



KitProg User Guide

Doc. No. 001-96359 Rev. *B

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intl): +1.408.943.2600
www.cypress.com

Copyrights

© Cypress Semiconductor Corporation, 2015. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Source Code

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

Trademarks

PSoC Creator is a trademark and CapSense and PSoC are registered trademarks of Cypress Semiconductor Corporation. All other trademarks are the property of their respective owners.

Purchase of I2C components from Cypress or one of its sublicensed Associated Companies conveys a license under the Philips I2C Patent Rights to use these components in an I2C system, provided that the system conforms to the I2C Standard Specification as defined by Philips. As from October 1st, 2006 Philips Semiconductors has a new trade name - NXP Semiconductors.

Flash Code Protection

Cypress products meet the specifications contained in their particular Cypress Datasheets. Cypress believes that its family of products is one of the most secure families of its kind on the market today, regardless of how they are used. There may be methods, unknown to Cypress that can breach the code protection features. Any of these methods, to our knowledge, would be dishonest and possibly illegal. Neither Cypress nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable." Cypress is willing to work with the customer who is concerned about the integrity of their code. Code protection is constantly evolving. We at Cypress are committed to continuously improving the code protection features of our products.

Contents



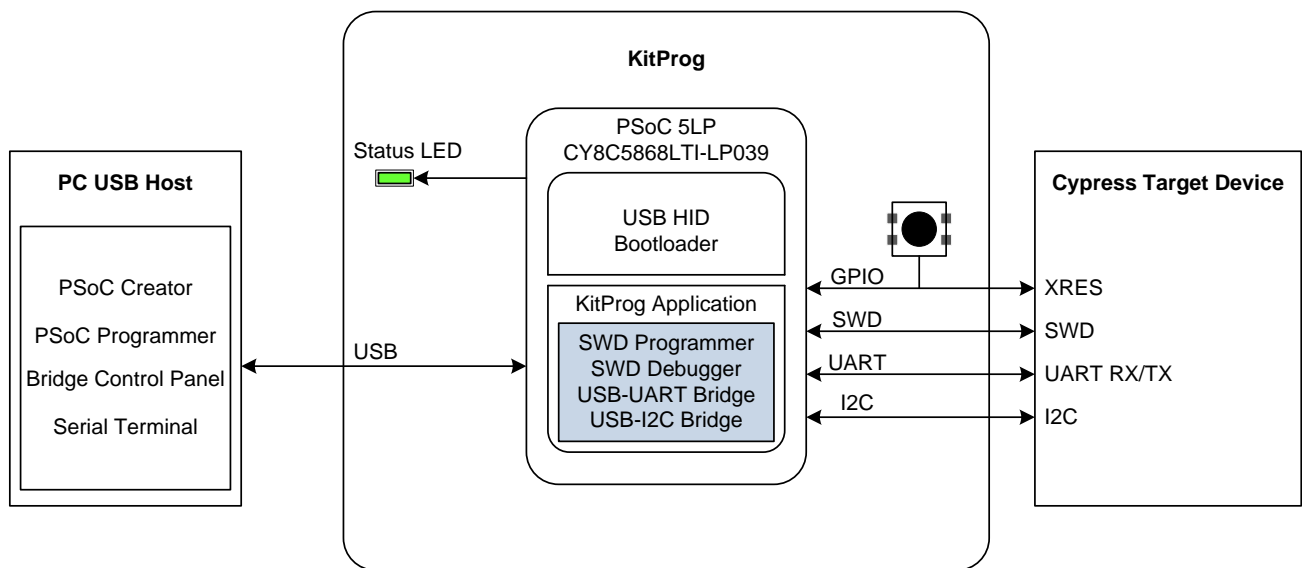
KitProg User Guide	1
Contents.....	3
1. Introduction.....	4
2. KitProg Ecosystem.....	5
3. Using the KitProg Programmer/Debugger	7
3.1 Programming Using PSoC Creator	7
3.2 Debugging Using PSoC Creator.....	9
3.3 Programming Using PSoC Programmer.....	10
3.4 KitProg Driver Installation	10
3.5 Updating the KitProg Firmware	11
4. Using the KitProg USB-UART Bridge	13
5. Using the KitProg USB-I2C Bridge.....	21
6. Developing Applications for PSoC 5LP	28
6.1 Building a Bootloadable Project for PSoC 5LP	28
6.2 Building a Normal Project for PSoC 5LP	35
7. Troubleshooting the KitProg	37
7.1 KitProg Status LED Indication	37
7.2 PSoC 5LP Factory Program Restore Instructions	38
7.2.1 PSoC 5LP is Programmed with a Bootloadable Application	38
7.2.2 PSoC 5LP is Programmed with a Standard Application (Restore using MiniProg3)	42
Appendix.....	43
A.1 Pin Assignment Tables.....	43
Revision History	44
Document Revision History.....	44

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 gives you 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

Development Kits	Target Device
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 PSoC 4 BLE Pioneer Kit	PSoC 4200 BLE
CY8CKIT-044 PSoC 4 M-Series Pioneer Kit	PSoC 4200M

Note: 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 [Using the KitProg USB-I2C Bridge](#) chapter, all 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/go/psoccreator
USB-I2C Bridge	Bridge Control Panel (BCP)	Installed along with PSoC Programmer
USB-UART Bridge	HyperTerminal/PuTTY	HyperTerminal is available as part of Microsoft Windows installation in Windows XP. PuTTY is 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

Function	Supported Speeds	Units
Programmer/Debugger	1.6	MHz
USB-I2C Bridge	50, 100, 400, 1000	kHz
USB-UART Bridge	1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200	Bauds

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 - www.cypress.com/go/AN79953
- Getting Started with PSoC 4 BLE - www.cypress.com/go/AN91267
- Getting Started with PSoC 5LP - www.cypress.com/go/AN77759
- Getting Started with CapSense – www.cypress.com/go/AN64846

3. Using the KitProg Programmer/Debugger

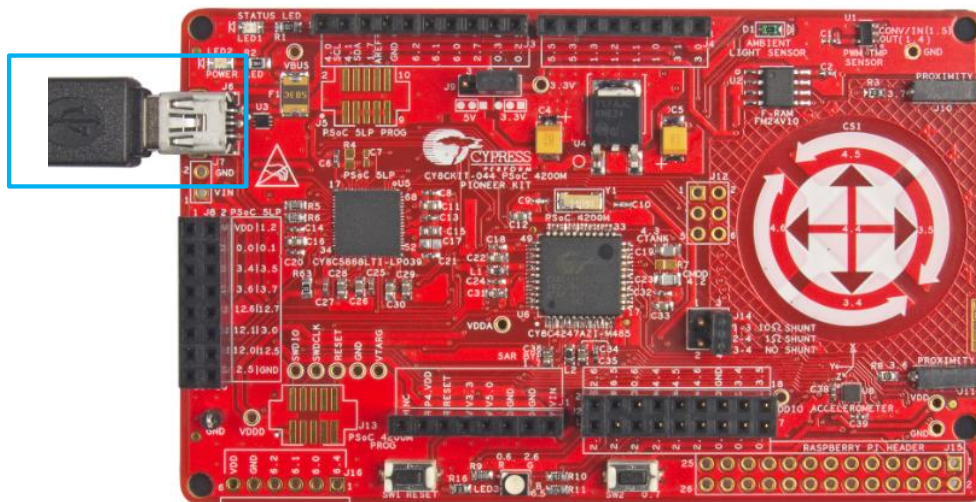


This section explains the method to use the KitProg programmer/debugger integrated onto the PSoC development kits. The KitProg supports the development kits listed in [Table 2-1](#). This section uses the PSoC 4 M-Series Pioneer Kit as an example.

3.1 Programming Using PSoC Creator

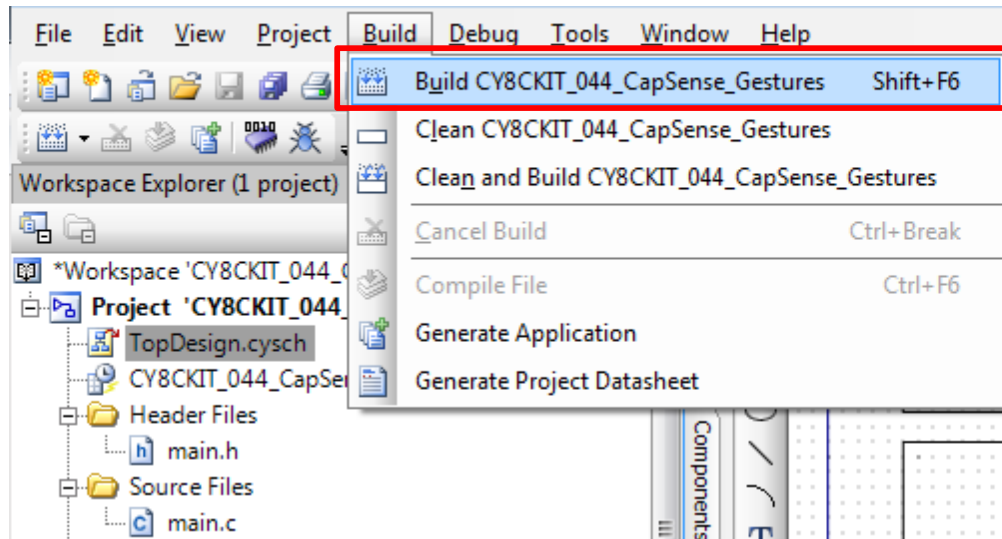
1. Connect the USB cable into the USB connector, J6, as shown in [Figure 3-1](#). If you are connecting the kit to your PC for the first time, it enumerates as a USB composite device and installs the required driver software. See the [KitProg Driver Installation](#) section for more information.

Figure 3-1. Connect USB Cable to J6



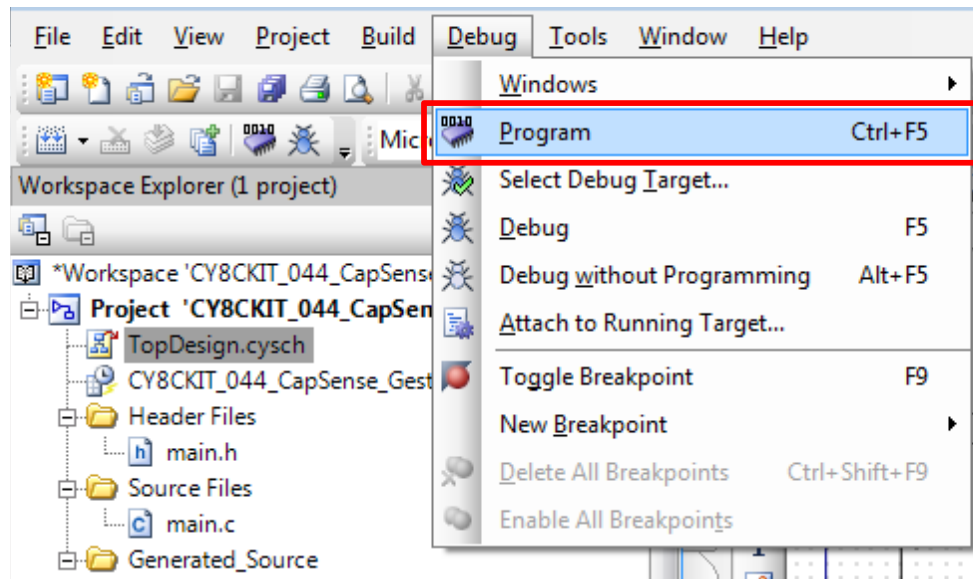
2. Launch PSoC Creator from the **Start > All Programs > Cypress > PSoC Creator <version> > PSoC Creator <version>**.
3. Select **File > Open > Project/Workspace** in PSoC Creator and browse to the desired project.
4. Select **Build > Build Project** or press **[Shift] [F6]** to build the project, as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

Figure 3-2. 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 3-3.

Figure 3-3. Programming Device from PSoC Creator

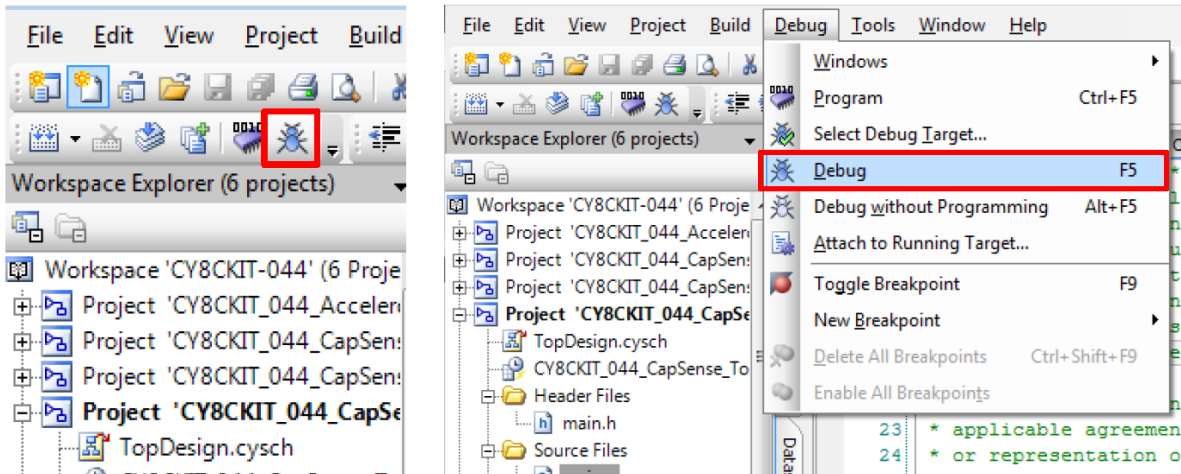


3.2 Debugging Using PSoC Creator

To debug the project using PSoC Creator, follow steps 1 to 4 from [Programming Using PSoC Creator](#). Then, follow these steps:

1. Click the Debug icon or press **[F5]**, as shown in [Figure 3-4](#). Alternatively, you can select **Debug > Debug**. This programs the device and starts the debugger.

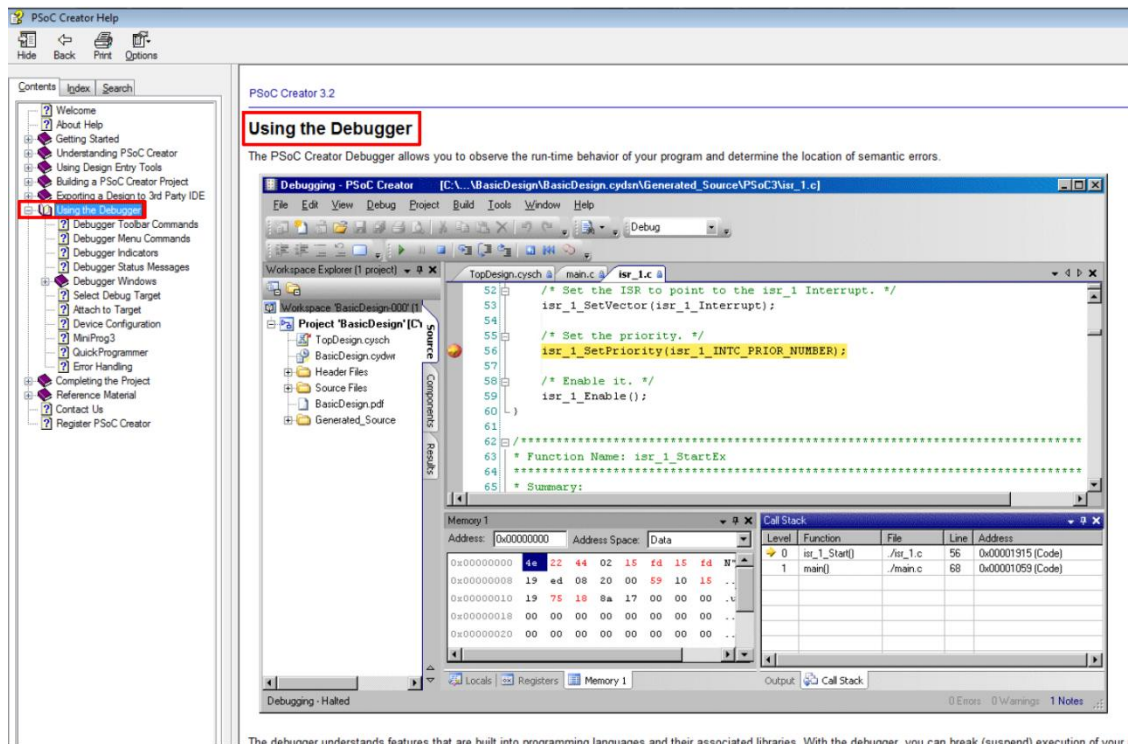
Figure 3-4. Debug Option in PSoC Creator



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 3-5](#).

Figure 3-5. Using the PSoC Creator Debugger



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 a 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 device selected, 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.

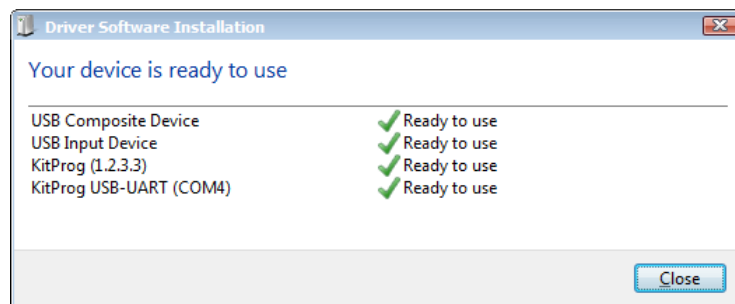
3.4 KitProg Driver Installation

The CY8CKIT-044 powers 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 they should be installed properly for its correct operation.

Table 3-1 Enumerated Interfaces

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

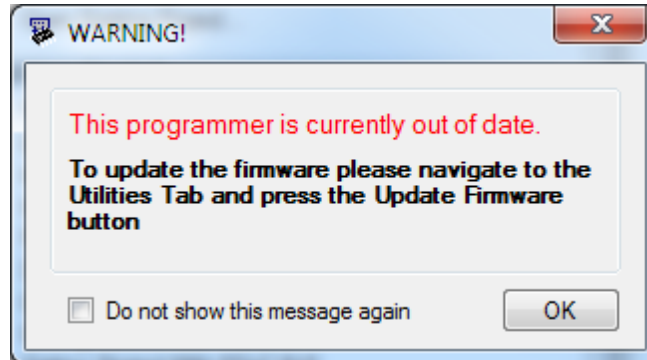
Figure 3-6 KitProg Driver Installation (appearance may differ depending on the Windows version)



3.5 Updating the KitProg Firmware

The KitProg firmware normally 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 3-7](#).

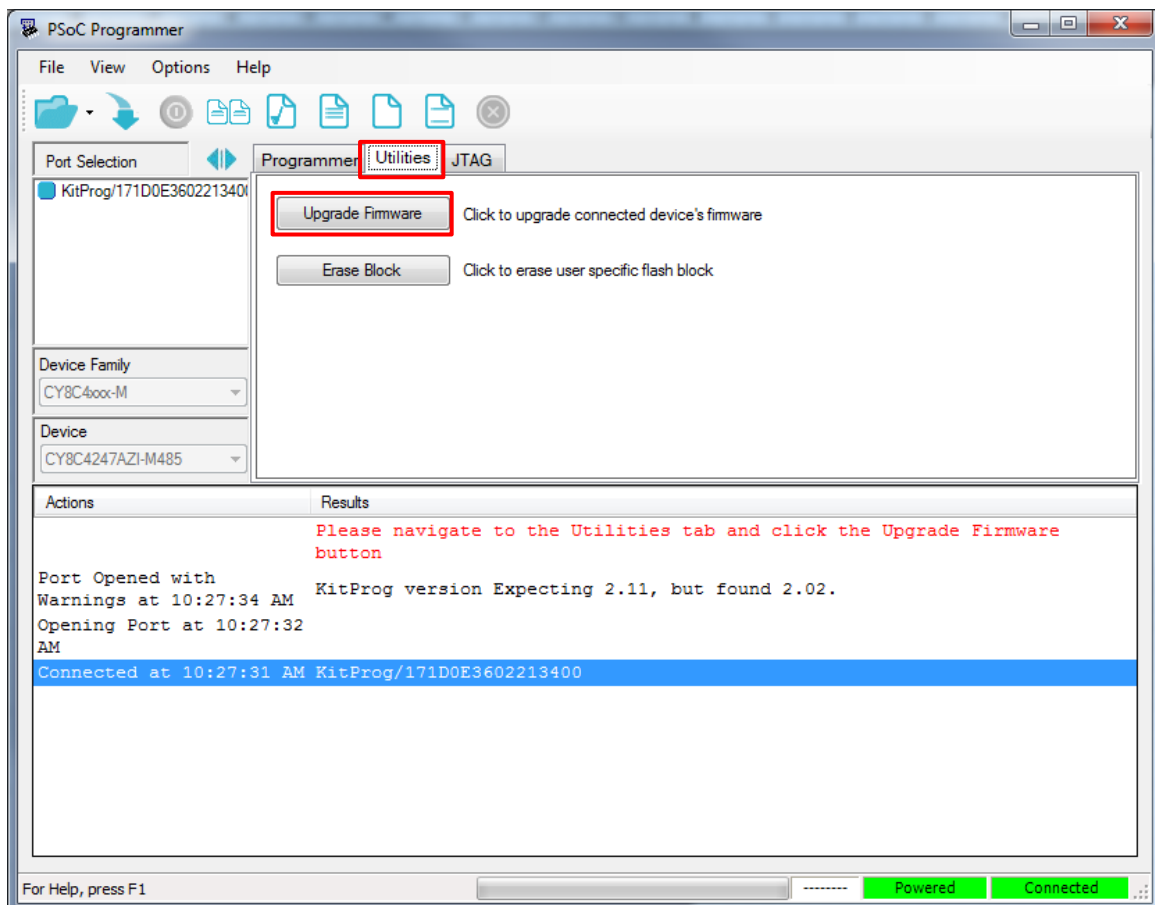
Figure 3-7. KitProg Firmware Update Warning



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 3-8](#).

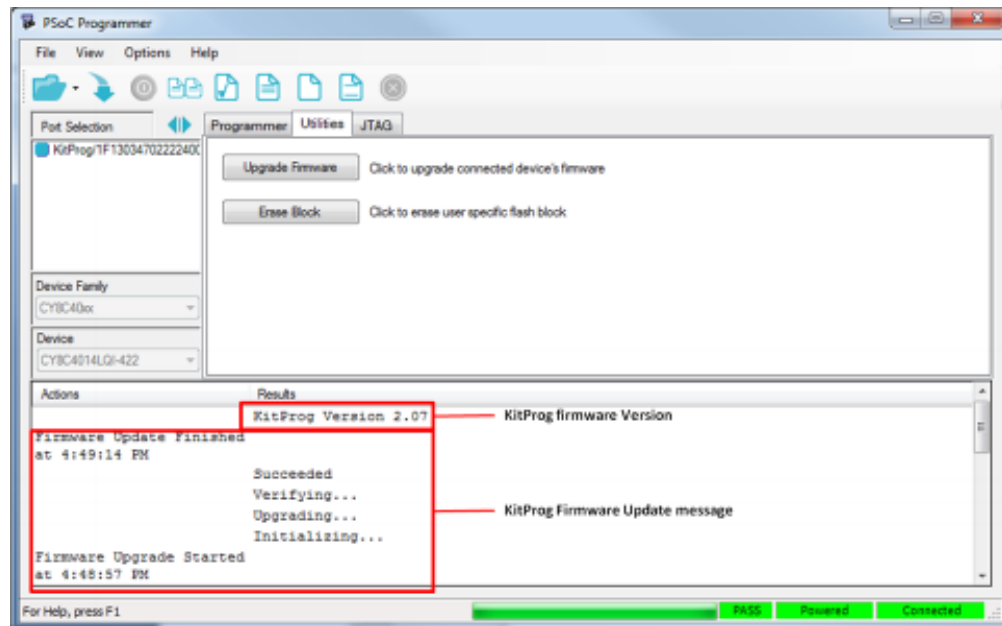
To update the KitProg, go to the **Utilities** tab on PSoC Programmer and click **Upgrade Firmware**, as shown in [Figure 3-8](#).

Figure 3-8. Upgrade Firmware in PSoC Programmer



On successful upgrade, the Actions and Results window displays the firmware update message with the KitProg version, as shown in [Figure 3-9](#).

Figure 3-9. Firmware Updated in PSoC Programmer



4. Using the KitProg USB-UART Bridge

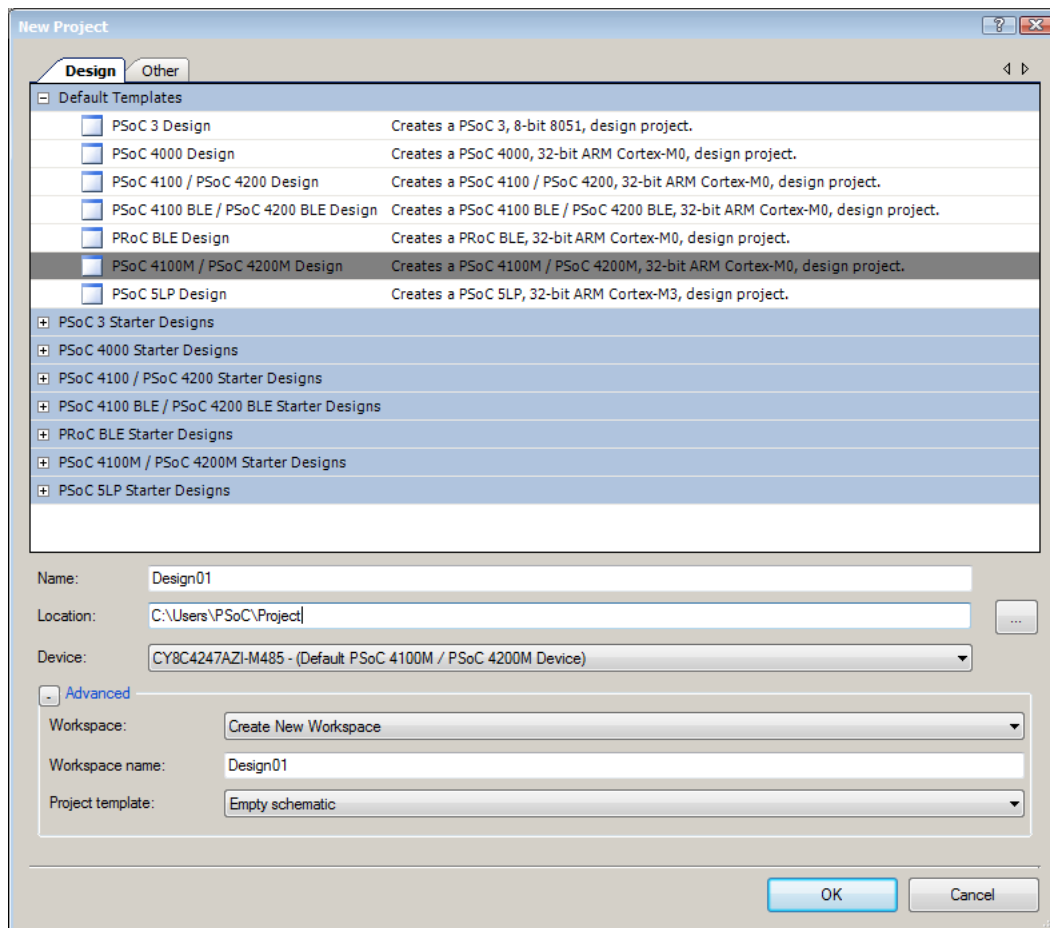


The KitProg can act as a USB-UART bridge. This feature of the KitProg is useful to send and receive data between the Cypress device on the kit and the PC. For example, in the PSoC 4 M-Series Pioneer Kit, the KitProg USB-UART can be used to print debug messages on a COM terminal software running on the 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 the Windows HyperTerminal as the COM terminal software. If you have a Windows operating system that does not have HyperTerminal, use an alternative terminal software such as PuTTY.

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

Figure 4-1. Create New Project in PSoC Creator



2. Drag and drop a UART (SCB mode) Component from the Component Catalog (Figure 4-2) 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 4-3.

Figure 4-2. UART Component in Component Catalog

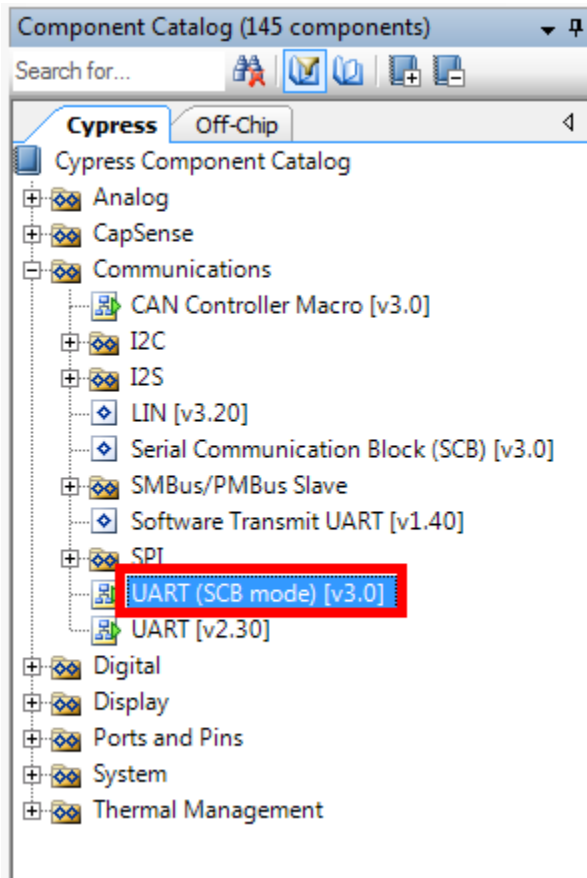
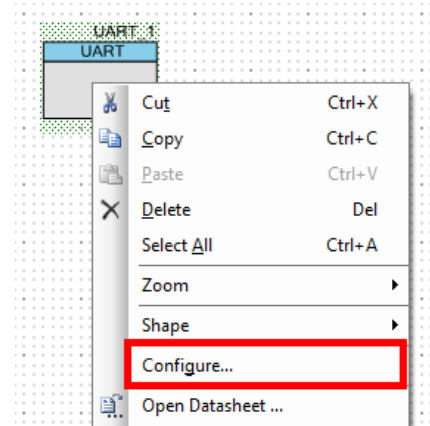


Figure 4-3. Open UART Configuration Window



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

Figure 4-4. UART Configuration Tab Window

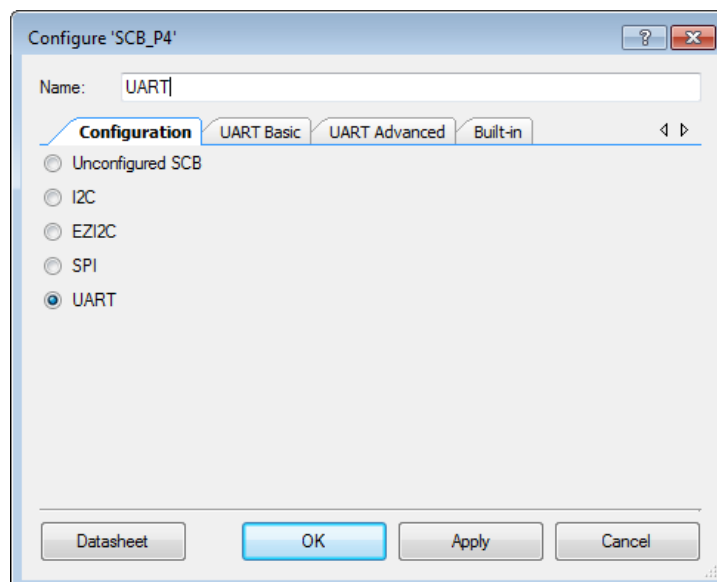


Figure 4-5. UART Basic Tab Window

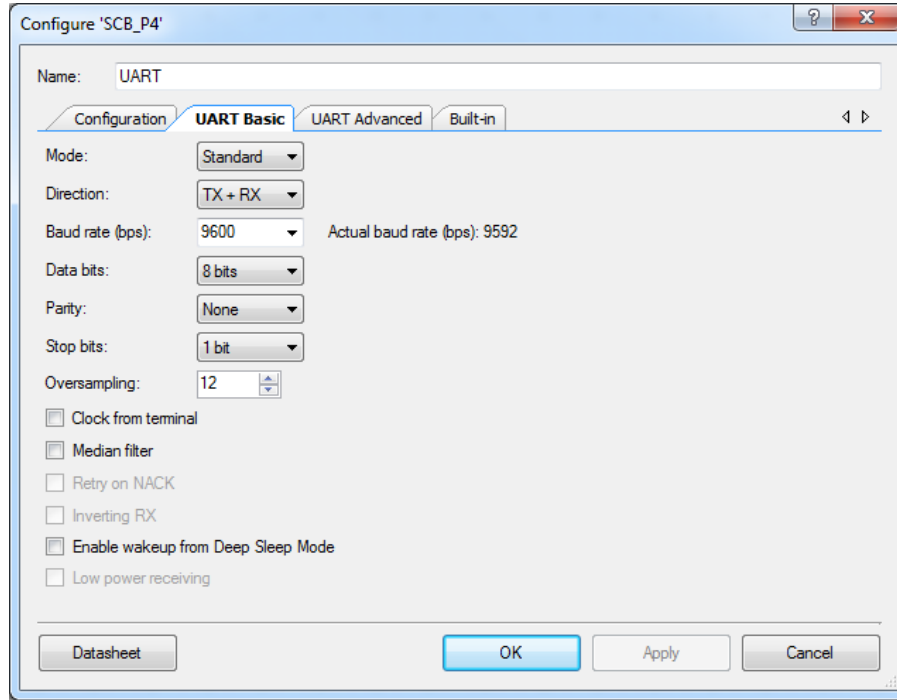
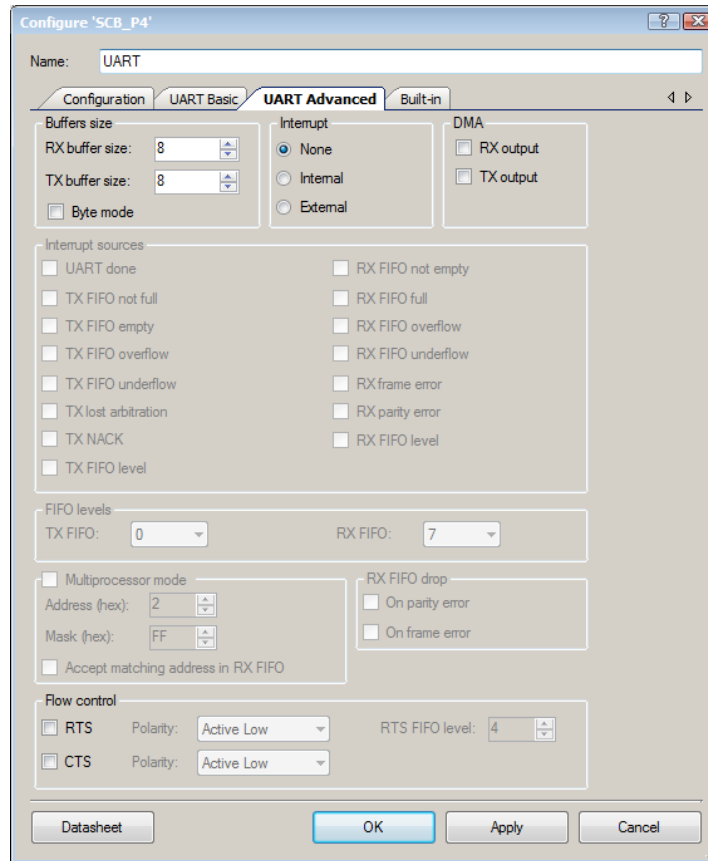


Figure 4-6. UART Advanced Tab Configuration Window



- Select P7[0] for UART RX and P7[1] for UART TX in the **Pins** tab of `<Project_Name>.cydwr`, as shown in Figure 4-7. 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.

Figure 4-7. Pin Selection

Alias	Name /	Port	Pin	Lock
\UART:rx\	P7[0]	TCPWM0:line_out, SCB3:uart_rx, SCB3:i2c scl, SCB3:spi mosi	37	<input checked="" type="checkbox"/>
\UART:tx\	P7[1]	TCPWM0:line_out_compl, SCB3:uart_tx, SCB3:i2c sda, SCB3:spi miso	38	<input checked="" type="checkbox"/>

- Place the following code in the `main.c` project 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);
        }
    }
}
```

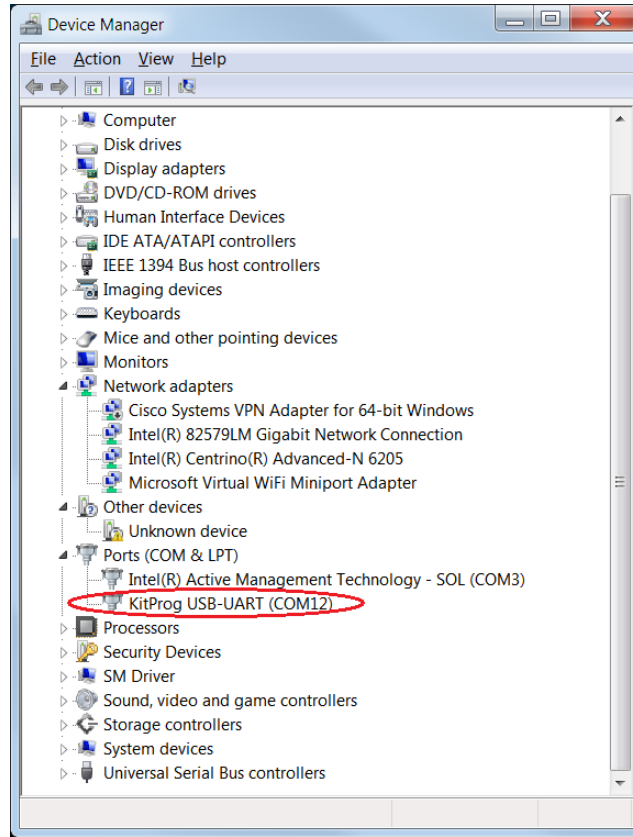
- Build the project by choosing **Build > Build [Project Name]** or pressing **[Shift] [F6]**. After the project is built without errors and warnings, program the project (by choosing **Debug > Program**) to PSoC 4200M using KitProg.

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.

- 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 4-8.

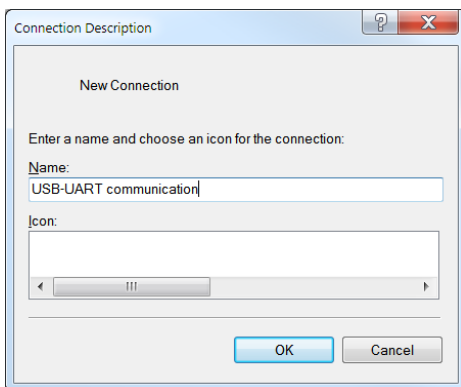
Figure 4-8. KitProg USB-UART in Device Manager



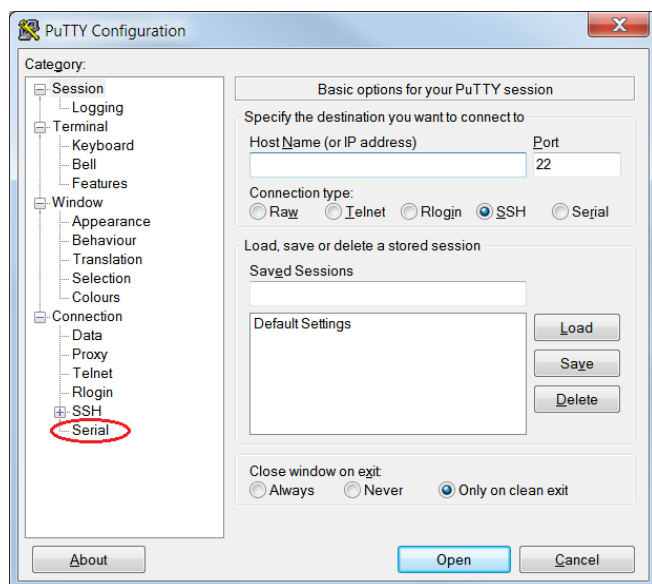
2. Open HyperTerminal, choose **File > New Connection**, enter a name for the new connection, and click **OK**, as shown in Figure 4-9. For PuTTY, double-click the PuTTY application and select **Serial** under **Category**.

Figure 4-9. Open New Connection

HyperTerminal



PuTTY

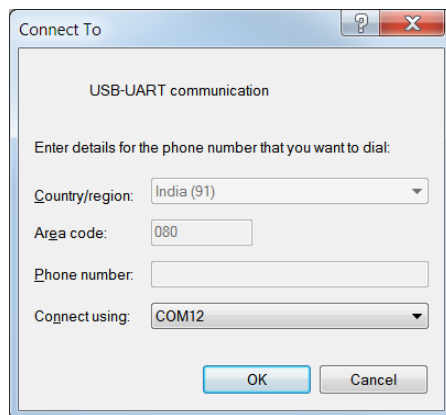


3. A new window opens, where the communication port can be selected. In HyperTerminal, select COMx (the specific communication port that is assigned to the KitProg USB-UART) in **Connect using** and click **OK**, as shown in Figure 4-10.

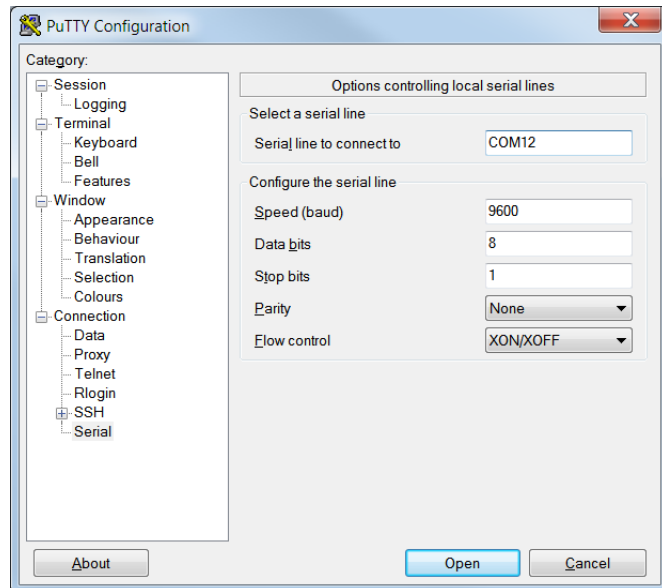
In PuTTY enter the COMx in **Serial line to connect to**. This example uses COM12.

Figure 4-10. Select Communication Port

HyperTerminal



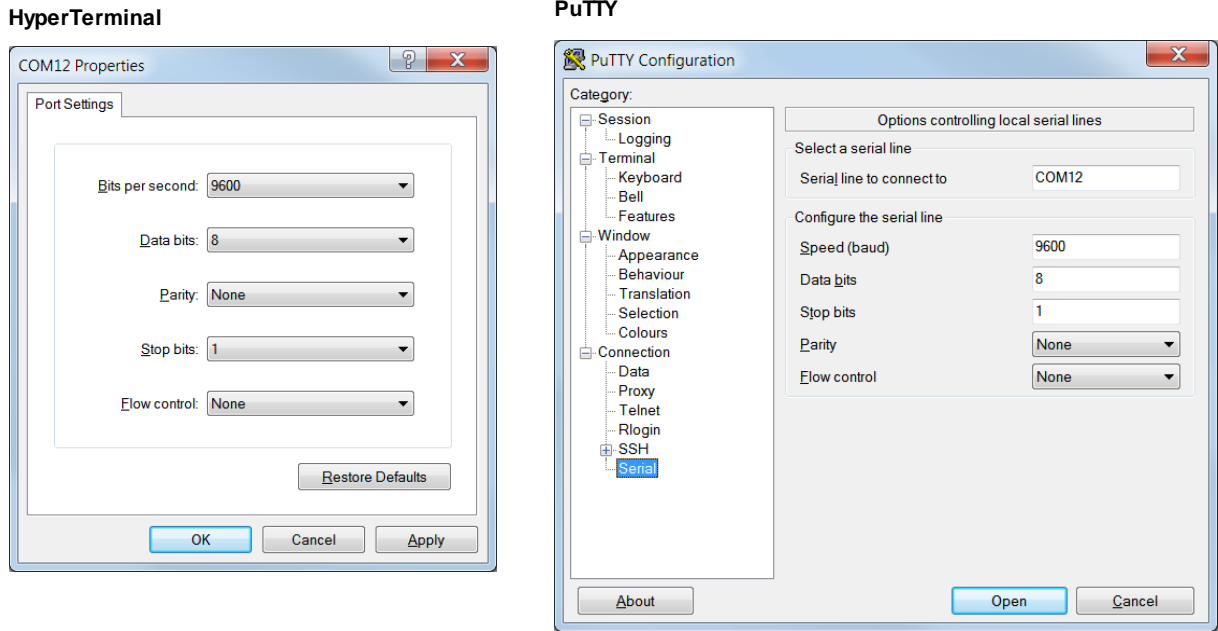
PuTTY



4. In HyperTerminal, select **Bits per second**, **Data bits**, **Parity**, **Stop bits**, and **Flow control** under **Port Settings** and click **OK**, as shown in Figure 4-11. Make sure that the settings are identical to the UART settings configured for the PSoC 4200M device.

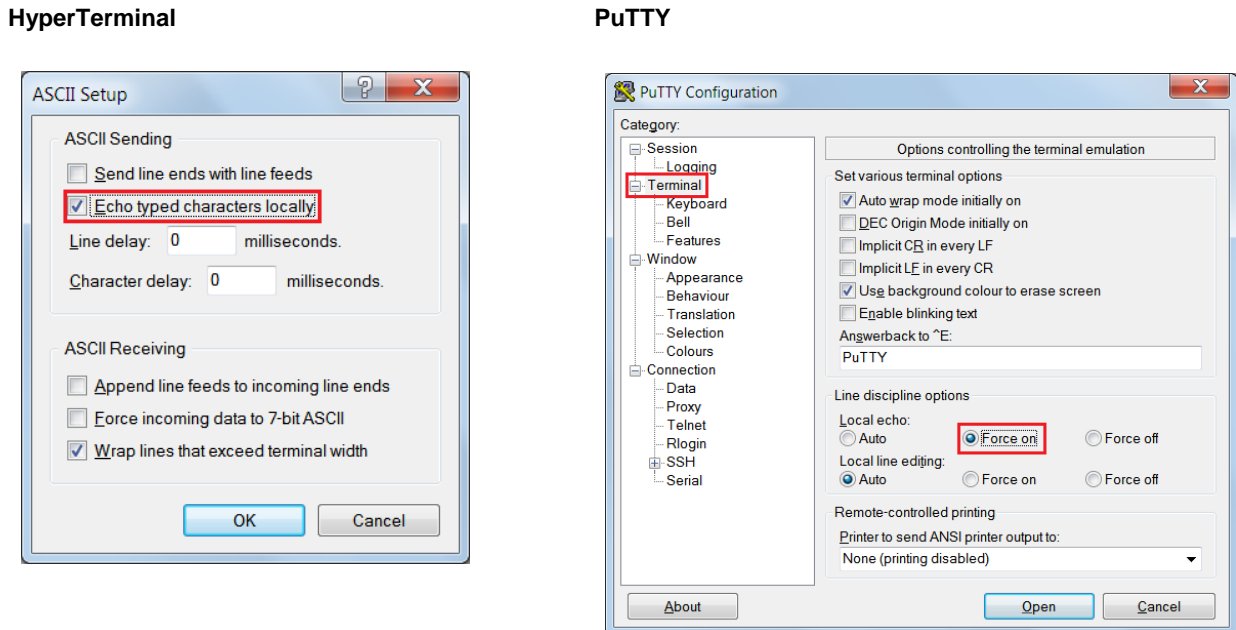
In PuTTY, select **Speed (baud)**, **Data bits**, **Stop bits**, **Parity**, and **Flow control** under **Configure the serial line**.

Figure 4-11. Configure the Communication Port



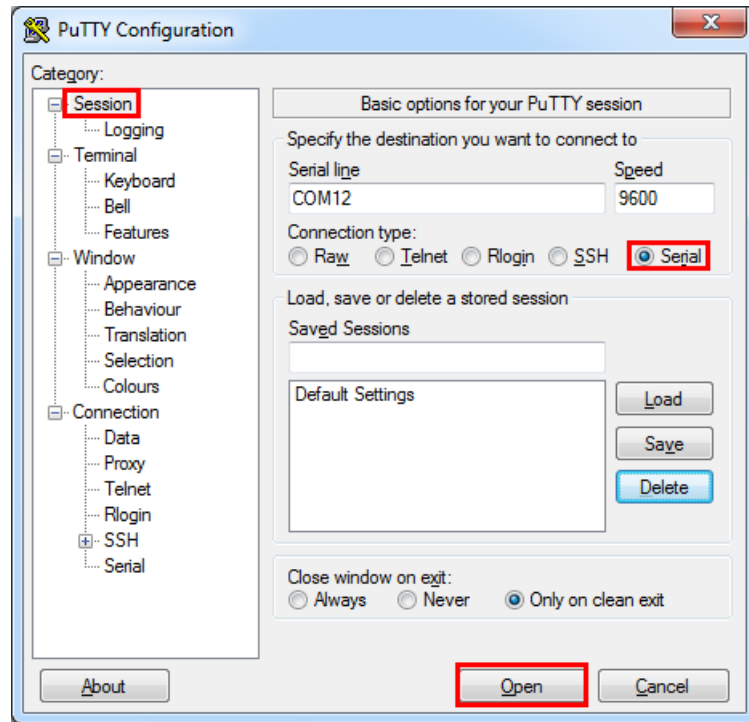
5. Enable **Echo typed characters locally** under **File > Properties > Settings > ASCII Setup** to display the typed characters on HyperTerminal, as shown in Figure 4-12. In PuTTY, select **Force on** under **Terminal > Line discipline options** to display the typed characters on PuTTY, as shown in Figure 4-12.

Figure 4-12. Enable Echo of Typed Characters in HyperTerminal and PuTTY



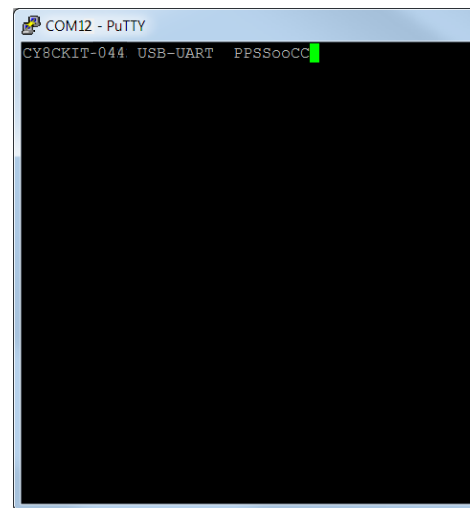
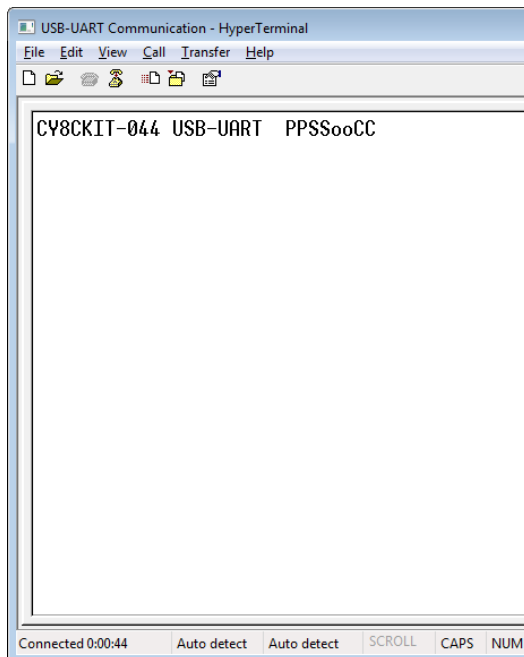
6. In PuTTY, click **Session** and select **Serial** under **Connection type**. **Serial line** shows the communication port (COM12) and **Speed** shows the baud rate selected. Click **Open** to start the communication, as shown in Figure 4-13.

Figure 4-13. Opening Port in PuTTY



- The COM terminal software displays both the typed data and the echoed data from the PSoC 4200M UART, as shown in Figure 4-14.

Figure 4-14. Data Displayed on HyperTerminal and PuTTY



5. Using the KitProg USB-I2C Bridge

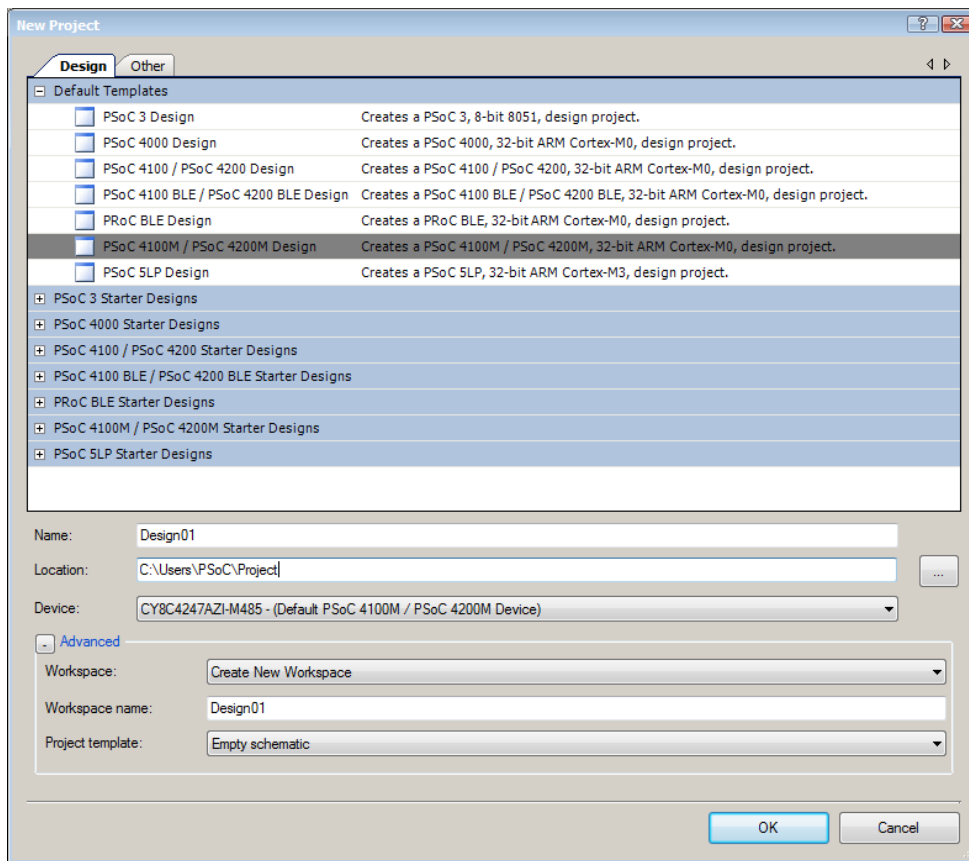


The KitProg serves as a USB-I2C bridge that can be used to communicate with the USB-I2C software running on the PC. For example, the KitProg USB-I2C Bridge can be used to tune the CapSense Component on a PSoC device. This feature is applicable to all kits listed in [Table 2-1](#). This section uses the 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 BCP software and the PSoC 4200 M device.

Note: For more information on how to use the KitProg USB-I2C Bridge to tune the CapSense Component, refer to section 5.4 Project: CapSense of the CY8CKIT-042 PSoC 4 Pioneer Kit Guide.

1. Create a new PSoC 4 project in PSoC Creator, as shown in [Figure 5-1](#). Select an appropriate location for your project and name the project as desired. You must select the appropriate device for this project depending on the kit, as shown 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 5-1. Create New Project in PSoC Creator



2. Drag and drop an EZI2C Slave (SCB mode) Component from the Component Catalog ([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 EZI2C Slave Component, double-click or right-click the EZI2C Slave Component and select **Configure**, as shown in [Figure 5-3](#).

Figure 5-2. EZI2C Slave Component in Component Catalog

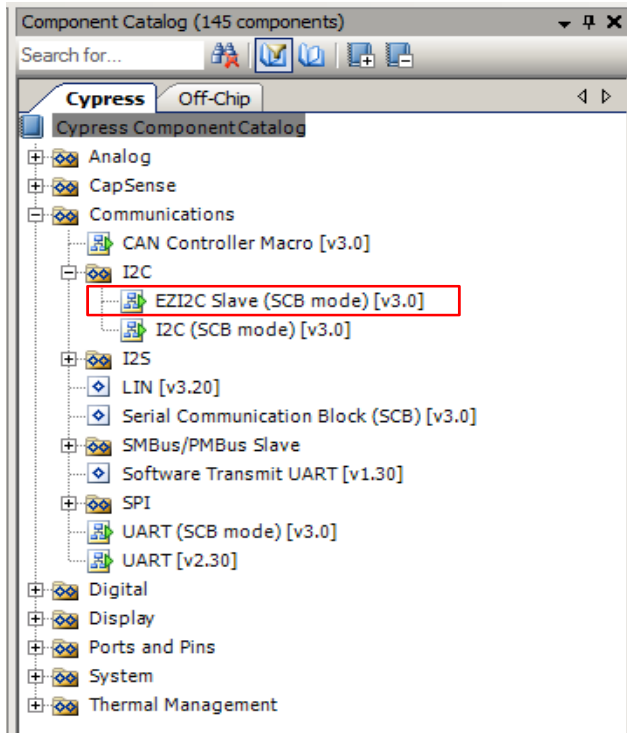
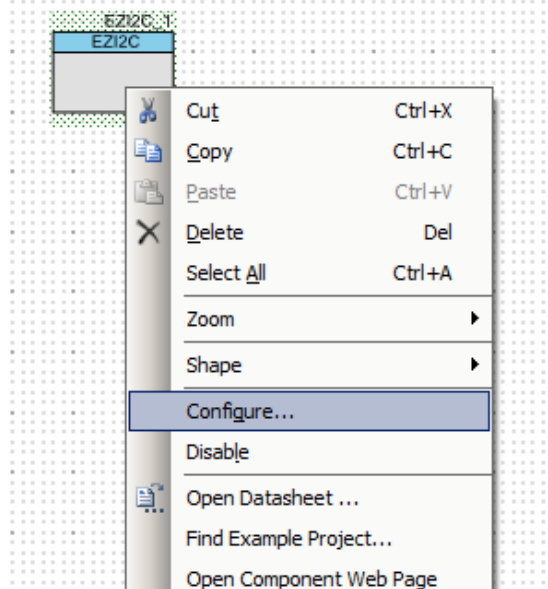


Figure 5-3. Open EZI2C Slave Configuration Window



3. Configure the EZI2C Slave Component as shown in [Figure 5-4](#) and [Figure 5-5](#) and click **OK**.

Figure 5-4. Configuration Tab

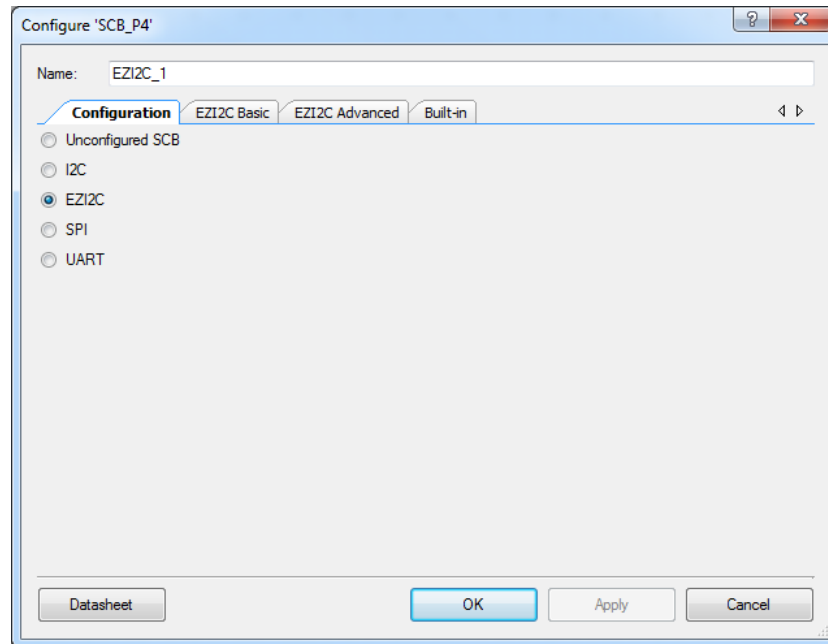
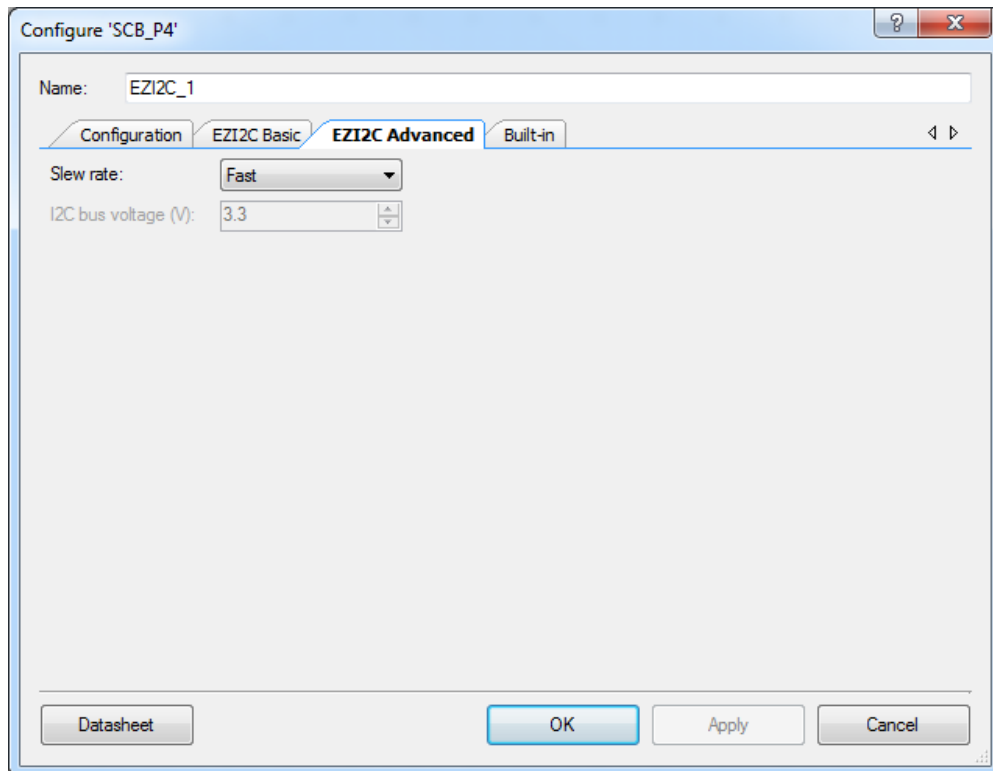
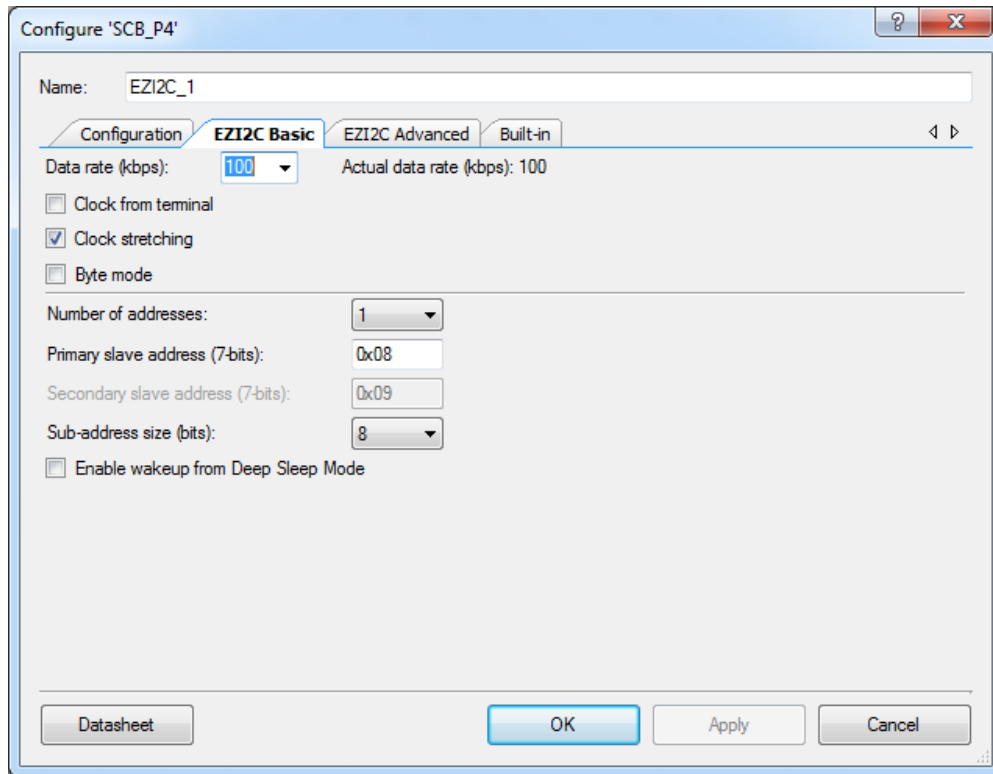


Figure 5-5. EZI2C Slave Basic and Advanced Tabs



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 5-6. 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.

Figure 5-6. 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	<input checked="" type="checkbox"/>
\EZI2C_1:sda\	P4[1]	SCB0:uart_tx, CAN0:can_tx, SCB0:i2c sda, SCB0:spi miso	28	<input checked="" type="checkbox"/>

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();

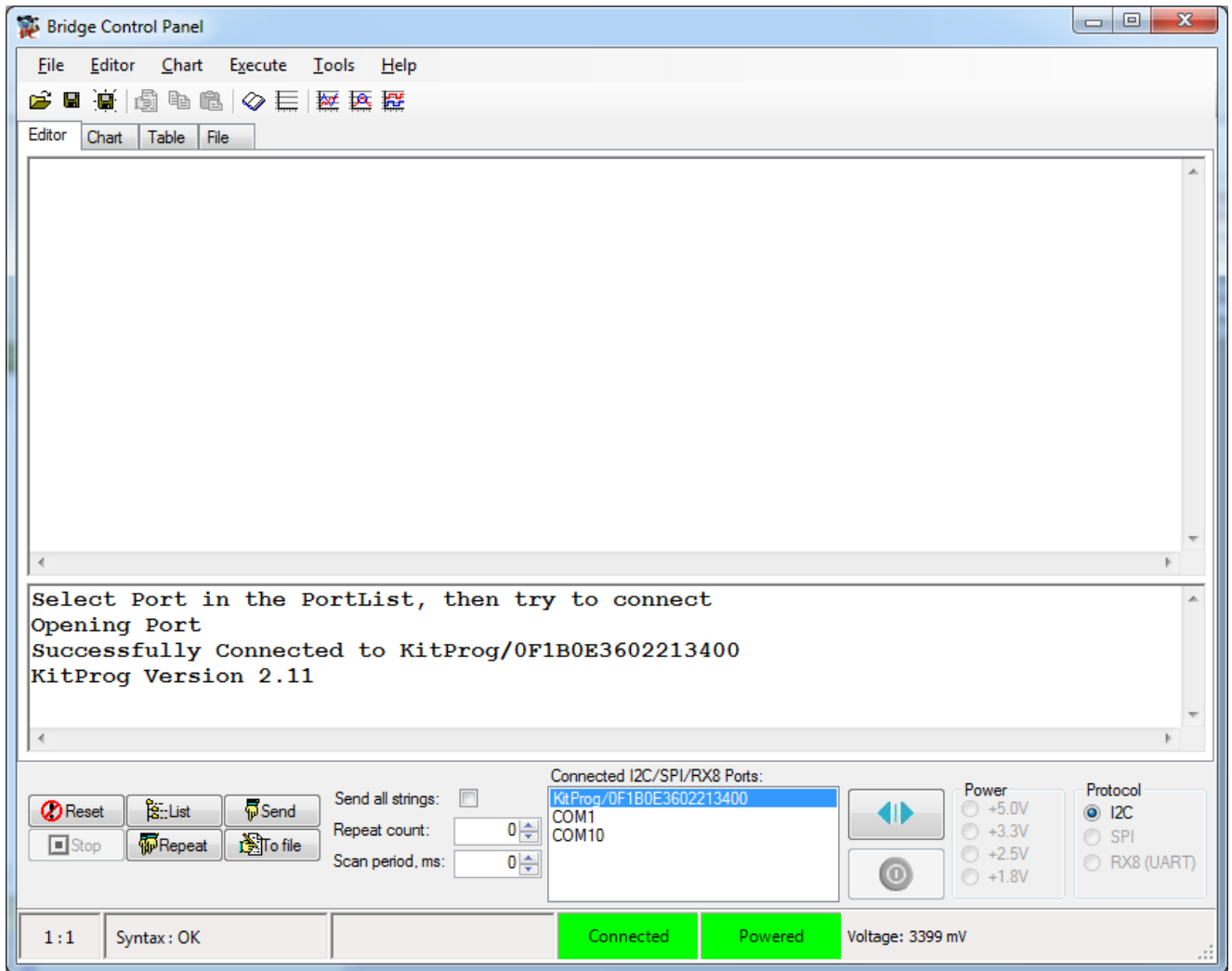
    /* This API sets the buffer and address boundary to which the external
     * master can communicate. In this example, external master can read
     * from and write to the first 5 bytes of the i2cBuffer and read bytes
     * from all the 10 bytes of the i2cBuffer array. */
    EZI2C_1_EzI2CSetBuffer1(BUF_SIZE,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

    for(;;)
    {

    }
}
```

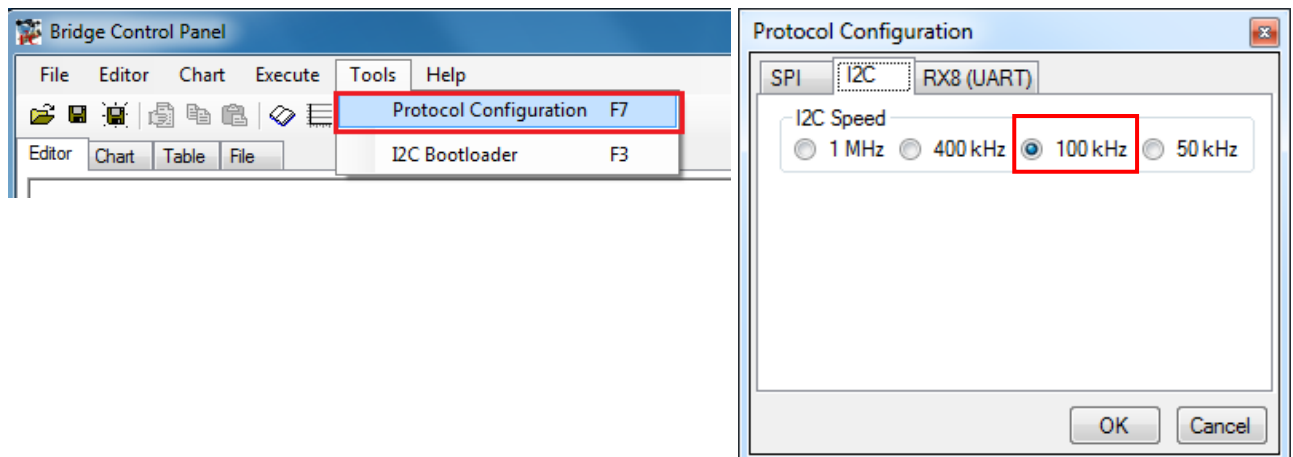
6. Build the project by choosing **Build > Build Project** or pressing **[Shift] [F6]**. After the project is built without errors and warnings, program (**[Ctrl] [F5]**) this project onto the PSoC 4200M using KitProg.
7. Open the BCP from **Start > All Programs > Cypress > Bridge Control Panel <version> > Bridge Control Panel <version>**.
8. Select **KitProg/<serial number>** under **Connected I2C/SPI/RX8 Ports**, as shown in Figure 5-7.

Figure 5-7. Connecting to KitProg/ in BCP



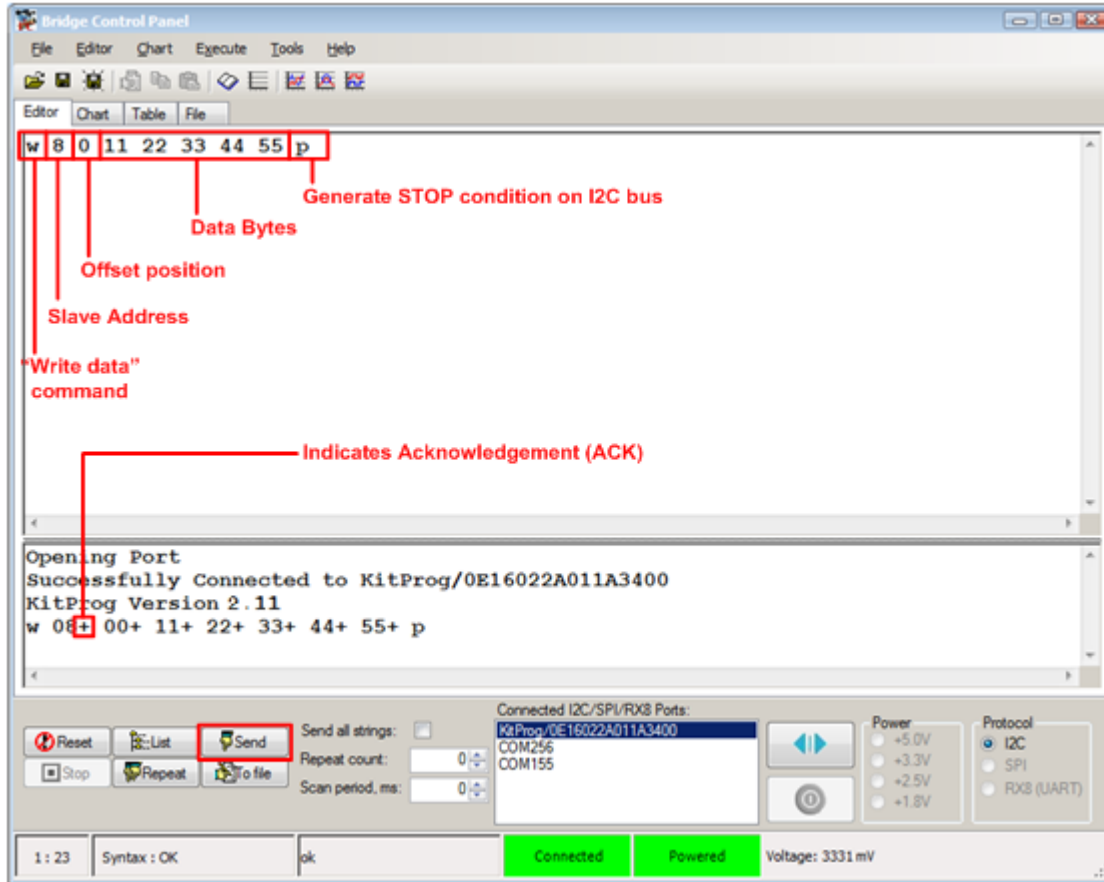
- Open **Protocol Configuration** from the **Tools** menu and select the appropriate **I2C Speed**, as shown in Figure 5-8. Make sure the I2C speed is the same as the one configured in the EZI2C Slave Component. Click **OK** to close the window.

Figure 5-8. Opening Protocol Configuration Window in BCP



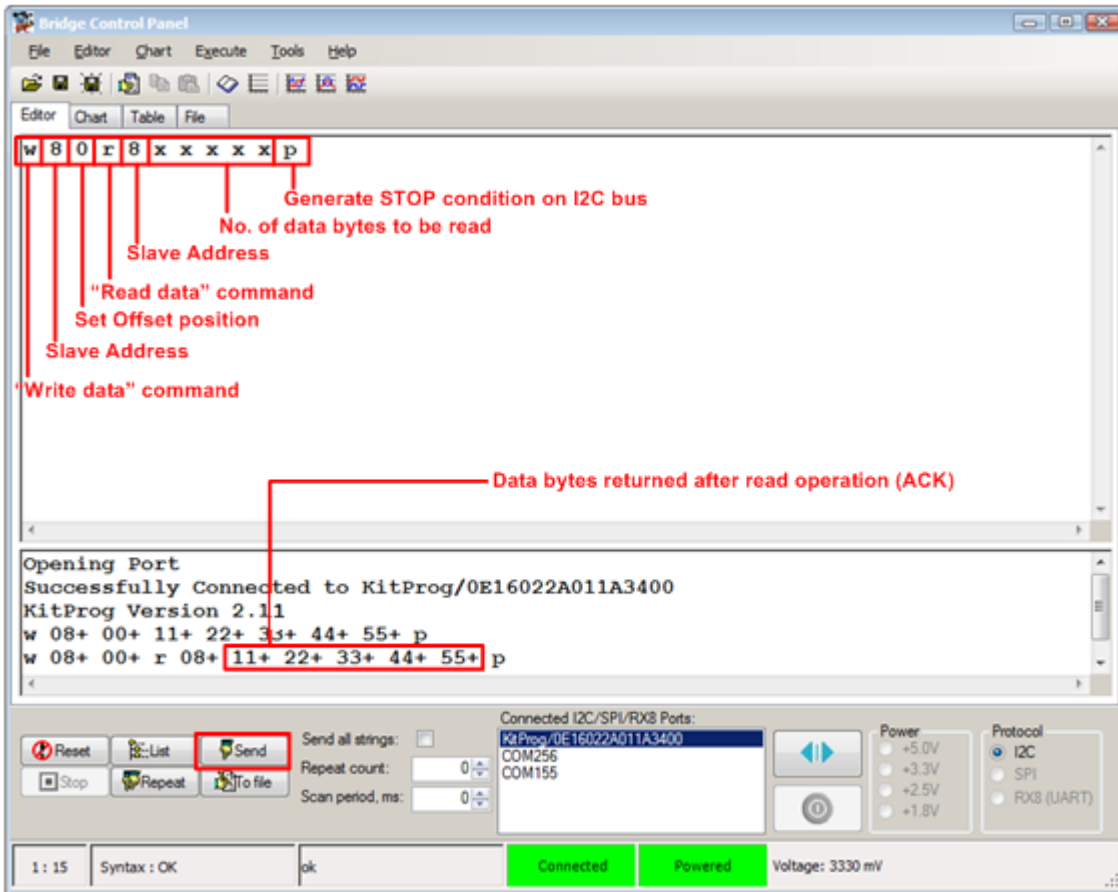
- From the BCP, transfer 5 bytes of data to the I2C device with slave address 0x08. The EZI2C Slave requires an additional parameter to be sent from the BCP to set the offset address from/to where the data bytes are read/written. Type the command shown in Figure 5-9 and press **[Enter]** or click the **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 5-9. Enter Commands in BCP



- 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 5-10.

Figure 5-10. Read Data Bytes from BCP



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.

6. 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](#) 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 give step by step directions for building a Bootloadable and a Normal project for PSoC 5LP.

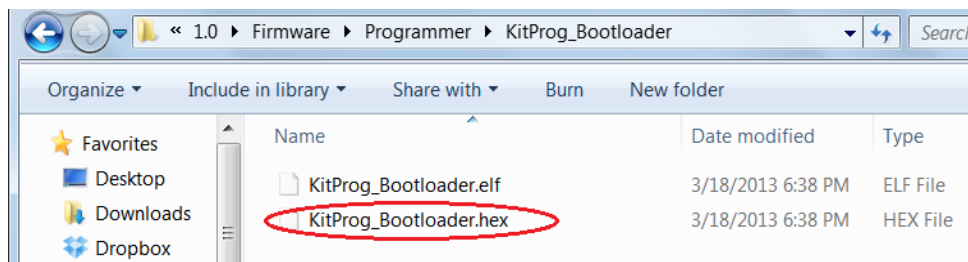
6.1 Building a Bootloadable Project for PSoC 5LP

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 6-1](#).

<Install_Directory>\<Kit_Name>\<version>\Firmware\Programmer\KitProg_Bootloader

Figure 6-1. KitProg Bootloader Hex File Location



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 6-3](#). 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 6-2. Open New Project in PSoC Creator

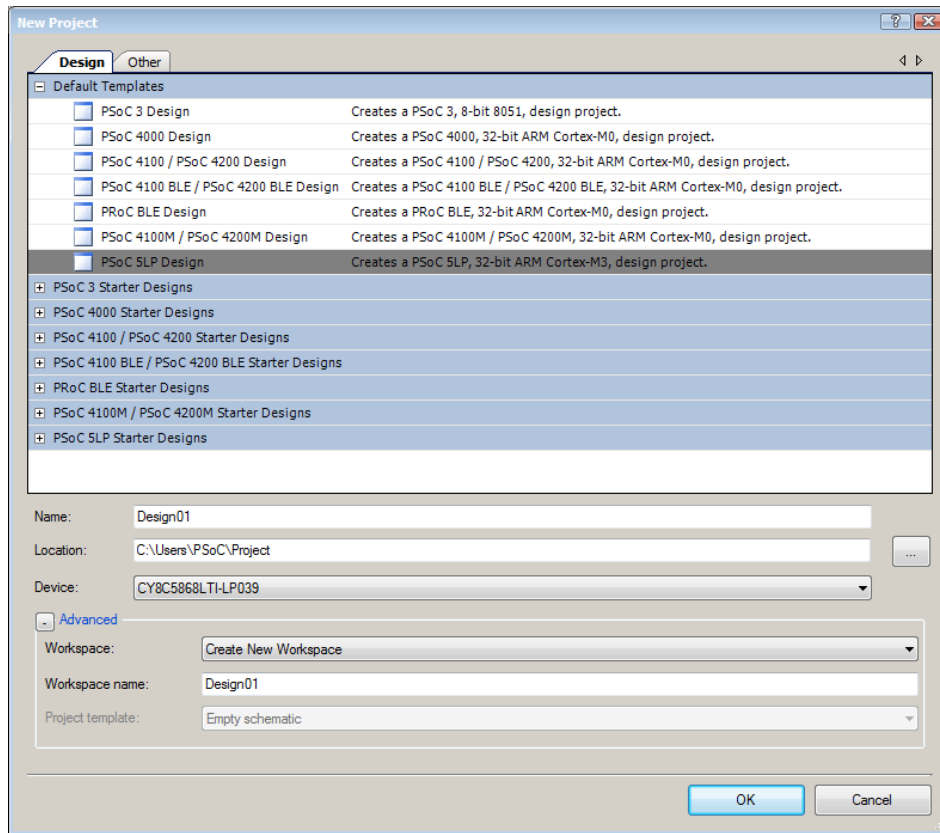
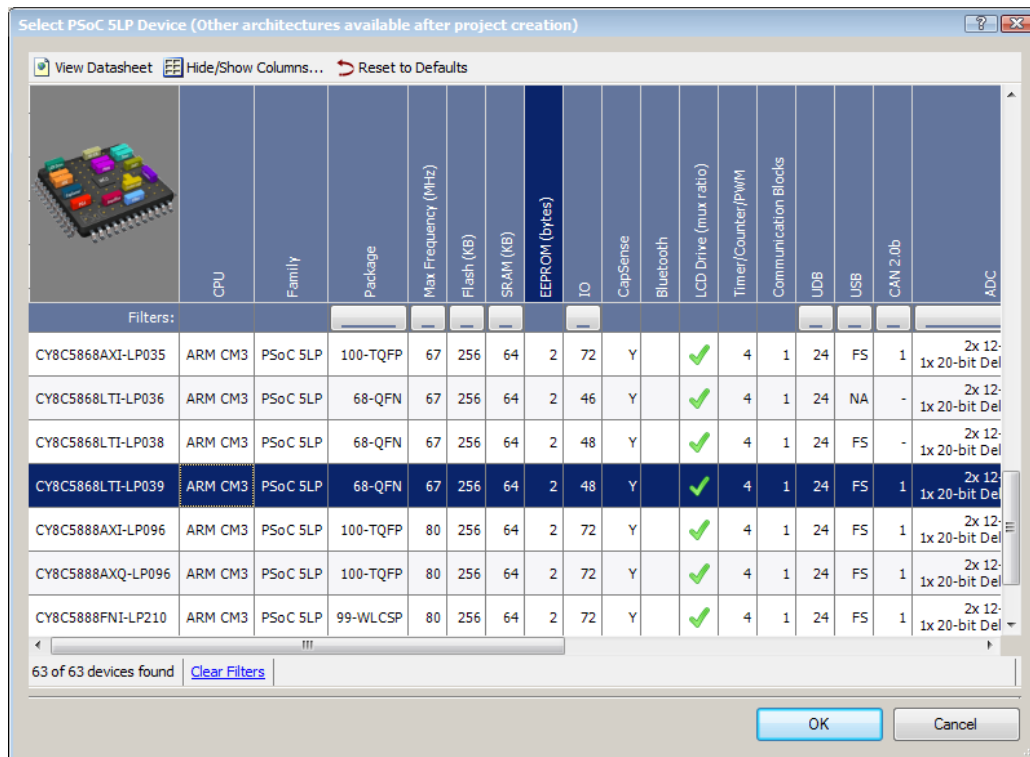
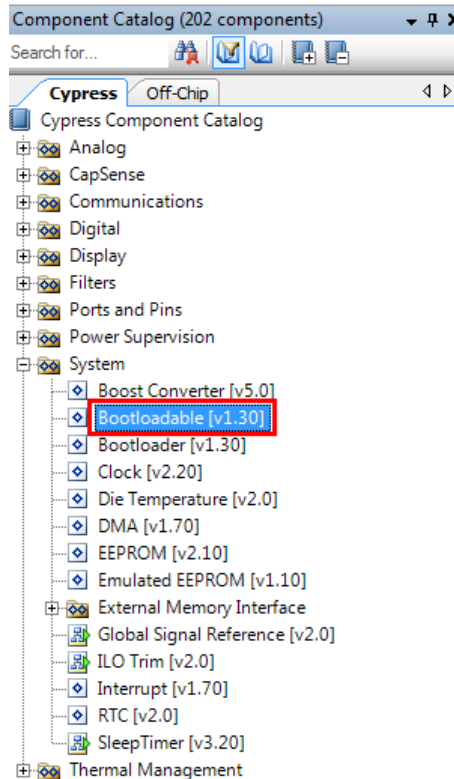


Figure 6-3. Select Device in PSoC Creator



- Navigate to the Schematic view and drag and drop a Bootloadable Component (Figure 6-4) on the TopDesign.

Figure 6-4. Bootloadable Component in Component Catalog



- 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 6-5. Select the *KitProg_Bootloader.hex* (Figure 6-6) 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 6-5. Configuration Window of Bootloadable Component

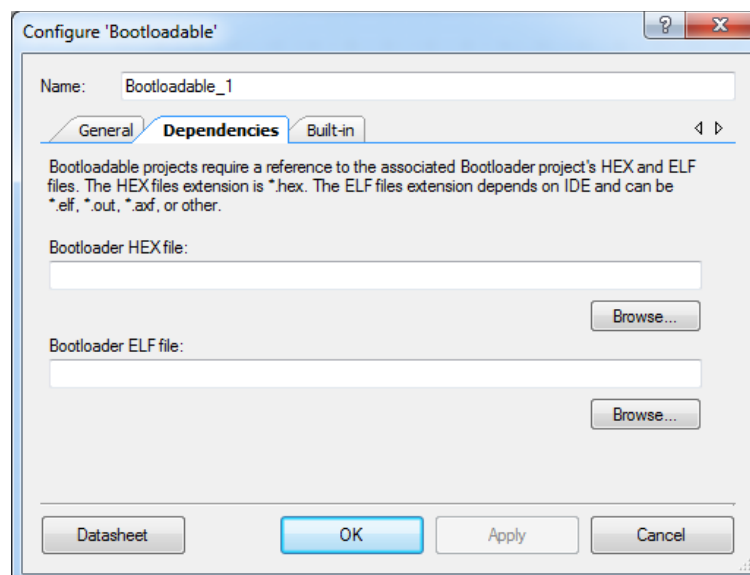
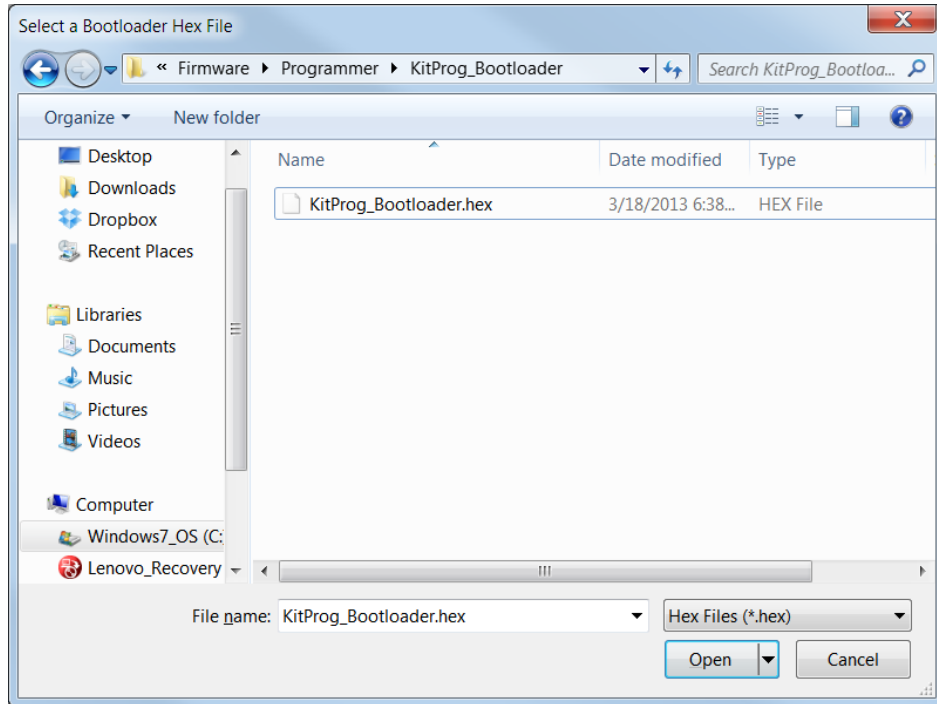
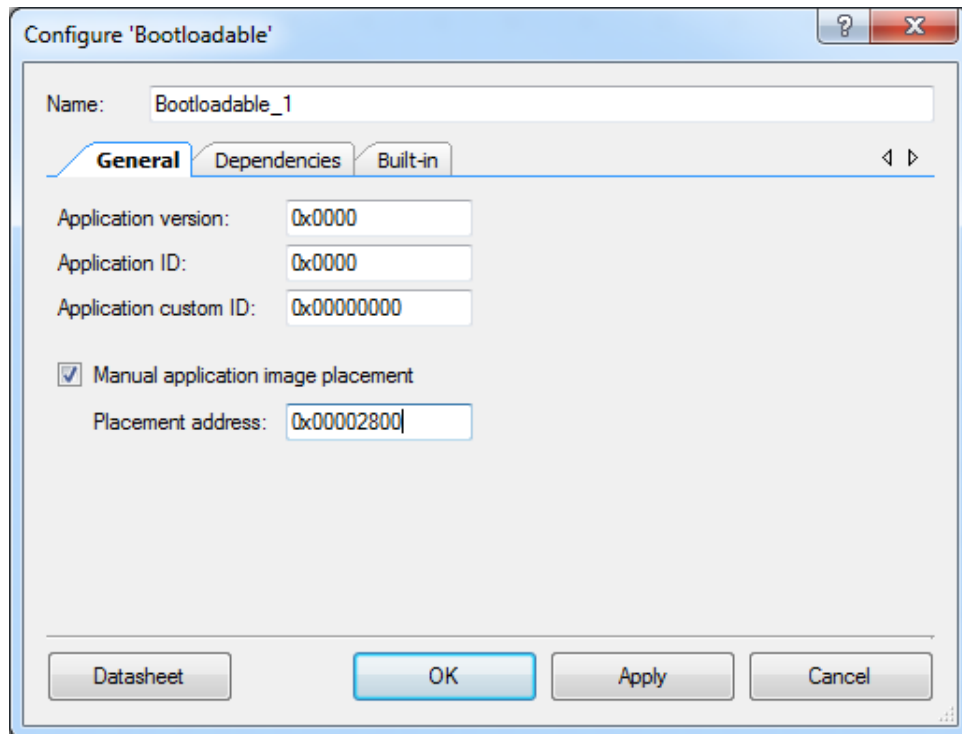


Figure 6-6. Select KitProg Bootloader Hex File



4. In the **General** tab, check the **Manual application image placement** checkbox and set the **Placement address** as '0x00002800' as shown in Figure 6-7.

Figure 6-7. Bootloadable Component-General Tab



5. Develop your custom project.

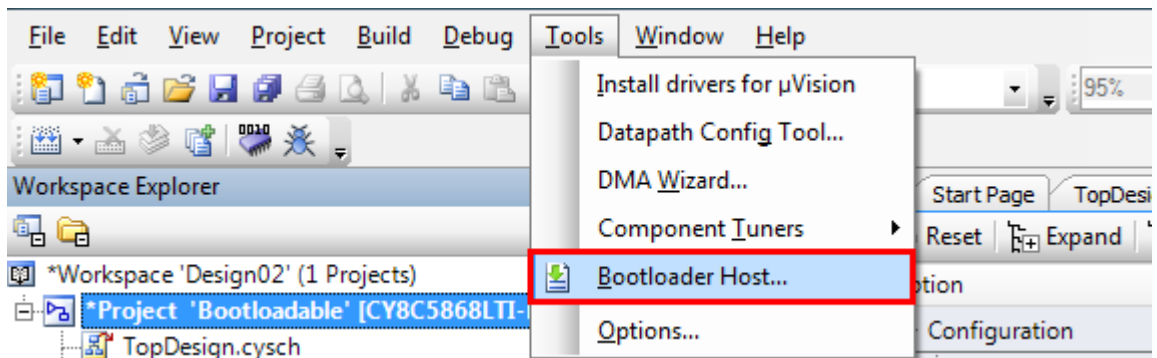
- Make sure that the `<project name>.cydwr System` setting of the Bootloadable project and the KitProg_Bootloader project is the same. Figure 6-8 shows the `KitProg_Bootloader.cydwr System` settings.

Figure 6-8. KitProg Bootloader System Settings

[-] Configuration	
... Device Configuration Mode	Compressed
... Enable Error Correcting Code (ECC)	<input type="checkbox"/>
... Store Configuration Data in ECC Memory	<input type="checkbox"/>
... Instruction Cache Enabled	<input checked="" type="checkbox"/>
... Enable Fast IMO During Startup	<input checked="" type="checkbox"/>
... Unused Bonded IO	Allow with info
... Heap Size (bytes)	0x1000
... Stack Size (bytes)	0x4000
... Include CMSIS Core Peripheral Library Files	<input checked="" type="checkbox"/>
[-] Programming/Debugging	
... Debug Select	GPIO
... Enable Device Protection	<input type="checkbox"/>
... Embedded Trace (ETM)	<input type="checkbox"/>
... Use Optional XRES	<input type="checkbox"/>
[-] Operating Conditions	
... Vddd (V)	5.0
... Vdda (V)	5.0
... Variable Vdda	<input type="checkbox"/>
... Vddio0 (V)	5.0
... Vddio1 (V)	5.0
... Vddio2 (V)	5.0
... Vddio3 (V)	5.0
... Temperature Range	-40C - 85/125C

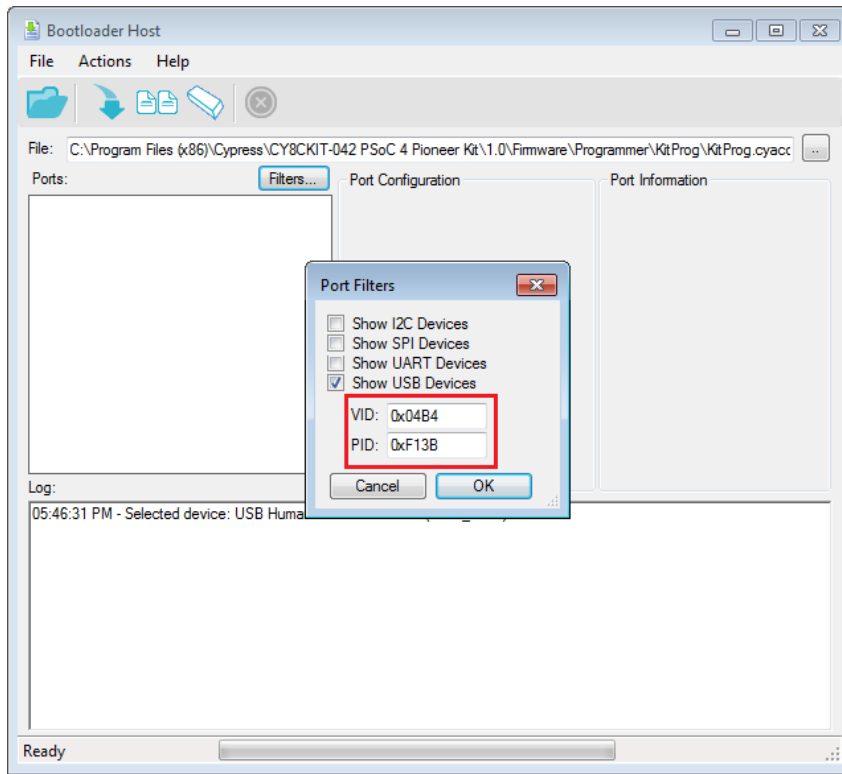
- Build the project in PSoC Creator by choosing **Build > Build Project** or pressing **[Shift] [F6]**.
- To program the project onto the PSoC 5LP device, open the Bootloader Host tool, which is available in PSoC Creator. Choose **Tools > Bootloader Host**, as shown in Figure 6-9.

Figure 6-9. Open Bootloader Host Tool in PSoC Creator



- 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.
- 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 6-10.

Figure 6-10. Port Filters Tab in Bootloader Host Tool



11. In the Bootloader Host tool, click the **Open File** button (Figure 6-11) to browse to the location of the bootloadable file (*.cyacd), as shown in Figure 6-12. This file is present in the project directory.

Figure 6-11. Open Bootloadable File in Bootloader Host Tool

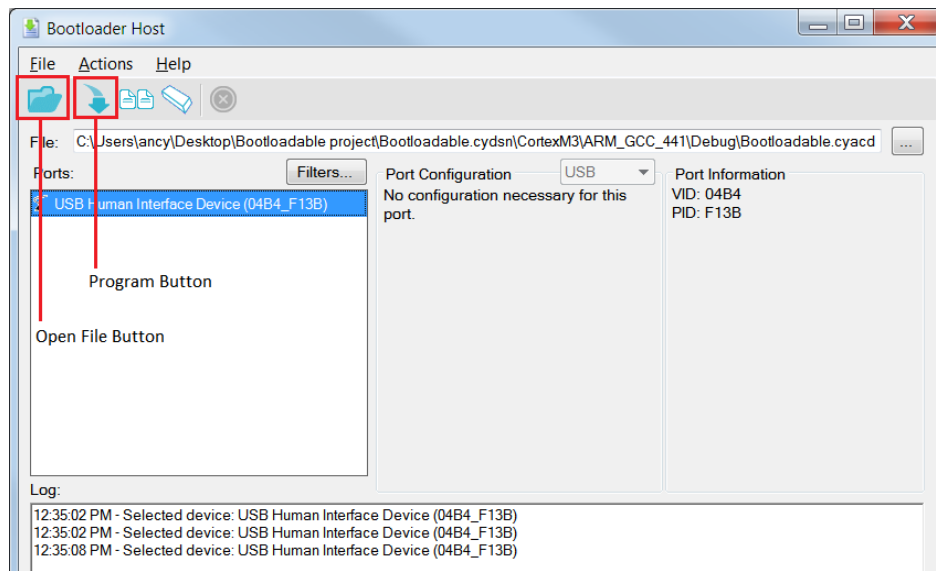
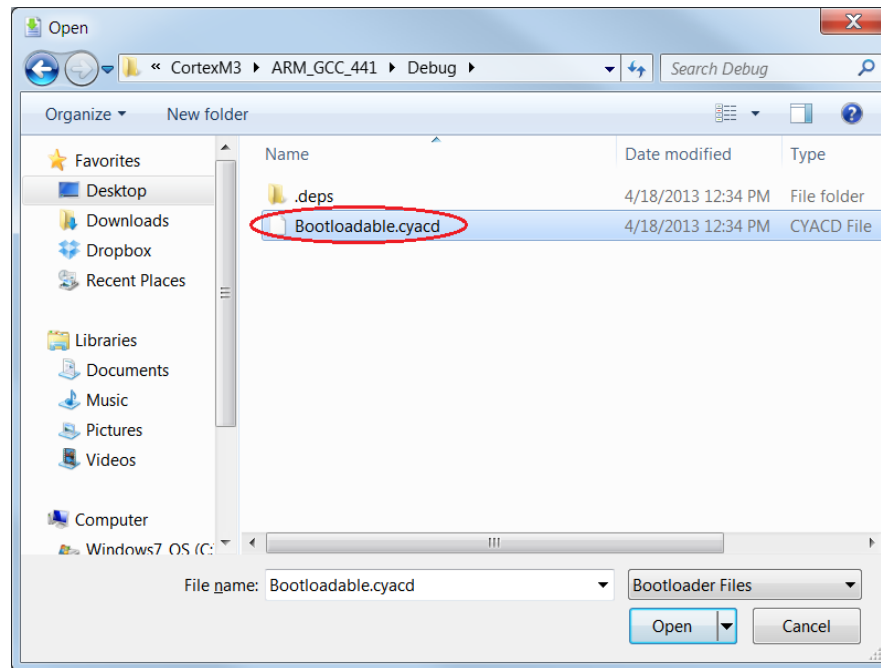


Figure 6-12. Select Bootloadable .cyacd File from Bootloader Host Tool



12. Select the **USB Human Interface Device** in the **Ports** list and click the **Program** button (Figure 6-11) 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.
- The status LED does not function unless it is used by the custom project.

For additional information on bootloaders, refer to the Cypress application note [AN73503 – USB HID Bootloader for PSoC 3 and PSoC 5LP](#).

6.2 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.

The *KitProg.hex* file is available at the following location:

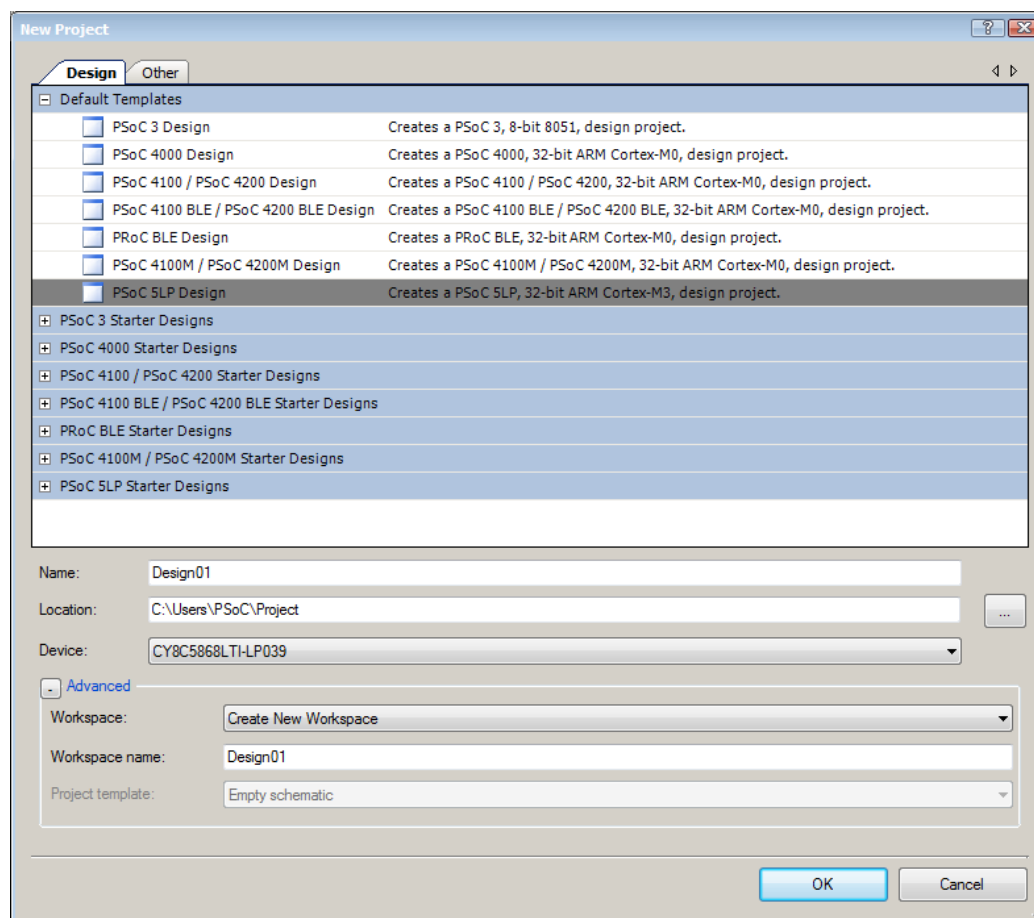
```
<Install_Directory>\<Name_of_the_Kit>\<version>\Firmware\Programmer\KitProg
```

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.

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**, as shown in Figure 6-13 and click **OK**.

Figure 6-13. Create New Project in PSoC Creator



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 10-pin PSoC 5LP programming header J5 (which needs to be populated).
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.

7. Troubleshooting the KitProg



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

7.1 KitProg Status LED Indication

The KitProg Status LED on the development kit indicates the status of the KitProg operation using different blink rates. Table 7-1 shows the KitProg LED indication and the corresponding status of the KitProg.

Table 7-1. 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 <i>KitProg.cyacd</i> 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 <i>.cyacd</i> 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 section Updating the KitProg Firmware for explanation on how to update the KitProg firmware.

7.2 PSoC 5LP Factory Program Restore Instructions

7.2.1 PSoC 5LP is Programmed with a Bootloadable Application

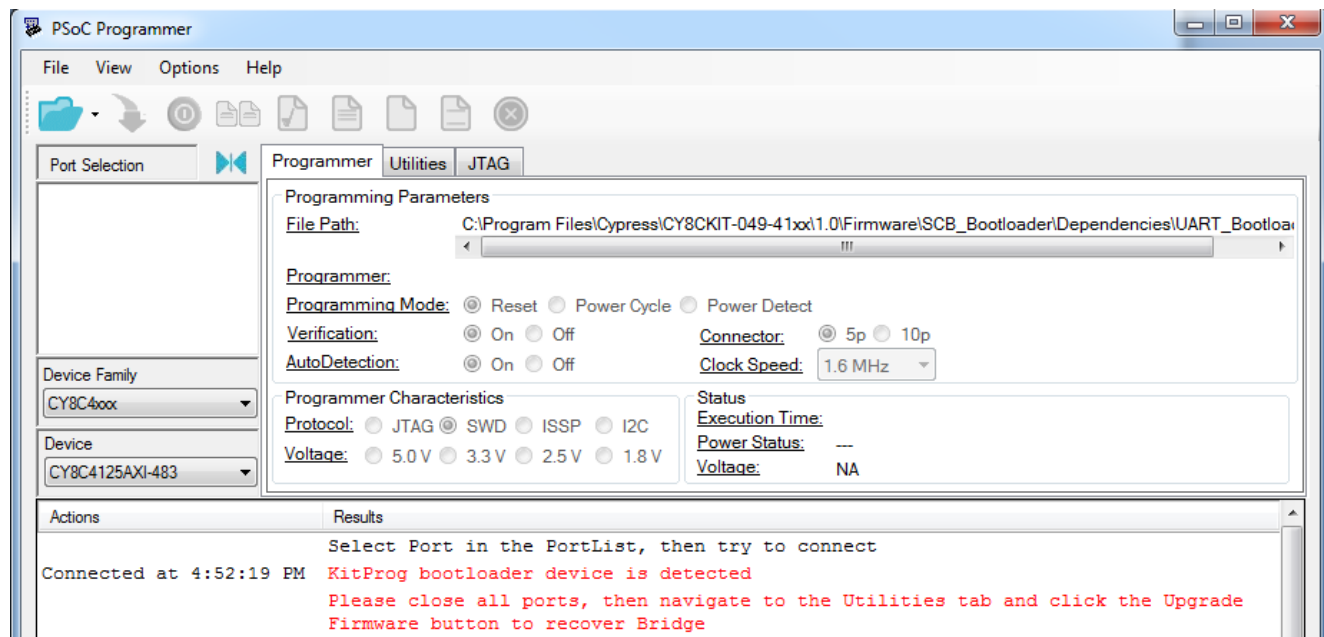
Reprogramming or bootloading the PSoC 5LP device with a new flash image will overwrite the KitProg and forfeit the ability to use the PSoC 5LP device as a programmer/debugger for the kit. If the PSoC 5LP is programmed with a bootloadable application, restore the KitProg by using one of the following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

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

7.2.1.1 Restore PSoC 5LP Factory Program Using PSoC Programmer

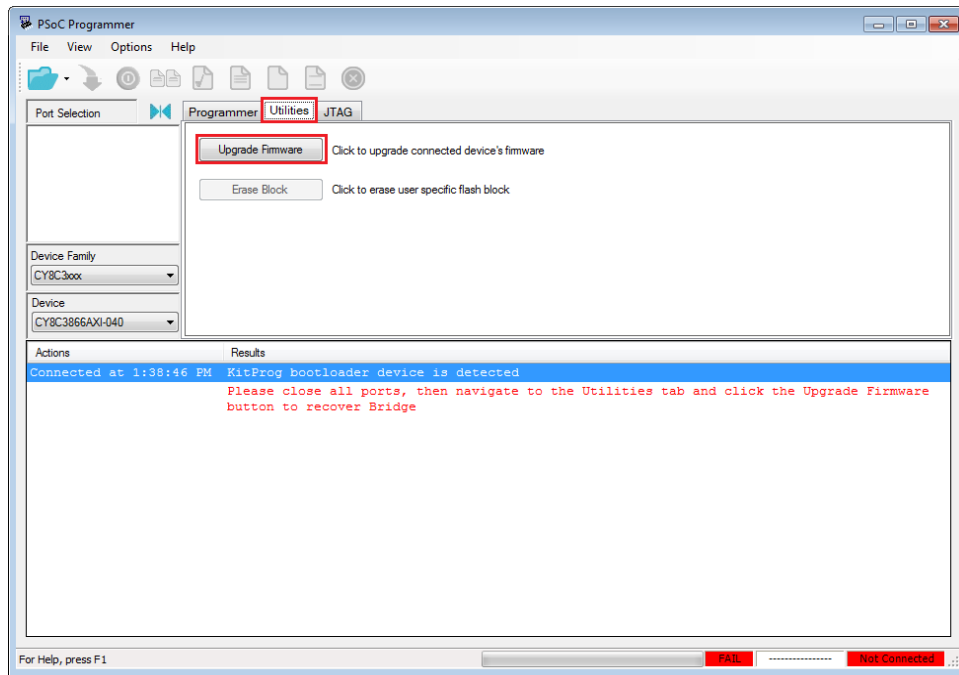
1. Launch PSoC Programmer from **Start > Cypress > PSoC Programmer <version> > PSoC Programmer <version>**.
2. Configure the PSoC 4 M-Series Pioneer Kit in bootloader mode. To do this, while pressing the reset button (SW1), connect the PSoC 4 M-Series Pioneer Kit to the computer using the included USB 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, as shown in [Figure 7-1](#): “KitProg Bootloader device is detected.”

Figure 7-1. PSoC Programmer Results Window



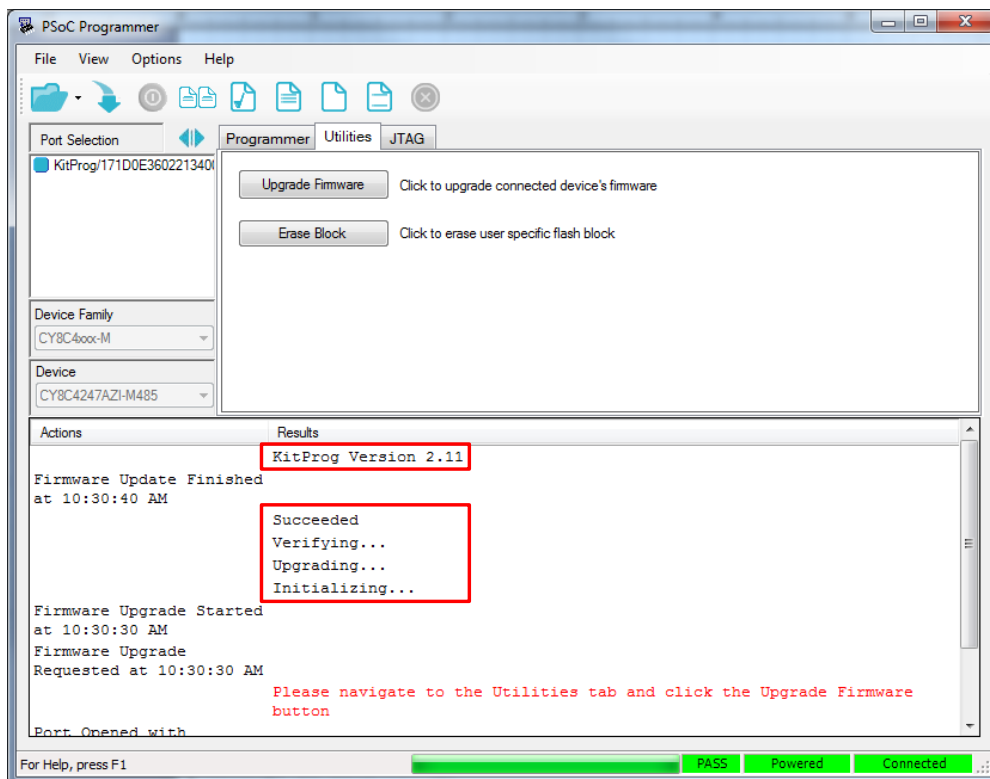
4. Switch to the **Utilities** tab in PSoC Programmer and click the **Upgrade Firmware** button, as shown in [Figure 7-2](#). Unplug all other PSoC programmers (such as MiniProg3 and DVKProg) from the PC before clicking the **Upgrade Firmware** button.

Figure 7-2. Upgrade Firmware



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 7-3.

Figure 7-3. Firmware Update Completed



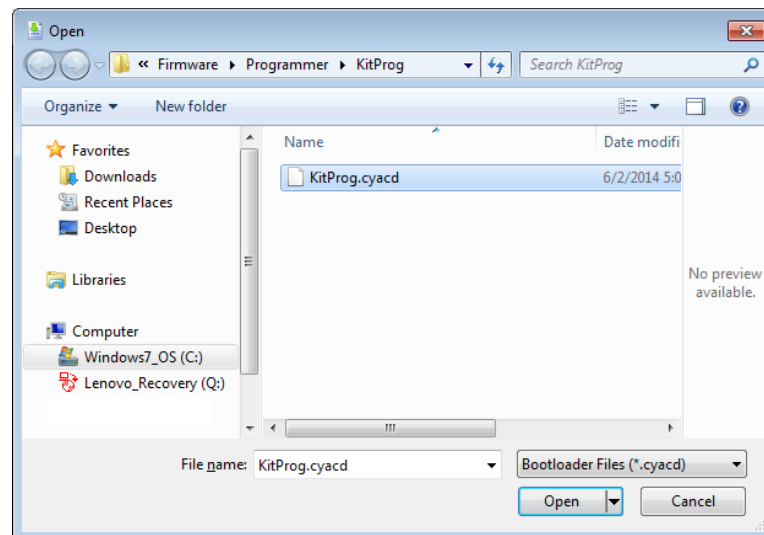
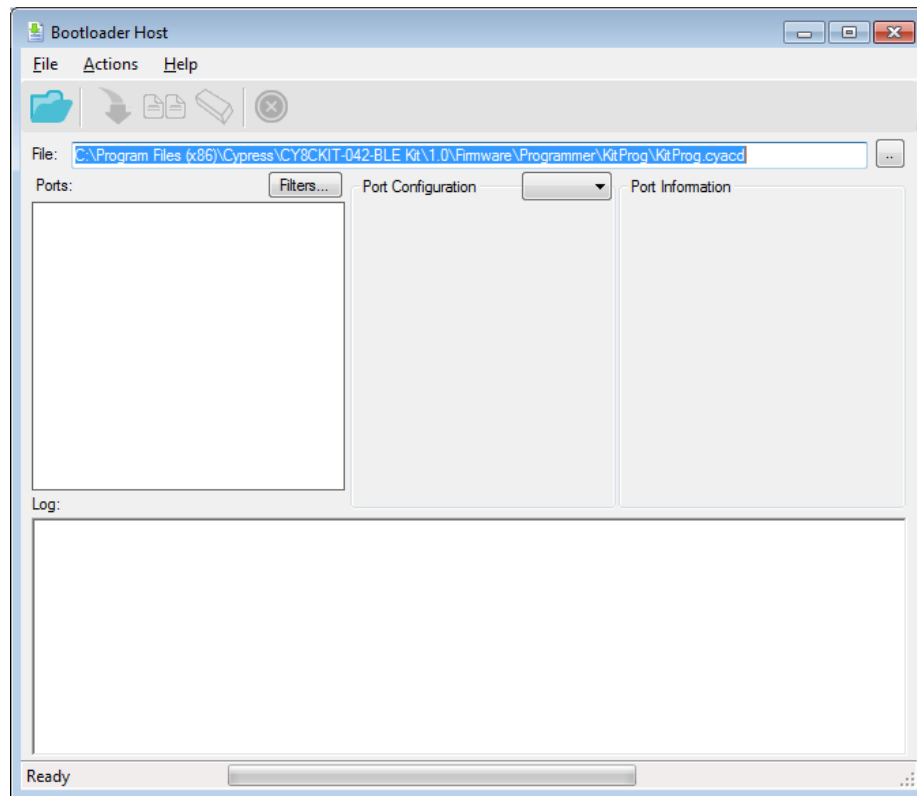
6. The factory program is now successfully restored on the PSoC 5LP. It can be used as the programmer/debugger for the PSoC 4200M device.

7.2.1.2 Restore PSoC 5LP Factory Program Using USB 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 7-4](#). The default location for this file is:

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

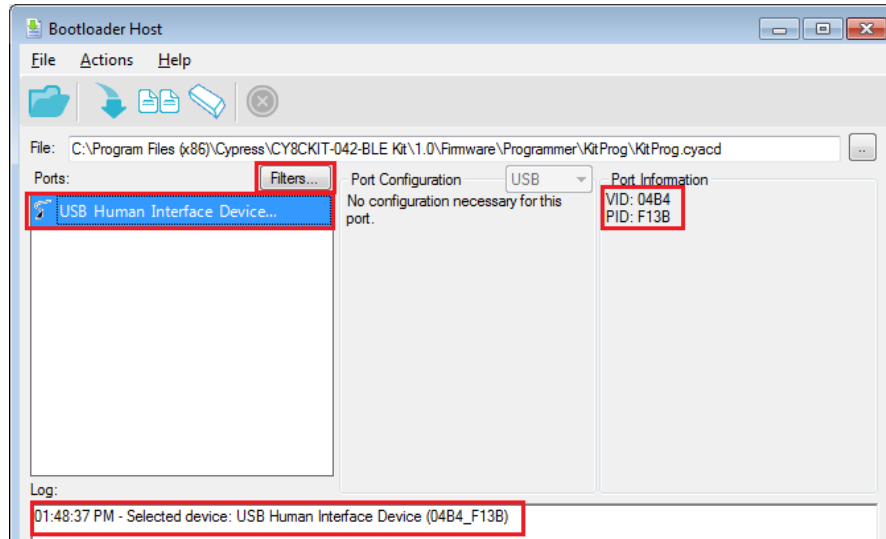
Figure 7-4. Load KitProg .cyacd File



3. Configure the PSoC 4 M-Series Pioneer Kit in bootloader mode. To do this, while holding down the reset button (SW1), 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.

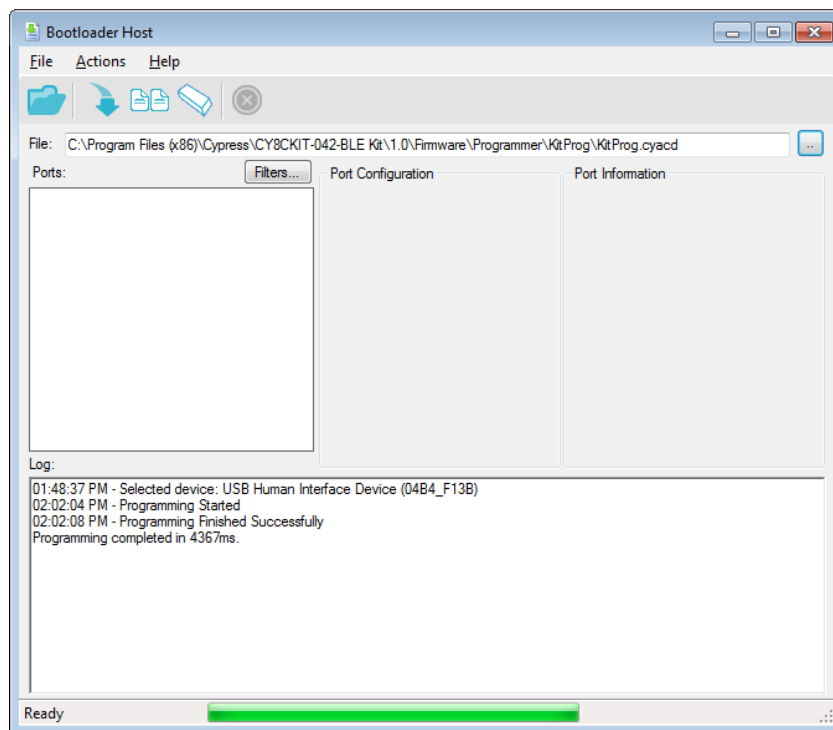
- In the Bootloader Host tool, set the filters for the USB devices with **VID: 04B4** and **PID: F13B**. The **USB Human Interface Device** port appears in the **Ports** list. Click that port to select it, as shown in Figure 7-5.

Figure 7-5. Select USB Human Interface Device



- Click the **Program** button (or choose **Actions > Program**) to restore the factory program by bootloading it onto the PSoC 5LP.
- After programming is completed, the message “Programming Finished Successfully” appears, as shown in Figure 7-6.

Figure 7-6. Programming Finished Successfully



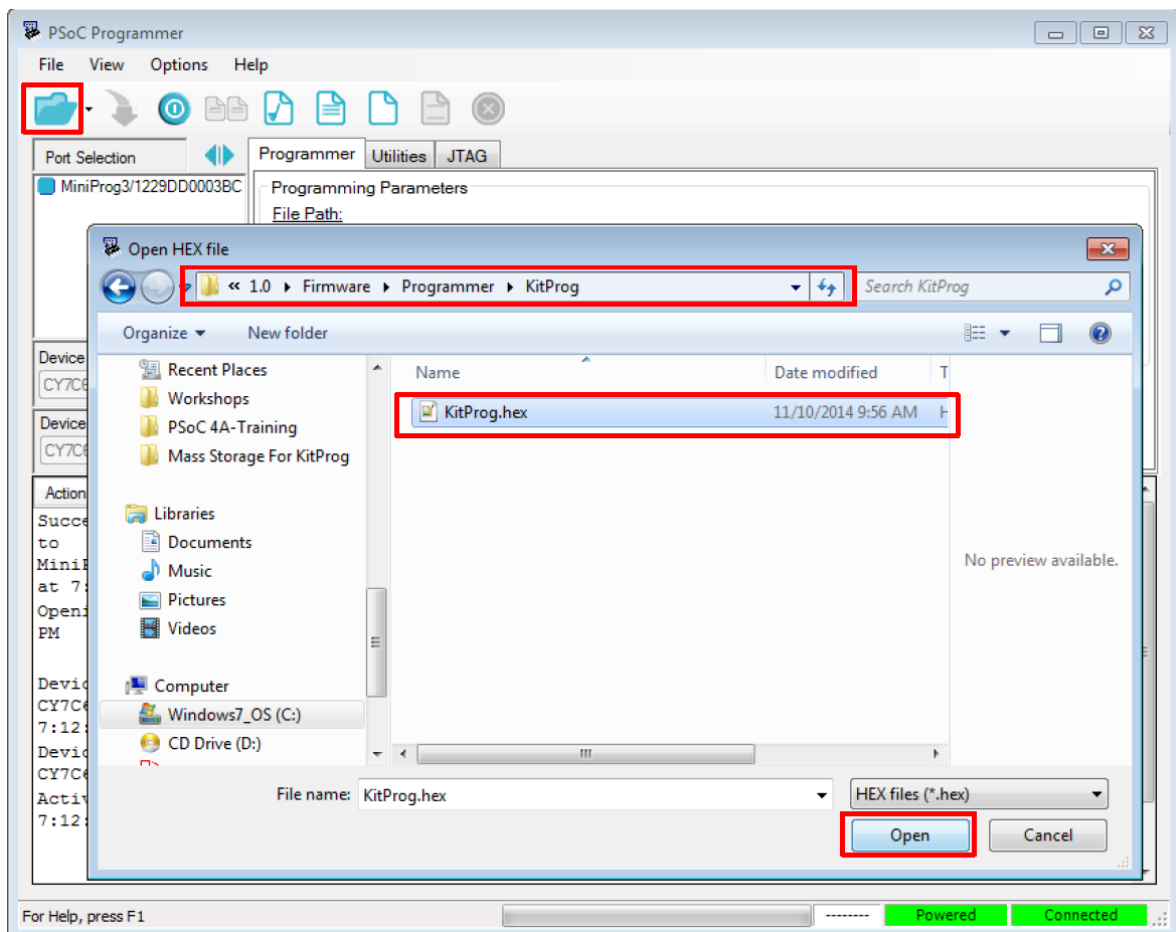
- The factory KitProg program is now successfully restored on the PSoC 5LP.

7.2.2 PSoC 5LP is Programmed with a Standard Application (Restore 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.

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 7-7](#). 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 7-7. Select the *KitProg.hex* File to Program the PSoC 5LP



Appendix



A.1 Pin Assignment Tables

PSoC 5LP GPIO Header (J8) for CY8CKIT-042-BLE and CY8CKIT-044

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

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	GND	J8_12	P3[0]	IDAC Output

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 3-7 , Figure 3-8 , Figure 6-8 , and Figure 7-3 .