Please note that Cypress is an Infineon Technologies Company. The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.
Contents

1. Introduction .......................... 5
   1.1 Switching between KitProg2 Modes ................................................................. 7
   1.2 Acronyms ......................................................................................................... 9

2. Ecosystem .................................. 10

3. KitProg2 Mode Programmer and Debugger ................................. 12
   3.1 KitProg2 Driver Installation ........................................................................... 12
   3.2 Programming Using PSoC Creator ................................................................. 13
   3.3 Debugging Using PSoC Creator ................................................................. 16
   3.4 Programming Using PSoC Programmer .......................................................... 17
   3.5 Updating the KitProg2 Firmware .................................................................... 17

4. CMSIS-DAP Mode Programming and Debugging ......................... 20
   4.1 Programming and Debugging using µVision ............................................... 20

5. Mass Storage Programmer .................................................. 27
   5.1 Enter or Exit the Mass Storage Programmer Mode ........................................... 27
   5.2 Programming Using the Mass Storage Programmer ........................................ 27
   5.3 Frequently Asked Questions on KitProg2 Mass Storage Programmer .............. 29

6. USB-UART Bridge ............................................. 30

7. USB-I2C Bridge .............................................. 41

8. USB-SPI Bridge .................................................. 52

9. Developing Applications for PSoC 5LP ........................................... 58
   9.1 Building a Custom Bootloadable Application for KitProg2 ............................. 58
   9.2 Building a Normal Project for PSoC 5LP ...................................................... 67

10. Troubleshooting the KitProg2 .................................................. 71
   10.1 KitProg2 Status LED Indication ...................................................................... 71
   10.2 PSoC 5LP Factory Program Restore Instructions ........................................ 73
       10.2.1 PSoC 5LP is Programmed with a Custom Application ......................... 73
       10.2.1.1 Restore PSoC 5LP Factory Program Using PSoC Programmer .......... 74
   10.2.2 Restore PSoC 5LP KitProg2 using MiniProg3 ........................................... 76
A. Appendix ................................................................. 78
  A.1 Pin Assignments .................................................................................. 78
  A.1.1 PSoC 5LP GPIO and Custom Application Header (J8 and J11)
       for CY8CKIT-041-40XX and CY8CKIT-041-41XX ................................. 78
  A.1.2 PSoC 5LP GPIO and Custom Application Header (J16 and J11)
       for CY8CKIT-048................................................................................ 79
  A.1.3 PSoC 5LP GPIO and Custom Application Header (J6 and J7)
       for CY8CKIT-145-40XX..................................................................... 79

Revision History ................................................................. 80
1. Introduction

The KitProg2 is an onboard programmer/debugger with USB-UART, USB-I2C and USB-SPI Bridge functionality. It is an update to the existing KitProg used for programming and debugging the target device. The KitProg2 is integrated onto most PSoC® development kits. This user guide provides comprehensive information on how to use the KitProg2 with PSoC development kits. Figure 1-1 shows the KitProg2 ecosystem. The Cypress PSoC 5LP device is used to implement the KitProg2 functionality.

Figure 1-1. KitProg2 Ecosystem
KitProg2 is an enhancement over KitProg. It follows the dual-image bootloading approach; the KitProg2 firmware is the first image and a custom application can be loaded as a second image. To learn more about the concept of dual-image bootloading, refer to application note AN73854 – PSoC 3, PSoC 4, and PSoC 5LP Introduction to Bootloaders.

Figure 1-2. KitProg2 Dual-Image Bootloader Architecture

![Custom Application Image](image)

KitProg2 Bootloader

Figure 1-2 shows the KitProg2 flash architecture based on the concept of dual-image bootloading. If you are building custom applications for PSoC 5LP in KitProg2, only the ‘Custom Application Image’ flash area can be bootloaded with the corresponding cyacd file. The KitProg2 image cannot be bootloaded. It can only be restored to factory settings as described in Restore PSoC 5LP Factory Program Using PSoC Programmer on page 74.

The KitProg2 image contains SWD, CMSIS-DAP, and Mass Storage Programmer functionality to program a target PSoC, and USB-UART/USB-I2C/USB-SPI Bridge functionality. The custom application image can be any application that PSoC 5LP can execute. To create custom applications, refer to the Developing Applications for PSoC 5LP chapter on page 58.
1.1 Switching between KitProg2 Modes

There are three types of kits that use KitProg2 programmers. These are: (1) prototyping kits (also known as stamp boards) which have a single mode switch and a single amber status LED; (2) development kits (also known as pioneer kits) which have a single mode switch and 3 status LEDs; and (3) development kits which have two mode switches and three status LEDs.

Figure 1-3 shows how to switch between modes in KitProg2. Part (a) of Figure 1-3 is valid for all kits with a single mode switch. This figure illustrates the switching workflow using the CY8CKIT-041-40XX kit as an example, which has SW3 as the mode switch. Part (b) of Figure 1-3 is valid for all kits with dual mode switches. This figure illustrates the switching workflow using the CY8CKIT-062 BLE Pioneer Kit. In either case the input on the mode switch or switches is evaluated; depending on the current mode of operation, the next mode of operation is ascertained.

On power-on reset or reset, PSoC 5LP enters bootloader entry mode. If the mode switch SW3 was pressed while the USB connector was plugged in and then released, KitProg2 enters bootloader mode. If the mode switch SW3 was not pressed, then depending on the current mode of operation, PSoC 5LP will enter PPCOM mode, Mass Storage Programming/CMSIS-DAP mode, or the custom application.

As illustrated in Figure 1-3 part (a) switching between KitProg2 and Mass Storage Programming/CMSIS-DAP mode can be achieved by pressing and releasing the mode switch within five seconds. Similarly, switching to the custom application from the PPCOM or Mass Storage Programming/CMSIS-DAP mode can be achieved by pressing and holding the mode switch for more than five seconds and then releasing. Switching from the custom mode back to PPCOM or Mass Storage Programming/CMSIS-DAP modes is dependent on the custom application implementation.

In order to recognize the various state changes the Amber LED shows different effects. When in KitProg2 Mode the Amber LED lights up, when in Mass Storage/CMSIS-DAP mode the Amber LED turns off and when in bootloader mode the Amber LED shows a blinking effect.
Figure 1-3. Switching between KitProg2 Modes

a) For Single Mode Switch Development Kits

b) For Two Mode Switch Development Kits

Note: In order to switch back from custom mode application to CMSIS-DAP/Mass Storage or KitProg2 within the firmware, refer Developing Applications for PSoC 5LP on page 58.

Note: After pressing and releasing the mode switch, it may take a few seconds for the kit to re-enumerate in the new mode. While enumeration is taking place, none of the status LEDs will be ON.
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCP</td>
<td>Bridge Control Panel</td>
</tr>
<tr>
<td>BLE</td>
<td>Bluetooth Low Energy</td>
</tr>
<tr>
<td>CMSIS-DAP</td>
<td>Cortex Microcontroller Software Interface Standard Debug Access Protocol</td>
</tr>
<tr>
<td>GPIO</td>
<td>General-Purpose Input/Output</td>
</tr>
<tr>
<td>HID</td>
<td>Human Interface Device</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>LED</td>
<td>Light-Emitting Diode</td>
</tr>
<tr>
<td>MISO</td>
<td>Master-In-Serial-Out</td>
</tr>
<tr>
<td>MOSI</td>
<td>Master-Out-Serial-In</td>
</tr>
<tr>
<td>NVL</td>
<td>Non Volatile Latch</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PPCOM</td>
<td>PSoC Programmer Component Object Module</td>
</tr>
<tr>
<td>PSoC</td>
<td>Programmable System-on-Chip</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
</tr>
<tr>
<td>SCB</td>
<td>Serial Communication Block</td>
</tr>
<tr>
<td>SCL</td>
<td>Serial Clock Line</td>
</tr>
<tr>
<td>SDA</td>
<td>Serial Data Line</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>SWD</td>
<td>Serial Wire Debug</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>UDB</td>
<td>Universal Digital Block</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>XRES</td>
<td>External Reset</td>
</tr>
</tbody>
</table>
2. Ecosystem

Table 2-1 lists the development kits that use the KitProg2. Table 2-2 lists the prerequisite Cypress software needed to use the KitProg2.

Table 2-1. Development Kits Supported by KitProg2

<table>
<thead>
<tr>
<th>Development Kits</th>
<th>Target Device</th>
<th>KitProg2 onboard mode switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>CY8CKIT-041-40XX PSoC 4 S-Series Pioneer Kit</td>
<td>PSoC 4000S</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-041-41XX PSoC 4100S Pioneer Kit</td>
<td>PSoC 4100S</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit</td>
<td>PSoC Analog Coprocessor</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-145-40XX PSoC 4 S-Series Prototyping Kit</td>
<td>PSoC 4000S</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-146 PSoC 4200DS Prototyping Kit</td>
<td>PSoC 4200DS</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-147 PSoC 4100PS Prototyping Kit</td>
<td>PSoC 4100PS</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-148 PSoC 4700S Inductive Sensing Evaluation Kit</td>
<td>PSoC 4700S</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-149 PSoC 4100S Plus Prototyping Kit</td>
<td>PSoC 4100S Plus</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CPROTO-063-BLE PSoC 6 BLE Prototyping Kit</td>
<td>PSoC 6 MCU (PSoC 63 Line)</td>
<td>Single Switch</td>
</tr>
<tr>
<td>CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit</td>
<td>PSoC 6 MCU (PSoC 63 Line)</td>
<td>Dual Switch</td>
</tr>
<tr>
<td>CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit</td>
<td>PSoC 6 MCU (PSoC 62 Line)</td>
<td>Dual Switch</td>
</tr>
</tbody>
</table>

Table 2-2. Prerequisite Software for KitProg2 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-SPI 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 on <a href="http://www.putty.org">www.putty.org</a>)</td>
</tr>
</tbody>
</table>
KitProg2 supports different speeds for communication interfaces. Table 2-3 summarizes the KitProg2 operating modes.

Table 2-3. KitProg2 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-UART Bridge</td>
<td>1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200</td>
<td>Baud</td>
</tr>
<tr>
<td>USB-I2C Bridge</td>
<td>50, 100, 400, 1000</td>
<td>kHz</td>
</tr>
<tr>
<td>USB-SPI Bridge</td>
<td>50–6000</td>
<td>kHz</td>
</tr>
</tbody>
</table>

This document assumes that you know the basics of using 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®
This section explains how the KitProg2 programmer/debugger is integrated onto the PSoC development kits. The KitProg2 supports the development kits listed in Table 2-1 on page 10. This section uses the PSoC 4 S-Series Pioneer Kit as an example.

KitProg2 uses two types of programming/debugging interfaces – PPCOM and CMSIS-DAP. The PPCOM interface supports Cypress tool chains such as PSoC Creator and PSoC Programmer. The PPCOM interface provides additional options such as programming NVLs, which are not available via a standard CMSIS-DAP interface.

CMSIS-DAP is an alternative programming/debugging interface in which the KitProg2 can be used with third-party tool chains to program/debug the target. This mode is selected when you press and release the mode switch for less than two seconds.

The amber status LED stays ON if the KitProg2 is in PPCOM interface mode and it turns off at a rate of 1 Hz if the KitProg2 is in CMSIS-DAP/Mass Storage Programming mode.

Refer to Switching between KitProg2 Modes on page 7 to understand the behavior of the status LED.

Note: PPCOM is the abbreviation for PSoC Programmer Component Object Module. It is the programming/debugging interface provided in Cypress’ PSoC Creator and PSoC Programmer.

3.1 KitProg2 Driver Installation

The kit is powered from a computer over the USB interface. It enumerates as a composite device. The USB drivers required for enumeration are part of the kit installer and should be appropriately installed for correct operation.

Figure 3-1 shows the driver installation in PPCOM programming mode and Figure 3-2 shows the driver installation in CMSIS-DAP programming mode.

Figure 3-1. KitProg2 Driver Installation in PPCOM Programming Mode
3.2 Programming Using PSoC Creator

1. Connect a USB cable to the USB connector, J6, as shown in Figure 3-3. 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.

2. Verify the KitProg2 is in PPCOM mode (Amber LED is ON). See Switching between KitProg2 Modes on page 7 for details.

3. Launch PSoC Creator from Start > All Programs > Cypress > PSoC Creator <version> > PSoC Creator <version>.

4. Select File > Open > Project/Workspace in PSoC Creator and browse to the desired project. You may also select File > Code Example to browse through a library of existing example projects. See PSoC Creator User Guide for details.
5. Select **Build > Build Project** or press **[Shift] [F6]** to build the project, as shown in **Figure 3-4**.

Figure 3-4. Build an Example Project

6. If there are no errors during build, program the PSoC 4000S device on the kit by choosing **Debug > Program** or pressing **[Ctrl] [F5]**, as shown in **Figure 3-5**.

Figure 3-5. Programming Device from PSoC Creator
7. If the device is already acquired, programming will complete automatically – the result will appear in the PSoC Creator status bar at the bottom left of the screen. If the device is yet to be acquired, the Select Debug Target window will appear. Select KitProg2/<serial number> and click the Port Acquire button, as shown in Figure 3-6.

Figure 3-6. Port Acquire

8. After the device is acquired, it is shown in a tree structure below the KitProg2/<serial number>. Click the Connect button and then OK to exit the window and start programming, as shown in Figure 3-7.

Figure 3-7. Connect Device from PSoC Creator and Program
3.3 Debugging Using PSoC Creator

To debug the project using PSoC Creator, follow steps 1 to 5 from section Programming Using PSoC Creator on page 13. Then, follow these steps:

1. Click the Debug icon or press [F5], as shown in Figure 3-8. Alternatively, you can select Debug > Debug. This programs the device and starts the debugger.

Figure 3-8. 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 3-9.

Figure 3-9. Using the PSoC Creator Debugger
3.4 Programming Using PSoC Programmer

PSoC Programmer (3.24.2 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 PSoC 4000S devices, the .hex file is located at: <Project Directory>\<Project Name.cydsn>\CortexM0p\<Compiler Name and Version>\<Debug> or <Release>\<Project Name.hex>.
3. Click the KitProg2/<serial number> in the Port Selection list to connect the kit to your computer.
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, PSoC Programmer will display a device mismatch error and terminate programming. Ensure that you have selected the correct .hex file.
5. When programming is completed successfully, indicated by a PASS message on the status bar, the kit is ready for use. Close PSoC Programmer.

3.5 Updating the KitProg2 Firmware

The KitProg2 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 3-10.

Figure 3-10. KitProg2 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 3-11.

To update the KitProg2, go to the **Utilities** tab on PSoC Programmer and click **Upgrade Firmware**, as shown in Figure 3-11.

Figure 3-11. Upgrade Firmware in PSoC Programmer
On successful upgrade, the Actions and Results window displays the firmware update message with the KitProg2 version, as shown in Figure 3-12.

Figure 3-12. Firmware Updated in PSoC Programmer
4. CMSIS-DAP Mode Programming and Debugging

This section explains how the CMSIS-DAP programmer/debugger is integrated onto the PSoC development kits. The KitProg2 supports the development kits listed in Table 2-1. This section uses the PSoC 4 S-Series Pioneer Kit as an example.

The KitProg2 programmer and debugger (PPCOM), KitProg2 USB-I2C Bridge, and KitProg2 USB-UART Bridge functionalities are not available in this configuration.

CMSIS-DAP is an alternative programming/debugging interface in which the KitProg2 can be used with third-party tool chains to program/debug the target. This mode is selected when you press and release the mode switch for less than two seconds.

The amber status LED stays ON if the KitProg2 is in PPCOM interface mode and it turns off at a rate of 1 Hz if the KitProg2 is in CMSIS-DAP/Mass Storage Programming mode.

Refer to Switching between KitProg2 Modes on page 7 to understand the behavior of status LED.

4.1 Programming and Debugging using μVision

CMSIS-DAP mode can be used for programming from many third party IDEs. As an example, the steps to program using μVision are shown below.

To use KitProg2 in CMSIS-DAP mode to program using μVision, do the following:

1. Connect the kit to PC and enter into CMSIS-DAP mode. To enter into CMSIS-DAP mode, refer Switching between KitProg2 Modes on page 7.

   Note: The KitProg2 should be in the CMSIS-DAP programming mode (press mode switch SW3 on CY8CKIT-041-40XX for less than two seconds to change modes). In this mode, the amber LED switches OFF.

   Note: in KitProg2 1.04 version, this LED will show breathing effect in Mass Storage or CMSIS-DAP mode.
2. Open the project in PSoC Creator and build the project. Right-click the project in the Workspace Explorer and select **Export to IDE**, as shown in Figure 4-1.

Figure 4-1. Export to IDE

3. Select the IDE as shown in Figure 4-2.

Figure 4-2. Select IDE
4. Select the tool chain as shown in Figure 4-3.

Figure 4-3. Select Tool Chain

5. Select the project files as shown in Figure 4-4. The next window that appears is Review export details. Click Export.

Figure 4-4. Select Project Files

6. Open the µVision project in the µVision IDE tool. Keil µVision version 4.74.0.22 is used in this illustration. Build the project in µVision.
7. Open the PSoC Programmer installation folder and look for the path 
   Programmer\3rd_Party_Configuration_Files\CY8C40xx\Prog_Algorithm.
   **Note:** The default path location of PSoC programmer installation folder is
   C:\Program Files (x86)\Cypress\Programmer.

8. Copy the CY8C40xx.FLM file to the \ARM\Flash folder inside the installation directory of Keil
   µVision. This will typically look similar to: C:\Keil\ARM\Flash\

9. Select **Project > Options for Target “project_name”**.

   ![Figure 4-5. Option for Target in µVision](image)

10. Make sure to set appropriate values for ROM and RAM areas of the target device as shown in
    **Figure 4-6**.

    ![Figure 4-6. Target Tab](image)
11. Select the **Utilities** tab and select **CMSIS-DAP Debugger**, as shown in Figure 4-7.

**Figure 4-7. Utilities Tab in µVision.**

12. A programming algorithm must be added in the IDE to program PSoC 4. Click the **Settings** option. The **Cortex-M Target Driver Setup** window will open as shown in Figure 4-8. Go to the **Flash Download** tab and click the **Add** button as shown in Figure 4-9. Select the option corresponding to PSoC 4 in the programming algorithm.

**Figure 4-8. Cortex-M Target Driver Setup**
13. Select the **Debug** tab. Make sure that KitProg2 is listed in the CMSIS-DAP – JTAG/SW adapter. Select the port as **SW**; the SW Device field will show **IDcode** as **0x0BB11477**. Set the clock frequency to **1MHz** and the Reset option to **VECTRESET** as shown in **Figure 4-10**. Click **OK** and program the device using the option **Flash > Download**.

**Figure 4-10.** Debug Tab in Target Driver Setup
14. For debugging, go to the **Debug** tab and select the CMSIS-DAP Debugger as shown in Figure 4-11.

![Debug Tab in Options for Target](image)

**Figure 4-11. Debug Tab in Options for Target**

15. Selecting **Debug > Start/Stop Debug Session** will start the debug session. Note that the green LED on the CY8CKIT-041-40XX blinks at an 8-Hz rate during the debug session.

![Debug Session in µVision](image)

**Figure 4-12. Debug Session in µVision**
5. **Mass Storage Programmer**

The KitProg2 can act as a USB Mass Storage Programmer. CMSIS-DAP and Mass Storage modes are the same modes though the method of programming is different. This is an alternative configuration of KitProg2. The KitProg2 programmer and debugger (PPCOM), KitProg2 USB-I2C Bridge, KitProg2 USB-SPI Bridge, and KitProg2 USB-UART Bridge functionalities are not available in this configuration.

### 5.1 Enter or Exit the Mass Storage Programmer Mode

Follow these steps to enter or exit the Mass Storage Programmer mode of KitProg2:

1. Connect the kit to the PC. Ensure that the amber status LED is ON and not blinking. See section 10.1 *KitProg2 Status LED Indication on page 71* for details on the status LED indications. Amber status LED ON indicates that the current configuration is KitProg2 Programmer and Debugger.

2. Press and release the mode switch within two seconds. The KitProg2 reenumerates as a Mass Storage Programmer/CMSIS-DAP programmer if the previous configuration is PPCOM programmer and debugger. The amber status LED stops glowing indicating that the kit is in Mass Storage Programming/CMSIS-DAP programming mode.

3. The KitProg2 remains in the selected mode until you change the mode manually using the mode switch. To exit the Mass Storage Programming mode, press and release the mode switch within two seconds. The amber status LED is continuously ON, indicating that the KitProg2 is in the KitProg2 Programmer and Debugger mode.

### 5.2 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 **Enter or Exit the Mass Storage Programmer Mode on page 27**. The KitProg2 is visible as a removable disk drive in the file explorer of the PC, as shown in *Figure 5-1*.

*Figure 5-1. KitProg2 Emulated as Mass Storage Device*
2. Open the KitProg2 drive to view the STATUS.TXT file, as shown in Figure 5-2. Note that the file extension .TXT is visible for the file only if it is enabled in your PC settings. The STATUS.TXT file shows the current status of the Mass Storage Programmer.

Figure 5-2. STATUS.TXT in the KitProg2 Drive

3. Copy any PSoC 4000S device based project .hex file to the KitProg2 drive to begin programming. Alternatively, you can drag and drop the .hex file onto the drive. The .hex file for a PSoC Creator project is available in the following path:

   <Project Directory>\<Project Name.cydsn>\CortexM0p\<Compiler Name and Version>\<Debug> or <Release>\<Project Name.hex>

Figure 5-3. Copy .hex File to KitProg2 Drive

4. The green status LED on the kit blinks during the programming operation. It stays ON after the programming operation completes successfully and the KitProg2 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 KitProg2 drive.

   Note: For prototyping kits the amber LED blinks during the programming in Mass Storage mode. For more information on switching of modes in prototyping kits, refer Switching between KitProg2 Modes on page 7.

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

Figure 5-4. Status Displayed in KitProg2 Drive after Programming
5.3 Frequently Asked Questions on KitProg2 Mass Storage Programmer

1. What are the Cypress kits supported by the KitProg2 Mass Storage Programmer?
   The KitProg2 Mass Storage Programmer supports all the kits mentioned in Table 2-1 on page 10.
   Note: Newer kits which are not listed in the table can be explored in Cypress website http://www.cypress.com.

2. What are the operating systems supported by KitProg2 Mass Storage Programmer?
   The KitProg2 Mass Storage Programmer works on Microsoft Windows (XP or later) and Apple Mac operating systems (OSX or later). The KitProg2 Mass Storage Programmer is currently not supported on Linux operating systems.

3. What happens if I copy an incorrect .hex file to the KitProg2 drive?
   If you copy a .hex file with invalid data (such as incorrect silicon ID and incorrect checksum), the KitProg2 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 KitProg2 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 file other than the ones specified above, and the file size does not exceed the KitProg2 drive size, the file will be visible in the KitProg2 drive until the drive is removed from the PC. Note that the file is not actually copied to the KitProg2 drive. Delete these files before attempting to program a new .hex file.

4. Why does my operating system display the “Disk Not Ejected Properly” pop-up after every programming operation in KitProg2 Mass Storage Programmer mode?
   The KitProg2 Mass Storage Programmer temporarily ejects two seconds after the programming operation. This can also cause the file explorer window of the KitProg2 drive to close after programming operation in some operating systems.

5. Is it possible to program an external PSoC other than the one on the kit using the KitProg2 Mass Storage Programmer?
   Yes. You need to remove several onboard zero-ohm resistors to disconnect the onboard target device. See the documentation for the kit that you are using for details.

6. Can I use .hex files generated by any other IDE other than PSoC Creator to program the PSoC 4000S using KitProg2 Mass Storage Programmer?
   Yes. You can use the .hex file generated by an external IDE such as Eclipse, IAR, or Keil µVision which supports PSoC 4 devices, to program using the KitProg2 Mass Storage Programmer.

7. Why does the programming time for different files vary?
   The KitProg2 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 30 seconds.
6. USB-UART Bridge

The KitProg2 can act as a USB-UART Bridge. This feature of the KitProg2 is useful to send and receive data between the Cypress device on the kit and a PC. For example, in the PSoC 4 S-Series Pioneer Kit, the KitProg2 USB-UART can be used to print debug messages on COM terminal software running on the PC.

This section demonstrates a method to create a PSoC 4 code example, which communicates with COM terminal software using the KitProg2 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 an alternative terminal software such as PuTTY.

1. Create a new PSoC 4000S 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 target hardware (kit) for this project. This example uses CY8CKIT-041-40XX (PSoC 4000S device) as the target hardware. Ensure that the Select project template option is set to 'Empty schematic'. This example uses PSoC 4000S as the target device and PSoC 4 S-Series Pioneer Kit as the target board.

Figure 6-1. Create New Project in PSoC Creator
2. Drag and drop a UART (SCB mode) Component from the Component Catalog (see Figure 6-2) to the TopDesign. The Component Catalog is located along the right 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 6-3.

Figure 6-2. UART Component in Component Catalog

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

Figure 6-4. UART Configuration Tab Window

![UART Configuration Tab Window](image)

Figure 6-5. UART Basic Tab Window

![UART Basic Tab Window](image)
Figure 6-6. UART Advanced Tab Configuration Window
4. Select P0\[4\] for UART RX and P0\[5\] for UART TX in the Pins tab of <Project_Name>.cydwr, as shown in Figure 6-7. This can be opened by double clicking on “Pins” under the “Design Wide Resources”. The <Project_Name>.cydwr file can be found in the Workspace Explorer window, which is located along the left of the PSoC Creator window. Note that these pins are for the USB-UART interface on the PSoC 4 S-Series Pioneer Kit. If you are using a different kit, refer to the respective kit guide for the appropriate pins.

**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. Here, the SCB implementation of the UART is used, 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.

Figure 6-7. UART Pin Assignment
5. Place the following code in the main.c file. The code echoes any data received through the UART.

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

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 4000S using KitProg2.

```c
#include <project.h>
int main()
{
    uint8 ch;
    /* Start SCB UART TX+RX operation */
    UART_1_Start();
    /* Transmit String through UART TX Line */
    UART_1_UartPutString("CY8CKIT-041 USB-UART");
    for(;;)
    {
        /* Get received character or zero if nothing has been received yet */
        ch = UART_1_UartGetChar();
        if(0u != ch)
        {
            /* Send the data through UART. This function is blocking and waits until there is an entry into the TX FIFO. */
            UART_1_UartPutChar(ch);
        }
    }
}
```
To communicate with the PSoC 4000S device from the terminal software, follow this procedure:

1. Connect the USB Micro-B cable to J6. The kit enumerates as a **KitProg2 USB-UART**, and is available in the **Device Manager** under **Ports (COM & LPT)**. A communication port is assigned to the **KitProg2 USB-UART**, as shown in **Figure 6-8**.

**Figure 6-8. KitProg2 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 6-9**. For PuTTY, double-click the PuTTY application and select **Serial** under **Category**.

**Figure 6-9. 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 KitProg2 USB-UART) in Connect using and click OK, as shown in Figure 6-10.

In PuTTY, enter the COMx in Serial line to connect to. This example uses COM43.

Figure 6-10. 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 6-11). Ensure that the settings are identical to the UART settings configured for the PSoC 4000S device.

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

Figure 6-11. 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 6-12. In PuTTY, select **Force on** under **Terminal > Line discipline options** to display the typed characters on PuTTY, as shown in Figure 6-12.

Figure 6-12. Enable Echo of Typed Characters in HyperTerminal and PuTTY

![HyperTerminal and PuTTY](image)

6. In PuTTY, click **Session** and select **Serial** under **Connection type**. The **Serial line** shows the communication port (COM43) and **Speed** shows the baud rate selected. Click **Open** to start the communication, as shown in Figure 6-13.

Figure 6-13. Opening Port in PuTTY
7. The COM terminal software displays both the typed data and the echoed data from the PSoC 4000S UART, as shown in Figure 6-14.

**Note:** The string “CY8CKIT-041 USB-UART” is transmitted when the kit is powered up or reset. If you open the terminal window after the kit has been plugged in, you will not see this message. Press the Reset button on the kit to see the message.

Figure 6-14. Data Displayed on HyperTerminal and PuTTY
7. USB-I2C Bridge

The KitProg2 serves as a USB-I2C Bridge that can be used to communicate with USB-I2C software running on a PC. For example, the KitProg2 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 on page 10. This section uses the PSoC 4 S-Series Pioneer Kit as an example to demonstrate the KitProg2 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 4000S device.

Note: For information on how to use the KitProg2 USB-I2C Bridge to tune the CapSense Component, refer to the Manual Tuning Process section in AN85951 - PSoC 4 CapSense Design Guide.

1. Create a new PSoC 4000S project in PSoC Creator. Select a specific location for your project and name the project as desired. You must select the appropriate target hardware (kit) for this project as shown in Figure 7-1. This example uses CY8CKIT-041-40XX (PSoC 4000S device) as the target hardware. Ensure that the Select project template option is set to ‘Empty schematic’ as shown in Figure 7-2. Create the workspace and project name as shown in Figure 7-3.

Figure 7-1. Create New Project in PSoC Creator
Figure 7-2. Select Empty Schematic

Figure 7-3. Create Workspace
2. Drag and drop an EZI2C Slave (SCB mode) Component from the Component Catalog (see Figure 7-4) to the TopDesign. The Component Catalog is located along the right 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 7-5.

Figure 7-4. EZI2C Slave Component in Component Catalog

Figure 7-5. Open EZI2C Slave Configuration Window
3. Configure the EZI2C Slave Component as shown in Figure 7-6 and Figure 7-7; then, click OK.

Figure 7-6. Configuration Tab
Figure 7-7. EZI2C Slave Basic and Advanced Tabs
4. Select pin P3[0] for the I2C SCL and pin P3[1] for the I2C SDA in the **Pins** tab of `<Project_Name>.cydwr`, as shown in **Figure 7-8**. This can be opened by double clicking on “Pins” under the “Design Wide Resources”. The `<Project_Name>.cydwr` file can be found in the Workspace Explorer window, which is located along the left of the PSoC Creator window. Note that these are the pins for the USB-I2C interface on the PSoC 4 S-Series Pioneer Kit. If you are using a different kit, refer to the respective kit guide for the appropriate pins.

**Figure 7-8. Select Pins in cydwr**
5. Place the following code in the main.c file. The code will enable the PSoC 4000S 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 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] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E};

    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 4000S using KitProg.

7. Open BCP from Start > All Programs > Cypress > Bridge Control Panel <version> > Bridge Control Panel <version>.
8. Select KitProg2/<serial number> under Connected I2C/SPI/RX8 Ports, as shown in Figure 7-9.

9. If the KitProg2 firmware is not the most recent version the connection will not work. See Updating the KitProg2 Firmware on page 17 to update the KitProg2 firmware.
10. Open **Protocol Configuration** from the Tools menu and select the appropriate **I2C Speed**, as shown in Figure 7-10. Ensure that the I2C speed is the same as the one configured in the EZI2C Slave Component. Click **OK** to close the window.

Figure 7-10. Opening Protocol Configuration Window in BCP
11. The buffer in the EZI2C slave is initialized with the data 0x01, 0x02, 0x03, 0x04, and 0x05 after power-on or reset. Send the Read command from the BCP to read the initial data with the slave address 0x08 (first command in Figure 7-11). The EZI2C Slave requires an additional write to be sent from the BCP to set the offset address from/to where the data bytes are read/written. Use a write command to set the read offset and then issue the read command. In this example, the offset is set to ‘0’ so the five data bytes are read starting from the beginning of the buffer. The log shows the initial data. After reading, transfer five bytes of data from the BCP to the I2C device with slave address 0x08. The five bytes sent are 0x11, 0x22, 0x33, 0x44, and 0xAA as shown in the second command of Figure 7-11. Type the command shown in Figure 7-11 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 failed.

**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.

Figure 7-11. Enter Commands in BCP
12. From the BCP, read back the five bytes of data just written from the I2C slave device with slave address 0x08. The log shows if the transaction was successful, as shown in Figure 7-12.

Figure 7-12. Read Data Bytes from BCP

Refer to Help > Help Contents in the BCP or press [F1] for more information on the I2C commands. Refer to the EZI2C component datasheet for more information on the EZI2C protocol.
8. USB-SPI Bridge

The KitProg2 serves as a USB-SPI bridge that can be used to communicate with USB-SPI software running on a PC. This section uses the PSoC 6 BLE Kit as an example to demonstrate the KitProg2 USB-SPI Bridge functionality. The following steps describe how to use the USB-SPI bridge, which can communicate between the Bridge Control Panel (BCP) software and the PSoC 6 BLE device.

1. Create a new PSoC 6 BLE project in PSoC Creator. Select a specific location for your project and name the project as desired. You must select the appropriate target hardware (kit) for this project as shown in Figure 8-1. This example uses CY8CKIT-062 kit (PSoC 6 BLE device) as the target hardware. Ensure that the Select project template option is set to 'Empty schematic' as shown in Figure 8-2. Create the workspace and project name as shown in Figure 8-3.

Figure 8-1. Create New Project in PSoC Creator
Figure 8-2. Select Empty Schematic

Figure 8-3. Create Workspace
2. Drag and drop an SPI component and digital output pin from the component catalog to the Top-Design. The component catalog is located along the right of the PSoC creator window by default.

3. Double-click the digital output pin to configure the pin. Set the pin name as Pin_LED and uncheck the hardware connection as shown in Figure 8-4.

Figure 8-4. Configuring the output pin

4. To configure the SPI component, double-click or right-click the SPI Component and select Configure, as shown in Figure 8-5.

Figure 8-5. SPI Component in Component Catalog and configuring SPI Component
5. Configure the SPI Slave Component as shown in Figure 8-6 then click OK.

Figure 8-6. Configuring SPI Component

6. Select pin P12[0] for SPI_MOSI, P12[1] for SPI_MISO, P12[2] for SPI_SCLK, P12[4] for SPI_SS and P0[3] for the digital output pin Pin_LED in the Pins tab of <Project_Name>.cydwr, as shown in Figure 8-7. The <Project_Name>.cydwr file is available in the Workspace Explorer window, which is located along the left of the PSoC Creator window by default. Double-click on the file to open it. Note that these are the pins for the USB-SPI interface on the PSoC 6 BLE Pioneer Kit. If you are using a different kit, refer to the respective kit guide for the appropriate pins.

Figure 8-7. Select Pins for SPI interface and the output pin
7. Place the following code in the *main_cm0p.c* file. The code will enable the PSoC 6 BLE device to communicate with the BCP application using the SPI Slave interface. The function of the code is to control LED ON/OFF (LED 5) on the CY8CKIT-062 kit based on the command received from the BCP. A command word of 0x00 turns OFF the LED and 0x01 turns ON the LED.

**Note:** The *main_cm0p.c* file can be found on the Workspace Explorer window, which is located along the left of the PSoC Creator window by default. Double click on the file to open it.

```c
#include <project.h>

#define BUFFER_SIZE 1
#define LED_ON 0
#define LED_OFF 1

int main(void)
{
    __enable_irq(); /* Enable global interrupts. */
    /* Enable CM4. CY_CORTEX_M4_APPL_ADDR must be updated if CM4 memory layout is changed. */
    Cy_SysEnableCM4(CY_CORTEX_M4_APPL_ADDR);
    int8 RxBuffer[BUFFER_SIZE];
    RxBuffer[0]=0;
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    SPI_P6_Start();
    for (;;)
    {
        while(SPI_P6_IsBusBusy());

        SPI_P6_ReadArray((void*)RxBuffer,BUFFER_SIZE);
        SPI_P6_ClearRxFifo();

        switch(RxBuffer[0])
        {
        case 0x00:
            Cy_GPIO_Write(Pin_LED_0,LED_OFF); // Turn the LED OFF
            break;
        case 0x01:
            Cy_GPIO_Write(Pin_LED_0,LED_ON);  // Turn the LED ON
            break;
        default:
            break;
        }
    }
}
```
8. 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 6 BLE using KitProg.

9. Open BCP from **Start > All Programs > Cypress > Bridge Control Panel <version> > Bridge Control Panel <version>**.

10. Select **KitProg2/<serial number>** under **Connected I2C/SPI/RX8 Ports**, as shown in Figure 8-8. Select the protocol configuration and type in the command as shown in Figure 8-8. Refer BCP help topic for SPI command formats. Click on the “Send” radio button to send the command from BCP.

**Figure 8-8. Connecting to Kitprog2 USB-SPI Bridge in BCP**

**Expected Output:** Sending a command word of 0x01, turns LED5 ON and a command word of 0x00 turns LED5 OFF.
9. Developing Applications for PSoC 5LP

The KitProg2 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 KitProg2: Custom (Bootloadable) Application and Normal. Custom application projects can be bootloaded 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 S-Series Pioneer Kit, this header is marked J11. See the respective kit guide for more information on the PSoC 5LP programming header. Go to section 9.2 Building a Normal Project for PSoC 5LP on page 67 for details on creating a normal project for PSoC 5LP.

Custom applications for KitProg2 are developed using the concept of dual image bootloading. To learn more about the bootloading concept, refer to the application note AN73854 – PSoC 3, PSoC 4, and PSoC 5LP: Introduction to Bootloaders.

The following sections provide step by step directions to build a Custom Application and a Normal project for KitProg2.

9.1 Building a Custom Bootloadable Application for KitProg2

Custom applications developed for the PSoC 5LP are programmed to the kit using the KitProg2 bootloader. Therefore, a bootloadable project should first be created. An example of a custom application is provided as a compressed archive KitProg2_Custom_App.zip in the PSoC Programmer installation folder under Examples\Misc\KitProg2_Custom_App\ This example demonstrates an LED with a breathing effect and also has the feature to switch between KitProg2 and custom application using the mode switch.
To build a custom application for the PSoC 5LP, follow this procedure:

1. In PSoC Creator, choose **File > New > Project** and select **Target device**; select `<Launch Device Selector...>` from the drop-down list as shown in **Figure 9-1**

   **Note:** The custom application must provide a means to switch back to KitProg2 mode, otherwise the programming capability will be lost unless KitProg2 firmware is re-bootoladed.

   **Figure 9-1. Open New Project in PSoC Creator**

2. Select **CY8C5868LTI-LP039**, as shown in **Figure 9-2**. Click **OK** and click **Next**.

   **Note:** In PSoC Creator 3.1 or earlier, you must either set the **Application Type** as **Bootloadable** in the New Project window under the Advanced section, or you can change it after project creation by selecting **Project > Build Settings** and clicking `<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 and later versions automatically recognizes the application type from the TopDesign schematic.

   **Figure 9-2. Select Device in PSoC Creator**
3. Choose **Empty schematic** in the **Select project template** dialog, as shown in **Figure 9-3**. Click **Next**.

Figure 9-3. Select Empty Schematic

![Select Empty Schematic](image)

4. In the **Create Project** dialog, choose the workspace name, location, and project name (**Figure 9-4**). Click **Finish**.

Figure 9-4. Choose Name and Location

![Create Project](image)
5. Navigate to the Schematic view and drag and drop a Bootloadable Component (see Figure 9-5) on the TopDesign.

Figure 9-5. Bootloadable Component in Component Catalog

6. All custom applications developed for the PSoC 5LP should be based on the KitProg2 bootloader .hex file, which is programmed onto the kit. Therefore, you need to provide the location of the bootloader .hex file inside the custom application project.

   The bootloader .hex file is included in the compressed archive KitProg2_Custom_App.zip in the PSoC Programmer installation folder under Examples\Misc\KitProg2_Custom_App\. They are also present in Kit installation directory at <Install_Directory>\<Kit_Name>\<version>\Firmware\Programmer\KitProg2_Bootloader\. Extract the Bootloader folder from the archive.
7. 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 9-6. Select the **KitProg2_Bootloader.hex** (see Figure 9-7) extracted in step 6 and click **Open**.

**Note:** The user should copy the `.hex` and `.elf` files into their custom project folder and refer to them from that location. The **KitProg2_Bootloader.elf** is selected automatically if it is available with the same name in the same path. Ensure that both `.hex` and `.elf` files exist in the same folder with the same name.

Figure 9-6. Configuration Window of Bootloadable Component

Figure 9-7. Select KitProg2 Bootloader Hex File.
8. In the General tab, check the Manual application image placement checkbox and set the Placement address as '0x00003200', as shown in Figure 9-8.

Figure 9-8. Bootloadable Component - General Tab

9. Develop your custom project. An example of a custom application is provided as a compressed archive KitProg2_Custom_App.zip in the PSoC Programmer installation folder under Examples\Misc\KitProg2_Custom_App\.

This example demonstrates an LED with a breathing effect and also has the feature to switch between KitProg2 and custom application using the mode switch.
10. Ensure that the `<project name>.cydwr System` setting of the bootloadable project and the `KitProg2_Bootloader` project is the same. Figure 9-9 shows the `KitProg2_Bootloader.cydwr System` settings.

Figure 9-9. KitProg2 Bootloader System Settings

11. Build the project in PSoC Creator by choosing `Build > Build Project` or pressing `[Shift] [F6]`.

12. 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 9-10. Alternatively, the Bootloader Host tool can be accessed by going to `Start > All Programs > Cypress > PSoC Creator 3.3 > Bootloader Host`, as shown in Figure 9-11.

Figure 9-10. Open Bootloader Host Tool in PSoC Creator
13. Press and hold the mode switch while connecting the kit to the computer. If the switch is pressed for more than 100 ms, the PSoC 5LP enters the bootloader. The amber status LED will start blinking to indicate that PSoC 5LP entered bootloader mode.

14. 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 04B4, PID as F146, and click OK, as shown in Figure 9-12.

Figure 9-12. Port Filters Tab in Bootloader Host Tool
15. In the Bootloader Host tool, click the **Open File** button (Figure 9-13) to browse to the location of the bootloadable file (*.cyacd), as shown in Figure 9-14. This file is present in the project directory. The .cyacd file is available in the following path:

`<Project Directory>\<Project Name>.cydsn\CortexM3\<Compiler Name and Version>\<Debug> or <Release>\<Project Name>_2.cyacd`

Note that there are 2 cyacd files in the project directory. `<Project_Name>_1.cyacd` is the first application (in this case the KitProg2 application) and `<Project_Name>_2.cyacd` is the second application (in this case the custom application). Therefore, we want to select `<Project_Name>_2.cyacd` to bootload the custom application.

The “Active application” selection shown in Figure 9-13 is used to determine which application will start once bootloading has completed. In this case, since we have selected Image 2, the custom application will run once bootloading finishes.

---

**Figure 9-13.** Open Bootloadable File in Bootloader Host Tool

![Bootloader Host tool interface with Open File button highlighted](image1.png)

**Figure 9-14.** Select `<Project_Name>_2>.cyacd` File from Bootloader Host Tool

![File selection interface showing cyacd file](image2.png)
16. Select the **USB Human Interface Device (04B4_F146) - USB** in the **Ports** list and click the **Program** button (Figure 9-13) in the Bootloader Host tool to program the device.

17. 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 section A.1 Pin Assignments on page 78 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 4000S in CY8CKIT-041-40XX. Refer to the respective kit schematics before allocating the pins.
- When a custom bootloadable project is the active application on the PSoC 5LP, the initial capability of the PSoC 5LP to act as a programmer, USB-UART Bridge or USB-I2C Bridge or USB-SPI Bridge is not available. To recover this functionality, switch the active application back to the KitProg2 image.
- The status LEDs do not function unless they are implemented and used by the custom project.

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

### 9.2 Building a Normal Project for PSoC 5LP

A normal project is a new project created for the PSoC 5LP device on the PSoC 4 S-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 or USB-SPI Bridge functionality, reprogram the PSoC 5LP device with the factory-set *KitProg2.hex* file, which is shipped with the kit installer.

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

The *KitProg2.hex* file is available at the following location:

<Install_Directory><Name_of_the_Kit><version>Firmware\Programmer\KitProg2\KitProg2.hex
To build a normal project for the PSoC 5LP, follow these steps:
1. In PSoC Creator, choose **File > New > Project** and select **Target device**; select **<Launch Device Selector…>** from the drop-down list and click **Next**. See Figure 9-15.

Figure 9-15. Create New Project in PSoC Creator
2. Select **CY8C5868LTI-LP039**, as shown in Figure 9-16. Click **OK**.

Figure 9-16. Select the Device

3. Develop your custom project.
4. Build the project in PSoC Creator by choosing **Build > Build Project** or pressing **[Shift] [F6]**.
5. Connect the MiniProg3 to the onboard PSoC 5LP Programming header J11 (which needs to be populated).
6. 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), select the device and click **Connect** to acquire and program.

If the PSoC device is not available, click the **Port Setting** button. Set **Active Protocol** to ‘SWD’, **Acquire Mode** to ‘Power Cycle’, and **Connector** to either 5 pin or 10 pin depending on the kit. Then, click the **Port Acquire** button for the PSoC 5LP device to appear. See **Figure 9-17**.

**Figure 9-17. Select Debug Target for Programming**

---

**Notes:**

- The 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 Assignments on page 78** 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 4000S in CY8CKIT-041-40XX. 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 or USB-SPI Bridge is not available.
- The status LEDs do not function unless they are implemented and used by the custom project.
10. Troubleshooting the KitProg2

This section explains the methods to troubleshoot the KitProg2 and recover the KitProg2 firmware if you modified it.

10.1 KitProg2 Status LED Indication

The KitProg2 status LEDs on the development kit indicate the status of the KitProg2 operation using different blink rates. Table 10-1 shows the KitProg2 LED indication and the corresponding status of the KitProg2.

Table 10-1. Meaning of KitProg2 LED Indications

<table>
<thead>
<tr>
<th>User Indication</th>
<th>Scenario</th>
<th>Action Required by User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amber LED</td>
<td>Green LED</td>
<td>Red LED</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>ON</td>
<td>Blinking</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>Frequency=8 Hz</td>
<td></td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>
### Troubleshooting the KitProg2

Table 10-1. Meaning of KitProg2 LED Indications (continued)

<table>
<thead>
<tr>
<th>User Indication</th>
<th>Scenario</th>
<th>Action Required by User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amber LED</td>
<td>Green LED</td>
<td>Red LED</td>
</tr>
<tr>
<td>OFF</td>
<td>8 Hz</td>
<td>OFF</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>Blinking</td>
<td>Frequency= 1 Hz</td>
<td>OFF</td>
</tr>
<tr>
<td>OFF</td>
<td>8 Hz (Blinking)</td>
<td>User defined</td>
</tr>
<tr>
<td>Blinking</td>
<td>Frequency= 15.0 Hz</td>
<td>OFF</td>
</tr>
</tbody>
</table>

**Note:** There is another category of Kits known as prototyping Kits which have single button and a single LED based KitProg2 System. To enter into various modes below points should be noted:

1. When the kit is in KitProg2 mode, the Amber LED will be on.
2. When the kit is in Mass Storage/CMSIS DAP mode, the Amber LED is turned off. To reach this mode, press and release the mode switch provided on the prototyping kit in less than 2 seconds.

**Note:** In 1.04 KitProg2 version, Amber LED shows breathing effect in Mass Storage or CMSIS DAP Mode.
3. When the kit is in Bootloader mode, the Amber LED shows a blinking effect. To reach this mode, hold down the mode switch and insert the kit into a USB port.

   **Note:** The various modes and their representation in terms of LEDs with different colors are not supported in these categories of kits eg. CY8CIT-146 PSoC 4200DS Prototyping Kit.

   **Note:** The BCP software cannot connect to the KitProg2, if the KitProg2 firmware version is outdated. See Updating the KitProg2 Firmware on page 17 to update the KitProg2 firmware.

   **Note:** The programming/debugging function and USB-I2C Bridge/USB-SPI Bridge function of the KitProg2 are mutually exclusive functions and cannot be used together. Therefore, to use one function the other function should be disconnected. For instance, to program the device while using the USB-I2C Bridge/USB-SPI Bridge in BCP, either close BCP or disconnect the USB-I2C Bridge/USB-SPI Bridge. The USB-UART Bridge function of the KitProg2, however, can run in parallel to both programming/debugging and USB-I2C Bridge/USB-SPI Bridge functions. USB-I2C Bridge and USB-SPI Bridge functionality cannot be used simultaneously.

### 10.2 PSoC 5LP Factory Program Restore Instructions

#### 10.2.1 PSoC 5LP is Programmed with a Custom Application

Reprogramming the PSoC 5LP device with a new flash image in the KitProg2 area of flash (i.e. application area 1) will forfeit the ability to use the PSoC 5LP device as a programmer/debugger for the kit. See section 10.2.1.1 Restore PSoC 5LP Factory Program Using PSoC Programmer on page 74, for details on restoring the KitProg2.

   **Note:** This method cannot be used to recover the KitProg2 if the PSoC 5LP was reprogrammed using a MiniProg3. See section 10.2.2 Restore PSoC 5LP KitProg2 using MiniProg3 on page 76 if you want to recover the KitProg2 functionality using a MiniProg3.
10.2.1.1 Restore PSoC 5LP Factory Program Using PSoC Programmer

1. Launch PSoC Programmer from Start > Cypress > PSoC Programmer <version> > PSoC Programmer <version>.

2. Configure the Kit in bootloader mode. To do this, while pressing the mode switch, connect the Kit to the computer using the included USB cable (USB Standard-A to Micro-B). This puts the PSoC 5LP into bootloader mode, which is indicated by the blinking amber status LED.

3. The following message appears in the PSoC Programmer Results window (see Figure 10-1): “KitProg2 Bootloader devices are detected”.

Figure 10-1. PSoC Programmer Results Window
4. Switch to the Utilities tab in PSoC Programmer and click the Upgrade Firmware button, as shown in Figure 10-2. Unplug all other PSoC programmers (such as MiniProg3 and DVKProg) from the PC before clicking the Upgrade Firmware button.

**Figure 10-2. Upgrade Firmware**

![Upgrade Firmware](image)

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 10-3.

**Figure 10-3. Firmware Update Completed**

![Firmware Update Completed](image)

6. The factory program is now successfully restored on the PSoC 5LP. It can be used as the programmer/debugger for the PSoC 4000S device.
10.2.2 Restore PSoC 5LP KitProg2 using MiniProg3

This section explains the method to reprogram the PSoC 5LP using a MiniProg3 to recover the KitProg2 functionality. This method must be used to recover the KitProg2 if the PSoC 5LP was completely reprogrammed. Note that this method of restoring KitProg2 will erase any custom applications loaded earlier.

1. Launch PSoC Programmer from Start > Cypress > PSoC Programmer <version> > PSoC Programmer <version>.

2. Connect the MiniProg3 to the PC. Connect the 5-pin or 10-pin connector (depending on the kit) of the MiniProg3 to the onboard PSoC 5LP programming header.

   Note: This header is not populated by default. You will need to populate it to connect the MiniProg3.

3. Select MiniProg3 from the Port Selection list in PSoC Programmer on your PC.

4. Using the File > Open menu or the File Load icon, load the KitProg2.hex file, which is installed with the kit software, as shown in Figure 10-4. The default location for this file is: C:\Program Files (x86)\Cypress\Programmer\KitProg2.hex.

Figure 10-4. Select the KitProg2.hex File to Program the PSoC 5LP

5. Select the Power Cycle option for Programming Mode, 5.0 V for voltage, 10p (or 5p, if applicable) for Connector, and SWD 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, and PASS is indicated on the status bar, as shown in Figure 10-5.

Figure 10-5. Firmware Programming Completed
A. Appendix

A.1 Pin Assignments

A.1.1 PSoC 5LP GPIO and Custom Application Header (J8 and J11) for CY8CKIT-041-40XX and CY8CKIT-041-41XX

<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>J8_02</td>
<td>PSoC 5LP_VDD</td>
<td>VDD</td>
<td>N.C</td>
<td>No Connect</td>
<td></td>
</tr>
<tr>
<td>J8_05</td>
<td>J8_06</td>
<td>P15[2]</td>
<td>SPI SCLK</td>
<td>P12[0]</td>
<td>I2C SCL</td>
<td></td>
</tr>
<tr>
<td>J8_07</td>
<td>J8_08</td>
<td>P15[3]</td>
<td>SPI SSEL</td>
<td>P3[0]</td>
<td>CUSTOM</td>
<td></td>
</tr>
<tr>
<td>J8_15</td>
<td>J8_16</td>
<td>P3[6]</td>
<td>CUSTOM</td>
<td>J8_16</td>
<td>GND</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>J11</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>J11_01</td>
<td>J11_02</td>
<td>P0[2]</td>
<td>CUSTOM</td>
<td>PSoC 5LP_VBUS</td>
<td>VBUS</td>
<td></td>
</tr>
<tr>
<td>J11_03</td>
<td>J11_04</td>
<td>P0[3]</td>
<td>CUSTOM</td>
<td>GND</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>J11_05</td>
<td>J11_06</td>
<td>P0[7]</td>
<td>CUSTOM</td>
<td>PSoC 5LP_XRES</td>
<td>XRES</td>
<td></td>
</tr>
<tr>
<td>J11_07</td>
<td>J11_08</td>
<td>P1[6]</td>
<td>INT_PIN</td>
<td>PSoC 5LP_SWCLK</td>
<td>SWD CLK</td>
<td></td>
</tr>
</tbody>
</table>
### A.1.2 PSoC 5LP GPIO and Custom Application Header (J16 and J11) for CY8CKIT-048

<table>
<thead>
<tr>
<th>J16 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
<th>J16 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J16_01</td>
<td>PSoC 5LP_VDD</td>
<td>VDD</td>
<td>J16_02</td>
<td>N.C</td>
<td>No Connect</td>
</tr>
<tr>
<td>J16_05</td>
<td>P15[2]</td>
<td>SPI SCLK</td>
<td>J16_06</td>
<td>P12[0]</td>
<td>I2C SCL</td>
</tr>
<tr>
<td>J16_07</td>
<td>P15[3]</td>
<td>SPI SSEL</td>
<td>J16_08</td>
<td>P3[0]</td>
<td>CUSTOM</td>
</tr>
<tr>
<td>J16_15</td>
<td>P3[6]</td>
<td>CUSTOM</td>
<td>J16_16</td>
<td>GND</td>
<td>GND</td>
</tr>
</tbody>
</table>

### J11

<table>
<thead>
<tr>
<th>J11 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
<th>J11 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J11_01</td>
<td>P0[2]</td>
<td>CUSTOM</td>
<td>J11_02</td>
<td>PSoC 5LP_VBUS</td>
<td>VBUS</td>
</tr>
<tr>
<td>J11_03</td>
<td>P0[3]</td>
<td>CUSTOM</td>
<td>J11_04</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>J11_05</td>
<td>P0[7]</td>
<td>CUSTOM</td>
<td>J11_06</td>
<td>PSoC 5LP_XRES</td>
<td>RST</td>
</tr>
<tr>
<td>J11_07</td>
<td>P1[6]</td>
<td>INT_PIN</td>
<td>J11_08</td>
<td>PSoC 5LP_SWDCLK</td>
<td>SWD CLK</td>
</tr>
</tbody>
</table>

### A.1.3 PSoC 5LP GPIO and Custom Application Header (J6 and J7) for CY8CKIT-145-40XX

<table>
<thead>
<tr>
<th>J6 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
<th>J6 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J6_01</td>
<td>PSoC 5LP_VBUS</td>
<td>VBUS</td>
<td>J6_02</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>J6_03</td>
<td>P12[5]</td>
<td>CUSTOM</td>
<td>J6_04</td>
<td>P12[0]</td>
<td>I2C SCL</td>
</tr>
<tr>
<td>J6_07</td>
<td>P12[6]</td>
<td>UART_TX</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>J7 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
<th>J7 Pin</th>
<th>PSoC 5LP Signal</th>
<th>PSoC 5LP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J7_01</td>
<td>GND</td>
<td>GND</td>
<td>J7_02</td>
<td>P3[0]</td>
<td>CUSTOM</td>
</tr>
<tr>
<td>J7_07</td>
<td>P0[1]</td>
<td>CUSTOM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For other kits, refer to the kit user guide to get this information.
# Revision History

## Document Revision History

**Document Title:** KitProg2 User Guide  
**Document Number:** 002-10738

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN Number</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>5097455</td>
<td>01/22/2016</td>
<td>VJYA/SNVN</td>
<td>Initial version of KitProg2 User Guide.</td>
</tr>
</tbody>
</table>
| *A       | 5201134     | 04/01/2016   | VJYA             | Updated Introduction chapter on page 5:  
Added “Acronyms” on page 9.  
Updated Appendix chapter on page 78:  
Updated “Pin Assignments” on page 78:  
Added “PSoC 5LP GPIO and Custom Application Header (J16 and J11) for CY8CKIT-048”.  
Added “PSoC 5LP GPIO and Custom Application Header (J6 and J7) for CY8CKIT-145-40XX”. |
| *B       | 5268884     | 05/12/2016   | VJYA             | Added “CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit” related information in all instances across the document.  
Added “CY8CKIT-145-40XX PSoC 4 S-Series Prototyping Kit” related information in all instances across the document. |
| *C       | 5392257     | 08/05/2016   | SRDS             | Added “CY8CKIT-041-41XX PSoC 4100S Pioneer Kit” related information in all instances across the document. |
| *D       | 5660193     | 03/24/2017   | NMIT / VKVK      | Added “USB-SPI Bridge” related information in all instances across the document.  
Updated Introduction chapter on page 5:  
Updated description.  
Updated “Switching between KitProg2 Modes” on page 7:  
Updated description.  
Updated Figure 1-3.  
Updated Ecosystem chapter on page 10:  
Updated Table 2-1.  
Updated Table 2-3. |
## Document Revision History (continued)

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN Number</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
</table>
| *D (cont.) | 5660193 | 03/24/2017 | NMIT / VKVK | Updated KitProg2 Mode Programmer and Debugger chapter on page 12:  
Updated description.  
Updated “Programming Using PSoC Creator” on page 13:  
Updated description.  
Updated Figure 3-4.  
Updated Figure 3-5.  
Updated “Debugging Using PSoC Creator” on page 16:  
Updated description.  
Updated Figure 3-8.  
Removed “Programming and Debugging using µVision”.  
Added CMSIS-DAP Mode Programming and Debugging chapter on page 20.  
Updated Mass Storage Programmer chapter on page 27:  
Updated description.  
Updated “Enter or Exit the Mass Storage Programmer Mode” on page 27:  
Updated description.  
Updated “Programming Using the Mass Storage Programmer” on page 27:  
Updated description.  
Updated “Frequently Asked Questions on KitProg2 Mass Storage Programmer” on page 29:  
Updated description.  
Updated USB-UART Bridge chapter on page 30:  
Updated description.  
Updated Figure 6-1.  
Updated USB-I2C Bridge chapter on page 41:  
Updated description.  
Updated Figure 7-1.  
Updated Figure 7-11.  
Added USB-SPI Bridge chapter on page 52. |
## Document Revision History (continued)

### Document Title: KitProg2 User Guide
### Document Number: 002-10738

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN Number</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>*D (cont.)</td>
<td>5660193</td>
<td>03/24/2017</td>
<td>NMIT / VKVK</td>
<td>Updated Developing Applications for PSoC 5LP chapter on page 58: Updated description. Updated “Building a Custom Bootloadable Application for KitProg2” on page 58: Updated description. Updated Figure 9-1. Updated Figure 9-2. Updated Figure 9-6. Updated Figure 9-10. Updated “Building a Normal Project for PSoC 5LP” on page 67: Updated description. Updated Figure 9-15. Updated Figure 9-16. Updated Troubleshooting the KitProg2 chapter on page 71: Updated “KitProg2 Status LED Indication” on page 71: Updated Table 10-1. Updated description. Updated “PSoC 5LP Factory Program Restore Instructions” on page 73: Updated “PSoC 5LP is Programmed with a Custom Application” on page 73: Updated description. Updated “Restore PSoC 5LP KitProg2 using MiniProg3” on page 76: Updated description. Updated Appendix chapter on page 78: Updated “Pin Assignments” on page 78: Updated description. Updated to new template.</td>
</tr>
<tr>
<td>*E</td>
<td>5767710</td>
<td>06/08/2017</td>
<td>NMIT</td>
<td>Updated Troubleshooting the KitProg2 chapter on page 71: Updated “KitProg2 Status LED Indication” on page 71: Updated Table 10-1.</td>
</tr>
<tr>
<td>*F</td>
<td>5940717</td>
<td>10/23/2017</td>
<td>NMIT</td>
<td>Updated Ecosystem chapter on page 10: Updated Table 2-1 (Added new kit CY8CKIT-149 to support S3 device). Updated Troubleshooting the KitProg2 chapter on page 71: Updated “KitProg2 Status LED Indication” on page 71: Updated Table 10-1.</td>
</tr>
<tr>
<td>*G</td>
<td>5960991</td>
<td>11/08/2017</td>
<td>NMIT</td>
<td>Fixed typos (Replaced “CMIS” with “CMSIS” and replaced “CMISIS” with “CMSIS” in all instances across the document).</td>
</tr>
</tbody>
</table>
## Document Revision History (continued)

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN Number</th>
<th>Issue Date</th>
<th>Origin of Change</th>
<th>Description of Change</th>
</tr>
</thead>
</table>
| *G (cont.) | 5960991 | 11/08/2017 | NMIT | Updated Introduction chapter on page 5:  
Updated “Switching between KitProg2 Modes” on page 7:  
Updated description.  
Updated KitProg2 Mode Programmer and Debugger chapter on page 12:  
Updated description.  
Updated CMSIS-DAP Mode Programming and Debugging chapter on page 20:  
Updated description.  
Updated “Programming and Debugging using μVision” on page 20:  
Updated description.  
Updated Mass Storage Programmer chapter on page 27:  
Updated “Enter or Exit the Mass Storage Programmer Mode” on page 27:  
Updated description.  
Updated Troubleshooting the KitProg2 chapter on page 71:  
Updated “KitProg2 Status LED Indication” on page 71:  
Updated Table 10-1.  
Updated description. |
| *H      | 6022878   | 01/10/2018 | NMIT | Updated Troubleshooting the KitProg2 chapter on page 71:  
Updated “KitProg2 Status LED Indication” on page 71:  
Updated Table 10-1.  
Completing Sunset Review. |
| *I      | 6043479   | 03/07/2018 | NMIT | Updated Ecosystem chapter on page 10:  
Updated Table 2-1.  
Updated Developing Applications for PSoC 5LP chapter on page 58:  
Updated “Building a Custom Bootloadable Application for KitProg2” on page 58:  
Updated description.  
Added Figure 9-8.  
Updated Figure 9-9. |
| *J      | 6216916   | 06/25/2018 | NMIT | Updated Ecosystem chapter on page 10:  
Updated Table 2-1. |
| *K      | 6367122   | 10/26/2018 | SRDS | Updated Ecosystem chapter on page 10:  
Updated Table 2-1. |