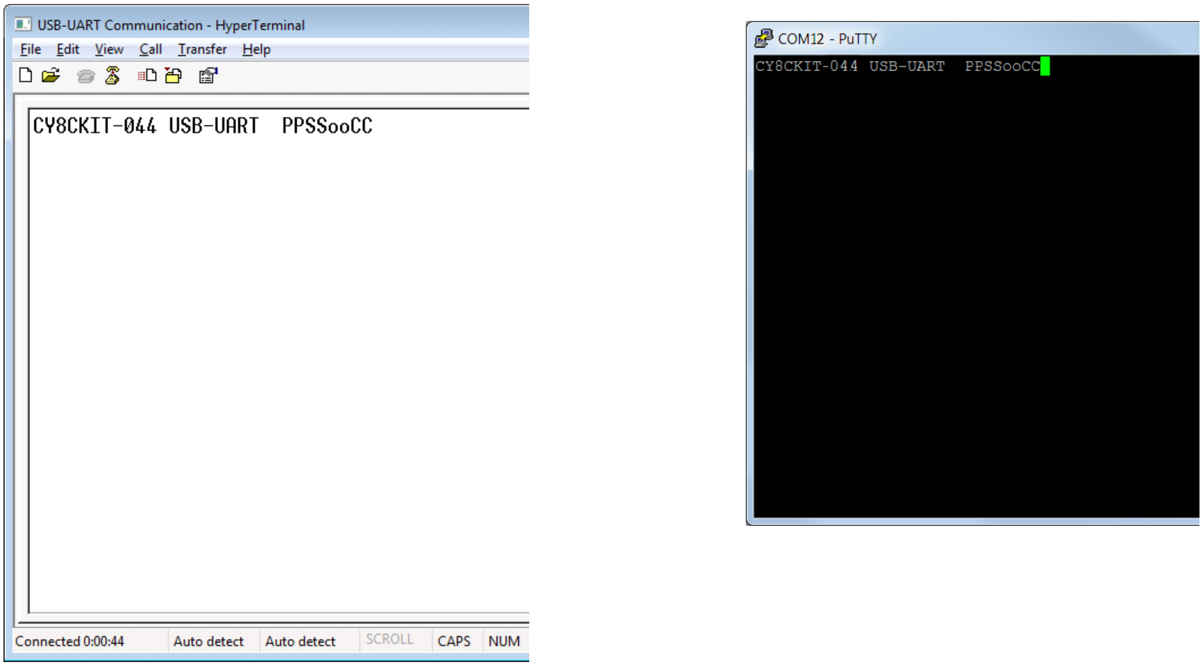
Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

7. The COM terminal software displays both the typed data and the echoed data from the PSoC 4200M UART, as shown in Figure 5-14.

Figure 5-14. Data Displayed on HyperTerminal and PuTTY


USB-UART Communication - HyperTerminal
File Edit View Call Transfer Help
Connected 0:00:44
Auto detect
Auto detect
SCROLL
CAPS
NUM
COM12 - PuTTY

```
CY8CKIT-044 USB-UART  PPSSooCC
```

```
CY8CKIT-044 USB-UART  PPSSooCC
```

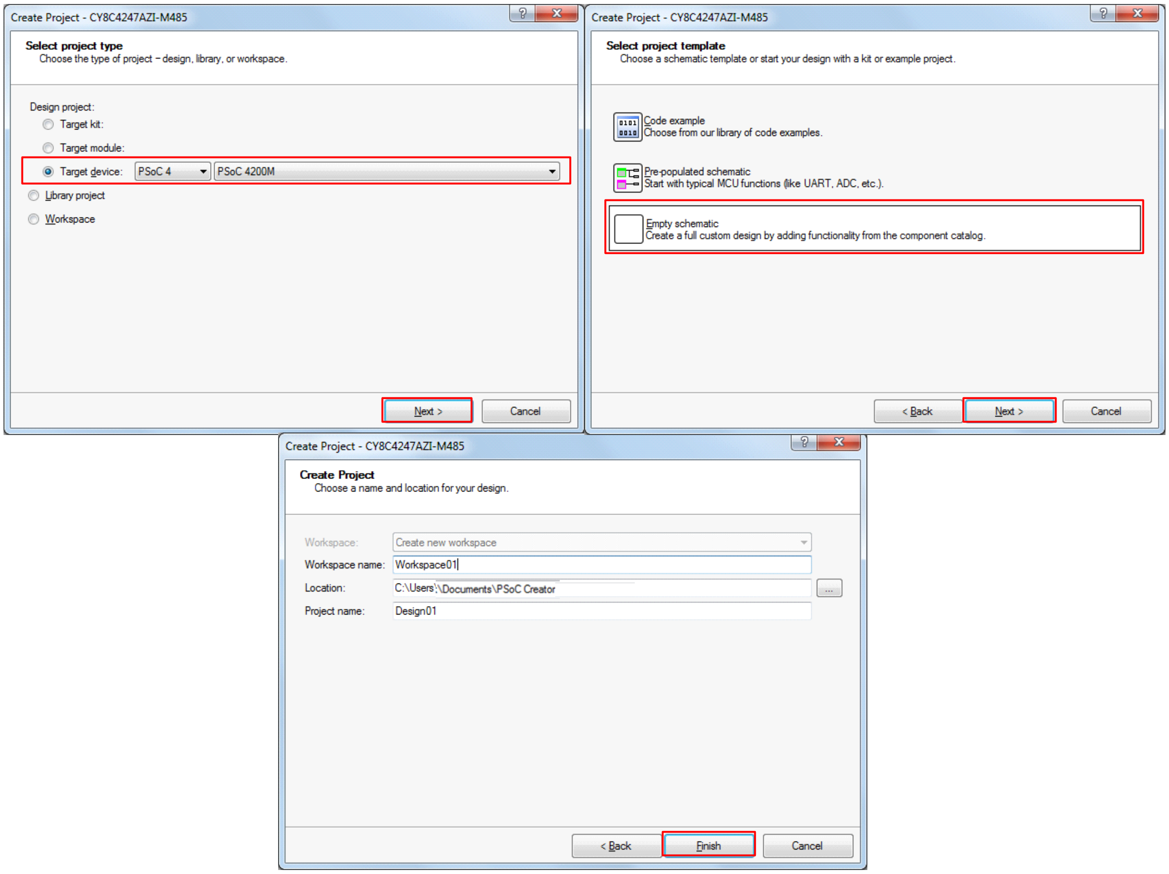
6. KitProg USB-I2C Bridge

The KitProg serves as a USB-I2C bridge that can be used to communicate with the USB-I2C software running on the PC. For example, the KitProg USB-I2C Bridge can be used to tune the CapSense Component on a PSoC device. This feature is applicable to all kits listed in Table 2-1. This section uses the PSoC 4 M-Series Pioneer Kit as an example to demonstrate the KitProg USB-I2C Bridge functionality. The following steps describe how to use the USB-I2C Bridge, which can communicate between the Bridge Control Panel (BCP) software and the PSoC 4200M device.

Note: For information on how to use the KitProg USB-I2C Bridge to tune the CapSense Component, refer to Section 5.2.2 **Manual Tuning Process** in AN85951 - PSoC 4 CapSense Design Guide.

1. Create a new PSoC 4 project in PSoC Creator, as shown in Figure 6-1. Select a specific location for your project and name the project as desired. You must select the appropriate device for this project depending on the kit, as provided in Table 2-1. Ensure that the option Project template is set to Empty schematic. This example uses PSoC 4200M as the target device.

Figure 6-1. Create New Project in PSoC Creator



Create Project - CY8C4247AZI-M485

Select project type
Choose the type of project – design, library, or workspace.

Design project:
○ Target kit:
○ Target module:
● Target device: PSoC 4 PSoC 4200M
○ Library project
○ Workspace

Create Project - CY8C4247AZI-M485

Select project template
Choose a schematic template or start your design with a kit or example project.

Code example
Choose from our library of code examples.

Pre-populated schematic
Start with typical MCU functions (like UART, ADC, etc.).

Empty schematic
Create a full custom design by adding functionality from the component catalog.

Create Project - CY8C4247AZI-M485

Create Project
Choose a name and location for your design.

Workspace: Create new workspace
Workspace name: Workspace01
Location: C:\Users\\Documents\PSoC Creator
Project name: Design01

2. Drag and drop an EZI2C Slave (SCB mode) Component from the Component Catalog (see Figure 6-2) to the TopDesign. The Component Catalog is located along the right side of the PSoC Creator window by default. To configure the EZI2C Slave Component, double-click or right-click the **EZI2C Slave Component** and select **Configure**, as shown in Figure 6-3.

Figure 6-2. EZI2C Slave Component in Component Catalog

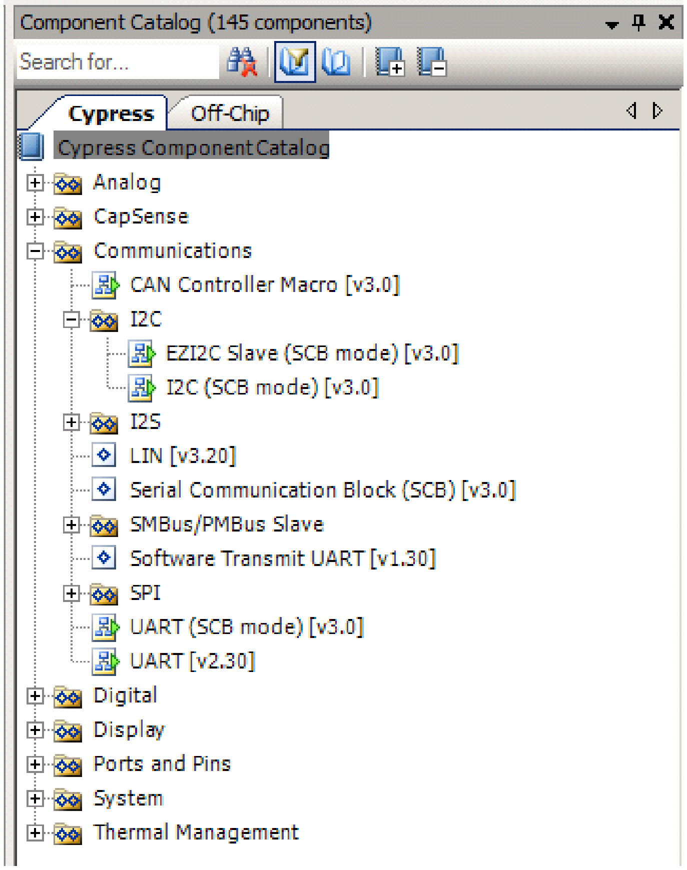
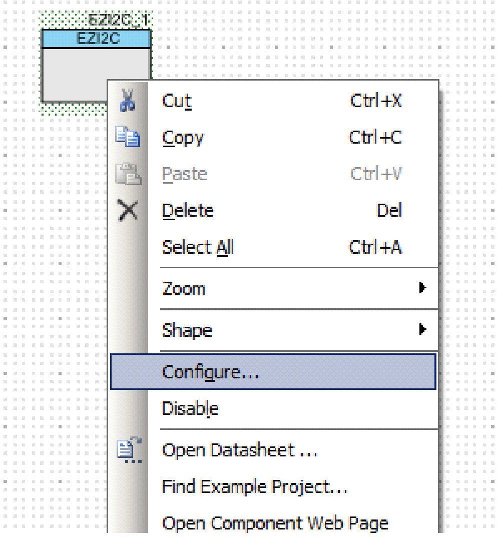
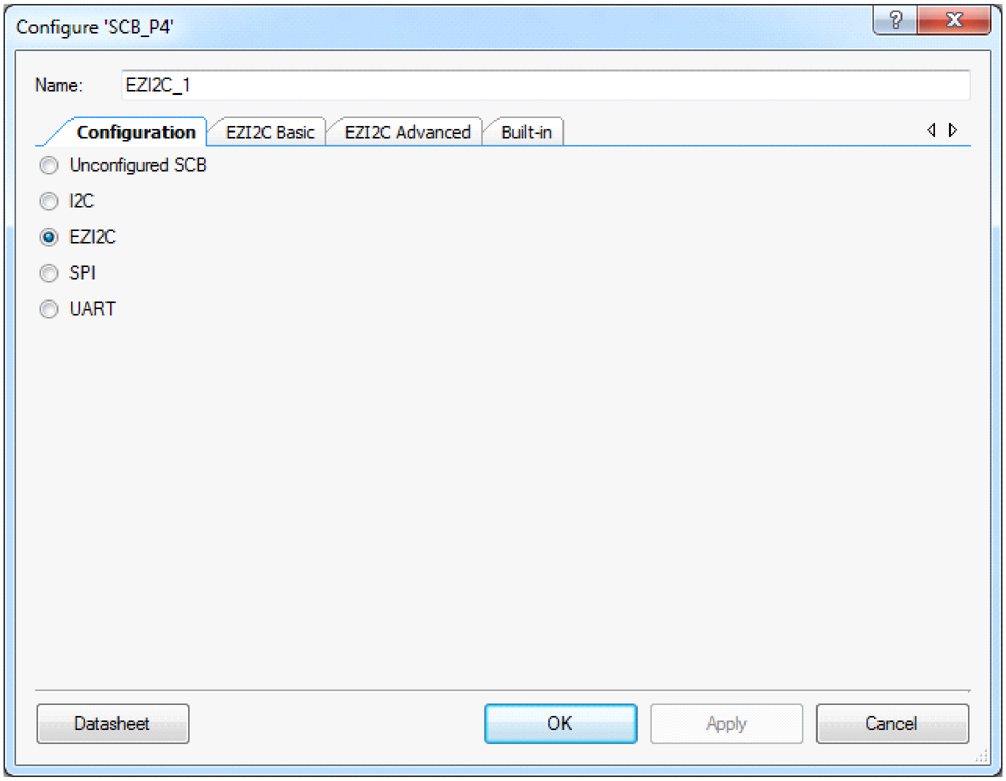


Figure 6-3. Open EZI2C Slave Configuration Window



3. Configure the EZI2C Slave Component as shown in Figure 6-4 and Figure 6-5, and then click **OK**.

Figure 6-4. Configuration Tab

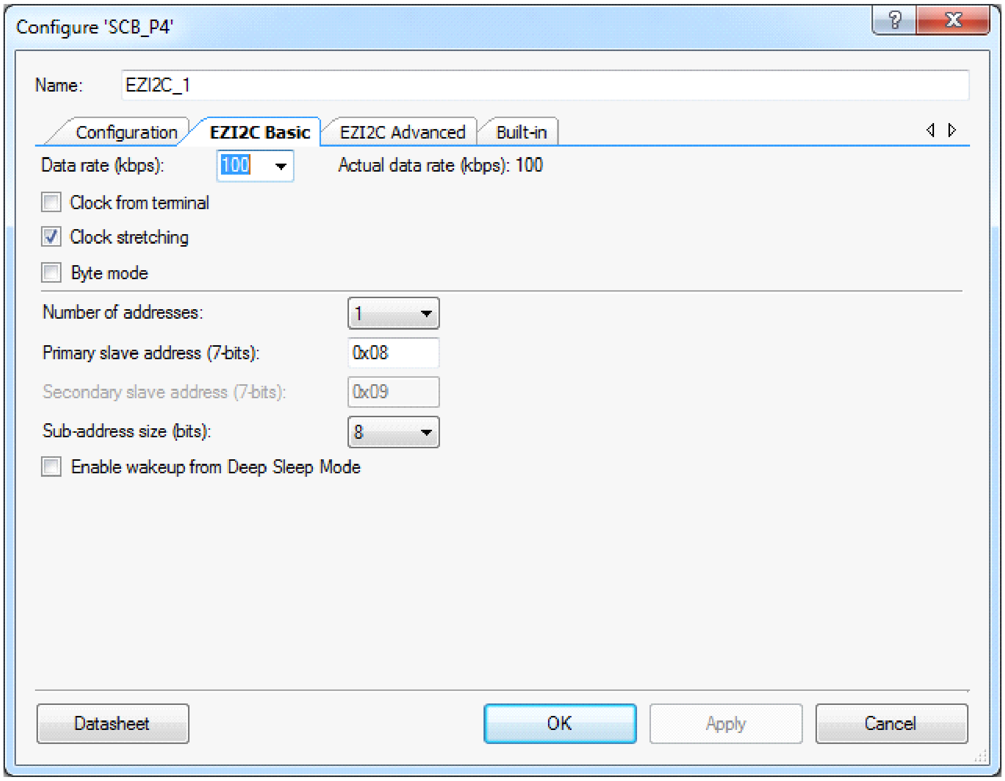


Configure 'SCB_P4'

Name: EZI2C_1

○ Unconfigured SCB
○ I2C
● EZI2C
○ SPI
○ UART

Figure 6-5. EZI2C Slave Basic and Advanced Tabs\



Name: EZI2C_1

Data rate (kbps): 100 Actual data rate (kbps): 100

□ Clock from terminal

☑ Clock stretching

□ Byte mode

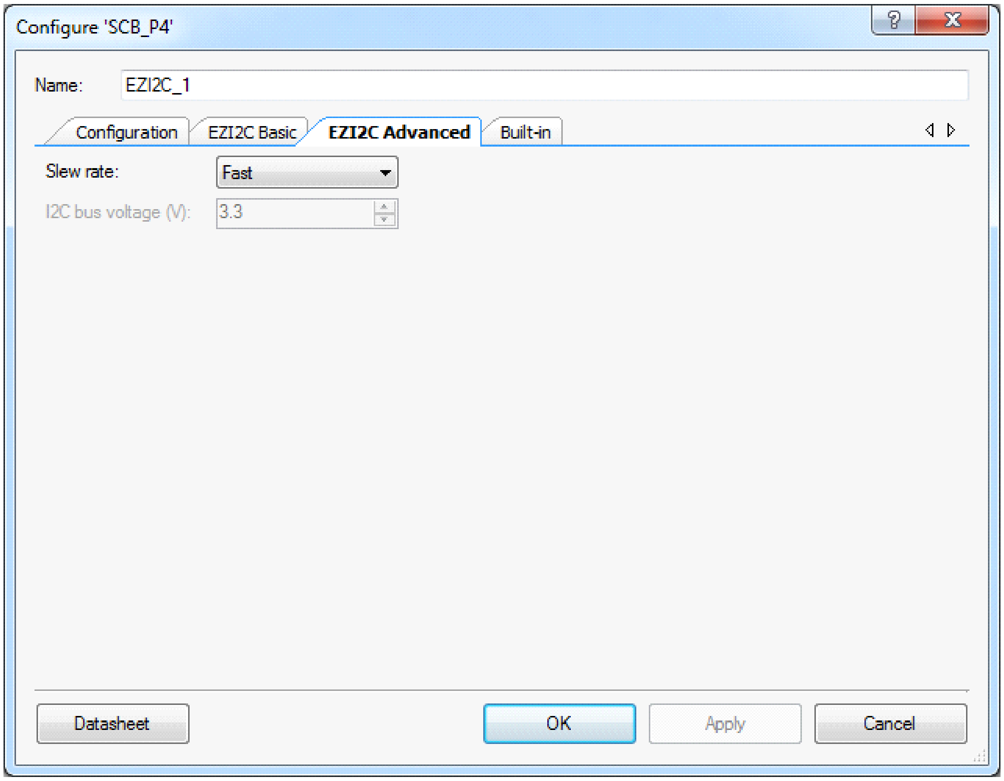
Number of addresses: 1

Primary slave address (7-bits): 0x08

Secondary slave address (7-bits): 0x09

Sub-address size (bits): 8

□ Enable wakeup from Deep Sleep Mode



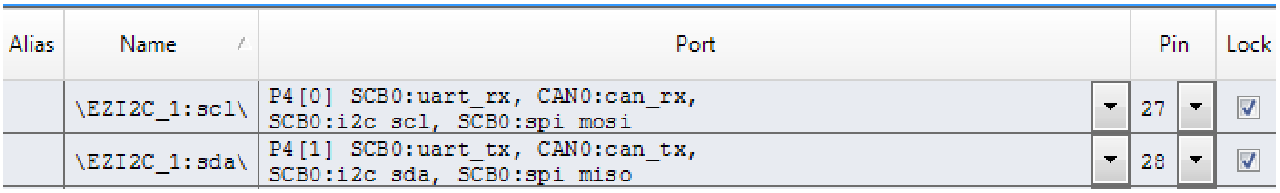
Name: EZI2C_1

Slew rate: Fast

I2C bus voltage (V): 3.3

4. Select pin P4[0] for the I2C SCL and pin P4[1] for the I2C SDA in the **Pins** tab of *<Project_Name>.cydwr*, as shown in Figure 6-6. The *<Project_Name>.cydwr* file is available in the Workspace Explorer window, which is located along the left side of the PSoC Creator window by default. Double-click on the file to open it. Note that these are the pins for the USB-I2C interface on the PSoC 4 M-Series Pioneer kit. If you are using a different kit, refer to the respective Kit Guide for the appropriate pins.

Figure 6-6. Pin Selection



Alias	Name	Port	Pin	Lock
	\EZI2C_1:scl\	P4[0] SCB0:uart_rx, CAN0:can_rx, SCB0:i2c scl, SCB0:spi mosi	27	☑
	\EZI2C_1:sda\	P4[1] SCB0:uart_tx, CAN0:can_tx, SCB0:i2c sda, SCB0:spi miso	28	☑

5. Place the following code in the *main.c* file. The code will enable the PSoC 4200M device with the BCP application using the EZI2C Slave interface.

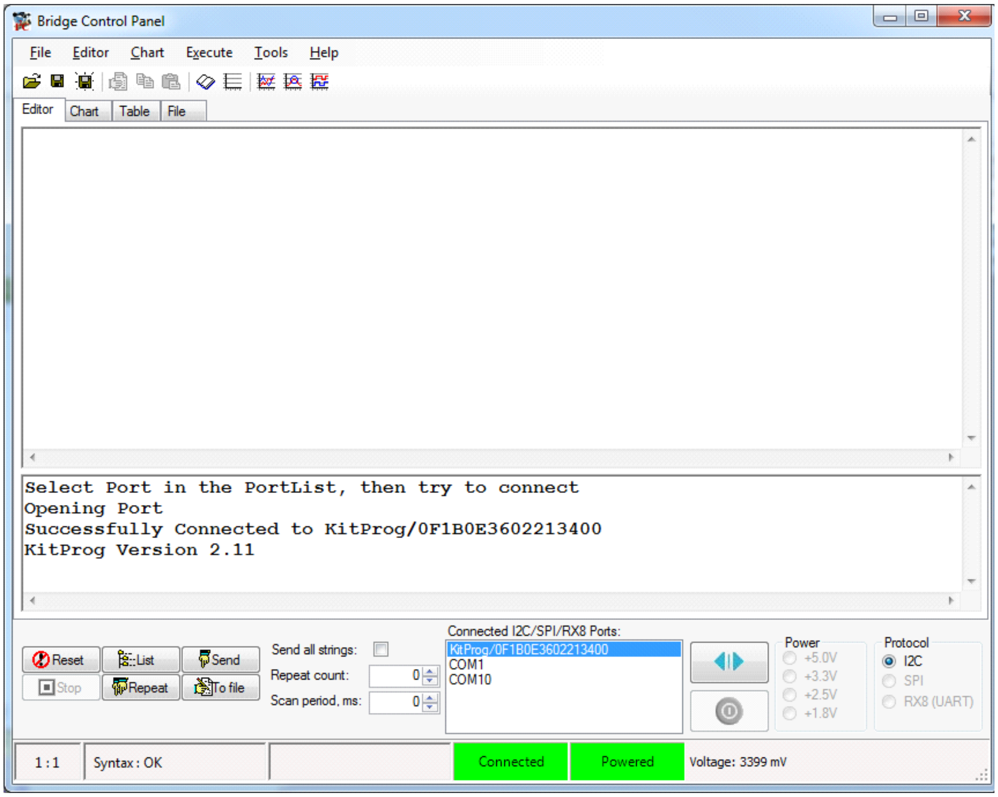
Note: The *main.c* file can be found on the Workspace Explorer window, which is located along the left side of the PSoC Creator window by default. Double-click on the file to open it.

```
#include <project.h>

#define BUF_SIZE                 0x0A
#define READ_WRITE_SIZE    0x05

int main()
{
       /* I2C Read/Write Buffer. */
       uint8 i2cBuffer[BUF_SIZE];

      CyGlobalIntEnable;

       EZI2C_1_Start();

       /* This API sets the buffer and address boundary to which the external
        * master can communicate. In this example, external master can read
        * from and write to the first 5 bytes of the i2cBuffer and read bytes
        * from all the 10 bytes of the i2cBuffer array. */
       EZI2C_1_EzI2CSetBuffer1(BUF_SIZE, READ_WRITE_SIZE, i2cBuffer);

      for(;;)
      {

      }
          }
```

6. Build the project by choosing **Build** > **Build Project** or pressing **[Shift] [F6]**. After the project is built without errors and warnings, program (**[Ctrl] [F5]**) this project onto the PSoC 4200M using KitProg.
7. Open the BCP from **Start** > **All Programs** > **Cypress** > **Bridge Control Panel <version>** > **Bridge Control Panel <version>**.

8. Select **KitProg/<serial number>** under **Connected I2C/SPI/RX8 Ports**, as shown in Figure 6-7.

Figure 6-7. Connecting to KitProg in BCP



```
Select Port in the PortList, then try to connect
Opening Port
Successfully Connected to KitProg/0F1B0E3602213400
KitProg Version 2.11
```

Send all strings: ☐
Repeat count: 0
Scan period, ms: 0

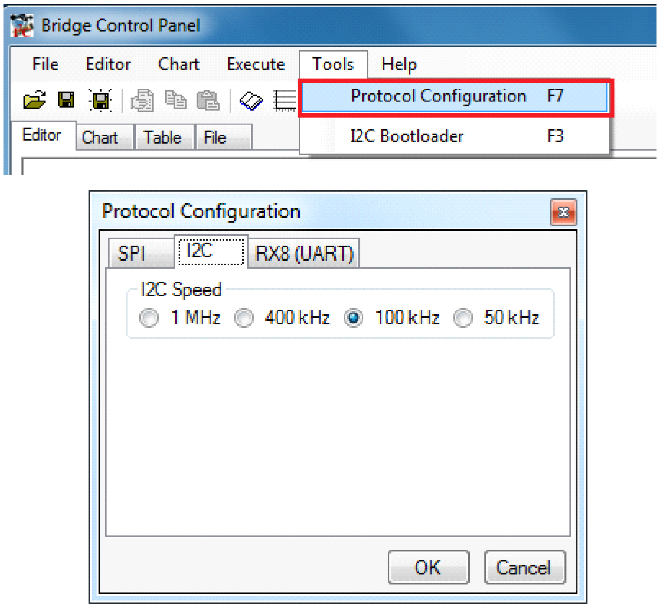
Connected I2C/SPI/RX8 Ports:
KitProg/0F1B0E3602213400
COM1
COM10

Protocol: ● I2C

1 : 1 | Syntax : OK | Connected | Powered | Voltage: 3399 mV

9. Open **Protocol Configuration** from the Tools menu and select the appropriate **I2C Speed**, as shown in Figure 6-8. Ensure that the I2C speed is the same as the one configured in the EZI2C Slave Component. Click **OK** to close the window.

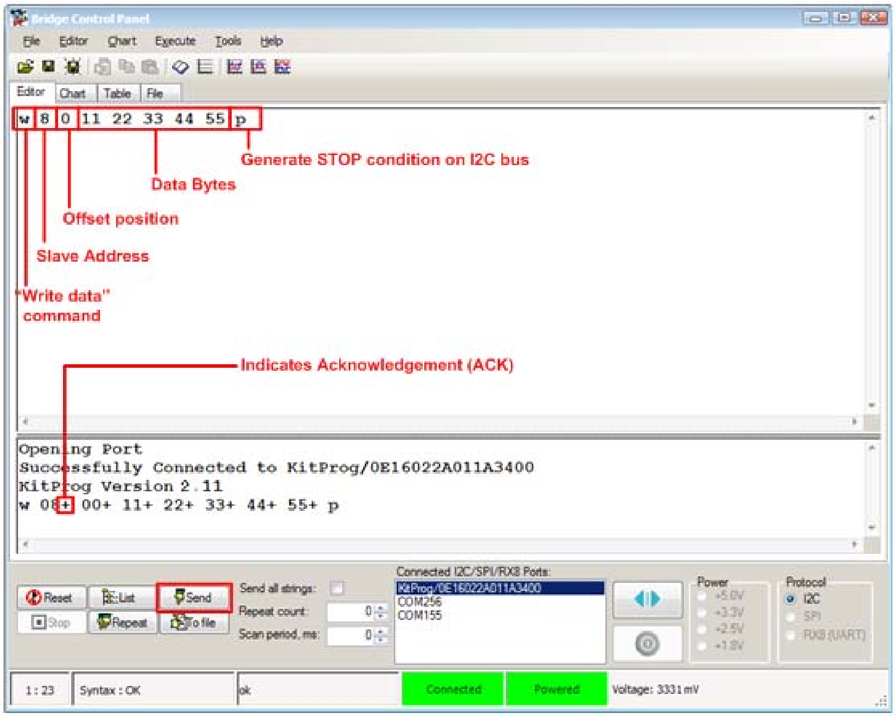
Figure 6-8. Opening Protocol Configuration Window in BCP



I2C Speed: ○ 1 MHz ○ 400 kHz ● 100 kHz ○ 50 kHz

10. From the BCP, transfer 5 bytes of data to the I2C device with slave address 0x08. The EZI2C Slave requires an additional parameter to be sent from the BCP to set the offset address from/to where the data bytes are read/written. Type the command shown in Figure 6-9 and press **[Enter]** or click the **Send** button in the BCP. The log shows whether the transaction was successful. A "+" after a byte indicates that the transaction was successful, and a "-" indicates that the transaction was a failure.

Figure 6-9. Enter Commands in BCP



```
w 8 0 11 22 33 44 55 p
```

```
Opening Port
Successfully Connected to KitProg/0E16022A011A3400
KitProg Version 2.11
w 08+ 00+ 11+ 22+ 33+ 44+ 55+ p
```

Send all strings: □
Repeat count: 0
Scan period, ms: 0

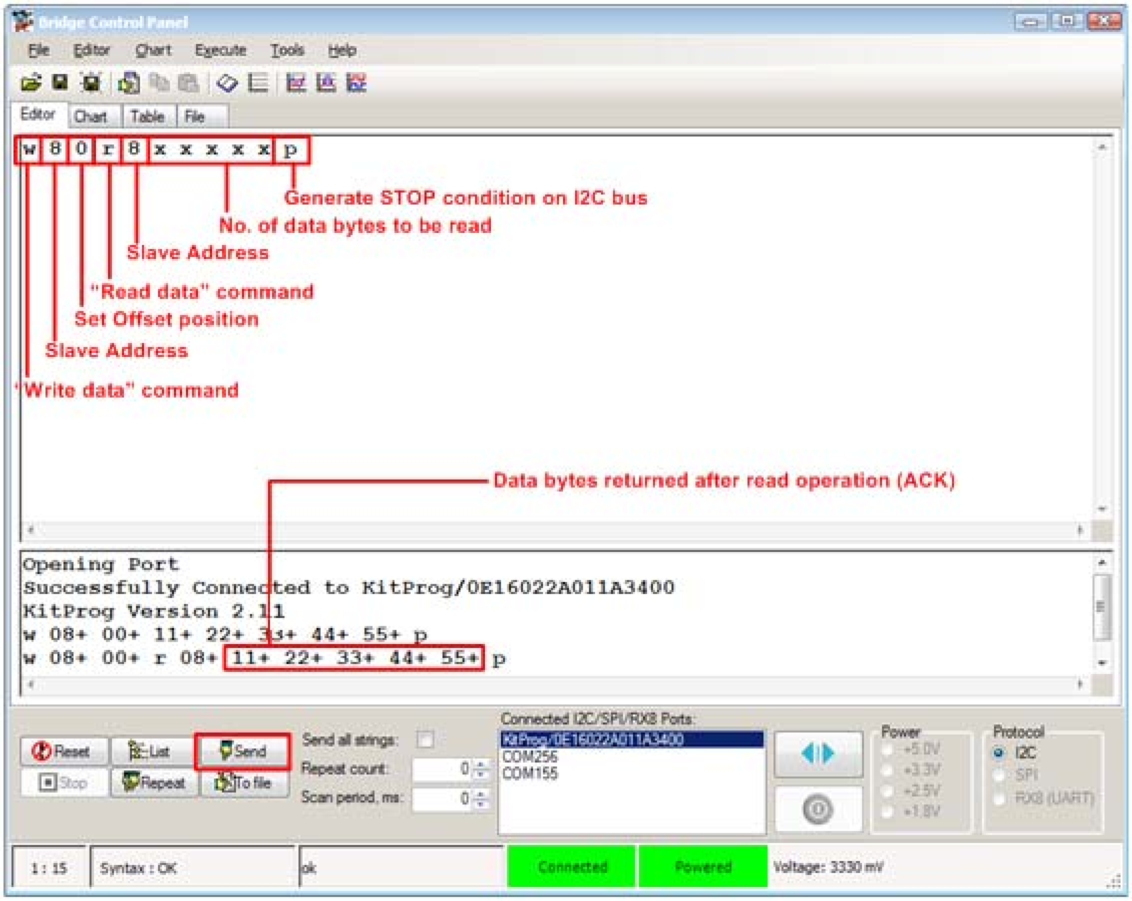
Connected I2C/SPI/RX8 Ports:
KitProg/0E16022A011A3400
COM256
COM155

Protocol: ● I2C

1 : 23 | Syntax : OK | ok | Connected | Powered | Voltage: 3331 mV

11. From the BCP, read 5 bytes of data from the I2C slave device with slave address 0x08. The log shows if the transaction was successful, as shown in Figure 6-10.

Figure 6-10. Read Data Bytes from BCP



```
w 8 0 r 8 x x x x x p
```

```
Opening Port
Successfully Connected to KitProg/0E16022A011A3400
KitProg Version 2.11
w 08+ 00+ 11+ 22+ 33+ 44+ 55+ p
w 08+ 00+ r 08+ 11+ 22+ 33+ 44+ 55+ p
```

Connected I2C/SPI/RX8 Ports: KitProg/0E16022A011A3400, COM256, COM155

Repeat count: 0
Scan period, ms: 0
Protocol: ● I2C

1 : 15 | Syntax : OK | ok | Connected | Powered | Voltage: 3330 mV

Note: You can add additional lines of commands by pressing **[Ctrl] [Enter]**. To execute any line, click on that line and press **[Enter]** or click the **Send** button.

Refer to **Help** > **Help Contents** in the BCP or press **[F1]** for more information on the I2C commands.

7. Developing Applications for PSoC 5LP

The KitProg is implemented using a PSoC 5LP device. You can also use the PSoC 5LP as a mixed-signal system-on-chip device to build your own custom projects. For example, the PSoC 5LP on the kit can be reprogrammed to act as a function generator for the kit. Refer to the application note AN69133 - PSoC® 3 / PSoC 5LP Easy Waveform Generation with the WaveDAC8 Component for details on how to create waveforms using a PSoC 5LP device.

Two types of projects can be created for a PSoC 5LP that runs KitProg: **Bootloadable** and **Normal**. Bootloadable projects can be programmed into the PSoC 5LP using the USB connection from a PC without any specialized hardware. To program Normal projects, you will require a MiniProg3. You also need to populate the PSoC 5LP programming header on the development kit. For the PSoC 4 M-Series Pioneer Kit, this header is marked **J5**. See the respective kit guide for more information on the PSoC 5LP programming header. Jump to the section Building a Normal Project for PSoC 5LP chapter on page 42, if you want to create a normal project for PSoC 5LP.

To learn more about the bootloading concept, refer to the application note AN73854 - PSoC® 3, PSoC 4, and PSoC 5LP Introduction to Bootloaders.

Note: The CY3280-MBR3 CapSense Evaluation Kit does not have a provision to populate the programming header for PSoC 5LP.

The following sections provide step by step directions for building a Bootloadable and a Normal project for PSoC 5LP.

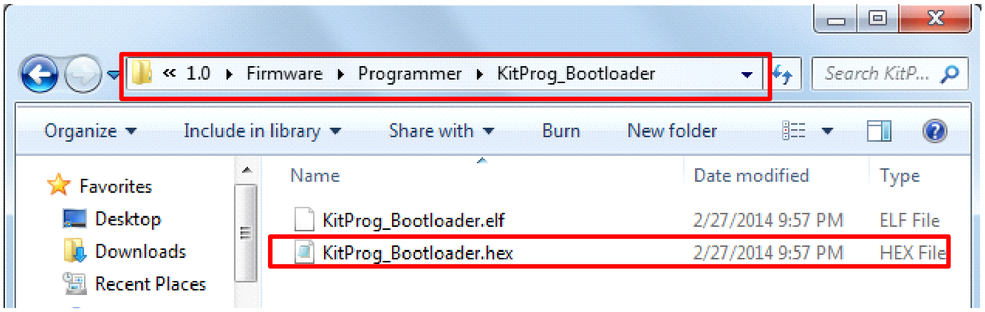
7.1 Building a Bootloadable Project for PSoC 5LP

All bootloadable applications developed for the PSoC 5LP should be based on the bootloader *.hex* file, which is programmed onto the kit. Therefore, you will need to provide the location of the bootloader *.hex* file inside the bootloadable project.

The bootloader *.hex* file is included in the kit installer directory in the following path, as shown in Figure 7-1:

```
<Install_Directory>\<Kit_Name>\<version>\Firmware\Programmer\KitProg_Boot
loader
```

Figure 7-1. KitProg Bootloader Hex File Location



« 1.0 ▸ Firmware ▸ Programmer ▸ KitProg_Bootloader

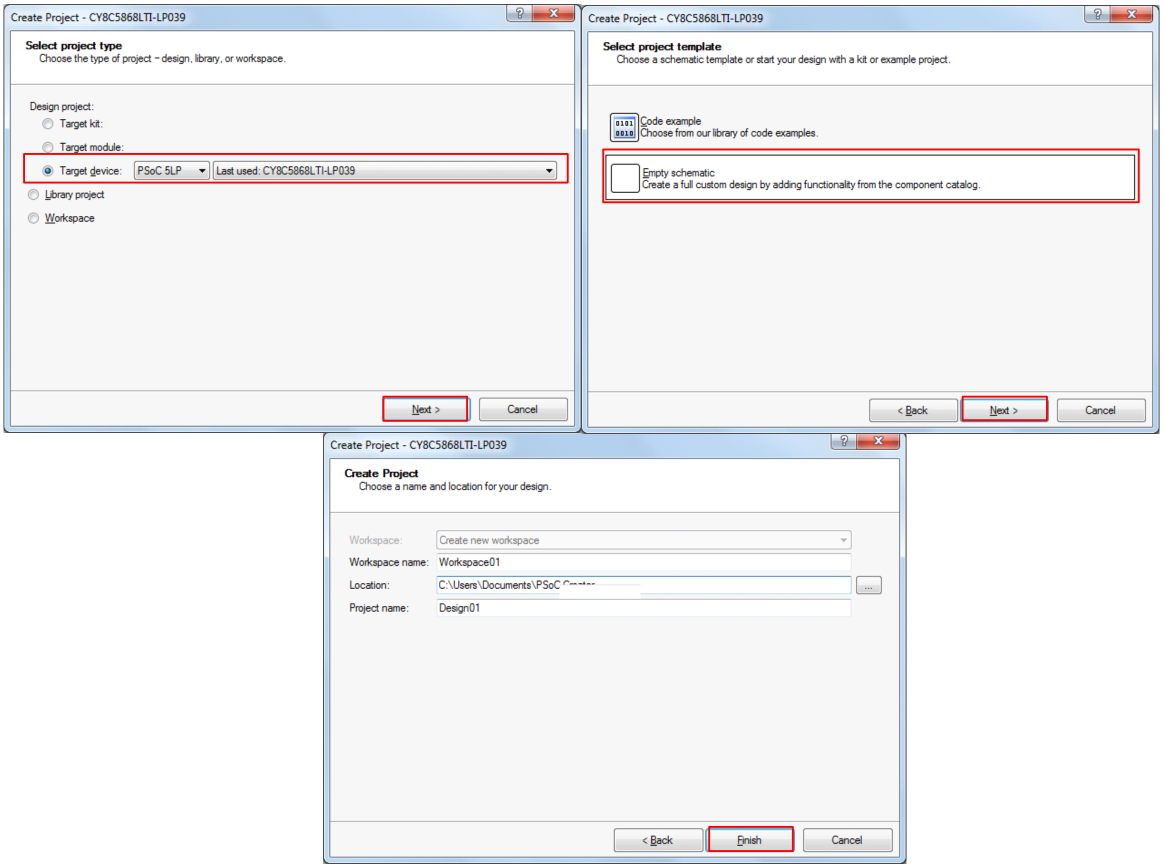
Name	Date modified	Type
KitProg_Bootloader.elf	2/27/2014 9:57 PM	ELF File
KitProg_Bootloader.hex	2/27/2014 9:57 PM	HEX File

To build a bootloadable application for the PSoC 5LP, follow this procedure:

1. In PSoC Creator, choose **New** > **Project** and click the **PSoC 5LP Design**; select **Launch Device Selector** from the drop-down list for **Device** to bring up the **Select PSoC 5LP Device** window and select **CY8C5868LTI-LP039**, as shown in Figure 7-3. Click **OK**.

Note: If you have not set the **Application Type** as **Bootloadable** in the New Project window under the Advanced section (in PSoC Creator 3.1 or earlier), you can change it in the existing project by selecting **Project** > **Build Settings** and click the **<Project Name>** > **Application Type** > **Bootloadable**. Beginning with PSoC Creator 3.2, the **Application Type** option is removed from the New Project window and the Build Settings menu. PSoC Creator 3.2 automatically recognizes the application type from the TopDesign schematic.

Figure 7-2. Open New Project in PSoC Creator



Create Project - CY8C5868LTI-LP039

Select project type

Choose the type of project – design, library, or workspace.

Design project:

○ Target kit:

○ Target module:

● Target device: PSoC 5LP | Last used: CY8C5868LTI-LP039

○ Library project

○ Workspace

Create Project - CY8C5868LTI-LP039

Select project template

Choose a schematic template or start your design with a kit or example project.

Code example
Choose from our library of code examples.

Empty schematic
Create a full custom design by adding functionality from the component catalog.

Create Project - CY8C5868LTI-LP039

Create Project

Choose a name and location for your design.

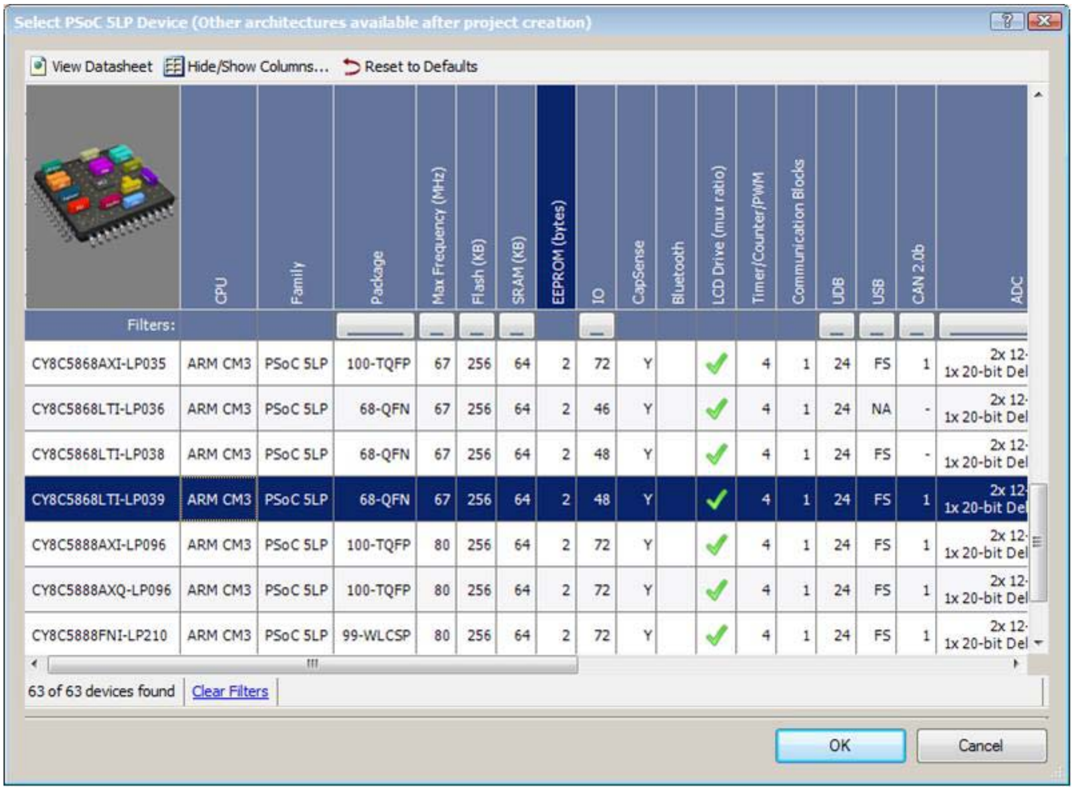
Workspace: Create new workspace

Workspace name: Workspace01

Location: C:\Users\Documents\PSoC Creator

Project name: Design01

Figure 7-3. Select Device in PSoC Creator



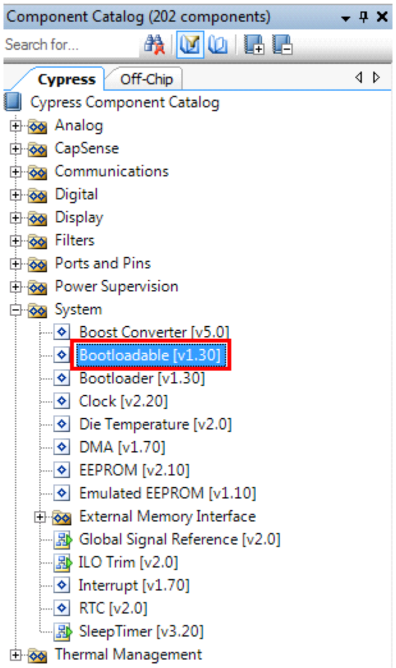
Select PSoC 5LP Device (Other architectures available after project creation)

	CPU	Family	Package	Max Frequency (MHz)	Flash (KB)	SRAM (KB)	EEPROM (bytes)	IO	CapSense	Bluetooth	LCD Drive (mux ratio)	Timer/Counter/PWM	Communication Blocks	UDB	USB	CAN 2.0b	ADC
CY8C5868AXI-LP035	ARM CM3	PSoC 5LP	100-TQFP	67	256	64	2	72	Y		✓	4	1	24	FS	1	2x 12- 1x 20-bit Del
CY8C5868LTI-LP036	ARM CM3	PSoC 5LP	68-QFN	67	256	64	2	46	Y		✓	4	1	24	NA	-	2x 12- 1x 20-bit Del
CY8C5868LTI-LP038	ARM CM3	PSoC 5LP	68-QFN	67	256	64	2	48	Y		✓	4	1	24	FS	-	2x 12- 1x 20-bit Del
CY8C5868LTI-LP039	ARM CM3	PSoC 5LP	68-QFN	67	256	64	2	48	Y		✓	4	1	24	FS	1	2x 12- 1x 20-bit Del
CY8C5888AXI-LP096	ARM CM3	PSoC 5LP	100-TQFP	80	256	64	2	72	Y		✓	4	1	24	FS	1	2x 12- 1x 20-bit Del
CY8C5888AXQ-LP096	ARM CM3	PSoC 5LP	100-TQFP	80	256	64	2	72	Y		✓	4	1	24	FS	1	2x 12- 1x 20-bit Del
CY8C5888FNI-LP210	ARM CM3	PSoC 5LP	99-WLCSP	80	256	64	2	72	Y		✓	4	1	24	FS	1	2x 12- 1x 20-bit Del

63 of 63 devices found | Clear Filters

2. Navigate to the Schematic view and drag and drop a Bootloadable Component (see Figure 7-4) on the TopDesign.

Figure 7-4. Bootloadable Component in Component Catalog



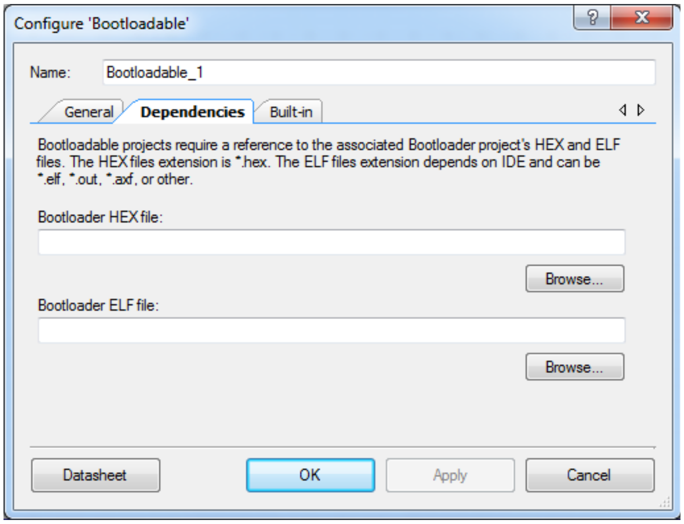
Component Catalog (202 components)

- Cypress Component Catalog
  - Analog
  - CapSense
  - Communications
  - Digital
  - Display
  - Filters
  - Ports and Pins
  - Power Supervision
  - System
    - Boost Converter [v5.0]
    - Bootloadable [v1.30]
    - Bootloader [v1.30]
    - Clock [v2.20]
    - Die Temperature [v2.0]
    - DMA [v1.70]
    - EEPROM [v2.10]
    - Emulated EEPROM [v1.10]
    - External Memory Interface
    - Global Signal Reference [v2.0]
    - ILO Trim [v2.0]
    - Interrupt [v1.70]
    - RTC [v2.0]
    - SleepTimer [v3.20]
  - Thermal Management

3. Set the dependency of the Bootloadable Component by selecting the **Dependencies** tab in the configuration window and clicking the **Browse** button, as shown in Figure 7-5. Select the *KitProg_Bootloader.hex* (see Figure 7-6) and click **Open**.

Note: The *KitProg_Bootloader.elf* is selected automatically if it is also available with the same name in the same path. Ensure that both *.hex* and *.elf* file exist in the same folder by the same name.

Figure 7-5. Configuration Window of Bootloadable Component



Configure 'Bootloadable'

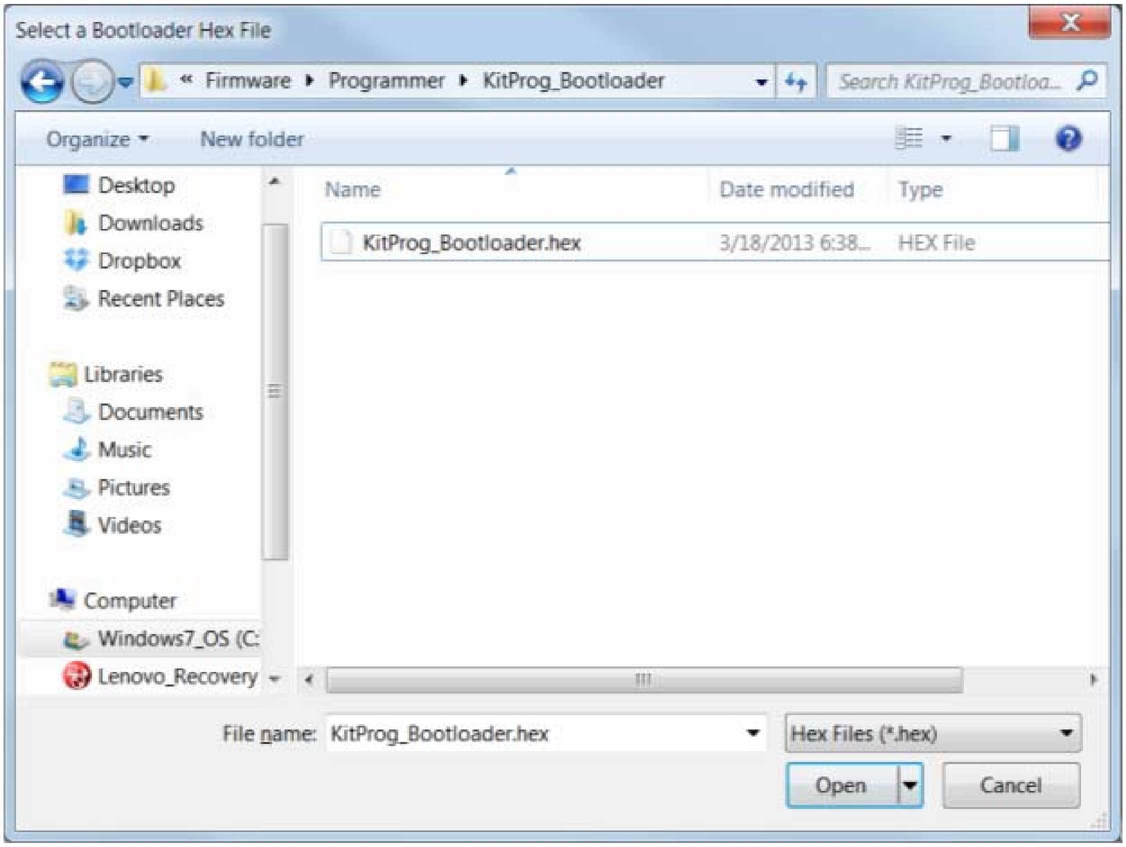
Name: Bootloadable_1

Bootloadable projects require a reference to the associated Bootloader project's HEX and ELF files. The HEX files extension is *.hex. The ELF files extension depends on IDE and can be *.elf, *.out, *.axf, or other.

Bootloader HEX file:

Bootloader ELF file:

Figure 7-6. Select KitProg Bootloader Hex File



Select a Bootloader Hex File

« Firmware ▸ Programmer ▸ KitProg_Bootloader

Search KitProg_Bootloa...

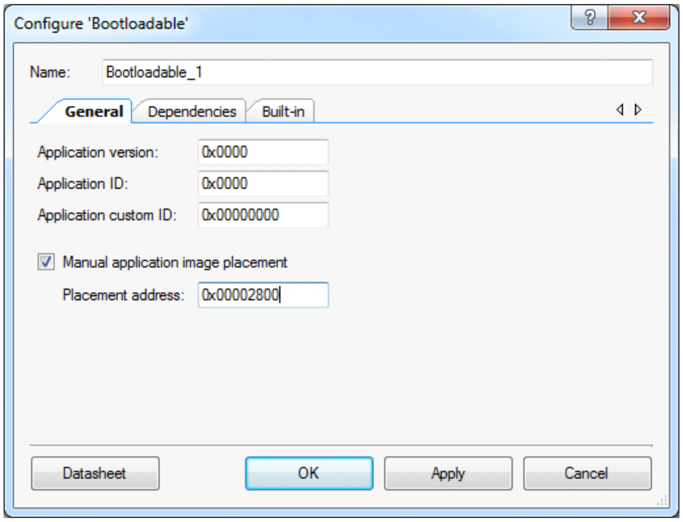
Name	Date modified	Type
KitProg_Bootloader.hex	3/18/2013 6:38...	HEX File

File name: KitProg_Bootloader.hex

Hex Files (*.hex)

4. In the **General** tab, check the **Manual application image placement** checkbox and set the **Placement address** as ‘0x00002800’, as shown in Figure 7-7.

Figure 7-7. Bootloadable Component-General Tab



Configure 'Bootloadable'

Name: Bootloadable_1

Application version: 0x0000

Application ID: 0x0000

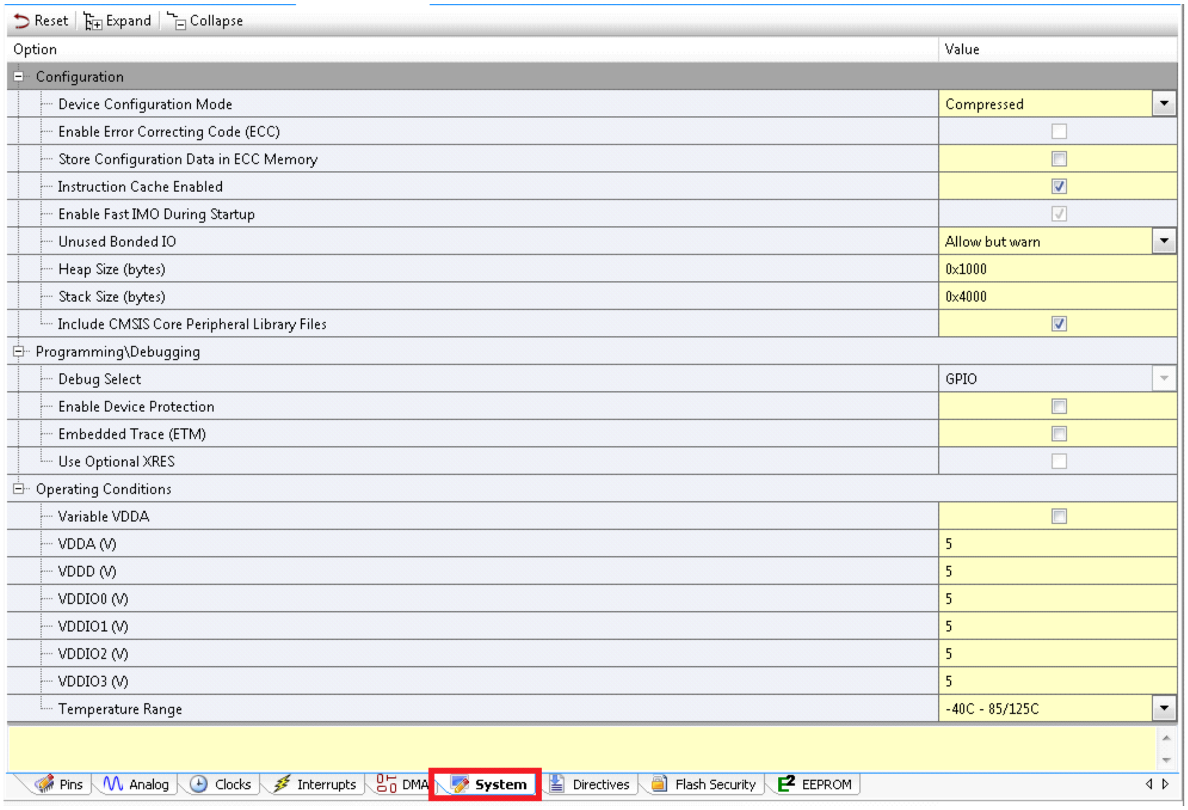
Application custom ID: 0x00000000

☑ Manual application image placement

Placement address: 0x00002800

5. Develop your custom project.
6. Ensure that the *<project name>.cydwr* **System** settings of the Bootloadable project and the KitProg_Bootloader project are the same. Figure 7-8 shows the *KitProg_Bootloader.cydwr* **System** settings.

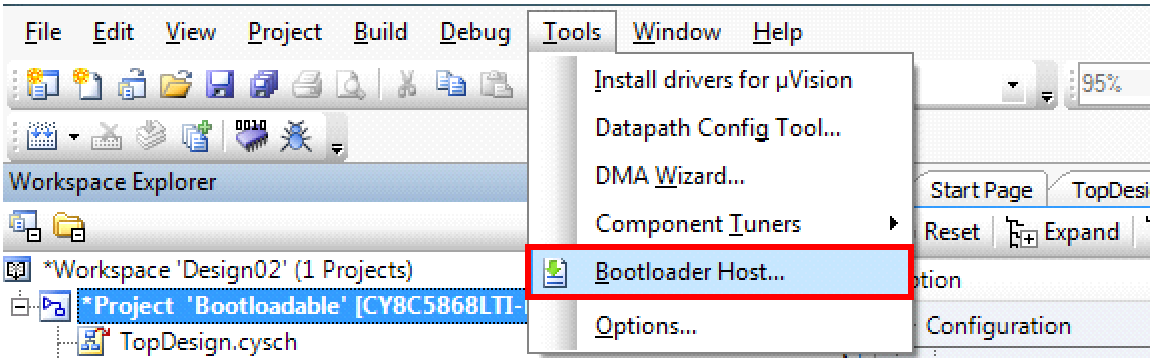
Figure 7-8. KitProg Bootloader System Settings



Option	Value
Configuration	
Device Configuration Mode	Compressed
Enable Error Correcting Code (ECC)	□
Store Configuration Data in ECC Memory	□
Instruction Cache Enabled	☑
Enable Fast IMO During Startup	☑
Unused Bonded IO	Allow but warn
Heap Size (bytes)	0x1000
Stack Size (bytes)	0x4000
Include CMSIS Core Peripheral Library Files	☑
Programming\Debugging	
Debug Select	GPIO
Enable Device Protection	□
Embedded Trace (ETM)	□
Use Optional XRES	□
Operating Conditions	
Variable VDDA	□
VDDA (V)	5
VDDD (V)	5
VDDIO0 (V)	5
VDDIO1 (V)	5
VDDIO2 (V)	5
VDDIO3 (V)	5
Temperature Range	-40C - 85/125C

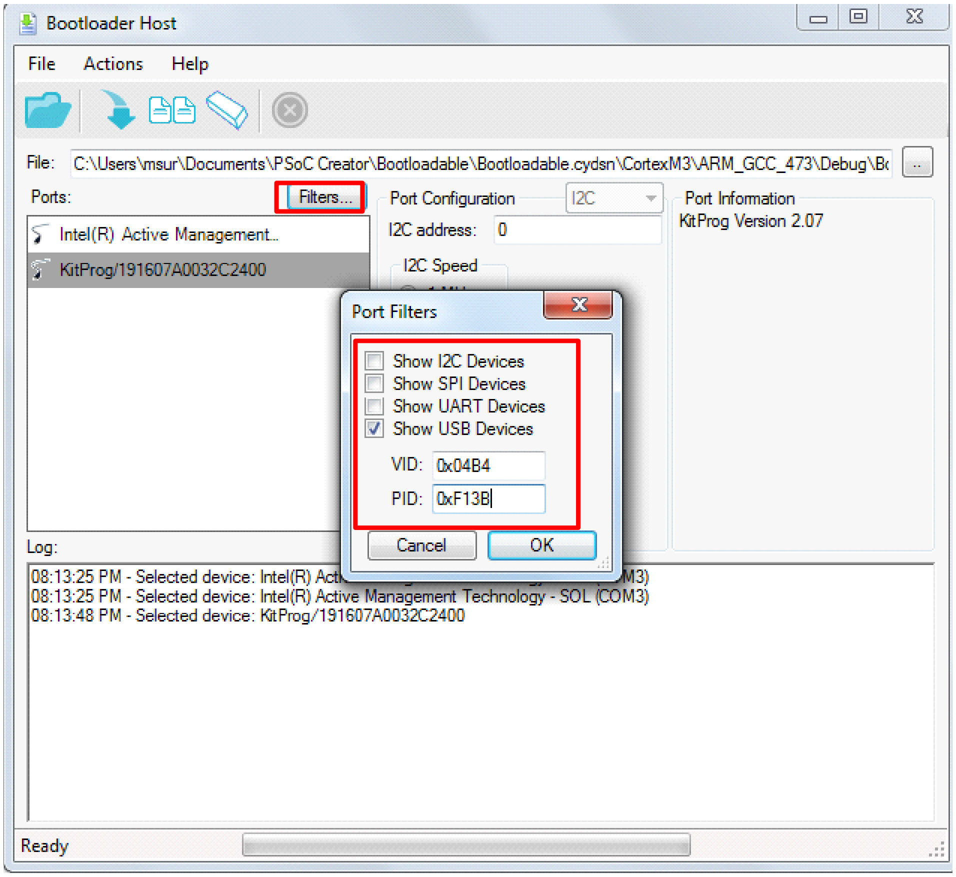
7. Build the project in PSoC Creator by choosing **Build** > **Build Project** or pressing **[Shift] [F6]**.
8. To program the project onto the PSoC 5LP device, open the Bootloader Host tool, which is available in PSoC Creator. Choose **Tools** > **Bootloader Host**, as shown in Figure 7-9.

Figure 7-9. Open Bootloader Host Tool in PSoC Creator



9. Keep the reset switch (**SW1**) pressed and connect the kit to the computer. If the switch is pressed for more than 100 ms, the PSoC 5LP enters the bootloader.
10. In the Bootloader Host tool, click **Filters** and add a filter to identify the USB device. Ensure that the check box for **Show USB Devices** is enabled. Set VID as **0x04B4**, PID as **0xF13B**, and click **OK**, as shown in Figure 7-10.

Figure 7-10. Port Filters Tab in Bootloader Host Tool



File: C:\Users\msur\Documents\PSoC Creator\Bootloadable\Bootloadable.cydsn\CortexM3\ARM_GCC_473\Debug\Bc

I2C address: 0

KitProg Version 2.07

□ Show I2C Devices
□ Show SPI Devices
□ Show UART Devices
☑ Show USB Devices

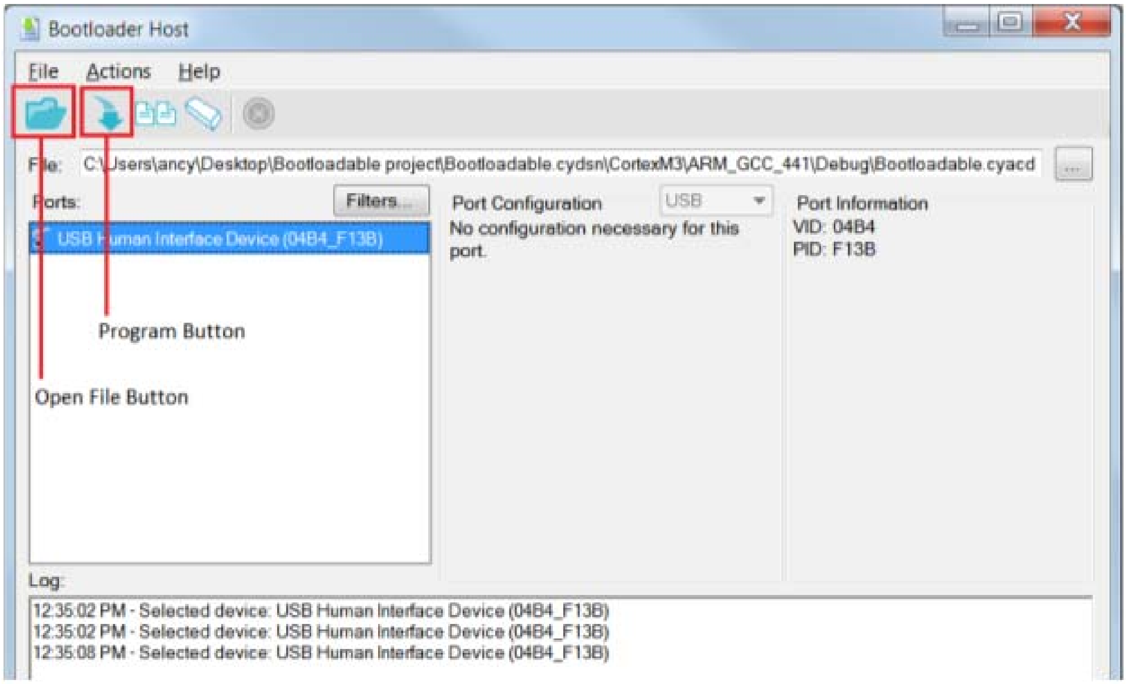
VID: 0x04B4

PID: 0xF13B

```
08:13:25 PM - Selected device: Intel(R) Acti...M3)
08:13:25 PM - Selected device: Intel(R) Active Management Technology - SOL (COM3)
08:13:48 PM - Selected device: KitProg/191607A0032C2400
```

11. In the Bootloader Host tool, click the **Open File** button (Figure 7-11) to browse to the location of the bootloadable file (*.*cyacd*), as shown in Figure 7-12. This file is present in the project directory.

Figure 7-11. Open Bootloadable File in Bootloader Host Tool



File: C:\Users\ancy\Desktop\Bootloadable project\Bootloadable.cydsn\CortexM3\ARM_GCC_441\Debug\Bootloadable.cyacd

Ports: USB Human Interface Device (04B4_F13B)

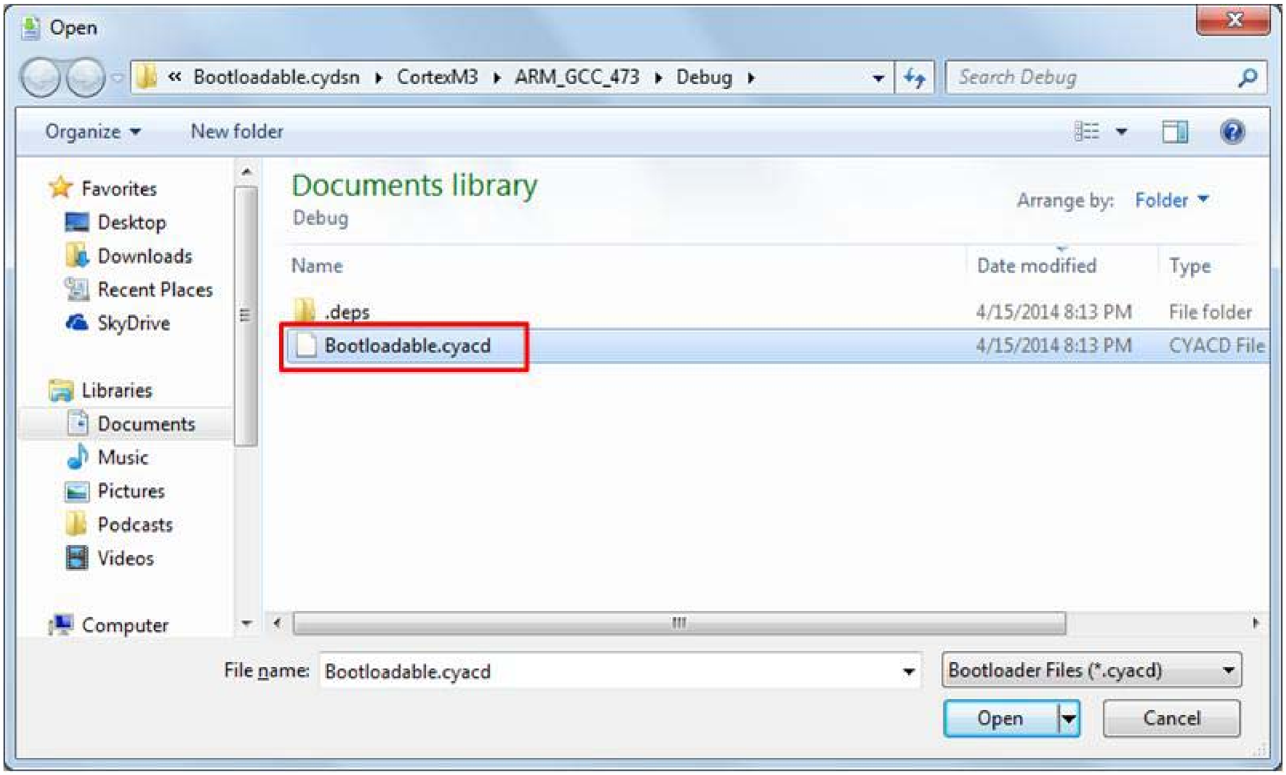
Port Configuration: USB
No configuration necessary for this port.

Port Information
VID: 04B4
PID: F13B

Log:

```
12:35:02 PM - Selected device: USB Human Interface Device (04B4_F13B)
12:35:02 PM - Selected device: USB Human Interface Device (04B4_F13B)
12:35:08 PM - Selected device: USB Human Interface Device (04B4_F13B)
```

Figure 7-12. Select Bootloadable .*cyacd* File from Bootloader Host Tool



Name	Date modified	Type
.deps	4/15/2014 8:13 PM	File folder
Bootloadable.cyacd	4/15/2014 8:13 PM	CYACD File

File name: Bootloadable.cyacd

Bootloader Files (*.cyacd)

12. Select the **USB Human Interface Device** in the **Ports** list and click the **Program** button (Figure 7-11) in the Bootloader Host tool to program the device.
13. If the bootload is successful, the log displays “Programming Finished Successfully”; otherwise, it displays “Failed” and a reason for the failure.

Notes:

- The PSoC 5LP pins are connected to the PSoC 5LP GPIO header. These pins are selected to support high-performance analog and digital projects. See A.1 Pin Assignment Tables for pin information.
- Take care when allocating the PSoC 5LP pins for custom applications. For example, P3[2]-P3[3] are dedicated for programming the PSoC 4200M in CY8CKIT-044. Refer to the respective kit schematics before allocating the pins.
- When a custom bootloadable project is programmed onto the PSoC 5LP, the initial capability of the PSoC 5LP to act as a programmer, USB-UART bridge, or USB-I2C bridge is not available. To recover this functionality, bootload the KitProg file back into the PSoC 5LP. The file is available in the kit installation directory at:
`<Install_Directory>\<Kit_Name>\<version>\Firmware\Programmer\KitProg\KitProg.cyacd`
- The status LED does not function unless it is used by the custom project.

For additional information on bootloaders, refer to the AN73503 - USB HID Bootloader for PSoC 3 and PSoC 5LP.

7.2 Building a Normal Project for PSoC 5LP

A normal project is a completely new project created for the PSoC 5LP device on the PSoC 4 M-Series Pioneer board. Here the entire flash of the PSoC 5LP is programmed, overwriting all bootloader and programming code. To recover the programmer, USB-UART bridge or USB-I2C bridge functionality, reprogram the PSoC 5LP device with the factory-set KitProg.hex file, which is shipped with the kit installer.

Note: You cannot program a normal PSoC 5LP project into the KitProg's PSoC 5LP device in prototyping kits such as CY8CKIT-059 and CY8CKIT-043. The PSoC 5LP device present in the KitProg of the prototyping kits supports programming through bootloading only.

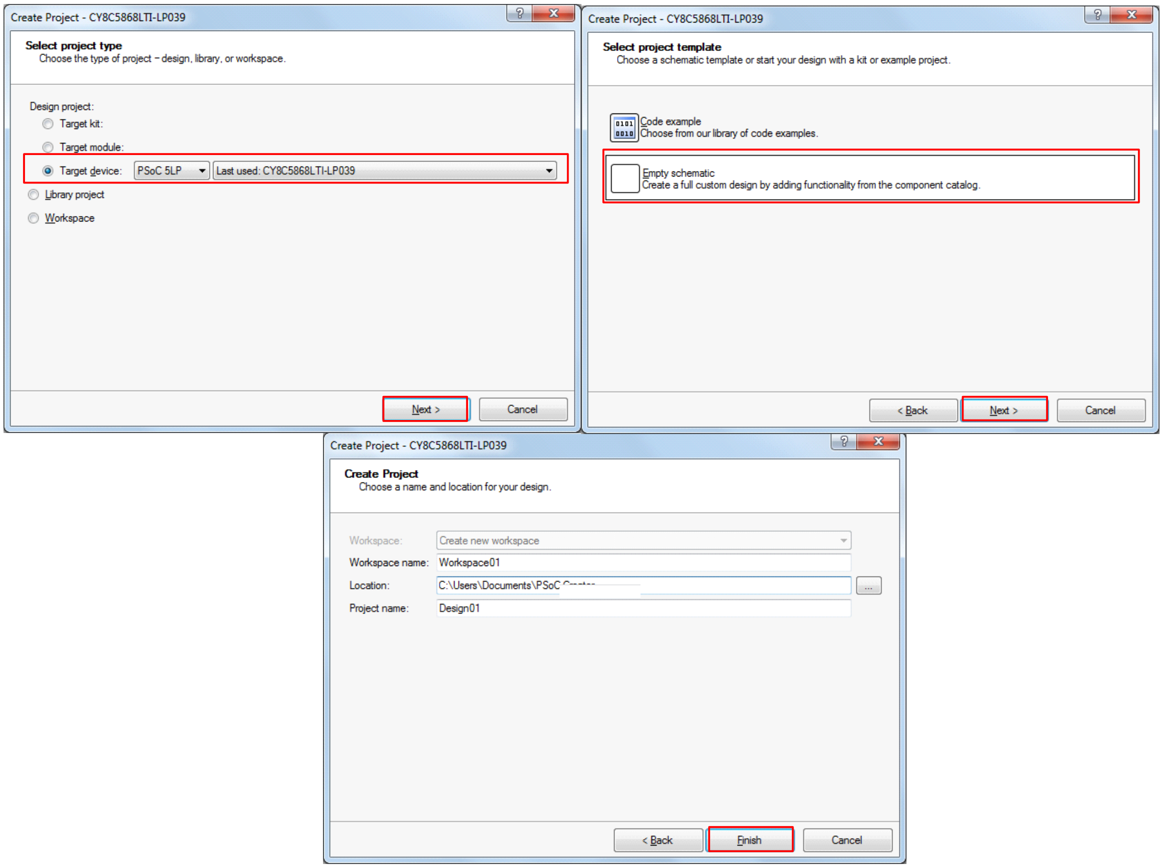
The *KitProg.hex* file is available at the following location:
`<Install_Directory>\<Name_of_the_Kit>\<version>\Firmware\Programmer\KitProg\KitProg.hex`

This advanced functionality requires a MiniProg3 programmer, which is not included with this kit. The MiniProg3 can be purchased from www.cypress.com/go/CY8CKIT-002. In addition, the 10-pin PSoC 5LP programming header (PSoC 5LP PROG) on the kit needs to be populated (refer to the kit’s Bill of Materials for ordering the part) to connect the MiniProg3 for programming and debugging the PSoC 5LP device.

To build a normal project for the PSoC 5LP, follow these steps:

1. In PSoC Creator, choose **New** > **Project** and click the **PSoC 5LP Design**; select **Device** as **CY8C5868LTI-LP039** (see Figure 7-13), and then click **OK**.

Figure 7-13. Create New Project in PSoC Creator



Create Project - CY8C5868LTI-LP039

Select project type
Choose the type of project – design, library, or workspace.

Design project:
○ Target kit:
○ Target module:
● Target device: PSoC 5LP Last used: CY8C5868LTI-LP039
○ Library project
○ Workspace

Create Project - CY8C5868LTI-LP039

Select project template
Choose a schematic template or start your design with a kit or example project.

Code example
Choose from our library of code examples.

Empty schematic
Create a full custom design by adding functionality from the component catalog.

Create Project - CY8C5868LTI-LP039

Create Project
Choose a name and location for your design.

Workspace: Create new workspace
Workspace name: Workspace01
Location: C:\Users\Documents\PSoC
Project name: Design01

2. Develop your custom project.
3. Build the project in PSoC Creator by choosing **Build** > **Build Project** or pressing **[Shift] [F6]**.
4. Connect the 10-pin connector of MiniProg3 to the onboard PSoC 5LP PROG header.
5. To program the PSoC 5LP with PSoC Creator, choose **Debug** > **Program** or press **[Ctrl] [F5]**. If the **Select Debug Target** window appears and shows MiniProg3 and the selected device in the project under it (CY8C5868LTI-LP039), click on the device and click **Connect** to program.

Notes:

- The 10-pin PSoC 5LP programming header is not populated.
- The PSoC 5LP pins are brought to the PSoC 5LP GPIO header. These pins are selected to support high-performance analog and digital projects. See A.1 Pin Assignment Tables for pin information.
- Take care when allocating the PSoC 5LP pins for custom applications. For example, P3[2]-P3[3] are dedicated for programming the PSoC 4200M in CY8CKIT-044. Refer to the respective kit schematics before allocating the pins.
- When a normal project is programmed onto the PSoC 5LP, the initial capability of the PSoC 5LP to act as a programmer, USB-UART bridge, or USB-I2C bridge is not available.
- The status LED does not function unless it is used by the custom project.

8. Troubleshooting the KitProg

This section explains the methods to troubleshoot the KitProg and recover the KitProg firmware if you modified it.

8.1 KitProg Status LED Indication

The KitProg Status LED on the development kit indicates the status of the KitProg operation using different blink rates. Table 8-1 shows the KitProg LED indication and the corresponding status of the KitProg.

Table 8-1. Meaning of KitProg LED Indications

User Indication	Scenario	Action Required by User
LED blinks fast: Frequency = 4.00 Hz	LED starts blinking at power up, if bootloadable file is corrupt.	Bootload the *KitProg.cyacd* file: In PSoC Programmer, connect to the kit, go to the **Utilities** tab, and press the **Upgrade Firmware** button.
LED blinks slow: Frequency = 0.67 Hz	Entered Bootloader mode by holding the Reset button during kit power-up.	Release the Reset button and re-plug the kit if you entered this mode by mistake. If the mode entry was intentional, bootload the new *.cyacd* file using the Bootloader Host tool available in PSoC Creator.
LED blinks very fast: Frequency = 15.0 Hz	SWD or I2C operation is in progress. The Kit's COM port connect / disconnect event (only one blink).	In PSoC Programmer, watch the log window for status messages for SWD operations. In the BCP, the LED blinks on I2C command requests. In BCP or any other serial port terminal program, distinguish the kit's COM port number by the blinking LED when the port is connected or disconnected.
LED is ON	USB enumeration successful. Kit is in the idle state waiting for commands.	PSoC Creator, PSoC Programmer, BCP, and any serial port terminal program can use the kit functions.
LED is OFF	Power LED is ON	This means that the USB enumeration was unsuccessful. This may happen if the kit is not powered from the USB host. Verify the USB cable and check if PSoC Programmer is installed on the PC.

Note: The Bridge Control Panel software cannot connect to the KitProg, if the KitProg firmware version is outdated. Refer to Updating the KitProg Firmware on how to update the KitProg firmware.

Note: The programming/debugging function and USB-I2C bridge function of the KitProg are mutually exclusive functions and cannot be used together. As a result, in order to use one function, the other function should be disconnected. For instance, in order to program the device while using USB-I2C bridge in BCP, either close BCP or disconnect the USB-I2C bridge in BCP. The USB-UART bridge function of the KitProg, however, can run in parallel to both programming/debugging and USB-I2C bridge functions.

8.2 PSoC 5LP Factory Program Restore Instructions

8.2.1 PSoC 5LP is Programmed with a Bootloadable Application

Reprogramming or bootloading the PSoC 5LP device with a new flash image will overwrite the KitProg and forfeit the ability to use the PSoC 5LP device as a programmer/debugger for the kit. If the PSoC 5LP is programmed with a bootloadable application, restore the KitProg by using one of the following two methods:

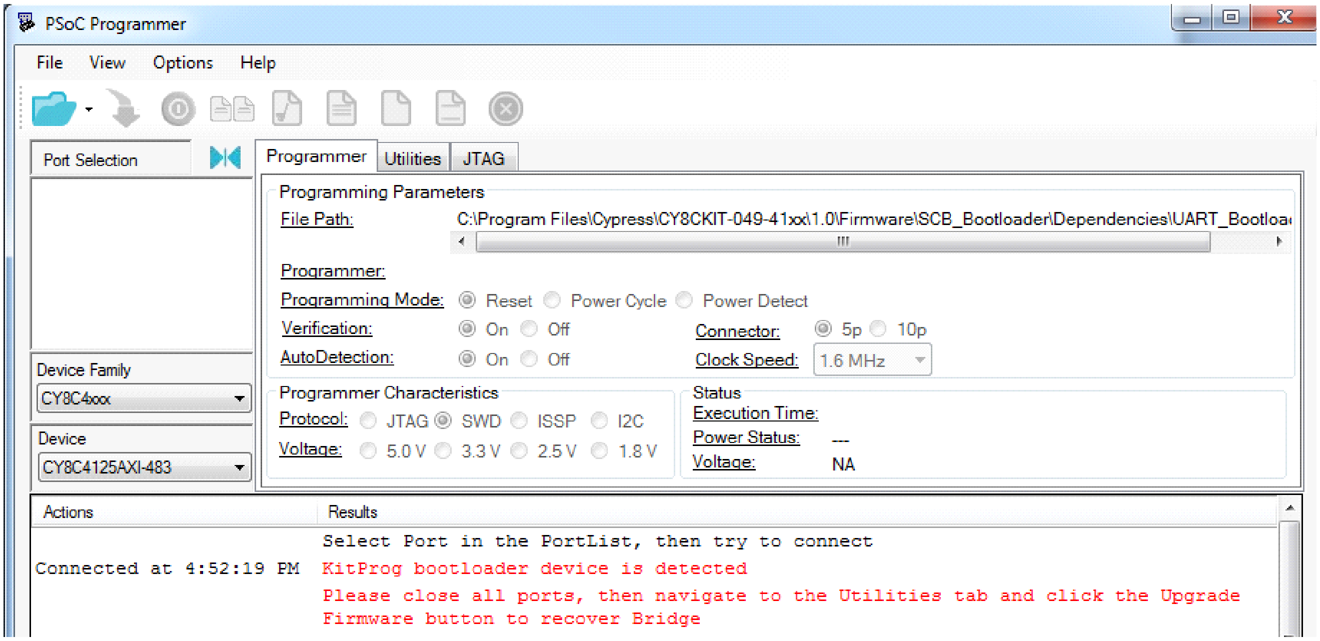
- Restore PSoC 5LP Factory Program Using PSoC Programmer
- Restore PSoC 5LP Factory Program Using Bootloader Host Tool

Note: This method cannot be used to recover the KitProg if the PSoC 5LP was reprogrammed using a MiniProg3. Jump to section Restore PSoC 5LP using MiniProg3 if you want to recover the KitProg functionality using a MiniProg3.

8.2.1.1 Restore PSoC 5LP Factory Program Using PSoC Programmer

1. Launch PSoC Programmer from **Start** > **Cypress** > **PSoC Programmer <version>** > **PSoC Programmer <version>**.
2. Configure the PSoC 4 M-Series Pioneer Kit in bootloader mode. To do this, while pressing the reset button (SW1 for pioneer kits and SW3 for prototyping kits), connect the PSoC 4 M-Series Pioneer Kit to the computer using the included USB cable (USB Standard-A to Mini-B). This puts the PSoC 5LP into bootloader mode, which is indicated by the blinking green status LED.
3. The following message appears in the PSoC Programmer **Results** window (see Figure 8-1): “KitProg Bootloader device is detected”.

Figure 8-1. PSoC Programmer Results Window



File Path: C:\Program Files\Cypress\CY8CKIT-049-41xx\1.0\Firmware\SCB_Bootloader\Dependencies\UART_Bootloa

Programmer:

Programming Mode: ● Reset ○ Power Cycle ○ Power Detect

Verification: ● On ○ Off

AutoDetection: ● On ○ Off

Connector: ● 5p ○ 10p

Clock Speed: 1.6 MHz

Protocol: ○ JTAG ● SWD ○ ISSP ○ I2C

Voltage: ○ 5.0 V ○ 3.3 V ○ 2.5 V ○ 1.8 V

Execution Time:

Power Status: ---

Voltage: NA

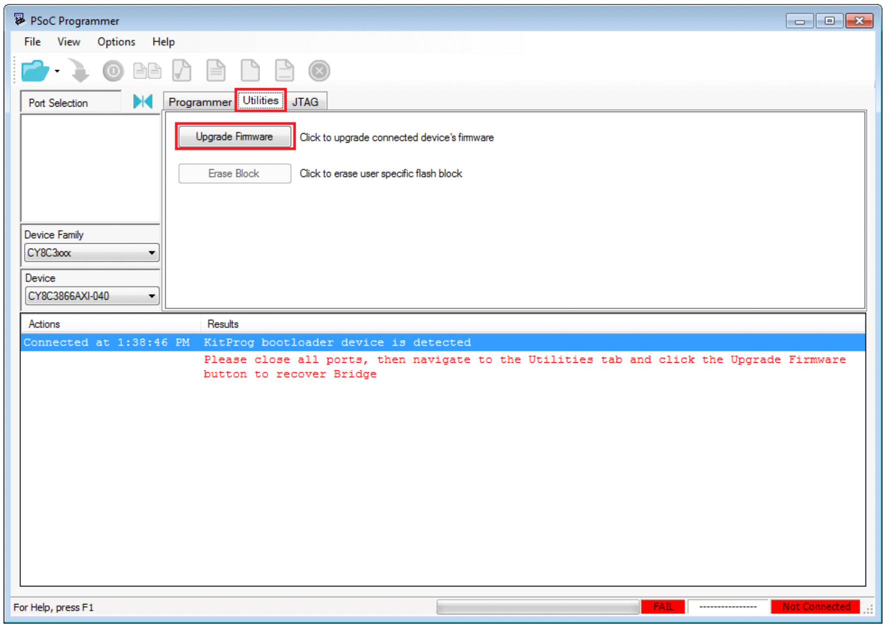
Device Family: CY8C4xxx

Device: CY8C4125AXI-483

```
Actions                    Results
                           Select Port in the PortList, then try to connect
Connected at 4:52:19 PM    KitProg bootloader device is detected
                           Please close all ports, then navigate to the Utilities tab and click the Upgrade
                           Firmware button to recover Bridge
```

4. Switch to the **Utilities** tab in PSoC Programmer and click the **Upgrade Firmware** button, as shown in Figure 8-2. Unplug all other PSoC programmers (such as MiniProg3 and DVKProg) from the PC prior to clicking the **Upgrade Firmware** button.

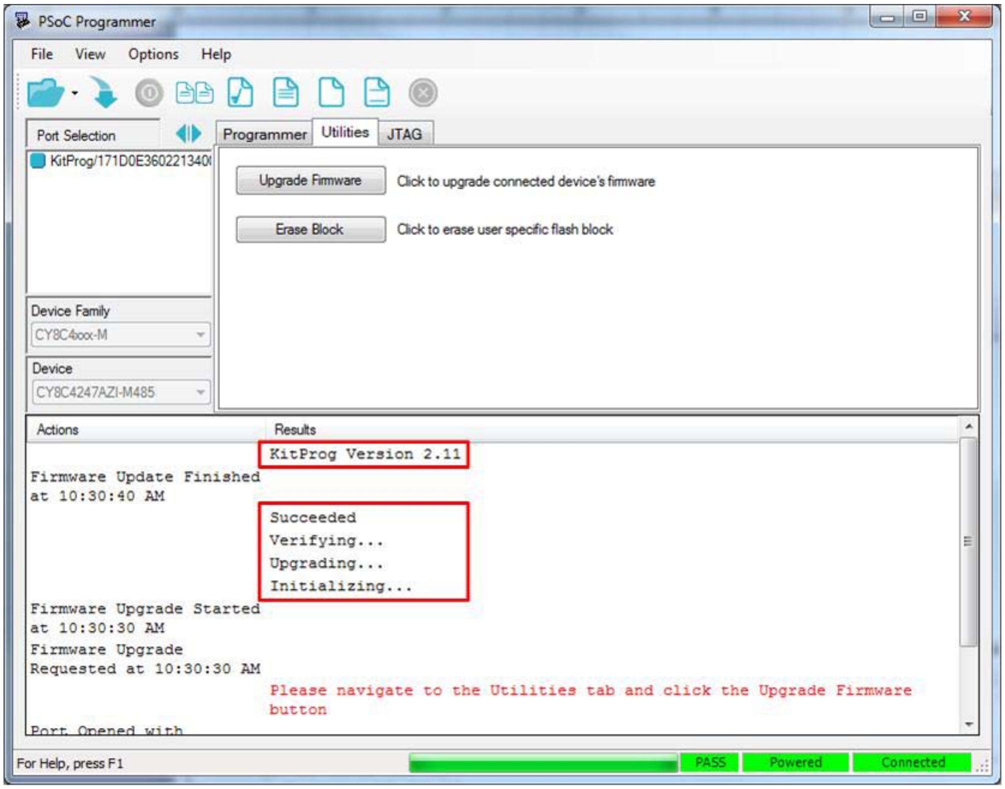
Figure 8-2. Upgrade Firmware



```
Connected at 1:38:46 PM  KitProg bootloader device is detected
                         Please close all ports, then navigate to the Utilities tab and click the Upgrade Firmware
                         button to recover Bridge
```

5. After programming is completed, the message “Firmware Update Finished at <time>” appears, and PASS message is indicated on the status bar, as shown in Figure 8-3.

Figure 8-3. Firmware Update Completed



```
                           KitProg Version 2.11
Firmware Update Finished
at 10:30:40 AM
                           Succeeded
                           Verifying...
                           Upgrading...
                           Initializing...
Firmware Upgrade Started
at 10:30:30 AM
Firmware Upgrade
Requested at 10:30:30 AM
                           Please navigate to the Utilities tab and click the Upgrade Firmware
                           button
Port Opened with
```

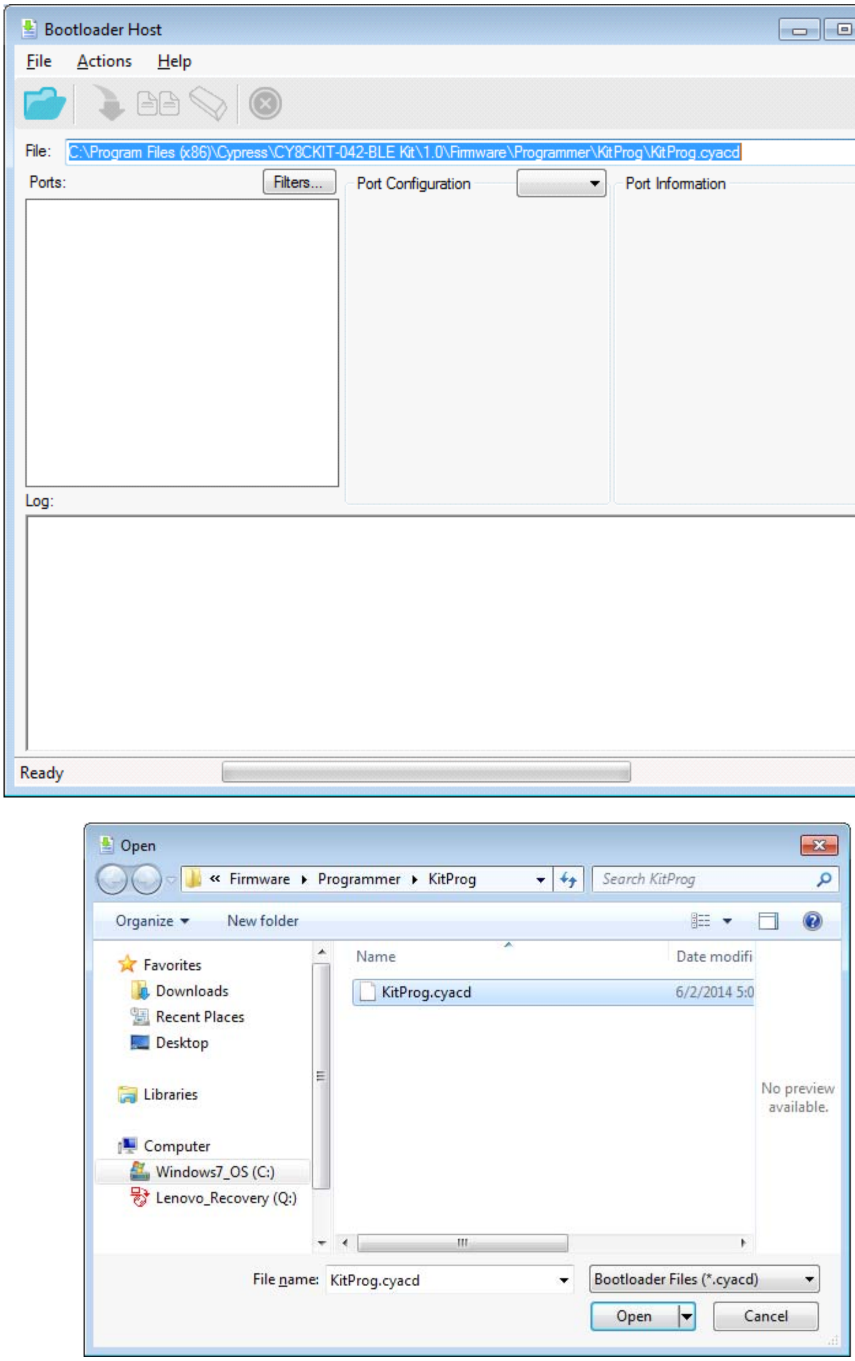
6. The factory program is now successfully restored on the PSoC 5LP. It can be used as the programmer/debugger for the PSoC 4200M device.

8.2.1.2 *Restore PSoC 5LP Factory Program Using Bootloader Host Tool*

1. Launch the Bootloader Host tool from **Start** > **Cypress** > **PSoC Creator <version>** > **Bootloader Host**.
2. Using the **File** > **Open** menu, load the *KitProg.cyacd* file, which is installed with the kit software, as shown in Figure 8-4. The default location for this file is:

```
<Install_Directory>\<Kit_Name>\<version>\Firmware\Programmer\KitProg\KitProg.cyacd
```

Figure 8-4. Load KitProg .cyacd File



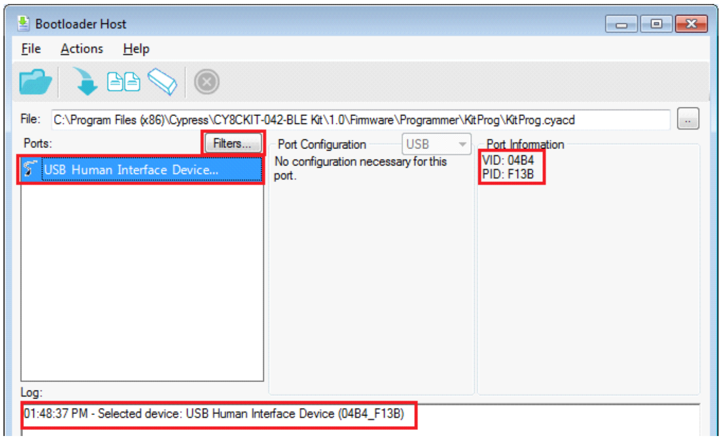
File: C:\Program Files (x86)\Cypress\CY8CKIT-042-BLE Kit\1.0\Firmware\Programmer\KitProg\KitProg.cyacd

File name: KitProg.cyacd

Bootloader Files (*.cyacd)

3. Configure the PSoC 4 M-Series Pioneer Kit in bootloader mode. To do this, while holding down the reset button (SW1 for pioneer kits and SW3 for prototyping kits), connect the PSoC 4 M-Series Pioneer Kit to the PC using the included USB cable (USB Standard-A to Mini-B). This puts the PSoC 5LP into bootloader mode, which is indicated by the blinking green status LED.
4. In the Bootloader Host tool, set the filters for the USB devices with **VID: 04B4** and **PID: F13B**. The **USB Human Interface Device** port appears in the **Ports** list. Click the port to select it, as shown in Figure 8-5.

Figure 8-5. Select USB Human Interface Device



File: C:\Program Files (x86)\Cypress\CY8CKIT-042-BLE Kit\1.0\Firmware\Programmer\KitProg\KitProg.cyacd

Ports: USB Human Interface Device...

Port Configuration: USB

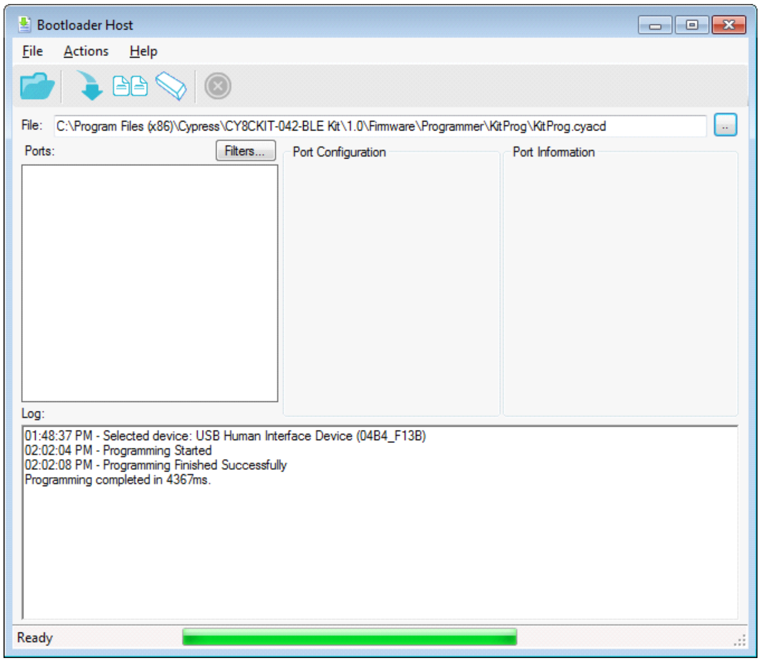
No configuration necessary for this port.

Port Information:
VID: 04B4
PID: F13B

```
01:48:37 PM - Selected device: USB Human Interface Device (04B4_F13B)
```

5. Click the **Program** button (or choose **Actions** > **Program**) to restore the factory program by bootloading it onto the PSoC 5LP.
6. After programming is completed, the message “Programming Finished Successfully” appears, as shown in Figure 8-6.

Figure 8-6. Programming Finished Successfully



File: C:\Program Files (x86)\Cypress\CY8CKIT-042-BLE Kit\1.0\Firmware\Programmer\KitProg\KitProg.cyacd

```
01:48:37 PM - Selected device: USB Human Interface Device (04B4_F13B)
02:02:04 PM - Programming Started
02:02:08 PM - Programming Finished Successfully
Programming completed in 4367ms.
```

7. The factory KitProg program is now successfully restored on the PSoC 5LP.

8.2.2 Restore PSoC 5LP using MiniProg3

This section explains the method to reprogram the PSoC 5LP using a MiniProg3 to recover the KitProg functionality. This method must be used to recover the KitProg if the PSoC 5LP was completely reprogrammed.

Note: Programming of KitProg through MiniProg3 is not possible in prototyping kits (CY8CKIT-043 and CY8CKIT-059).

1. Launch PSoC Programmer from **Start** > **Cypress** > **PSoC Programmer <version>** > **PSoC Programmer <version>**.
2. Connect the MiniProg3 to the PC. Connect the 10-pin connector of MiniProg3 to the onboard PSoC 5LP programming header.

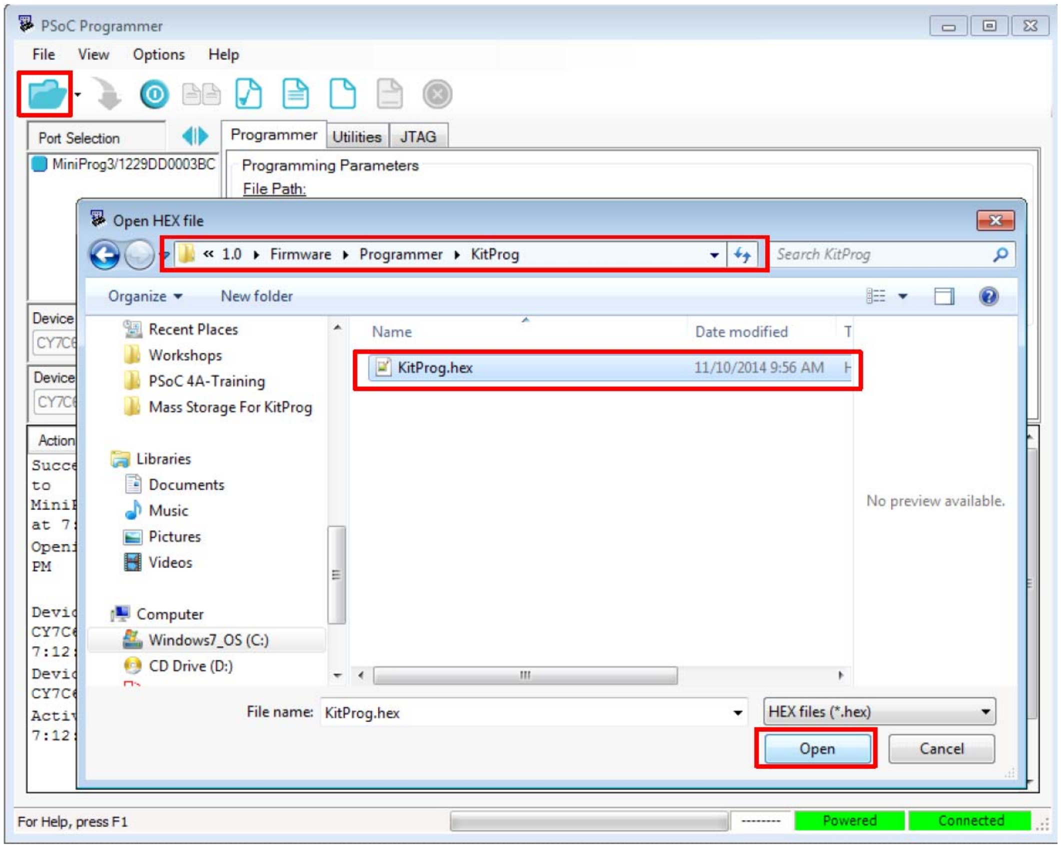
Note: This header is not populated by default. You will need to populate this header in order to connect a MiniProg3.

3. Select the **MiniProg3** from the **Port Selection** list in the PSoC Programmer on your PC.
4. Using the **File** > **Open** menu or using the **File Load** icon, load the *KitProg.hex* file, which is installed with the kit software, as shown in Figure 8-7. The default location for this file is:

```
<Install
Path>\<Name_of_the_Kit>\<version>\Firmware\Programmer\KitProg\KitProg.hex
```

5. Select the **Power Cycle** option for Programming Mode, **10p** (10 pin) option for Connector, and the **SWD** option for Protocol.
6. Click the **Program** button or **File** > **Program** to program the PSoC 5LP device.
7. After programming is complete, the “Program Finished at <time>” message is displayed.

Figure 8-7. Select the *KitProg.hex* File to Program the PSoC 5LP



8.3 Leakage Current Reduction on Target MCU through SWD Lines

A current bounce in the range of several hundred microamps may be observed on a target in low power modes, because the SWD clock and data lines are left floating on the kit. Because of that the target on the kit may wake up unexpectedly. If it is necessary to prevent this situation in your design, you can use one of the following solutions:

1. Re-purpose the SWD clock and data lines to GPIO.
2. Use resistive pull up on these lines.

A. Appendix

A.1 Pin Assignment Tables

A.1.1 PSoC 5LP GPIO Header (J8) for CY8CKIT-042-BLE, CY8CKIT-044 and CY8CKIT-046

J8					
Pin	**PSoC 5LP Signal**	**PSoC 5LP Description**	**Pin**	**PSoC 5LP Signal**	**PSoC 5LP Description**
J8_01	PSoC 5LP_VDD	VDD	J8_02	P1[2]	Digital I/O
J8_03	P0[0]	Delta Sigma ADC + Input	J8_04	P0[1]	Delta Sigma ADC - Input
J8_05	P3[4]	SAR - Input	J8_06	P3[5]	SAR + Input
J8_07	P3[6]	Buffered VDAC	J8_08	P3[7]	Buffered VDAC
J8_09	P12[6]	UART RX	J8_10	P12[7]	UART TX
J8_11	P12[1]	SPI MISO/I2C SDA	J8_12	P3[0]	IDAC Output
J8_13	P12[0]	SPI SCLK/I2C SCL	J8_14	P12[5]	SPI MOSI
J8_15	P2[5]	SPI SSEL	J8_16	GND	GND

A.1.2 PSoC 5LP GPIO Header (J8) for CY8CKIT-042 and CY8CKIT-040

J8					
Pin	**PSoC 5LP Signal**	**PSoC 5LP Description**	**Pin**	**PSoC 5LP Signal**	**PSoC 5LP Description**
J8_01	PSoC 5LP_VDD	VDD	J8_02	P1[2]	Digital I/O
J8_03	P0[0]	Delta Sigma ADC + Input	J8_04	P0[1]	Delta Sigma ADC - Input
J8_05	P3[4]	SAR - Input	J8_06	P3[5]	SAR + Input
J8_07	P3[6]	Buffered VDAC	J8_08	P3[7]	Buffered VDAC
J8_09	P12[6]	UART RX	J8_10	P12[7]	UART TX
J8_11	GND	SPI MISO/I2C SDA	J8_12	P3[0]	IDAC Output

A.1.3 PSoC 5LP GPIO Header (J8 and J9) for CY8CKIT-059 and CY8CKIT-043

J9			J8		
Pin	PSoC 5LP Signal	PSoC 5LP Description	Pin	PSoC 5LP Signal	PSoC 5LP Description
J9_01	VBUS	Power/VDD	J8_01	GND	Ground
J9_02	GND	Ground	J8_02	P3[0]	GPIO
J9_03	P12[5]	GPIO	J8_03	P3[4]	GPIO
J9_04	P12[0]	GPIO/I2C_SCL	J8_04	P3[5]	GPIO
J9_05	P12[1]	GPIO/I2C_SDA	J8_05	P3[6]	GPIO
J9_06	P12[6]	GPIO/UART_RX	J8_06	P0[0]	GPIO
J9_07	P12[7]	GPIO/UART_TX	J8_07	P0[1]	GPIO

Revision History

Document Revision History

Document Title: KitProg User Guide Document Number: 001-96359			
Revision	**Issue Date**	**Origin of Change**	**Description of Change**
**	02/25/2015	RNJT	Initial version of KitProg User Guide.
*A	03/27/2015	RNJT	Updated the kit name to PSoC 4 M-Series Pioneer Kit. Updated link to PSoC 4200M webpage.
*B	04/02/2015	RNJT	Updated the incorrect links. Updated Figure 3-8, Figure 3-9, Figure 7-8 and Figure 8-3.
*C	05/29/2015	RNJT	Updated Figure 1-1. Added a Note in Introduction, on page 4. Updated Table 2-1. Updated the KitProg description in Table 3-1. Added the chapter KitProg Mass Storage Programmer, on page 14. Updated Step 3 in Enter or Exit the Mass Storage Programmer Mode, on page 14. Updated Steps 2 and 3 in Programming Using the Mass Storage Programmer, on page 14. Updated Frequently Asked Questions on KitProg Mass Storage Programmer.
*D	06/12/2015	RNJT	Added Figure 3-2. Added Table A.1.3 in A.1 Pin Assignment Tables.
*E	06/25/2015	MSUR	Added a note in Building a Normal Project for PSoC 5LP. Updated Step 2 in Restore PSoC 5LP Factory Program Using PSoC Programmer, on page 45. Updated Step 3 in Restore PSoC 5LP Factory Program Using Bootloader Host Tool, on page 47. Updated the following in Restore PSoC 5LP using MiniProg3, on page 49: Updated title, added a Note.
*F	09/14/2015	MSUR	Removed the Note in Chapter Introduction, on page 4. Updated Table 2-1, Table 2-2 and Figure 7-8. Updated 4.3 Frequently Asked Questions on KitProg Mass Storage Programmer. Updated 7.2 Building a Normal Project for PSoC 5LP. Updated Step 4 in Chapter KitProg USB-UART Bridge, on page 18. Updated the Note and Step 4 in Chapter KitProg USB-I2C Bridge, on page 27. Updates Notes in 7.1 Building a Bootloadable Project for PSoC 5LP. Added a Note in 8.1 KitProg Status LED Indication. Updated the Note in 8.2.1 PSoC 5LP is Programmed with a Bootloadable Application.

Document Title: KitProg User Guide Document Number: 001-96359			
Revision	**Issue Date**	**Origin of Change**	**Description of Change**
*G	09/21/2015	SRDS	Updated the following sections: KitProg Programmer/Debugger, on page 7 KitProg USB-I2C Bridge, on page 27 Troubleshooting the KitProg, on page 44 PSoC 5LP GPIO Header (J8) for CY8CKIT-042-BLE, CY8CKIT-044 and CY8CKIT-046, on page 51. Updated Table 2-1.
*H	05/17/2017	AESATMP8	Updated logo and Copyright.
*I	04/24/2018	VKVK	Added a note in 3.4 KitProg Driver Installation. Updated KitProg Mass Storage Programmer, on page 14. Added 8.3 Leakage Current Reduction on Target MCU through SWD Lines. Updated Figure 5-1, Figure 6-1, Figure 7-2, and Figure 7-13. Updated Copyright information.