Copyrights

© Cypress Semiconductor Corporation, 2015-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

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.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoc, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.
# Contents

1. Introduction 4

2. KitProg Ecosystem 5

3. 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 .......................................................................... 11
   - 3.5 Updating the KitProg Firmware .................................................................. 11

4. KitProg Mass Storage Programmer 14
   - 4.1 Enter or Exit the Mass Storage Programmer Mode ....................................... 14
   - 4.2 Programming Using the Mass Storage Programmer ..................................... 14
   - 4.3 Frequently Asked Questions on KitProg Mass Storage Programmer .......... 17

5. KitProg USB-UART Bridge 18

6. KitProg USB-I2C Bridge 27

7. Developing Applications for PSoC 5LP 35
   - 7.1 Building a Bootloadable Project for PSoC 5LP ........................................... 35
   - 7.2 Building a Normal Project for PSoC 5LP .................................................. 42

8. Troubleshooting the KitProg 44
   - 8.1 KitProg Status LED Indication .................................................................... 44
   - 8.2 PSoC 5LP Factory Program Restore Instructions ......................................... 45
   - 8.3 Leakage Current Reduction on Target MCU through SWD Lines ............ 50

Appendix 51

Revision History 53
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

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

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

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Pre-requisite Software</th>
<th>Download Link/Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer</td>
<td>PSoC Programmer</td>
<td><a href="http://www.cypress.com/psocprogrammer">www.cypress.com/psocprogrammer</a></td>
</tr>
<tr>
<td>Debugger</td>
<td>PSoC Creator</td>
<td><a href="http://www.cypress.com/psoccreator">www.cypress.com/psoccreator</a></td>
</tr>
<tr>
<td>USB-I2C Bridge</td>
<td>Bridge Control Panel (BCP)</td>
<td>Installed along with PSoC Programmer</td>
</tr>
<tr>
<td>USB-UART Bridge</td>
<td>Terminal Emulator Program</td>
<td>Any terminal emulator program can be used such as HyperTerminal (available as part of Microsoft Windows XP installation) or PuTTY (available from <a href="http://www.putty.org">www.putty.org</a>)</td>
</tr>
</tbody>
</table>
The KitProg supports different speeds for communication interfaces. Table 2-3 summarizes the KitProg operating modes.

Table 2-3. KitProg Operating Modes

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Supported Speed</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer</td>
<td>1.6</td>
<td>MHz</td>
</tr>
<tr>
<td>USB-I2C Bridge</td>
<td>50, 100, 400, 1000</td>
<td>kHz</td>
</tr>
<tr>
<td>USB-UART Bridge</td>
<td>1200, 2400, 4800, 9600, 9200, 38400, 57600, and 115200</td>
<td>Baud</td>
</tr>
</tbody>
</table>

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.
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.
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>.
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB Composite Device</td>
<td>USB Composite device</td>
</tr>
<tr>
<td>USB Input Device</td>
<td>USB-I2C bridge, KitProg command interface</td>
</tr>
<tr>
<td>KitProg</td>
<td>Programmer and debugger</td>
</tr>
<tr>
<td>KitProg USB-UART</td>
<td>USB-UART bridge, which appears as the COM# port</td>
</tr>
</tbody>
</table>

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)

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
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

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 firmware version

KitProg firmware upgrade message
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

- Hard Disk Drives (2)
  - Windows7_OS (C:)
    - 34.7 GB free of 281 GB
  - Lenovo_Recovery (Q:)
    - 5.65 GB free of 15.6 GB
- Devices with Removable Storage (2)
  - CD Drive (D:)
    - 339 KB free of 340 KB
  - KitProg (E:)

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

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

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

```
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.
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
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
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
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.
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.

```c
#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

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
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

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
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

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 5-13.

Figure 5-13. Opening Port in PuTTY
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
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
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

![Component Catalog](image)

Figure 6-3. Open EZI2C Slave Configuration Window

![Configuration Window](image)
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
Figure 6-5. EZI2C Slave Basic and Advanced Tabs

Configure 'SCB_P4'

Name: EZI2C_1

Configuration: EZI2C Basic  EZI2C Advanced  Built-in

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 wake-up from Deep Sleep Mode

Datasheet  OK  Apply  Cancel

Configure 'SCB_P4'

Name: EZI2C_1

Configuration: EZI2C Basic  EZI2C Advanced  Built-in

Slow rate: Fast

I2C bus voltage (V): 3.3

Datasheet  OK  Apply  Cancel
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](image)

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.

```c
#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

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
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

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

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_Bootloader

**Figure 7-1. 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 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
Figure 7-3. Select Device in PSoC Creator

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
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

![Configuration Window of Bootloadable Component](image)

Figure 7-6. Select KitProg Bootloader Hex File

![Select KitProg Bootloader Hex File](image)
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

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
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.

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.
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

![Open Bootloadable File in Bootloader Host Tool](image_url)

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

![Select Bootloadable .cyacd File from Bootloader Host Tool](image_url)
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 bootloader 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](http://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**

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

<table>
<thead>
<tr>
<th>User Indication</th>
<th>Scenario</th>
<th>Action Required by User</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED blinks fast: Frequency = 4.00 Hz</td>
<td>LED starts blinking at power up, if bootloadable file is corrupt.</td>
<td>Bootload the <code>KitProg.cyacd</code> file: In PSoC Programmer, connect to the kit, go to the Utilities tab, and press the Upgrade Firmware button.</td>
</tr>
<tr>
<td>LED blinks slow: Frequency = 0.67 Hz</td>
<td>Entered Bootloader mode by holding the Reset button during kit power-up.</td>
<td>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 <code>.cyacd</code> file using the Bootloader Host tool available in PSoC Creator.</td>
</tr>
<tr>
<td>LED blinks very fast: Frequency = 15.0 Hz</td>
<td>SWD or I2C operation is in progress. The Kit’s COM port connect / disconnect event (only one blink).</td>
<td>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.</td>
</tr>
<tr>
<td>LED is ON</td>
<td>USB enumeration successful. Kit is in the idle state waiting for commands.</td>
<td>PSoC Creator, PSoC Programmer, BCP, and any serial port terminal program can use the kit functions.</td>
</tr>
<tr>
<td>LED is OFF</td>
<td>Power LED is ON</td>
<td>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.</td>
</tr>
</tbody>
</table>

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”.
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.
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

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:

\(<\text{Install Directory}>\backslash<\text{Kit_Name}>\backslash<\text{version}>\backslash\text{Firmware}\backslash\text{Programmer}\backslash\text{KitProg}\backslash\text{KitProg.cyacd}\)

**Figure 8-4. 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 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

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

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

<table>
<thead>
<tr>
<th>J8</th>
<th>Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
<th>Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J8_01</td>
<td>PSoC 5LP_VDD</td>
<td>VDD</td>
<td>J8_02</td>
<td>P1[2]</td>
<td>Digital I/O</td>
<td></td>
</tr>
<tr>
<td>J8_03</td>
<td>P0[0]</td>
<td>Delta Sigma ADC + Input</td>
<td>J8_04</td>
<td>P0[1]</td>
<td>Delta Sigma ADC - Input</td>
<td></td>
</tr>
<tr>
<td>J8_15</td>
<td>P2[5]</td>
<td>SPI SSEL</td>
<td>J8_16</td>
<td>GND</td>
<td>GND</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>J8</th>
<th>Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
<th>Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J8_01</td>
<td>PSoC 5LP_VDD</td>
<td>VDD</td>
<td>J8_02</td>
<td>P1[2]</td>
<td>Digital I/O</td>
<td></td>
</tr>
<tr>
<td>J8_03</td>
<td>P0[0]</td>
<td>Delta Sigma ADC + Input</td>
<td>J8_04</td>
<td>P0[1]</td>
<td>Delta Sigma ADC - Input</td>
<td></td>
</tr>
<tr>
<td>J8_11</td>
<td>GND</td>
<td>SPI MISO/I2C SDA</td>
<td>J8_12</td>
<td>P3[0]</td>
<td>IDAC Output</td>
<td></td>
</tr>
</tbody>
</table>
### A.1.3 PSoC 5LP GPIO Header (J8 and J9) for CY8CKIT-059 and CY8CKIT-043

<table>
<thead>
<tr>
<th>J9</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
<th>J8</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J9_01</td>
<td>VBUS</td>
<td>Power/VDD</td>
<td>J8_01</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>J9_02</td>
<td>GND</td>
<td>Ground</td>
<td>J8_02</td>
<td>P3[0]</td>
<td>GPIO</td>
</tr>
<tr>
<td>J9_04</td>
<td>P12[0]</td>
<td>GPIO/I2C_SCL</td>
<td>J8_04</td>
<td>P3[5]</td>
<td>GPIO</td>
</tr>
<tr>
<td>J9_06</td>
<td>P12[6]</td>
<td>GPIO/UART_RX</td>
<td>J8_06</td>
<td>P0[0]</td>
<td>GPIO</td>
</tr>
</tbody>
</table>
# Document Revision History

## Document Title: KitProg User Guide

**Document Number: 001-96359**

<table>
<thead>
<tr>
<th>Revision</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>02/25/2015</td>
<td>RNJT</td>
<td>Initial version of KitProg User Guide.</td>
</tr>
</tbody>
</table>
| *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.                                                                                                                                      |
<table>
<thead>
<tr>
<th>Revision</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>*G</td>
<td>09/21/2015</td>
<td>SRDS</td>
<td>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.</td>
</tr>
<tr>
<td>*H</td>
<td>05/17/2017</td>
<td>AESATMP8</td>
<td>Updated logo and Copyright.</td>
</tr>
<tr>
<td>*I</td>
<td>04/24/2018</td>
<td>VKVK</td>
<td>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.</td>
</tr>
</tbody>
</table>