

CY8CKIT-040

PSoC[®] 4000 Pioneer Kit Guide

Doc. # 001-91316 Rev. **

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



Copyrights

© Cypress Semiconductor Corporation, 2014. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

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

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

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

PSoC and CapSense are registered trademarks and PSoC Components, PSoC Creator, and SmartSense are trademarks of Cypress Semiconductor Corporation. All other products and company names mentioned in this document may be the trademarks of their respective holders.

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

Flash Code Protection

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

Cypress is willing to work with the customer who is concerned about the integrity of their code. Code protection is constantly evolving. We at Cypress are committed to continuously improving the code protection features of our products.

Contents



	Safety I	nformation	5
1.	Introdu	ction	7
	1.1	Kit Contents	7
	1.2	PSoC Creator	
	1.3	Getting Started	
	1.4	Additional Learning Resources	
		1.4.1 PSoC 4 Datasheets	
		1.4.2 Learning PSoC Creator	
		1.4.3 Application Notes	
		1.4.4 Design Guide	
		1.4.5 Technical Reference Manuals (TRM)	
	1.5	Technical Support	
	1.6	Documentation Conventions	
2.	Softwa	re Installation	13
	2.1	Before You Begin	13
	2.2	Install Software	13
	2.3	Install Hardware	15
	2.4	Uninstall Software	15
2	Kit One	vision	40
э.	Kit Ope		16
Э.	ки Оре 3.1	Kit Overview	-
5.			16
5.	3.1	Kit Overview	16 18
5.	3.1 3.2	Kit Overview Kit USB Connection	
5.	3.1 3.2	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000	16 18 19 19
5.	3.1 3.2	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger	
5.	3.1 3.2 3.3	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger	
5.	3.1 3.2 3.3 3.4	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge	
	3.1 3.2 3.3 3.4 3.5	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware	
	3.1 3.2 3.3 3.4 3.5 3.6	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware	
	3.1 3.2 3.3 3.4 3.5 3.6 Hardwa	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware Ire Board Details	
	3.1 3.2 3.3 3.4 3.5 3.6 Hardwa 4.1	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware	
	3.1 3.2 3.3 3.4 3.5 3.6 Hardwa 4.1 4.2	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware Ire Board Details Block Diagram Kit Component Details	
	3.1 3.2 3.3 3.4 3.5 3.6 Hardwa 4.1 4.2	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware Board Details Block Diagram	
4.	3.1 3.2 3.3 3.4 3.5 3.6 Hardwa 4.1 4.2 4.3	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware Ire Board Details Block Diagram Kit Component Details 4.3.1 CY8CKIT-040 Baseboard Components	
4.	3.1 3.2 3.3 3.4 3.5 3.6 Hardwa 4.1 4.2 4.3	Kit Overview Kit USB Connection Programming and Debugging PSoC 4000 3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger 3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger USB-I2C Bridge USB-UART Bridge Updating the Onboard Programmer Firmware Ire Board Details Block Diagram Kit Component Details 4.3.1 CY8CKIT-040 Baseboard Components 4.3.2 CY8CKIT-040 CapSense Trackpad Shield Board	



	5.2	Project: Blinking LED	50
		5.2.1 Project Overview	
		5.2.2 Project Description	
		5.2.3 Verify Output	52
	5.3	Project: CapSense Proximity and UART	
		5.3.1 Project Overview	
		5.3.2 Project Description	54
		5.3.3 Verify Output	62
	5.4	Project: CapSense Touchpad with I2C Tuner	68
		5.4.1 Project Overview	68
		5.4.2 Project Description	70
		5.4.3 Verify Output	73
	5.5	Project: Color Palette	77
		5.5.1 Project Overview	77
		5.5.2 Project Description	
		5.5.3 Verify Output	93
6	Advanc	ed Topics	95
υ.	6.1	-	
	6.2	Using PSoC 5LP as a USB-I2C Bridge Using FM24W256 F-RAM	
	0.2	6.2.1 Address Selection	
		6.2.2 Write/Read Operation	
		6.2.3 Example Firmware	
	6.3	Using PSoC 5LP as a USB-UART Bridge	
	6.4	Developing Applications for PSoC 5LP	
	0.4	6.4.1 Building a Bootloadable Project for PSoC 5LP	
		6.4.2 Building a Normal Project for PSoC 5LP	
	6.5	PSoC 5LP Factory Program Restore Instructions	
	0.0	6.5.1 PSoC 5LP Programmed with a Bootloadable Application	
		6.5.2 PSoC 5LP Programmed with a Standard Application	
Α.	Append	lix	136
	A.1	CY8CKIT-040 Schematics	136
	A.2	Pin Assignment Table	141
	A.3	Program and Debug Headers	143
	A.4	Use of Zero-ohm Resistors and No Load	
	A.5	Error in Firmware/Status Indication in Status LED	145
	A.6	Bill of Materials	146
	A.7	Trackpad/Touchpad Sticker Details	149

Safety Information



Regulatory Compliance

The CY8CKIT-040 PSoC[®] 4000 Pioneer Kit is intended for use as a development platform for hardware or software in a laboratory environment. The board is an open system design, which does not include a shielded enclosure. For this reason, the board may cause interference to other electrical or electronic devices in close proximity. In a domestic environment, this product may cause radio interference. In such cases, the user may be required to take adequate preventive measures. Also, this board should not be used near any medical equipment or RF devices.

Attaching additional wiring to this product or modifying the product operation from the factory default may affect its performance and cause interference with other apparatus in the immediate vicinity. If such interference is detected, suitable mitigating measures should be taken.

The CY8CKIT-040 as shipped from the factory has been verified to meet with requirements of CE as a Class A product.





The CY8CKIT-040 contains electrostatic discharge (ESD) sensitive devices. Electrostatic charges readily accumulate on the human body and any equipment, and can discharge without detection. Permanent damage may occur to devices subjected to high-energy discharges. Proper ESD precautions are recommended to avoid performance degradation or loss of functionality. Store unused CY8CKIT-040 boards in the protective shipping package.

End-of-Life/Product Recycling

The end of life for this kit is five years from the date of manufacture mentioned on the back of the box. Contact your nearest recycler to discard the kit.



General Safety Instructions

ESD Protection

ESD can damage boards and associated components. Cypress recommends that the user perform procedures only at an ESD workstation. If an ESD workstation is not available, use appropriate ESD protection by wearing an antistatic wrist strap attached to the chassis ground (any unpainted metal surface) on the board when handling parts.

Handling Boards

CY8CKIT-040 boards are sensitive to ESD. Hold the board only by its edges. After removing the board from its box, place it on a grounded, static free surface. Use a conductive foam pad if available. Do not slide the board over any surface.



Thank you for your interest in the PSoC[®] 4000 Pioneer Kit. The kit is designed as an easy-to-use and inexpensive development kit, highlighting the unique flexibility of the PSoC 4000 architecture. Designed for flexibility, this kit offers footprint compatibility with several third-party Arduino[™] shields. In addition, the board features an RGB LED, integrated USB programmer/debugger, a program/ debug header, USB-UART/I²C bridges, a proximity header, and an Arduino-compatible CapSense[®] Trackpad shield. This kit supports either 5 V or 3.3 V as power supply voltages.

The PSoC 4000 Pioneer Kit is based on the PSoC 4000 device family, delivering a programmable platform for a wide range of embedded applications. The PSoC 4000 is the smallest member of the PSoC 4 platform with support for CapSense, Timer Counter Pulse Width Modulator (TCPWM), I²C master or slave, and up to 20 GPIOs. PSoC 4000 is a cost-optimized, entry-level PSoC 4 device targeted as socket replacements for obsolete and/or proprietary 8-bit and 16-bit MCUs. PSoC 4000 with its ARM Cortex-M0 core provides 32 programmable peripherals including CapSense.

1.1 Kit Contents

The PSoC 4000 Pioneer Kit contains the following (see Figure 1-1):

- PSoC 4000 Pioneer Kit board
- Trackpad shield board with a color palette sticker
- Quick start guide
- USB Standard A to Mini-B cable
- 6 jumper wires

Note: Trackpad and Touchpad denote the same in the context of this document and can be used interchangeably.



Figure 1-1. Kit Contents



Inspect the contents of the kit; if you find any part missing, contact your nearest Cypress sales office for help: www.cypress.com/go/support.

Download the latest version of the kit setup file from www.cypress.com/CY8CKIT-040.



1.2 **PSoC Creator**

PSoC Creator[™] is a state-of-the-art, easy-to-use integrated design environment (IDE). It introduces revolutionary hardware and software codesign, powered by a library of preverified and precharacterized PSoC Components[™].

With PSoC Creator, you can:

- Drag and drop PSoC components to build a schematic of your custom design
- Automatically place and route components and configure GPIOs
- Develop and debug firmware using the included component APIs

PSoC Creator also enables you to tap into an entire tool ecosystem with integrated compiler chains and production programmers for PSoC devices.

For more information, visit www.cypress.com/psoccreator.

1.3 Getting Started

This guide helps acquaint you with the PSoC 4000 Pioneer Kit.

- The Software Installation chapter on page 13 describes the installation of the kit software.
- The Kit Operation chapter on page 16 explains how to program the PSoC 4 with a programmer and debugger, either the onboard PSoC 5LP or the external MiniProg3 (CY8CKIT-002).
- The Hardware chapter on page 27 details the hardware operation.
- The Example Projects chapter on page 45 describes the example projects that are provided with the kit.
- The Advanced Topics chapter on page 95 deals with topics such as building projects for PSoC 5LP, using onboard F-RAM, USB-UART functionality, and USB-I²C functionality of PSoC 5LP.
- The Appendix on page 136 provides schematics, pin assignments, information on the use of zero-ohm resistors, troubleshooting details, and the bill of materials (BOM).

1.4 Additional Learning Resources

1.4.1 PSoC 4 Datasheets

PSoC 4 datasheets list the features and electrical specifications of all PSoC 4 device families.

1.4.2 Learning PSoC Creator

Visit the PSoC Creator home page to download the latest version of PSoC Creator.

Launch PSoC Creator and navigate to the following items:

Quick Start Guide: Choose Help > Documentation > Quick Start Guide (see Figure 1-2). This guide gives you the basics for developing PSoC Creator projects.



Figure 1-2. Quick Start Guide

He	P			
	<u>T</u> opics <u>D</u> ocument Manager System Reference <u>Update Manager</u> PSoC Creator T <u>r</u> aining Cypress Dev Co <u>m</u> munity	•	<pre>main.c CapSense_Proximity_Design main.c 1.0</pre>	***
	Docum <u>e</u> ntation	•	Quick Start Guide	
	Register	`	Release Notes	

Simple Component Example Projects: Choose File > Example projects (see Figure 1-3). These example projects demonstrate how to configure and use PSoC Creator Components.

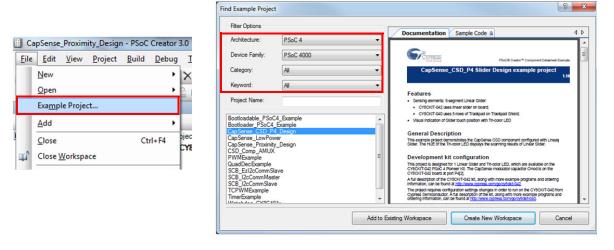


Figure 1-3. Example Projects

Starter Designs: Choose File > New > Project > PSoC 4 Starter Designs (see Figure 1-4). These starter designs demonstrate the unique features of PSoC 4.

Each starter design contains a PDF document that explains the features of the project and its configuration.

Note: The example projects and starter designs are designed for CY8CKIT-040 PSoC 4000 Pioneer Kit. However, these projects can be converted to use with other PSoC 4000 hardware setup by following the "Schematic and Pin Mapping" section in the help document provided with the example project.



2 X

Figure 1-4. Starter Designs

		Des	sign Other	4 Þ
		Ē	DelSig_SPIM	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over SPI.
		ē	Filter_ADC_VDAC	Shows how to filter an analog input all in hardware, and provides all the DMA setup to transfer the data from the ADC directly to the Digital Filter Block, bypassing the processor.
		ē	HW_Fan_Control_with_Alert	Shows how performing fan control in hardware completely frees up the CPU.
		E PSol	C 4 Starter Designs	
<u>F</u> ile	Edit View Project Build Debug Tools Window	ē	ADC_Differential_Preamplifier	Shows how to sample four different channels with Sequencing ADC and send results to PC using UART.
	New Project	Ē	CapSense_LowPower	For PSoC 4000 devices only, this project demonstrates a low power CapSense system using PSoC 4.
	Open Pile	P	CapSense_Proximity_Design	For PSoC 4000 devices only, this project demonstrates a CapSense based proximity sensing design to control brightness of a LED.
			Hibernate_and_Stop_PowerModes	Shows how to enter and wake up from hibernate and stop low power modes.
	Example Project 🗸 🗸 🛪	ē	Multiplexed_Comparator	Shows how to monitor multiple inputs using multiplexed comparator without CPU intervention.
1		ē	Opamp_Dynamic_Gain_Switching	Shows how to sample three ADC channels, change the input and gain on the fly, and send the result to PC using UART.
		ē	PMBusThermExample	Demonstrates usage of the PMBus Slave component in a simulated Thermal Management application.
		■ PSol	C 5LP Starter Designs	•
		Name:	Design01	
		Location	: C:\Users\msur\Desktop\Kits\C	Y8CKIT-040\Firmware\Example_FRAM
		→ Adva	anced	
				OK Cancel

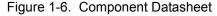
New Project

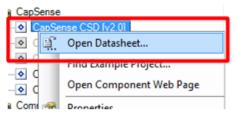
 System Reference Guide: Choose Help > System Reference > System Reference Guide (see Figure 1-5). This guide lists and describes the system functions provided by PSoC Creator.

Figure 1-5. System Reference Guide

Hel	p		
1	Topics		
	Document Manager		
	System Reference	System	n Reference Guide
	<u>U</u> pdate Manager	Open	Web Page

- Component Datasheets: Right-click a Component and select Open Datasheet (see Figure 1-6). Visit the following page for a list of all PSoC 4 Component datasheets:
 - PSoC 4 Component Datasheets





1.4.3 Application Notes

Application notes assist you in understanding specific features of the device and designing your PSoC application.

Visit the following page for a complete list of PSoC 4 application notes: PSoC 4 Application Notes

A few application notes that can help you get started with developing PSoC 4 applications are:

- AN79953 Getting Started with PSoC® 4
- AN54460 PSoC® 3, PSoC 4, and PSoC 5LP Interrupts



1.4.4 Design Guide

Visit the following page to download the PSoC 4 CapSense Design Guide, which shows how to design capacitive touch-sensing applications with the PSoC 4 family of devices.

■ PSoC 4 CapSense Design Guide

1.4.5 Technical Reference Manuals (TRM)

The TRMs provide detailed descriptions of the internal architecture of the PSoC 4 device.

PSoC 4 Technical Reference Manuals

1.5 Technical Support

If you have any questions, you can create a support request at the Cypress Technical Support page.

If you are in the United States, you can talk to our technical support team by calling our toll-free number: +1-800-541-4736. Select option 2 at the prompt. If you are outside United States, you can talk to our technical support team by calling: +1 (408) 943-2600 Ext. 2.

You can also use the following support resources if you need quick assistance.

- Self-help
- Local Sales Office Locations

1.6 Documentation Conventions

Convention	Usage
Courier New	Displays file locations, user-entered text, and source code: C:\cd\icc\
Italics	Displays file names and reference documentation: Read about the <i>sourcefile.hex</i> file in the <i>PSoC Creator User Guide</i> .
[Bracketed, Bold]	Displays keyboard commands in procedures: [Enter] or [Ctrl] [C]
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths, and icon names in procedures: Click the File icon and then click Open .
Times New Roman	Displays an equation: 2+2=4
Text in gray boxes	Describes cautions or a unique functionality of the product.

Table 1-1. Document Conventions for Guides



This section describes the installation of the CY8CKIT-040 PSoC 4000 Pioneer Kit software and the prerequisites.

2.1 Before You Begin

2.

All Cypress software installations require administrator privileges. However, this is not the case for installed software. Before you install the kit software, close any other Cypress software that is currently running.

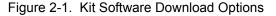
2.2 Install Software

Follow these steps to install the CY8CKIT-040 PSoC 4000 Pioneer Kit software:

1. Download the CY8CKIT-040 software.

Software Installation

The CY8CKIT-040 software is available in three different formats for download (see Figure 2-1):





- a. CY8CKIT-040 Kit Setup: This installation package contains the files related to the kit. However, it does not include the Windows Installer or Microsoft .NET framework packages. If these packages are not on your computer, the installer directs you to download and install them from the Internet.
- b. CY8CKIT-040 Kit Only: This executable file installs only the kit contents, which include kit code examples, hardware files, and user documents. This package can be used if all the software prerequisites listed in step 5 are installed on your PC.
- c. CY8CKIT-040 CD ISO: This file is a complete package, stored in a CD-ROM image format that you can use to create a CD or extract using ISO extraction programs, such as WinZip or WinRAR. The file can also be mounted like a virtual CD using virtual drive programs such as Virtual CloneDrive and MagicISO. This file includes all the required software, utilities, drivers, hardware files, and user documents.



- If you have downloaded the ISO file, mount it in a virtual drive. Extract the ISO contents if you do
 not have a virtual drive to mount. Double-click *cyautorun.exe* in the root directory of the extracted
 content or mounted ISO if 'Autorun from CD/DVD' is not enabledd in the PC. The installation window shown in Figure 2-2 will appear automatically. Note: If you are using the 'Kit Setup' or 'Kit
 Only' file, then go to step 6 for installation.
- 3. Click Install CY8CKIT-040 to start the kit installation, as shown in Figure 2-2.

Figure 2-2. Kit Installer Startup Screen



- 4. Select the folder in which you want to install the CY8CKIT-040 kit-related files. Choose the directory and click **Next**.
- 5. When you click **Next**, the CY8CKIT-040 ISO installer automatically installs the required software, if it is not present on your computer.

Following is the required software:

- a. PSoC Creator 3.0 SP1 or later: Download the latest version from www.cypress.com/psoccreator.
- b. PSoC Programmer 3.20.1 or later: Download the latest version from www.cypress.com/programmer.
- 6. Choose the **Typical/Custom/Complete** installation type in the Product Installation Overview window, as shown in Figure 2-3. Click **Next** after you select the installation type.





CyInstaller for CY8CKIT-040 PSoC 4000 Pioneer Kit	? x
Product Installation Overview Choose the install type that best suits your needs	
Choose the type of installation Product: CY8CKIT-040 PSoC 4000 Pioneer Kit. Installation Type: Installs the most common features of CY8CKIT-040 PSoC 4000 Pioneer Kit.	
Contact Us	<u>C</u> ancel

- 7. When the installation begins, a list of packages appears on the installation page. A green check mark appears next to each package after successful installation.
- 8. Enter your contact information or select the check box **Continue Without Contact Information**. Click **Finish** to complete the CY8CKIT-040 kit installation.
- 9. After the installation is complete, the kit contents are available at the following location: <Install_Directory>\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version>

Default location:

Windows 7 (64-bit):

C:\Program Files (x86)\Cypress\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version> Windows 7 (32-bit):

C:\Program Files\Cypress\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version>

Note: For Windows 7/8/8.1 users, the installed files and the folder are read only. To change the property, right-click the folder and choose **Properties > Attributes**; disable the **Read-only** check box. Click **Apply** and **OK** to close the window.

2.3 Install Hardware

There is no additional hardware installation required for this kit.

2.4 Uninstall Software

You can uninstall the CY8CKIT-040 PSoC 4000 Pioneer Kit software using one of the following methods:

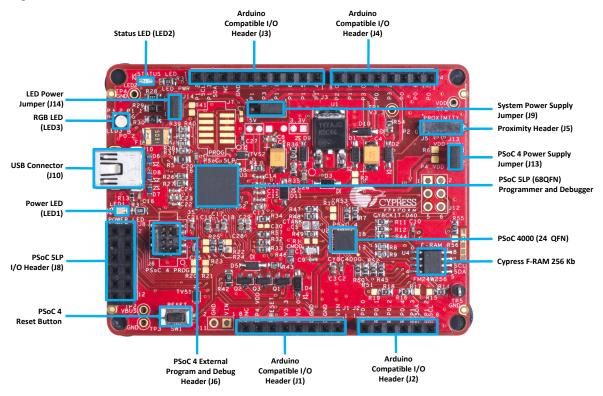
- Go to Start > All Programs > Cypress > Cypress Update Manager > Cypress Update Manager. Select the Uninstall button that corresponds to the kit software.
- Go to Start > Control Panel > Programs and Features (or Add/Remove Programs for Windows XP). Select the Uninstall/Change button that corresponds to the kit software.



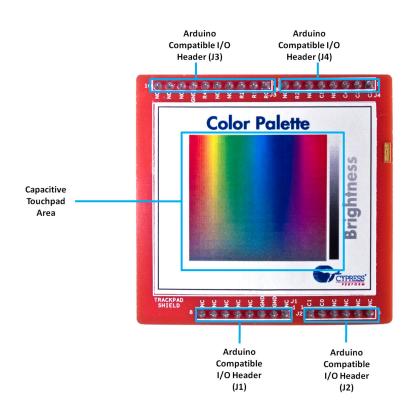
3.1 Kit Overview

The PSoC 4000 Pioneer Kit can be used to develop applications using the PSoC 4000 family of devices. The kit includes two boards – an Arduino-compatible baseboard and a CapSense-based Trackpad shield board. Figure 3-1 is an image of the PSoC 4000 Pioneer Kit baseboard and shield board with a markup of the onboard components.

Figure 3-1. CY8CKIT-040 Kit Details









3.2 Kit USB Connection

The PSoC 4000 Pioneer Kit connects to the PC over a USB interface (see Figure 3-2). The kit enumerates as a composite device and three separate devices appear under the **Device Manager** in the Windows operating system. See Table 3-1, Figure 3-3, and Figure 3-4.

Figure 3-2. Kit USB Connection

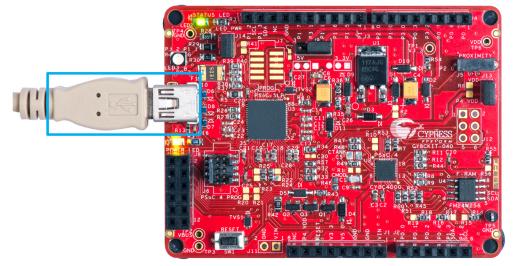


Table 3-1. PSoC 4000 Pioneer Kit in Device Manager After Enumeration

Port	Description
USB Input Device	USB-I ² C bridge
KitProg	Programmer and debugger
KitProg USB-UART	USB-UART bridge appears as a COM# port

Figure 3-3. KitProg Driver Installation

Driver Software Installation		×
Installing device driver softwar	e	
USB Composite Device KitBridge KitProg Programmer KitProg USBUART Obtaining device driver software from W Skip obtaining driver software from Win		
	(Close





Driver Software Installation		×
Your device is ready to use		
USB Composite Device USB Input Device KitProg (3.4.1.20) KitProg USB-UART (COM16)	 Ready to use 	
		Close

3.3 **Programming and Debugging PSoC 4000**

The kit allows programming and debugging of the PSoC 4 device in two modes:

- Using the Onboard PSoC 5LP Programmer and Debugger
- Using the CY8CKIT-002 MiniProg3 Programmer and Debugger

3.3.1 Using the Onboard PSoC 5LP Programmer and Debugger

The default programming interface for the kit is a USB-based, onboard programming interface. Before trying to program the device, PSoC Creator and PSoC Programmer must be installed. See Install Software on page 13 for information on installing the kit software.

 To program the device, plug the USB cable into the programming USB connector J10, as shown in Figure 3-5. The kit will enumerate as a composite device. See Kit USB Connection on page 18 for details.

<image>

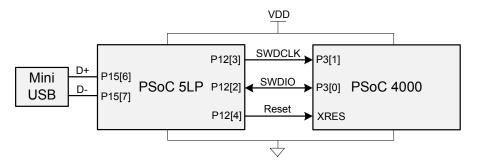
Figure 3-5. Connect USB Cable to J10



 The onboard PSoC 5LP uses serial wire debug (SWD) to program the PSoC 4 device. See Figure 3-6.

Note: Figure 3-6 is provided only for reference, all connections are hardwired on the board itself.

Figure 3-6. SWD Programming of PSoC 4000 Using PSoC 5LP



 The kit's onboard programmer will enumerate on the PC and in the software tools as KitProg. Open an example project in PSoC Creator (such as Project: Blinking LED on page 50) and initiate the build by choosing Build > Build Project or pressing [Shift] [F6]. See Figure 3-7.

Figure 3-7. Build Project in PSoC Creator

				•	-	-		
t	<u>B</u> uil	d	<u>D</u> ebug	<u>T</u> ools	<u>W</u> indow	<u>H</u> elp	0	
Γ		B	<u>u</u> ild CY8C	KIT_040_	Blinking_LE	D	Shift+F6	
		C	lean CY80	.KIT_040	Blinking_L	Ð		Ľ
:)	**	C	lea <u>n</u> and E	Build CY8	CKIT_040_E	Blinkin	ig_LED	
	×	<u>C</u>	ancel Buil	d		Ct	trl+Break	- 1
BI 40		С	ompile Fil	e			Ctrl+F6	
+0	1	G	enerate A	pplicatio	n			Ē
:in		G	enerate Pr	roject Da	tasheet			

 After the project is built without errors and warnings, choose Debug > Program or press [Ctrl] [F5] to program the device. See Figure 3-8.

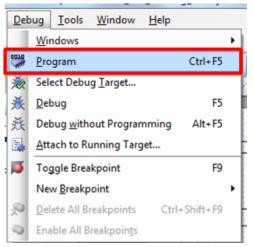


Figure 3-8. Program Device From PSoC Creator



The onboard programmer supports only the RESET programming mode. When using the onboard programmer, the board can either be powered by the USB (VBUS) or by an external source such as an Arduino shield (see Power Supply System on page 34). If the board is already powered from another source, plugging in the USB programmer does not damage the board.

3.3.2 Using the CY8CKIT-002 MiniProg3 Programmer and Debugger

The PSoC 4 on the kit can also be programmed using a MiniProg3 (CY8CKIT-002). To use MiniProg3 for programming, use the J6 connector on the board, as shown in Figure 3-9. With MiniProg3, programming is similar to the onboard programmer; however, it enumerates as MiniProg3 instead of KitProg.

The board can also be powered from the MiniProg3. To do so, choose **Tool** > **Options** in PSoC Creator. In the Options window, expand **Program/Debug** > **Port Configuration**; click **MiniProg3** and select the settings shown in Figure 3-10. Choose **Debug** > **Program** to program and power the board.

Note The CY8CKIT-002 MiniProg3 is not part of the PSoC 4000 Pioneer Kit contents. It can be purchased from the Cypress Online Store.

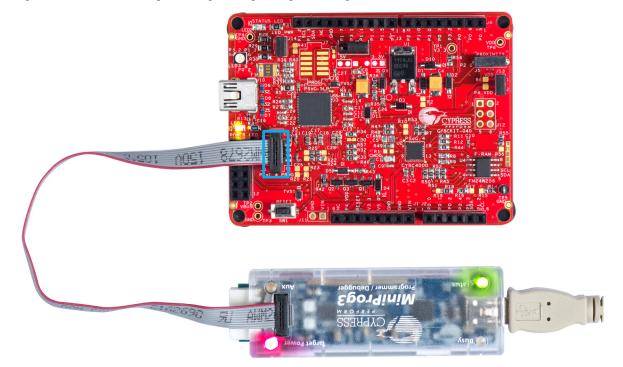


Figure 3-9. PSoC 4 Programming/Debug Using MiniProg3



Options	? ×
Project Management Text Editor Design Entry Language Support Program/Debug General Fonts and Colors Device Recognition Prof Configuration MiniProg3 FX2LP-SWD True TouchBridge DVKProg1 KitProg Environment	Active Protocol: SWD Clock Speed: 1.6 MHz Power Acquire Mode 5.0 V Reset 3.3 V Power Cycle 2.5 V Connector 1.8 V 5 pin External 10 pin
Restore Defaults	OK Apply Cancel

Figure 3-10. MiniProg3 Configuration in PSoC Creator

Note: Ensure that both MiniProg3 (with or without power) on header J6 and KitProg are not connected to the onboard PSoC 4 at the same time. This will result in failed device acquisition from both.



3.4 USB-I²C Bridge

The PSoC 5LP also functions as a USB-I²C bridge. The PSoC 4 communicates with the PSoC 5LP using an I²C interface and the PSoC 5LP transfers the data over the USB to the USB-I²C software utility on the PC, called the Bridge Control Panel (BCP).

The BCP is available as part of the PSoC Programmer installation. This software can be used to send and receive USB-I²C data from the PSoC 5LP. When the USB Mini-B cable is connected to header J10 on the PSoC 4000 Pioneer Kit, the **KitProg USB-I²C** is available under **Connected I2C/ SPI/RX8 Ports** in the BCP, as shown in Figure 3-11.

🌠 Bridge	e Control Panel				
<u>F</u> ile	<u>E</u> ditor <u>C</u> hart E <u>x</u> ecute <u>T</u>	ools <u>H</u> elp			
🖻 🖬	🗑 🕼 🖻 🛍 🔷 🧮 🛿	× 🕰 🕰			
Editor (Chart Table File				
					~
•					Þ
Seled	ct Port in the Po	ortList, then try			*
			Select	Connect/	_
٠			KitProg	Disconnect Button	- F
& Res Sto		Send all strings:	nnected I2C/SPI/RX8 Ports: Prog/191607A0032C2400 M43	▶ Power +5.0V +3.0V +2.5V +2.5V ● +1.8V	Protocol il2C SPI RX8 (UART)
1:1	Syntax : OK		Not Connected	Voltage: -	

Figure 3-11. Bridge Control Panel

To use the USB-I²C functionality, select the **KitProg USB-I²C** in the BCP (Figure 3-11). On successful connection, the **Connected** and **Powered** status boxes turn green (Figure 3-12).



🗱 Bridge Control Panel				
<u>File Editor Chart Execute Tools H</u> elp				
☞ 🖩 🚊 🗟 № 🛍 🔷 듬 🖾 🧮				
Editor Chart Table File				
	*			
Alast Dant in the Dantrick then two to service				
Select Port in the PortList, then try to connect Opening Port	<u></u>			
Successfully Connected to KitProg/191607A0032C2400				
KitProg Version 2.07				
	*			
	4			
Connected I2C/SPI/RX8 Ports:	Protocol			
COM3	I2C SPI			
Instead code to the set of	RX8 (UART)			
1:1 Syntax: OK Connected Powered Voltage: 5032 mV				

Figure 3-12. KitProg USB-I²C Connected in Bridge Control Panel

USB-I²C is implemented using the USB and I²C components of PSoC 5LP. The SCL (P12_0) and SDA (P12_1) lines from the PSoC 5LP are connected to the SCL (P1_2) and SDA (P1_3) lines of the PSoC 4 I²C. The USB-I²C bridge currently supports I²C speed of 50 kHz, 100 kHz, 400 kHz, and 1 MHz.

See Using PSoC 5LP as a USB-I2C Bridge on page 95 for building a project that uses the USB-I²C bridge functionality.

3.5 USB-UART Bridge

The onboard PSoC 5LP can also act as a USB-UART bridge to transfer and receive data from the PSoC 4 device to the PC via the COM terminal software. When the USB Mini-B cable is connected to J10 of the PSoC 4000 Pioneer Kit, a device named **KitProg USB-UART** is available under **Ports (COM & LPT)** in the Device Manager. For more information about the USB-UART functionality, see Using PSoC 5LP as a USB-UART Bridge on page 108.

To use the USB-UART functionality in the COM terminal software, select the corresponding COM port as the communication port for transferring data to and from the COM terminal software.

The UART lines from PSoC 5LP are brought to the P12[6] (J8_9) and P12[7] (J8_10) pins of header J8. This interface can be used to send or receive data from any design/device that has a UART by connecting the pins on header J8 to the RX and TX pins available on the connecting device.



Note: The PSoC 4000 family that is featured in the kit board does not support a full-duplex UART; it can support only a software-based UART transmit on any pin. On the board, P3[0] of the PSoC 4000 device is hardwired to the UART bridge's RX line through zero-ohm resistor R57.

Table 3-2 lists the specifications supported by the USB-UART bridge.

Table 3-2. Specifications Supported by USB-UART Bridge

Parameter	Supported Values
Baud rate	1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200
Data bits	8
Parity	None
Stop bits	1
Flow control	None
File transfer protocols supported	Xmodem, 1K Xmodem, Ymodem, Kermit, and Zmodem (only speeds greater than 2400 baud).

3.6 Updating the Onboard Programmer Firmware

The firmware of the onboard programmer and debugger (KitProg), PSoC 5LP, can be updated from PSoC Programmer. When a new firmware is available or when the KitProg firmware is corrupt (see Error in Firmware/Status Indication in Status LED on page 145), PSoC Programmer displays a warning indicating that new firmware is available.

Open PSoC Programmer from **Start > All Programs > Cypress > PSoC Programmer<version>**. When PSoC Programmer opens, a WARNING! window pops up saying that the programmer is currently out of date, as shown in Figure 3-13.

Figure 3-13. 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 3-14.



-	PSoC Programmer				
	File View Options He	lp			
	🖆 · 🔰 💿 🔒				
	Port Selection	Programme Utilities TAG			
	KitProg/040D0DA0032C24C	Upgrade Firmware Click to upgrade connected device's firmware Upgrade Firmware button			
		Erase Block Click to erase user specific flash block			
	Device Family	Update Firmware Message			
	CY8C40xx 👻				
	Device				
	CY8C4014LQI-422 -				
	Actions	Results			
		Please navigate to the Utilities tab and click the Upgrade Firmware button			
	Port Opened with Warnings at 4:20:22	KitProg version Expecting 2.07, but found 1.00.			
	Opening Port at 4:20 PM				
	Connected at 4:20:19	KitProg/040D0DA0032C2400			
		Select Port in the PortList, then try to connect			
	Disconnected at 4:19 PM	116 DVKProg1/170B168B01131400			
Successfully Connected					
	to	DVKProg1 Version 1.01			
H					
Ŀ	or Help, press F1	Powered Connected			

Figure 3-14. Upgrade Firmware Message in PSoC Programmer

Click the **Utilities** tab and click the **Upgrade Firmware** button. On successful upgrade, the **Actions** and **Results** window displays the firmware update message with the KitProg version, as shown in Figure 3-15.

PSoC Programmer			
File View Options Help			
Port Selection IV Programmer Vtilities JTAG			
KitProg/1F13034702222400 Upgrade Firmware Click to upgrade connected device's firmware			
Erase Block Click to erase user specific flash block			
Device Family			
CY8C40xx *			
Device			
CY8C4014LQI-422 v			
Actions Results KitProg Version 2.07 KitProg firmware Version			
KitProg Version 2.07 KitProg firmware Version			
Firmware Update Finished at 4:49:14 PM Succeeded			
Verifying Upgrading KitProg Firmware Update message			
Initializing			
Firmware Upgrade Started at 4:48:57 PM			
For Help, press F1 Connected			

Figure 3-15. Firmware Updated in PSoC Programmer

4. Hardware



4.1 Board Details

The PSoC 4000 Pioneer Kit consists of the following blocks:

- CY8CKIT-040 baseboard (see Figure 4-1) -
 - □ PSoC 4 (4000 family)
 - PSoC 5LP
 - Power supply system
 - □ Coin cell battery holder (BT1)
 - □ Programming interfaces (J6, J7, and J10)
 - □ Arduino compatible headers (J1, J2, J3, J4, and J12)
 - □ PSoC 5LP GPIO header (J8)
 - □ Proximity header (J5)
 - Pioneer board LEDs
 - Push button (Reset button)
 - Cypress ferroelectric RAM (F-RAM)
- CY8CKIT-040 CapSense Trackpad shield board (see Figure 4-2)



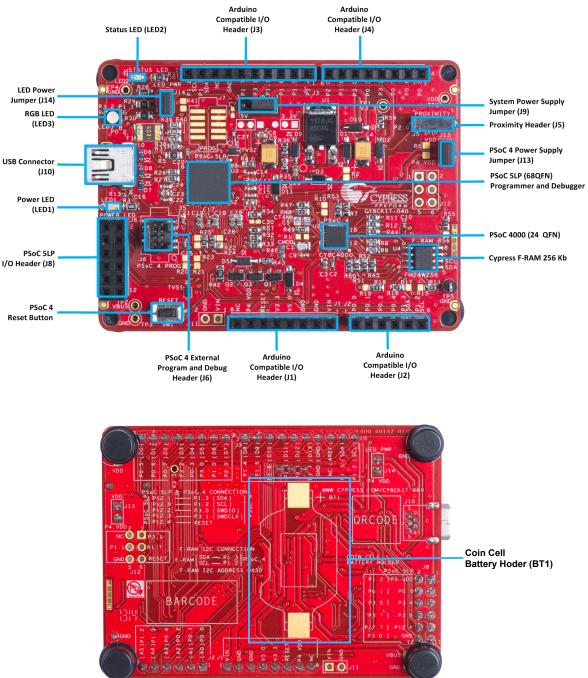


Figure 4-1. CY8CKIT-040 - Baseboard Details



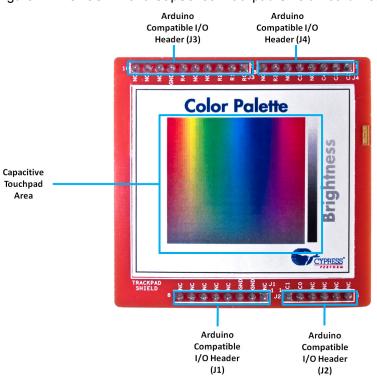
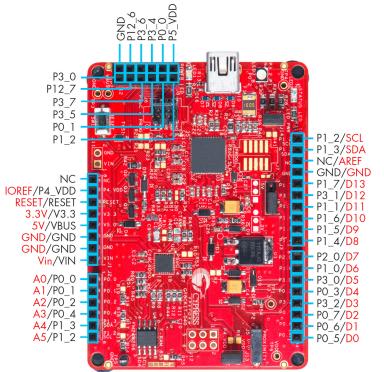


Figure 4-2. CY8CKIT-040 CapSense Trackpad Shield Board Details





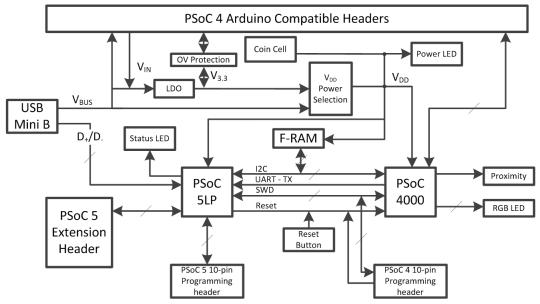




4.2 Block Diagram

This section provides the block-level description of the PSoC 4000 Pioneer Kit, as illustrated in Figure 4-4.





PSoC 4 is a new generation of programmable system-on-chip devices from Cypress for embedded applications. The PSoC 4000 family is the smallest member of the PSoC 4 platform with CapSense, TCPWM, I²C master or slave, and up to 20 GPIOs support. PSoC 4000 is a cost-optimized, entry-level PSoC4 device targeted as socket replacements for obsolete and/or proprietary 8-bit and 16-bit MCUs. PSoC 4000 with its ARM Cortex-M0 core brings 32 CapSense and programmable peripherals.

The kit features an onboard PSoC 5LP, which communicates through the USB to program and debug the PSoC 4 using SWD. The PSoC 5LP also functions as a USB-I²C bridge and USB-UART bridge. It can also be used to develop PSoC 5LP based applications (see Developing Applications for PSoC 5LP on page 119).

The kit includes an RGB LED, a status LED, and a power LED. The RGB LED is connected to the PSoC 4, and the status LED is connected to the PSoC 5LP. This kit also includes a Reset button that connects to the PSoC 4 XRES, a CapSense proximity header, and a 6x5 elements Trackpad board that is Arduino shield-compatible and can be used to develop touch-based applications. The PSoC 4 pins are brought out onto headers J1 to J4 on the kit to support Arduino shields. The PSoC 5LP pins are brought out onto header J8 to enable using the onboard PSoC 5LP to develop custom applications.

The PSoC 4000 Pioneer Kit can be powered from the USB Mini-B, the Arduino compatible header, an external power supply, or an optional coin cell battery. The input voltage is regulated by a low drop out (LDO) regulator to 3.3 V. You can select between VBUS (5 V) and 3.3 V by suitably plugging the jumper onto the voltage selection header J9. VDD can also be supplied by a coin cell battery placed in the BT1 coin cell holder. The voltage supplied by the coin cell is directly connected to the VDD line through a diode, D11. Take care to ensure that VDD does not exceed the device's VDD specification (1.8 V to 5.5 V).



4.3 Kit Component Details

4.3.1 CY8CKIT-040 Baseboard Components

4.3.1.1 PSoC 4

This kit uses the PSoC 4000 family device. PSoC 4 is the architecture of programmable embedded system controllers with an ARM Cortex-M0 CPU. PSoC 4 delivers a programmable platform for embedded applications. The CY8C40 family is the smallest member of the PSoC 4 family of devices and is upward compatible with larger members of PSoC 4.

For more information, refer to the PSoC 4 web page and the PSoC 4000 family datasheet.

Features

- 32-bit MCU Subsystem
 - □ 16-MHz ARM Cortex-M0 CPU
 - Up to 16 KB of flash with Read Accelerator
 - □ Up to 2 KB of SRAM

Programmable Analog

- □ Two current DACs (IDACs) for general-purpose or capacitive sensing applications
- □ One low-power comparator with internal reference

■ Low Power 1.71-V to 5.5-V Operation

- Deep Sleep mode with wake-up on interrupt and I2C address detect
- Capacitive Sensing
 - Cypress Capacitive Sigma-Delta (CSD) provides best-in-class signal-to-noise ratio (SNR) and water tolerance
 - Cypress-supplied software component makes capacitive sensing design easy
 - □ Automatic hardware tuning (SmartSense[™])

Serial Communication

- Multimaster I2C block with the ability to do address matching during Deep Sleep and generate a wake-up on match
- Timing and Pulse-Width Modulation
 - □ One 16-bit Timer/Counter/Pulse-Width Modulator (TCPWM) block
 - □ Center-aligned, Edge, and Pseudo-Random modes
 - Comparator-based triggering of Kill signals for motor drive and other high-reliability digital logic applications
- Up to 20 Programmable GPIO Pins
 - □ 24-pin QFN, 16-pin SOIC, 16-pin QFN, and 8-pin SOIC packages
 - □ GPIO pins on Ports 0, 1, and 2 can be CapSense or have other functions
 - □ Drive modes, strengths, and slew rates are programmable

PSoC Creator Design Environment

- Integrated Development Environment (IDE) provides schematic design entry and build (with analog and digital automatic routing)
- Applications Programming Interface (API) component for all fixed-function and programmable peripherals
- Industry-Standard Tool Compatibility



After schematic entry, development can be done with ARM-based industry-standard development tools

4.3.1.2 PSoC 5LP

An onboard PSoC 5LP (CY8C5868LTI-LP039) is used to program and debug PSoC 4. The PSoC 5LP connects to the USB port of the PC through a USB Mini-B connector and to the SWD interface of the PSoC 4 device.

PSoC 5LP is a true system-level solution providing MCU, memory, analog, and digital peripheral functions in a single chip. The CY8C58LPxx family offers a modern method of signal acquisition, signal processing, and control with high accuracy, high bandwidth, and high flexibility. Analog capability spans the range from thermocouples (near DC voltages) to ultrasonic signals. For more information, refer to the PSoC 5LP web page.

Features

- 32-bit ARM Cortex-M3 CPU core
 - DC to 67-MHz operation
 - Flash program memory up to 256 KB, 100,000 write cycles, 20-year retention, and multiple security features
 - □ Up to 32-KB flash error correcting code (ECC) or configuration storage
 - □ Up to 64-KB SRAM
 - 2-KB electrically erasable programmable read-only memory (EEPROM) memory, 1M cycles, and 20 years' retention
 - 24-channel direct memory access (DMA) with multilayer AHB bus access
 Programmable chained descriptors and priorities
 High-bandwidth 32-bit transfer support
- Low voltage, ultralow power
 - □ Wide operating voltage range: 0.5 V to 5.5 V
 - □ High-efficiency boost regulator from 0.5-V input to 1.8-V to 5.0-V output
 - □ 3.1 mA at 6 MHz (2.7 V to 5.5 V)
 - □ Low-power modes including:
 - $2-\mu$ A sleep mode with real-time clock (RTC) and low-voltage detect (LVD) interrupt 300-nA hibernate mode with RAM retention
- Versatile I/O system
 - □ 28 to 72 I/Os (62 GPIOs, 8 SIOs, 2 USBIOs)
 - □ Any GPIO to any digital or analog peripheral routability
 - □ LCD direct drive from any GPIO, up to 46×16 segments
 - CapSense support on any GPIO
 - □ 1.2-V to 5.5-V I/O interface voltages, up to four domains
 - Maskable, independent IRQ on any pin or port
 - □ Schmitt-trigger transistor-transistor logic (TTL) inputs
 - □ All GPIOs configurable as open drain high/low, pull-up/pull-down, High-Z, or strong output
 - Configurable GPIO pin state at power-on reset (POR)
 - □ 25-mA sink on SIO
- Digital peripherals
 - □ 20 to 24 programmable logic device (PLD)-based universal digital blocks (UDBs)



- □ Full CAN 2.0b 16 RX, 8 TX buffers
- □ Full-Speed (FS) USB 2.0 12 Mbps using internal oscillator
- □ Four 16-bit configurable timers, counters, and PWM blocks
- 67-MHz, 24-bit fixed point digital filter block (DFB) to implement finite impulse response (FIR) and infinite impulse response (IIR) filters
- Library of standard peripherals
 - 8-, 16-, 24-, and 32-bit timers, counters, and PWMs

Serial peripheral interface (SPI), universal asynchronous transmitter receiver (UART), and I²C Many others available in Component catalog available in PSoC Creator IDE

- Library of advanced peripherals
 Cyclic redundancy check (CRC)
 Pseudo random sequence (PRS) generator
 Local interconnect network (LIN) bus 2.0
 Quadrature decoder
- Analog peripherals (1.71 V \leq VDDA \leq 5.5 V)
 - □ 1.024 V ±0.1 percent internal voltage reference across –40 °C to +85 °C
 - Configurable delta-sigma ADC with 8- to 20-bit resolution Sample rates up to 192 ksps
 Programmable gain stage: ×0.25 to ×16
 12-bit mode, 192 ksps, 66-dB signal-to-noise and distortion ratio (SINAD), ±1-bit INL/DNL
 16-bit mode, 48 ksps, 84-dB SINAD, ±2-bit INL, ±1-bit DNL
 - □ Up to two SAR ADCs, each 12-bit at 1 Msps
 - □ Four 8-bit 8 Msps current IDACs or 1 Msps voltage VDACs
 - □ Four comparators with 95 ns response time
 - □ Four uncommitted opamps with 25 mA drive capability
 - Four configurable multifunction analog blocks; example configurations are programmable gain amplifier (PGA), transimpedance amplifier (TIA), mixer, and sample and hold
 - CapSense support
- Programming, debug, and trace
 - □ JTAG (4-wire), SWD (2-wire), single-wire viewer (SWV), and TRACEPORT interfaces
 - Cortex-M3 flash patch and breakpoint (FPB) block
 - □ Cortex-M3 Embedded Trace Macrocell[™] (ETM[™]) that generates an instruction trace stream
 - □ Cortex-M3 data watchpoint and trace (DWT) that generates data trace information
 - □ Cortex-M3 Instrumentation Trace Macrocell (ITM) that can be used for printf-style debugging
 - DWT, ETM, and ITM blocks that communicate with off-chip debug and trace systems via the SWV or TRACEPORT
 - □ Bootloader programming supportable through I²C, SPI, UART, USB, and other interfaces
- Precision, programmable clocking
 - □ 3- to 62-MHz internal oscillator over full temperature and voltage range
 - □ 4- to 25-MHz crystal oscillator for crystal PPM accuracy
 - □ Internal PLL clock generation up to 67 MHz
 - □ 32.768-kHz watch crystal oscillator
 - □ Low-power internal oscillator at 1, 33, and 100 kHz

For more information, see the CY8C58LPxxx family datasheet.

4.3.1.3 Power Supply System

The power supply system on this board is versatile, allowing the input supply to come from the following sources:

- 5-V power from onboard USB programming header J10
- 5-V to 12-V power from Arduino shield using J1_01 (VIN on J1) header
- VTARG power from the onboard SWD programming using J6 or J7
- VIN J11 (not populated by default)
- Coin cell battery BT1 (not populated by default)

The PSoC 4 and PSoC 5LP are powered with either a 3.3 V or 5 V source. The selection between 3.3 V and 5 V is made through the J9 jumper. The board can supply 3.3 V and 5 V to the I/O headers and receive 3.3 V from the I/O headers (J9 should select 3.3 V for this). The board can also be powered with an external power supply through the VIN (J11) header; the allowed voltage range for the VIN is 5 V to 12 V. The LDO regulator regulates the VIN down to 3.3 V. Figure 4-5 shows the power supply block diagram and circuitry. In addition, there is a coin cell battery holder (BT1), which can power the VDD line directly. The allowed voltage range supported through the coin cell battery is between 1.8 V and 5.5 V (VDD specification of PSoC 4000 family). The BT1 holder is not populated on the board by default. BU2032SM-BT-GTR (from Keystone Electronics) can be used for BT1. This part supports CR2032 type coin cell batteries. Refer to the Bill of Materials on page 146 for details on other parts that can be used for BT1.

Note: The 5-V domain is directly powered by the USB (VBUS). For this reason, this domain is unregulated.

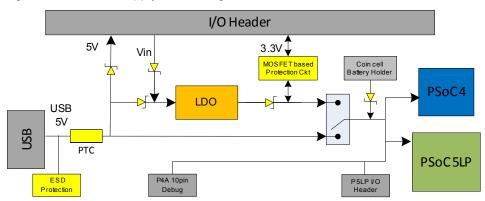
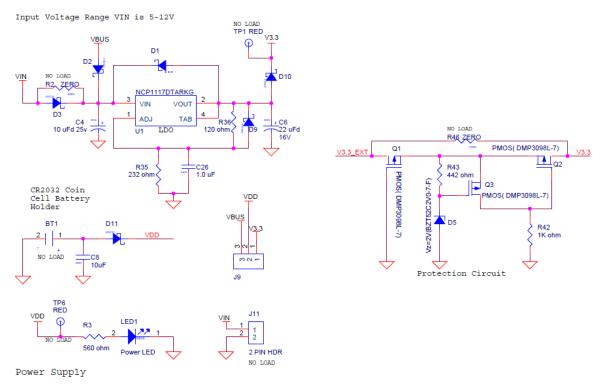


Figure 4-5. Power Supply Block Diagram and Schematic with Protection Circuits





Protection Circuit

The power supply rail has reverse-voltage, overvoltage, short circuits, and excess current protection features, as seen in Figure 4-5.

- The Schottky diode (D1) ensures power cannot be supplied to the 5-V domain of the board from the I/O header.
- The series protection diode (D2) ensures VIN (power supply from the I/O header) does not back power the USB.
- The Schottky diode (D3) ensures 3.3 V from I/O header does not back power the LDO.
- The series protection diode (D4) ensures that the reverse voltage cannot be supplied from the VIN to the regulator input.
- A PTC resettable fuse is connected to protect the computer's USB ports from shorts and over current.
- The MOSFET-based protection circuit provides overvoltage and reverse-voltage protection to the 3.3-V rail. The PMOS Q1 protects the board components from a reverse-voltage condition. The PMOS Q2 protects the PSoC from an overvoltage condition. The PMOS Q2 will turn off when a voltage greater than 4.2 V is applied, protecting the PSoC 4.
- The output voltage of the LDO is adjusted such that it takes into account the voltage drop across the Schottky diode and provides 3.3 V.
- Populating R46 with a zero-ohm resistor will bypass the VIN protection circuitry.

Procedure to Measure PSoC 4 Current Consumption

The following three methods are supported for measuring current consumption of the PSoC 4 device.

When the board is powered through the USB port (J10), remove jumper J13 and connect an ammeter, as shown in Figure 4-6.

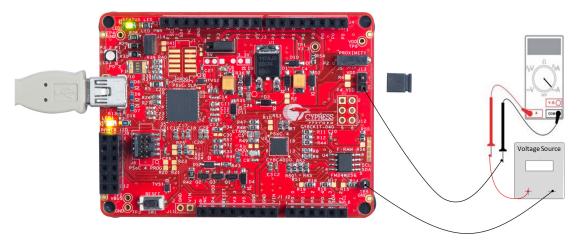




Figure 4-6. PSoC 4 Current Measurement When Powered From USB Port

When using a separate power supply for the PSoC 4 with USB powering (regulator output on the USB supply must be within 0.5 V of the separate power supply), remove jumper J13. Connect the positive terminal of voltage supply to the positive terminal of the ammeter and the negative terminal of the ammeter to the lower pin (P4.VDD) of J13. Figure 4-7 shows the required connections.

Figure 4-7. PSoC 4 Current Measurement When Powered Separately



Note: The RGB tricolor LED is powerd from PSoC 4 VDD only. Remove jumper J14 to measure power consumed by PSoC 4 alone.

4.3.1.4 Programming Interface

The kit allows programming and debugging of the PSoC 4 in two modes:

- Using the Onboard PSoC 5LP Programmer and Debugger on page 19
- Using the CY8CKIT-002 MiniProg3 Programmer and Debugger on page 21

4.3.1.5 Arduino Compatible Headers (J1, J2, J3, J4, and J12)

This kit has five Arduino compatible headers: J1, J2, J3, J4, and J12. You can develop applications based on the Arduino shield's hardware. An Arduino shield compatible Trackpad board is also supplied with the kit.



Figure 4-8. Arduino Header

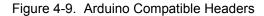


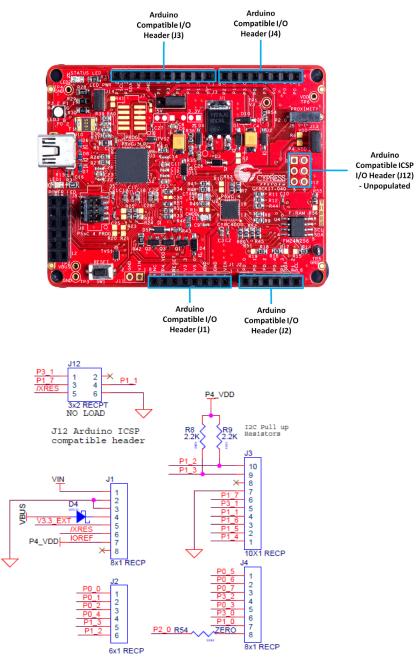
Note: The graphic LCD shield in Figure 4-8 is shown for reference only and not part of the kit.

- The J1 header contains I/O pins for reset and power supply line.
- The J2 header is an analog port. Because of limited analog support in PSoC 4000 family, it contains general-purpose digital I/O pins only.
- The J3 header is primarily a digital port. It contains I/O pins for PWM, I²C and general-purpose digital.
- The J4 header is also a digital port.
- The J12 header is an Arduino ICSP compatible header for the SPI interface. This header is not populated. Refer to the "No Load Components" section of Bill of Materialsfor the header part number.

Note: The PSoC 4000 family does not support SPI in hardware, but SPI master can be implemented on any pin using firmware bit banging.









Functionality of Unpopulated Header J12

The J12 header is a 2×3 header that supports Arduino shields. This header is used on a small subset of shields and is unpopulated on the PSoC 4000 Pioneer Kit.

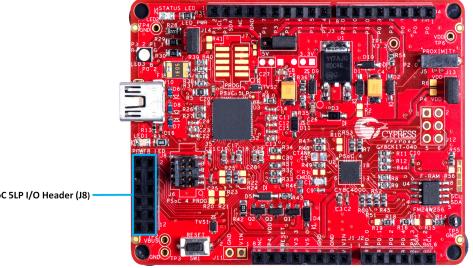
Note: The J12 header functions only in 5 V mode. For proper shield functionality, ensure the power jumper (J9) is connected in 5 V mode.



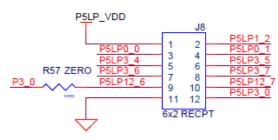
4.3.1.6 PSoC 5LP GPIO Header (J8)

A limited set of PSoC 5LP pins are brought to this header. Refer to Developing Applications for PSoC 5LP on page 119 for details on how to develop custom applications. See Pin Assignment Table on page 141 for pin details.

Figure 4-10. PSoC 5LP GPIO Header (J8)



PSoC 5LP I/O Header (J8)



PSoC 5LP GPIO Extension Header

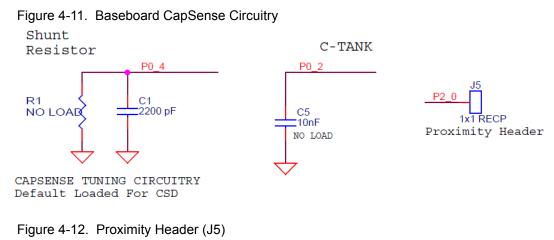
4.3.1.7 CapSense Circuit

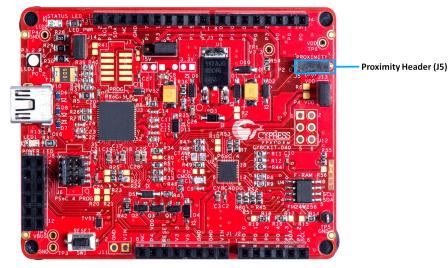
The baseboard contains a header (J5) for CapSense proximity wire connection (see Figure 4-12). A 2.2-nF capacitor (C1) is present on CMOD pin, P0[4], for CapSense operation. An optional resistor R1 can be loaded to convert the current output from IDAC to a voltage output in non-CapSense applications.

The board optionally supports CapSense designs that require waterproofing. Any pin that supports CapSense in the device can be configured as a shield signal to enable waterproof operation. However, if a shield tank capacitor is required in the design, Capacitor C5 (CTANK) on the board needs to be populated with the desired tank capacitor value and R30 connecting the Blue LED to P0 2 needs to be removed. Refer to the CapSense Design Guide for further details related to CapSense.

Note: The kit does not demonstrate the waterproof feature using the Trackpad shield board that ships with the kit because of limited I/O availability after Trackpad and RGB LED implementation. However, a custom shield board can be designed to use the feature.







4.3.1.8 Board LEDs

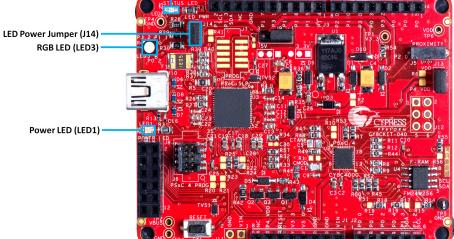
The PSoC 4000 Pioneer Kit board has three LEDs. A green LED (LED2) indicates the status of the programmer. See Error in Firmware/Status Indication in Status LED on page 145 for a detailed list of LED indications. An amber LED (LED1) indicates the status of power supplied to the board. The kit also has a general-purpose tricolor (RGB) LED (LED3) for user applications that connect to specific PSoC 4 pins. Jumper J14 is provided to enable/disable power to the RGB LED (LED3). The RGB LED is powered from PSoC 4 VDD, so jumper J14 needs to be removed to measure PSoC 4 power accurately without leakage and LED power.

Figure 4-13 shows the indication of all these LEDs on the board. Figure 4-14 and Figure 4-15 detail the LED schematic.



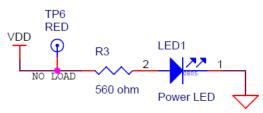


Hardware

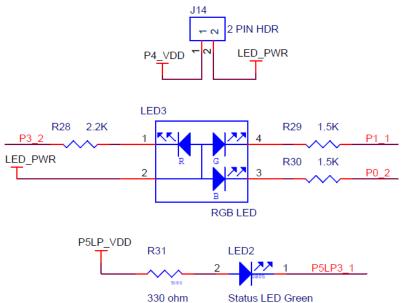


Status LED (LED2)







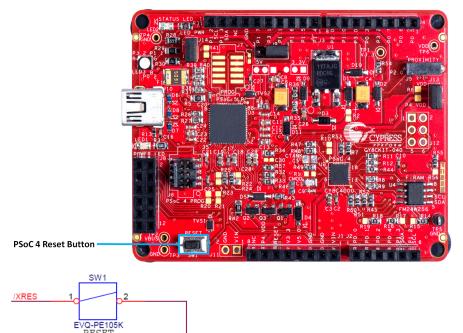




4.3.1.9 Push Buttons

The kit contains only a Reset push button, as shown in Figure 4-16. The Reset button is connected to the XRES pin of PSoC 4 and is used to reset the onboard PSoC 4 device. The push button connects to ground on activation (active low).



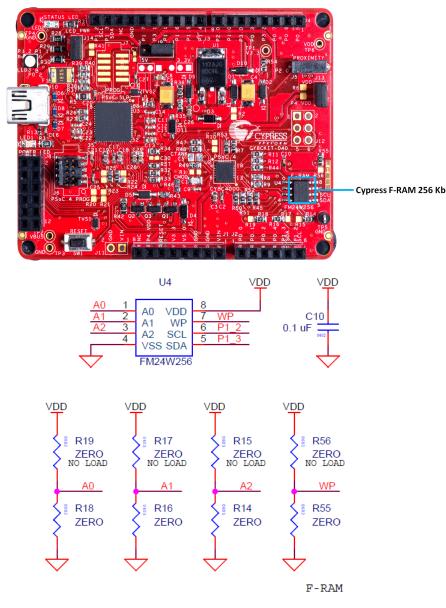


4.3.1.10 Cypress Ferroelectric RAM (F-RAM)

The baseboard contains an F-RAM device (FM24W256); see Figure 4-17, which can be accessed through I²C lines by either of the PSoC devices – PSoC 5 LP or PSoC 4 – or by both. The F-RAM is 256 Kb (32 KB) in size with the I²C speed up to 1 Mbps. The I²C slave address of the F-RAM device is 7-bit wide, and the LSB three bits are configurable through physical pins and are hardwired to 000 on the board. By default, the address of the F-RAM device used on the board is 0x50. This can be modified by changing the R19/R18, R17/R16, and R15/R14 pairs. Refer to Use of Zero-ohm Resistors and No Load on page 144 for details on how to change the F-RAM address using these resistors. The Using FM24W256 F-RAM on page 103 provides an example implementation showing how to use this F-RAM device with PSoC 4 and share it between Bridge Control Panel over the PSoC 5LP USB-I²C bridge.



Figure 4-17. Cypress F-RAM



4.3.2 CY8CKIT-040 CapSense Trackpad Shield Board

The kit also includes an Arduino-compatible CapSense Trackpad shield board. The Trackpad in the kit is a 6x5 elements capacitive sensing array. Figure 4-18 shows the pin mapping of the Trackpad.

The modulation capacitor (Cmod) used for CapSense is connected to pin P0[4], and an optional bleeder resistor (R1) can be connected across the Cmod. The Trackpad shield is Arduino-compatible and can also be used with the PSoC 4 Pioneer Kit (CY8CKIT-042). The sticker on the Trackpad shield can be redesigned according to user requirement and can be pasted on top of the Trackpad to implement any custom, application-specific UI. The Trackpad/Touchpad Sticker Details on page 149 provides the sticker template.

Refer to the CapSense Design Guide for further details related to CapSense.



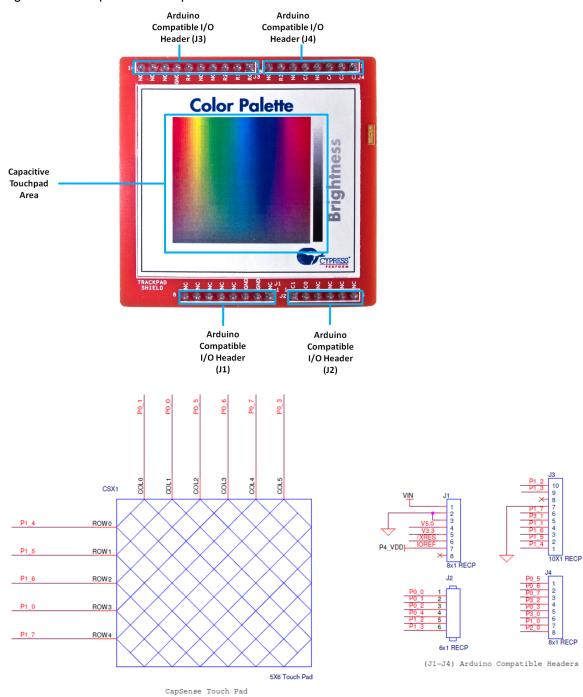


Figure 4-18. CapSense Trackpad Shield Board

5. Example Projects



5.1 Overview

The example projects described in this chapter introduce the functionality of the PSoC 4000 device and the onboard components. To access the examples, download and install the CD ISO image or setup files from the kit web page as explained in Install Software on page 13. The example projects are available at <Install_Directory>\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version>\Firmware\ after installation.

5.1.1 Programming the Example Projects

This section is provided as a reference for programming any example project into PSoC 4 on the board. The description of example projects shipped with the kit is from Project: Blinking LED on page 50. Follow these steps to open and program an example project:

- 1. Launch PSoC Creator from the Windows Start menu (Start > All Programs > Cypress > PSoC Creator<version> > PSoC Creator<version>).
- Open the example project by clicking <*Project_name>.cywrk* below Examples and Kits > Kits > CY8CKIT-040, as shown in Figure 5-1. CY8CKIT_040_Blinking_LED.cywrk is used as reference here.

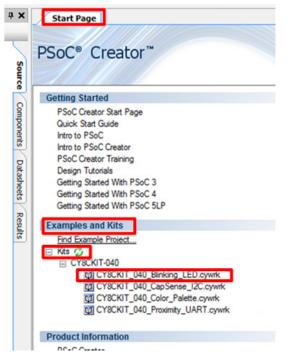
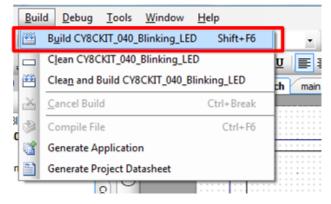


Figure 5-1. Open Code Example from PSoC Creator



3. Build the code example by choosing **Build > Build CY8CKIT_040_Blinking_LED.cywrk** to generate the hex file, as shown in Figure 5-2.

Figure 5-2. Build Project from PSoC Creator



- 4. To program, connect the board to a computer using the USB cable connected to port J10, as described in Kit USB Connection on page 18. The board is detected as KitProg.
- 5. Choose **Debug > Program** from PSoC Creator, as shown in Figure 5-3.

Figure 5-3. Program Device from PSoC Creator

Deb	oug <u>T</u> ools <u>W</u> indow <u>H</u> elp	
	Windows •	1
***	Program Ctrl+F5	
激	Select Debug <u>T</u> arget	1
煮	Debug F5	
羝	Debug without Programming Alt+F5	
	Attach to Running Target	Ŀ
ø	Toggle Breakpoint F9	
	New Breakpoint	
<u>چ</u>	Delete All Breakpoints Ctrl+Shift+F9	H
0	Enable All Breakpoints	H

6. If the device is not yet acquired, PSoC Creator will open the **Select Debug Target** window. Select **KitProg/<ID>** and click the **Port Acquire** button, as shown in Figure 5-4.



Select Debug Target	१
🚰 KitProg/191607A0032C2400	KitProg/191607A0032C2400
	POWER = 3 VOLTAGE_ADC = 5034 FREQUENCY = 200000 PROTOCOL = SWD
	KitProg Version 2.07
Show all targets	Port Setting Port Acquire
	ок

Figure 5-4. Acquire Device from PSoC Creator

7. After the device is acquired, it is listed in a tree structure below the KitProg (see Figure 5-5). Click the **Connect** button.

Select Debug Target	8 ×
E-5 KitProg/191607A0032C2400	PSoC 4000 CY8C4014LQI-422
└─ <i>ॣ</i> PS₀C 4000 CY8C4014LQI-422	PSoC 4000 (ARM CM0) Silicon ID: 0x0BB11477 Cypress ID: 0x0A48129A Revision: PRODUCTION
	Target unacquired
Show all targets	<u>C</u> onnect
	ОК

Figure 5-5. Connect Device from PSoC Creator

8. Click **OK** to exit the window and start programming, as shown in Figure 5-6.



Select Debug Target	ହ <mark>×</mark>
E-5 KitProg/191607A0032C2400	PSoC 4000 CY8C4014LQI-422
PSoC 4000 CY8C4014LQI-422 (Connected)	PSoC 4000 (ARM CM0) Silicon ID: 0x0BB11477 Cypress ID: 0x0A48129A Revision: PRODUCTION Target acquired
Show all targets	Disconnect
	ок

Figure 5-6. Program Device from PSoC Creator

Notes:

- The Debug port is disabled by default in one of the example projects (Project: CapSense Proximity and UART on page 52) because it uses the P3[0] (SWDIO) pin for software TX output. If debug is required, then change the **Debug Select** setting in the *cydwr* file to SWD, as shown in Figure 5-8. and disable software TX in the project by commenting out the TX_ENABLE macro present in the *main.h* file. The example project, Project: Color Palette on page 77, includes a software TX, but the TX port is disabled by default. To enable the TX port, change the **Debug Select** setting in the *cydwr* file to **GPIO** and uncomment the TX_ENABLE macro in the *main.h* file. If TX is required along with SWD debug, then follow these steps:
 - a. Route TX pin to any other available pin by modifying TX_PORT/TX_PIN macro available in main.h file of the projects.
 - b. Remove resistor R57 (Figure 5-7) on the board.
 - c. Route the TX pin selected in step 1 to pin J8_9 (P12[6]/RX line of PSoC 5LP available in J8).
- Reset the device after plugging in the USB cable for the first time (if kit drivers are installing, then after driver installation) when using SmartSense Auto-tuning in the project. This is because SmartSense tunes the sensors during power-on and the presence of hand or power fluctuations during the USB plugging will affect the tuning algorithm; it can render stuck or insensitive touch sensors.
- By default, when the example projects are opened for the first time, an inline error can pop up in the main.c or main.h file against the '#include <project.h>' line. This error is temporary and will go off when the project is built. The project.h file is generated only when the project is built, hence the error is shown before building the project.







R57



ption	Value
- Configuration	
Device Configuration Mode	Compressed 💌
···· Unused Bonded IO	Allow but warn
····· Heap Size (bytes)	0x0100
···· Stack Size (bytes)	0x0400
Include CMSIS Core Peripheral Library Files	
- Programming\Debugging	
Chip Protection	Open 👻
Debug Select	SWD (serial wire debug)
· Operating Conditions	SWD (serial wire debug) GPIO
Vddd (V)	5.0
····· Vdda (V)	5.0
Introlls whether or not to reserve pins for debugging. If SWD is selected, the debugging VO is selected the pins are available for general purpose use. When set to GPIO the dr	

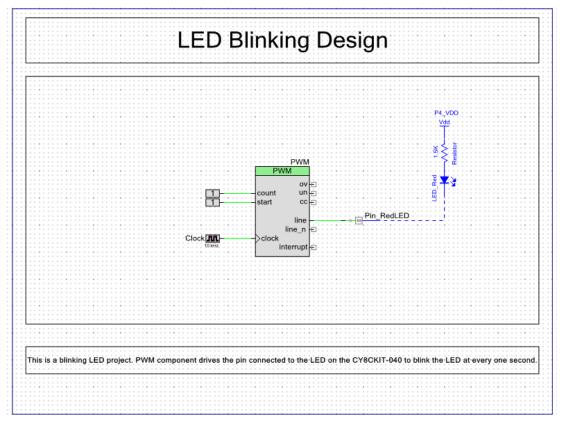


5.2 Project: Blinking LED

5.2.1 Project Overview

The CY8CKIT_040_Blinking_LED.cyprj example uses a PWM block to blink the red LED of the RGB LED, as shown in Figure 5-9. The PWM output is connected to pin P3_2 (red) of the RGB LED. The PWM block is configured as a digital clock signal generator with a frequency of 1 Hz. The blinking rate can be varied by changing the period and compare value of the PWM.

Figure 5-9. PSoC Creator Schematic Design of Blinking LED Project



- 5.2.2 Project Description
- 5.2.2.1 PSoC Creator Component Configuration

PWM (TCPWM mode)

The TCPWM Component is configured as a PWM with the parameters shown in Figure 5-10.



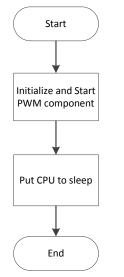
Configure 'TCPWM_P4'							2	X
Name: PWM								
Configuration	PWM Built-in						<	1 Þ
Prescaler:	1x •	Input Pre	ent M	lode				
PWM align:	Left align 🔹	reload		sing edge		-		
PWM mode:	PWM			vel		8		
Dead time cycle:	0	stop		sing edge		- H		
Stop signal event:	Don't stop on kill 🔻	switch count		sing edge vel		8		
Kill signal event:	Asynchronous							
Output line signal:	Direct output		egister	Swap	RegisterBu	π.		
Output line_n signal:	Inverse output	Compare 5			65535	-		
On terminal count								
On compare/capt	ure count							
	PWM, left al	ionod				-		
10000	Pww, ien al	ignea						
449.600								
counter 50	00	/						
ov °	+ + +	+ +						
UN TC TC TC	+ + +	+						
	+ + +	+						
line ——								
line_n								
Datasheet			ОК		Apply		Cancel	

Figure 5-10. TCPWM Component Parameters

5.2.2.2 Firmware Details

Figure 5-11 shows the flow chart of code implemented in main.c.

Figure 5-11. Blinking LED Project Flow Chart





5.2.2.3 Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the board. Open *CY8CKIT_040_Blinking_LED.cydwr* under the **Source** vertical tab in the **Workspace Explorer** and select the suitable pin, as shown in Figure 5-12.

Table 5-1. Pin Connection

Pin Name	Port Name
PWM	P3_2 (Red)

Figure 5-12. Pin Selection for Blinking LED Project

Alias	Name /	Port		Pin	l
	Pin_RedLED	P3[2] TCPWM0:P	▼ 23		•
	-				

5.2.3 Verify Output

Build and program the code example onto the device. Observe the frequency and duty cycle of the blinking LED. Change the period and compare value in the PWM Component, as shown in Figure 5-10. Rebuild and reprogram the device to change the blinking rate.

5.3 Project: CapSense Proximity and UART

5.3.1 Project Overview

The project CY8CKIT_040_Proximity_UART.cyprj implements a capacitive proximity sensor controlling the brightness of a LED. The project configures the sensor as a CapSense proximity widget with SmartSense Auto-tuning. Firmware Details on page 56 presents the firmware flow and explains the firmware blocks in detail.

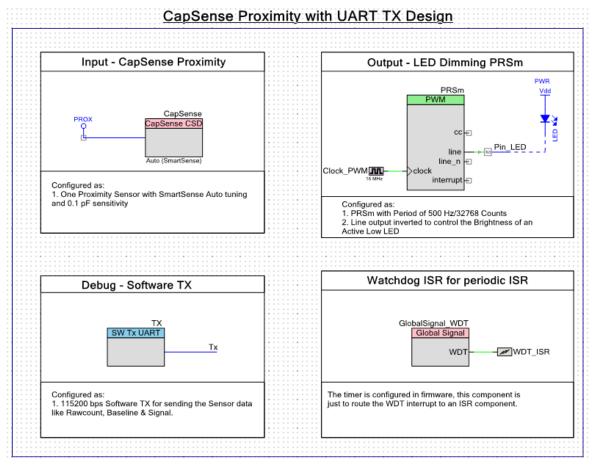


Figure 5-13. PSoC Creator Schematic Design of PWM Project



5.3.2 Project Description

5.3.2.1 PSoC Creator Component Configuration

CapSense

The CapSense Component is configured in SmartSense Auto-tuning mode with one proximity sensor for the design with the parameters shown in Table 5-2.

Table 5-2.	CapSense	Component	Parameters
------------	----------	-----------	------------

Parameter	Tab Present	Value	Rationale
Tuning method		Auto (SmartSense)	Automatically adjust sensitivity for different system environments.
Threshold mode	General	Automatic	To enable run-time threshold calculation for 5:1 SNR
Raw data noise filter		First Order IIR 1/4	Filter out noise or unwanted spikes in raw count. This setting can be tweaked based on require- ment.
ProximitySensor0	Widgets Config	-	Add a proximity sensor by clicking Proximity Sensors and then clicking Add Proximity Sensor . The only parameter that is available to modify in this tab is debounce. This can be set or adjusted based on system requirements.
Analog switch drive source		PRS-12b	Reduce EMI emission and enhance EMC immu- nity.
Sensor auto-reset		Disabled	Not required in the design. Add if required by the application.
Low baseline reset		5	System dependent number. Configure according to user needs.
Inactive sensor con- nection	Advanced	Ground	Make the proximity loop not pick up any charge when not scanned
Shield		Disabled	Not used in the design.
Guard sensor]	Disabled	Not used in the design.
Cmod precharge		Precharge by Vref buffer	Vref is enough for precharging, as there is only one sensor. Cmod voltage will not drop too low for a fast GPIO precharge.
Sensitivity	Scan Order	1	Obtain the maximum possible sensitivity using SmartSense. The parameter controls the scan time, so for a lower number sensitivity setting, the scan rate will be higher. This parameter can be adjusted depending on the response rate and proximity range needed.
Enable Tuner helper	Tuner helper	Unchecked	No tuner used.

PRSm (TCPWM Mode)

The TCPWM Component is used to control the LED brightness. The CapSense proximity sensor's signal output is used to deduce the LED brightness. Figure 5-14 shows the parameters for the TCPWM Component. The TCPWM block is configured as a PWM in Pseudo Random Sequence modulator (PRSm) mode with a resolution of 15 bits (fixed by the TCPWM block architecture). This 15-bit resolution of the PRSm along with a 16-MHz input clock generates a period of 500 Hz (PRS repeat period). The output line is inverted to drive the Active Low LED. A period of 32767 is set in the



Component to generate the proper period macro for the 15-bit PRSm. Though the output of PRSm has a variable frequency with a maximum frequency of 8 MHz (16 MHZ/2), the repeat rate of PRSm is considered to be the period in this context.

Note: The Compare value should be a minimum of '1'; '0' will leave the LED on.

Figure 5-14. TCPWM Component Configuration - 'PWM' Tab

Configure 'TCPWM_P4'						? X
Name: PRSm						
Configuration	PWM Built-in					۹ ۵
Prescaler:	1x 🔻	Input	Present	Mode		
PWM align:	Left align 👻	reload		Rising edge	-	
PWM mode:	Pseudo random PWM	start		Rising edge	-	
Run mode:	Continuous 🔻	stop		Rising edge	-	E
		switch		Rising edge	-	_
Stop signal event:	Don't stop on kill ▼	count		Level	-	
Kill signal event:	Asynchronous 🔻		Regist	er Swap	RegisterBuf	1
Output line signal:	Inverse output	Period	32767		65535	
Output line_n signal:	Direct output	Compare	e 1		65535	
Interrupt						
On terminal count						
On compare/capt	ure count					
						Ŧ
Datasheet		ОК		Apply		Cancel

TX (Software TX UART)

The software transmit TX is used to send out proximity sensor related data for debugging. The configuration for the component is shown in Figure 5-15. The TX pin is selected in firmware through TX_PORT/TX_PIN macros defined in the *main.h* file. The SW TX data can be sent to the PC using either an RS232 connector (with a voltage level translator in between) or through the USB-UART bridge available in the CY8CKIT-040 PSoC 5 LP UART bridge, or the CY3240 bridge configured as a UART bridge as documented in AN2397.



onfigure 'SW_Tx_UART'	8	X
Name: TX		
Basic Built-in		4 Þ
Parameter	Value	
BaudRate	115200	
PinAssignmentMethod	Dynamic	
	Dynamic	
Parameter Information	Dynamic	

Figure 5-15. Software UART TX Component Parameters

Pin_LED (Digital Output Pin)

The digital output pin is used to drive the PWM output to the LED. It is a standard strong drive output pin.

Clock_PWM (Cy_Clock)

Clock_PWM provides the clock that drives the PWM block. The clock is configured to be the maximum possible or allowed (16 MHz), so that the repeat rate of the PRSm is as high as possible for reduced LED flickers.

GlobalSignal_WDT (Global Signal Reference with ISR)

Component used to route the WDT ISR to an ISR component. This ISR is then configured in firmware for generating periodic wakeup signal using WDT during Sleep_Scan mode.

5.3.2.2 Firmware Details

Firmware Structure

The firmware is written in a modular format, with different aspects of the functionality provided as separate functions for easy understanding. The header provides a list of handy macros for configuring the project's key aspects according to user requirements. The comments in the header file provide the details on the macro.

File Names	ile Names Purpose				
main.c This file contains all the function definitions used in the firmware.					
main.h	This file contains all the macros used in the firmware. The details of the macros and their usage can be found in the comments above each macro.				

Firmware Flow Chart

Figure 5-16 shows the flow chart of code implemented in main.c.



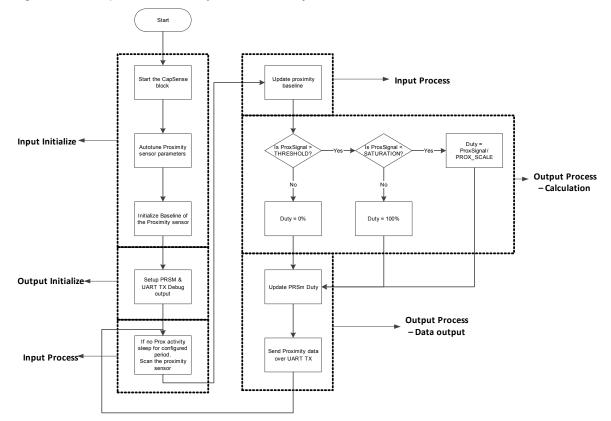


Figure 5-16. CapSense Proximity and UART Project Flow Chart

The firmware does the following:

- The CapSense/input initialization part tunes the CapSense system parameters using SmartSense.
- The output initialization part configures the PWM and the software UART TX output.
- The infinite loop code is divided into two phases: input process and output process.
 - The input process phase scans the proximity sensor and processes the sensor signal, such as applying filter, calculating baseline, and signal.
 - The output process phase is also split into two phases: the data calculation and data output subphases.

i. The data calculation subphase compares the proximity signal with a minimum and maximum threshold defined for an approaching hand. The LED brightness is then calculated based on the sensor's signal value relative to the thresholds. The minimum threshold generates the lowest LED brightness, and the maximum threshold generates the highest LED brightness.

ii. The data output subphase updates the PWM compare value with the calculated brightness. The system data such as sensor raw data, baseline, signal, and calculated LED brightness are sent over the UART TX line.

The device monitors the activity on the proximity sensor, and if there is no activity, that is, if the hand is out of range of the proximity, then the device enters a sleep-scan mode. The time for which the device checks for a no activity on the sensor before entering sleep-scan mode is set to 5 seconds and is configurable in the project (ENTER_SLEEP_COUNTS macro in *main.h*). In the sleep-scan mode, the device wakes up every 100 ms and checks for any activity on the proximity sensor. This



wakeup rate is configurable by modifying the WATCHDOG_TIMER_COUNT macro in the *main.h* file.

Note: There are macros defined in *main.h* for configuring the project based on user requirements. The details of each macro are mentioned in comments above its definition.

Useful CapSense Component Functions/Variables

Table 5-4, Table 5-5, and Table 5-6 provide details of some useful component APIs, variables, and macro definitions. These details are also available in the CapSense component datasheet.

Note: All reference to the API/variable/macro assume the CapSense component instance name as 'CapSense'.

Table 5-4 provides a quick reference to some key CapSense component variables/arrays.

Variable/Array Name	Description	Usage		
uint16 CapSense_SensorRaw []	This array contains the raw data for each sensor. The array size is equal to the total number of sen- sors (CapSense_TOTAL_SENSOR_COUNT). The CapSense_SensorRaw [] data is updated by these functions: • CapSense_ScanSensor() • CapSense_ScanEnabledWidgets() • CapSense_InitializeSensorBaseline() • CapSense_InitializeAllBaselines() • CapSense_UpdateEnabledBaselines()	The variable can be accessed in any file by importing it using 'extern uint16 CapSense_SensorRaw[];' in the '.c' or '.h' file. It is recommended not to alter the arrays manually.		
uint16 CapSense_SensorBas eline[]	This array holds the baseline data of each sensor. The historical count value, calculated indepen- dently for each sensor is called the sensor's base- line. The array's size is equal to the total number of sensors. The CapSense_SensorBaseline[] array is updated by these functions: • CapSense_InitializeSensorBaseline() • CapSense_InitializeAllBaselines() • CapSense_UpdateSensorBaseline() • CapSense_UpdateEnabledBaselines().	The variable can be accessed in any file by importing it using 'extern uint16 CapSense_SensorBaseline[];' in the '.c' or '.h' file. It is recommended not to alter the arrays manually.		
uint8 CapSense_SensorSign al[]	This array holds the sensor signal count com- puted by subtracting the previous baseline from the current raw count of each sensor. Each array element corresponding to a sensor will have the difference value only if the value is above the noise threshold of the sensor. Otherwise, it will be 0. The array size is equal to the total number of sensors. The CapSense_SensorSignal[] array is updated by these functions: • CapSense_InitializeSensorBaseline() • CapSense_InitializeAllBaselines() • CapSense_UpdateSensorBaseline() • CapSense_UpdateEnabledBaselines().	The variable can be accessed in any file by importing it using 'extern uint8 CapSense_SensorSignal[];' in the '.c' or '.h' file. It is recommended not to alter the arrays manually.		

Table 5-4. CapSense Component Key Variables



Table 5-5 provides a quick reference to some key CapSense component macro definitions.

Table 5-5. CapSense Component Macros

Macro Format	Sample	Description		
CapSense_TOTAL_SENS OR_COUNT	_	Defines the total number of sensors within the CapSense component.		
CapSense_SENSOR_"WI DGET_NAME"_< <i>ELE- MENT+ELEMENT_NUMB</i> <i>ER</i> >"WIDGET_TYPE"	• CapSense_SENSOR_TP1_ROW0TP • CapSense_SENSOR_LS0_E0LS • CapSense_Sensor_BTN1BTN	The constant denotes the sensor number of a sensor in the CapSense block. WIDGET_TYPE: BTN – buttons LS – linear sliders RS – radial sliders TP – touchpads PROX – proximity sensors MB – matrix buttons GEN – generic sensors GRD – guard sensors		
CapSense_"WIDGET_NA ME""WIDGET_TYPE"	• CapSense_TP1TP • CapSense_LS0LS	The constant denotes the widget number of a widget in the CapSense block.		
	CapSense_BTN1BTN			

Table 5-6 provides a quick reference to some key CapSense component APIs and their usage.

API	Description/Usage			
void	The API enables the selected widget sensors to be part of the scanning process.			
CapSense_EnableWidget(uint32 widget)	Proximity widgets are disabled by default in the component; the user needs to call this API along with the proximity widget number to enable the same to be included in the scanning process.			
void CapSense_Start(void)	The API enables the CapSense block and tunes the sensors if SmartSense or Auto-calibration is used.			
	It should be called before using the CapSense block for sensing.			
void	The API initializes the CapSense_sensorBaseline[] array with values obtained by scanning all sensors.			
CapSense_InitializeAllBaselines(voi d)	It should be called after starting the CapSense block and before starting the scan for detecting touches.			
void CapSense_InitializeSensorBaseline	The API initializes the CapSense_sensorBaseline[sensor] array ele- ment with values obtained by scanning the selected sensor.			
(uint32 sensor)	It can be used initialize each baseline individually.			
void CapSense_ScanEnabledWidgets(v oid)	The API starts scanning a sensor within the enabled widgets. The ISF continues scanning sensors until all enabled widgets are scanned. Us of the ISR ensures this function is non-blocking. After each sensor sca is complete, the ISR copies the measured sensor raw data to the CapSense SensorRaw[] array.			
	This is the preferred scanning method if there are multiple widgets in the design.			



Table 5-6. CapSense Component APIs

API	Description/Usage			
void CapSense_ScanWidget(uint32	The API sets the CapSense block settings for the selected widget and starts scanning the widget.			
widget)	It can be used if scanning of only an individual widget is desired.			
void CapSense_ScanSensor(uint32 sensor)	The API scans the selected sensor. After scanning is complete, the ISR copies the measured sensor raw data to the CapSense_SensorRaw[sensor] array element.			
	It can be used to perform individual sensor scanning.			
	The API returns the status of sensor scanning.			
uint22 CanSanaa JaBuay(yaid)	After calling any of the scan APIs, this API can be used to check whether the triggered scan is complete.			
uint32 CapSense_IsBusy(void)	Returns			
	'1' if scan is in progress			
	'0' if scan is complete			
void CapSense_UpdateSensorBaseline(uint32 sensor)	The API filters the CapSense_SensorRaw[sensor] element using the fil- ter selected in the component. It updates the CapSense_SensorBaseline[sensor] element using a low-pass filter with k = 256 on the filtered CapSense_SensorRaw[sensor] value.			
	This API should be called after completion of the ScanSensor() API, before checking for any activity on the sensor.			
void CapSense_UpdateEnabledBaseline	The API applies selected filter to the CapSense_SensorRaw[] array and updates the CapSense_SensorBaseline[] array of all the sensors present in the enabled widgets.			
s(void)	This API should be called after the completion of the ScanEnabledWid- gets() API before checking for any activity on any of the sensors.			
	The API compares the selected sensor CapSense_Signal[] array value to its finger threshold. Hysteresis and debounce are applied to determine if a sensor in the selected widget is active.			
uint32 CapSense_CheckIsWidgetActive(ui	This API should be called after the UpdateSensorBaseline() or Upda- teEnabledBaselines() API to check if any sensor in the widget is active.			
nt32 widget)	Returns			
	'1' if one or more sensors within the widget are active			
	'0' if all sensors within the widget are inactive			
	The API performs the same task as CapSense_CheckIsWidgetActive() on all the enabled widgets.			
uint32 CapSense_CheckIsAnyWidgetActiv	This API should be called after the UpdateEnabledBaselines() API or after updating the baseline of all enabled sensors/widgets, to check if any of the sensors is active in all the enabled widgets.			
e(void)	Returns			
	'1' if any widget is active			
	'0' if all the widgets are inactive			

5.3.2.3 Hardware Connections

A wire in the form of a loop is connected to jumper J5 (P2_0), as shown in Figure 5-17. To enable the UART TX connection to the PSoC 5LP USB-UART bridge, make sure R57 is populated on the board (by default it is populated on the board). No other hardware connections are required for this project. All other connections are hardwired on the board.



Note: The proximity distance depends on the diameter of the wire loop. The larger the diameter, the greater the distance. Take care while creating the loop because a larger loop tends to pick up more noise. If the wire shipped with the kit (4 inches in length) is wound to form a loop of 1 to 2 inch diameter, the proximity range will be approximately the same as the loop diameter for a fast approaching hand. To obtain a higher range, use a longer wire/bigger loop. Also, do not plug the wire loop after the device is programmed/powered, as the firmware tunes the proximity sensor during reset. Plugging the wire after a reset will be detected as change in capacitance, and the LED will be always on. Always do a reset after plugging in the wire loop if the device was already programmed.

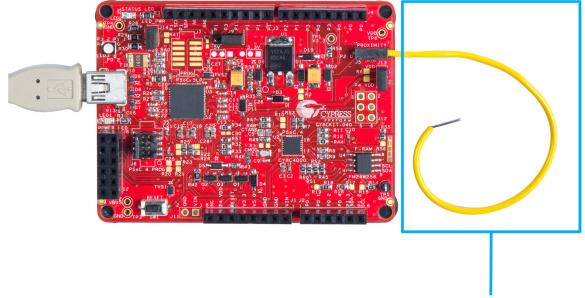


Figure 5-17. CapSense Proximity Example - Hardware Setup

Proximity Loop

Open CY8CKIT_040_Proximity_UART.cydwr in the Workspace Explorer and select the suitable pins, as shown in Figure 5-18.

Pin Name	Port Name		
Proximity Pin	P2_0		
CMOD Pin	P0_4		
LED Pin	P3_2 (Red)		
UART TX Pin	P3_0 ¹		

1. Selected in firmware

Figure 5-18. Pin Selection for Proximity Project

2		- 4 ▷	×
Alias	Name /	Port	
Cmod	\CapSense:Cmod\	P0[4] EXTCLK	-
ProximitySensor0_0_PROX	\CapSense:Sns\	P2[0]	-
	Pin_LED	P3[2] TCPWM0:P	-



5.3.3 Verify Output

Build and program the code example, and reset the device. Observe the red LED intensity changing as you move your palm toward the proximity wire loop. The UART TX data can be viewed through the BCP, as explained in 5.3.3.1 UART Data Viewing.

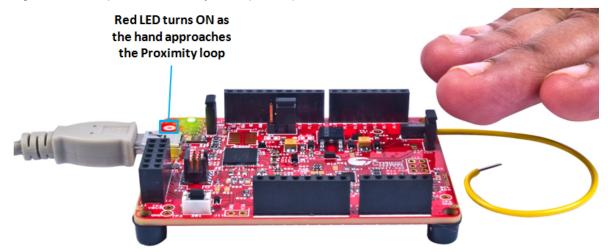


Figure 5-19. CapSense Proximity Example Output

5.3.3.1 UART Data Viewing

One UART packet size is 13 bytes, which includes 8 bytes of data, 2 bytes of header, and 3 bytes of footer. The 2-byte header precedes the data bytes; in the design it is 0x0D and 0x0A. The 3-byte footer follows the data bytes and in this design consists of 0x00, 0xFF, and 0xFF. The data bytes consists of proximity sensor raw counts (RC), baseline (BL), and signal (SIG) along with the calculated PWM duty (DUTY). Table 5-8 shows the UART TX data packet structure.

Header		Data				
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6
0x0D	0x0A	RC_MSB	RC_LSB	BL_MSB	BL_LSB	SIG_MSB
Data				Footer	_	
BYTE 7	BYTE 8	BYTE 9	BYTE 10	BYTE 11	BYTE 12	
SIG_LSB	DUTY_MSB	DUTY_LSB	0x00	0xFF	0xFF	

Table 5-8. UART TX Data Packet Structure

Follow these steps to set up the BCP for viewing the data:

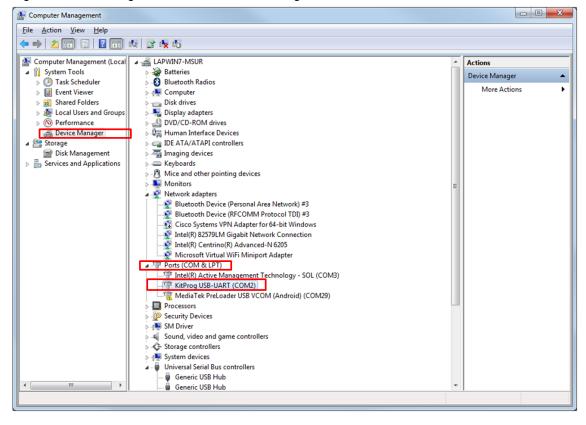
- Open the BCP software available from All Programs > Cypress > Bridge Control Panel <version> > Bridge Control Panel <version>.
- 2. Route the TX pin of the device to any available RX that can connect to the PC COM port. CY3240 (refer to AN2397), or KitProg in CY8CKIT-040 can be used for this purpose.

Note: In CY8CKIT-040, Pin 3_0 is routed directly to the RX pin of the PSoC 5 LP USB-UART bridge through a zero-ohm resistor. Pin 3_0 can be used as TX, and is available only by disabling the debug feature of the chip. If the debug feature is required along with TX, then the zero-ohm resistor connecting the TX and RX should be removed. Any free GPIO can be used as the TX for the device and can be routed to the RX pin of PSoC 5 LP externally (see the last step of Programming the Example Projects on page 45 for details)



3. In the BCP software, click the COM port to which you have connected the data. In this case, it is KitProg's COM, as shown in Figure 5-20.

Figure 5-20. KitProg COM Port in Device Manager



4. Select the **COM** port and **RX8** as the protocol, as shown in Figure 5-21.



Ũ	•						
羄 Brid	dge Control Panel						
<u>F</u> ile	<u>E</u> ditor <u>C</u> hart Execute <u>T</u> e	ools <u>H</u> elp					
B	∎ <u>@</u> 10 10 ⊘ ⊑ ∎	<u>≠ 18. 187</u>					
Editor	Chart Table File]
							~
							-
٠.							Þ
	ect Port in the Po ning Port	rtList, then tr	y to connec	t			^
Suc	icessfully Connecte 2 Serial Port	d to COM2					E V
			Connected I2C/SPI/R		- Pov	vor	Protocol
	Reset St.:List Send	Send all strings: Repeat count: Scan period, ms:	KtProg/191607A0032 COM2 COM2	C2400		+5.0V +3.3V +2.5V +1.8V	RX8 (UART)
1:1	Syntax : OK		Connected		Voltage: -		.4

Figure 5-21. Bridge Control Panel - COM Port and Protocol Selection

 Choose Tools > Protocol Configuration or press [F7] and configure the RX8 protocol parameters, as shown in Figure 5-22.

Protocol Configuration		
SPI I2C	RX8 (UART)	
Bit rate (bps):	115,200 👻	
Data bits:	8 🔻	
Parity type:	None	
Stop bit:	One 🔻	
Flow control:	None	
	OK Cancel	
	Cancer	

Figure 5-22. RX8 Protocol Configuration

6. Choose Chart > Variable Settings and set the variable names and types, as shown in Figure 5-23. Or click Load and then select the CapSense Proximity UART - Variable.ini file supplied with the project (...\Firmware\PSoC 4\Bridge Files\) in the Open window that appears. Click OK to exit.



Figure 5-23.	Bridge Control Panel - Vari	iable Settings
--------------	-----------------------------	----------------

	Dpen	
🚱 Variable Settings	C:\Program Files (x86)\Cypress\CY8CKIT-040 PSoC 4000 Pioneer Kit\1.0\Fi	rmware\PSOC 4\Bridge File
Variables Rags	Organize New folder	8 · · 1
N Active Variable Name Tops Sign Scale Offset Color V RevCourt int 1 0 Block II 2 V Beschen int 1 0 Block II 3 V Signal int 1 0 Marcon 4 V Duty int 1 0 Red 5 Kry5 byte 1 0 BlockVolet 6 Kry5 byte 1 0 LawnGreen	Favorites Forwinds CapSense Color Palette (Minimal TX) - Variables.ini CapSense Proximity UART - Variable.ini CapSense Proximity UART - Variable.ini CapSense Proximity UART - Variable.ini	Date modified Type Size 2/15/2014 5:33 PM Configuration sett 8 KB 2/14/2014 2:15 PM Configuration sett 8 KB
7 Key7 byte 1 0 Magenta 8 Var8 byte 1 0 Magenta Print packet every 1 0 AddxX is a count IV Acto Range of AddxY Sonal 0 ∞ AddxX is a time Min Max 500	 J Music Pictures Podcasts Videos 	
Lood Save VK X Cancel	Image: Computer Computer Image: Computer Image: Computer Image: Computer File game	ini (*in) Qpen Cancel

7. Choose File > Open File and select the CapSense Proximity UART - Editor.iic file supplied with the project from the ...\Firmware\PSoC 4\Bridge Files\ folder (as shown in Figure 5-24). Alternatively, go to Editor and type or copy the following command: rx8 [h=0D 0A] @1RawCount @0RawCount @1Baseline @0Baseline @1Signal @0Signal @1Duty @0Duty [t=00 ff ff]

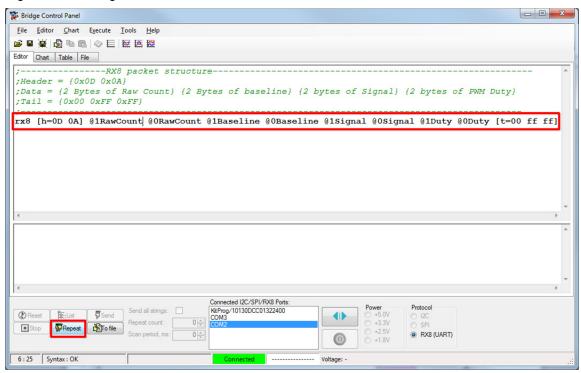
Figure 5-24. Open *.iic File in Bridge Control Panel

		🗱 Bridge Control Panel	Sa ma
	W Bridge Control	File Editor Chart Execute	e Tools
	Eile Editor OR		rl+B a
		🗋 🗃 Open I2C MBM File Ctrl	
		🕇 🚅 Open File Ctr	l+O -
🎬 Open Script File			×
C:\Program	m Files (x86)\Cypress\CY8CKIT-040 PSOC 4000 Pioneer Kit\Firr	nware\PSOC 4\Bridge Files 🔻 🔶 🛛 Sear	rch Bridge Files 🔎
Organize 🔻 New fol	der		i 🕶 🗖 🔞
📃 Desktop 🔦	Name	Date modified Type	Size
Downloads	CapSense Proximity UART - Editor.iic	2/14/2014 2:15 PM IIC File	1 KB
SkyDrive			
📜 Libraries			
Documents			
J Music			
E Pictures ⁽ 型 ⁹⁾ Podcasts			
Videos			
🖳 Computer			
🚢 Windows7_OS (I			
😽 Lenovo_Recover			
😡 msur (\\simba3) 🔻			
File	name: CapSense Proximity UART - Editor.iic	▼ IIC Fil	les (*.iic) 🔻
			pen 🔽 Cancel

8. Click the command line and then click **Repeat**, as shown in Figure 5-25 to start receiving the packets (make sure you have powered the device and programmed with the project's firmware, the TX is connected to the RX line of the COM, and the COM port is selected in the BCP).







9. You should start receiving data. Click the Chart tab to view the graph, as shown in Figure 5-26.
Figure 5-26. Bridge Control Panel - Chart for Viewing Debug Data
Signal





LED Intensity (PRSm duty cycle)



Raw count and Baseline





5.4 **Project: CapSense Touchpad with I²C Tuner**

5.4.1 Project Overview

The project *CY8CKIT_40_CapSense_I2C.cyprj* demonstrates the implementation of a CapSense Trackpad using SmartSense and an EzI2C-based CapSense Tuner window for viewing the Trackpad coordinates. The project is a simple implementation using SmartSense (minimal tuning). The EzI2C block of PSoC 4000 is interfaced through the PSoC 5LP based USB-I²C bridge to the PC GUI. The project uses the SmartSense feature, which sets all CapSense parameters to the optimum values automatically. The parameter settings can be monitored in the GUI but cannot be altered. In the manual tuning method, parameter settings can be changed in the GUI, and the resulting output can be seen (refer to the CapSense Design Guide and CapSense Component datasheet for more details on manual tuning).

Note: This project requires the Trackpad shield board to be plugged into the PSoC 4000 Pioneer Kit baseboard, as shown in Figure 5-27.

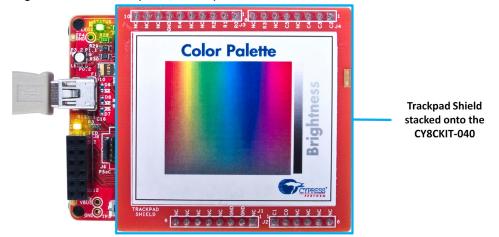


Figure 5-27. Kit Setup With Trackpad



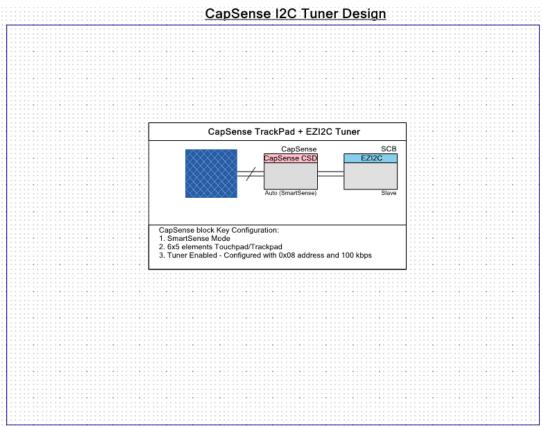


Figure 5-28. PSoC Creator Schematic Design of CapSense Trackpad Project with I²C Tuner



5.4.2 Project Description

5.4.2.1 PSoC Creator Component Configuration

CapSense

The CapSense Component is configured in SmartSense Auto-tuning mode with a 6x5 CapSense touchpad widget for the design, according to the parameters given in Table 5-9.

Parameter	Tab Present	Value	Rationale
Tuning method		Auto(SmartS ense)	Automatically adjust sensitivity for different system environ- ments.
Threshold mode	General	Automatic	Enable run-time threshold calculation for 5:1 SNR
Raw data noise filter		First Order IIR 1/4	Filter out noise or unwanted spikes in raw count. This setting can be tweaked based on requirement.
TouchPad0	Widgets Con- fig	-	Add and configure the touchpad as shown in Figure 5-29.
Analog switch drive source		PRS-Auto	Reduce EMI emission and enhance EMC immunity.
Sensor auto- reset		Disabled	Not required in the design. Add if required by the application.
Low baseline reset		5	System-dependent number. Configure according to user needs.
Inactive sensor connection	Advanced	Ground	Make the proximity loop not pick up any charge when not scanned.
Shield		Disabled	Not used in the design.
Guard sensor		Disabled	Not used in the design.
Cmod precharge		Precharge by Vref buffer	Vref is enough for precharging, as there is only one sensor. Cmod voltage will not drop too low for a fast GPIO precharge.
Sensitivity	Scan Order	5	Select all the sensors by pressing and holding [CTRL] or [Shift] key and clicking on all the sensors. When selected, enter a value in the Sensitivity field.
Enable Tuner helper	Tune helper	Enabled/ Checked	Name = 'SCB'



ame: CapSense				
General Widgets Cor	nfig	Scan Order Advanced Tune Help	er Built-in	4
🗄 Add touchpad 🛛 🔣 Ren	nove	Rename		
he Component is configured in	n the Sm	artSense mode, so some widget properti	es are not available.	
Buttons	E	Column Tuning		
🔤 🔤 Linear sliders		Column Finger threshold (FT)	100	
Radial sliders		Column Noise threshold	20	
Matrix buttons		Column Scan resolution	12 bits	
🖃 🛃 Touchpads		General		
Touchpad0		Number of sensor rows	5	
Proximity sensors		Number of sensor columns	6	
Generics		Row API Resolution	100	
		Column API Resolution	120	
		Position Noise Filter	Averaging	
		Row Tuning		
		Row Finger threshold (FT)	100	
		Row Noise threshold	20	
		Row Scan resolution	12 bits	

Figure 5-29. CapSense Touchpad Parameters

SCB (EzI2C Mode)

The Serial Communication Block (SCB) configured in EzI2C mode is used for the CapSense Tuner. The parameters of the component are shown in Figure 5-30.

Figure 5-30. SCB (EZI2C Mode) Component Parameters

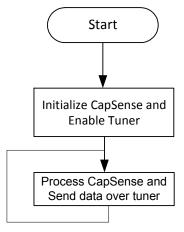
Configure 'SCB_P4'	? ×
Name: SCB	
Configuration EZI2C Built-in	4 ۵
Data rate (kbps): 100 - Actual data rate (kbps): 100	
Oversampling factor: 16	
Clock from terminal	
Clock stretching	
V Median filter	
Number of addresses: 1	
Primary slave address (7-bits): 0x08	
Secondary slave address (7-bits): 0x09	
Sub-address size (bits):	
Enable wakeup from Sleep Mode	
Datasheet OK Apply	Cancel
	.#



5.4.2.2 Firmware Details

Figure 5-31 shows the flow chart of code implemented in main.c.

Figure 5-31. CapSense Touchpad With I²C Tuner Project Flow Chart



5.4.2.3 Hardware Connections

Plug the Trackpad shield board into the Arduino headers of the kit, as shown in Figure 5-27. Other connections are hardwired on the board.

Open CY8CKIT_040_CapSense_I2C.cydwr in the Workspace Explorer and select the suitable pin.

Pin Name	Port Name
CMOD	P0_4
Trackpad_X0	P0_3
Trackpad_X1	P0_7
Trackpad_X2	P0_6
Trackpad_X3	P0_5
Trackpad_X4	P0_0
Trackpad_X5	P0_1
Trackpad_Y0	P1_4
Trackpad_Y1	P1_5
Trackpad_Y2	P1_6
Trackpad_Y3	P1_0
Trackpad_Y4	P1_7
I2C_SCL	P1_2
I2C_SDA	P1_3

Table 5-10. Pin Connection¹

1. Trackpad_X0 (Touchpad0_Col0_TP) to Trackpad_X5 (Touchpad0_Col5_TP) maps to COL5 to COL0 of the Trackpad so as to make the Trackpad x-axis left aligned.



Alias	Name /	Port	_	Pin		Lock
Cmod	\CapSense:Cmod\	PO[4] EXTCLK	-	5	-	V
Touchpad0_Col0TP	\CapSense:Sns[0]\	P0[3]	-	4	-	
Touchpad0_Col1TP	\CapSense:Sns[1]\	P0[7]	-	11	-	
Touchpad0_Col2TP	\CapSense:Sns[2]\	P0[6]	-	10	-	V
Touchpad0_Col3TP	\CapSense:Sns[3]\	P0[5]	-	9	-	
Touchpad0_Col4TP	\CapSense:Sns[4]\	P0[0]	-	1	•	V
Touchpad0_Col5TP	\CapSense:Sns[5]\	P0[1]	-	2	-	V
Touchpad0_Row0TP	\CapSense:Sns[6]\	P1[4]	-	16	-	V
Touchpad0_Row1TP	\CapSense:Sns[7]\	P1[5]	-	17	-	V
Touchpad0_Row2TP	\CapSense:Sns[8]\	P1[6] TCPWM0:N	-	18	-	V
Touchpad0_Row3TP	\CapSense:Sns[9]\	P1[0]	-	12	-	V
Touchpad0_Row4TP	\CapSense:Sns[10]\	P1[7] EXTCLK	-	19	-	V
	\SCB:scl\	P1[2] SCB0:I2C:SCL	-	14	-	V
	\SCB:sda\	P1[3] SCB0:I2C:SDA	-	15	-	V
`an Can						
III						

Figure 5-32. Pin Selection for CapSense I²C Project

5.4.3 Verify Output

Build and program the code example and reset the device. Launch the CapSense Tuner window as explained in the following steps.

5.4.3.1 Launching Tuner Window

The Tuner window from PSoC Creator must be up and running for the code example to work. To launch the GUI, follow these steps:

1. Go to the project's *TopDesign.cysch* file, as shown in Figure 5-33.

Figure 5-33. Top Design File

Workspace Explorer (1 project)
1 C
Workspace 'CY8CKIT_040_CapSense_I2C' (1 Projects)
E Project 'CY8CKIT_040_CapSense_12C' [CY8C4]
CY8CKIT_040_CapSense_I2C.cydwr
Header Files
🖻 🧰 Source Files

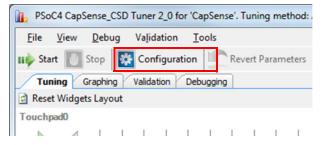


2. To launch the Tuner, right-click the CapSense Component in PSoC Creator and click Launch Tuner, as shown in Figure 5-34.

Figu	re 5	-34. Launch Tu	ner	
\leftarrow	*	Cut	Ctrl+X	
0.555		<u>C</u> opy	Ctrl+C	
	12	<u>P</u> aste	Ctrl+V	
figuratio	\times	<u>D</u> elete	Del	
d/Track ured wi		Select <u>A</u> ll	Ctrl+A	
		Zoom	+	
		Shape	+	
		Select	•	
		Configure		
	Ð,	Open Datasheet		
		Find Example Project		
		Open Component We	b Page	
		Launch Tuner		
		Generate Macro		

3. The Tuner window opens. Click **Configuration** to open the configuration window, as shown in Figure 5-35.

Figure 5-35. Tuner Window



4. Set the I²C communication parameters, as shown in Figure 5-36.

Figure 5-36. I²C Communication

Tuner Communication Setup		? ×
Ports: KitProg/191607A0032C2400 Port Information KitProg Version 2.07	Port Configural I2C address: Sub-address: I2C Speed 1 MHz 400 kHz 100 kHz 50 kHz	8 2-Đytes
		OK Cancel



5. Click **OK** to apply the settings.

5.4.3.2 Verify Output

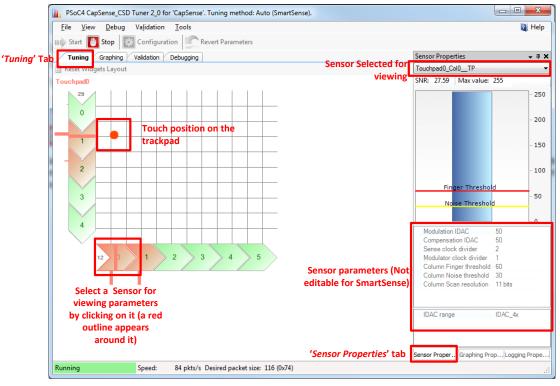
1. To start the scanning and communication process, click **Start** in the Tuner window, as shown in Figure 5-37.

Figure 5-37. Start Communication



2. Select a sensor in the **Tuning** tab. A red outline is displayed on the selected sensor. Different CapSense parameters are shown on the bottom right. You cannot edit the settings because auto-tuning is used in this project; auto-tuning automatically sets all the parameters. Touch the selected sensor and observe the response in the Tuner window.

Figure 5-38. Widget Testing



- 3. In the **Graphing** tab, the CapSense data Raw counts, Baseline, and Signal (difference count) for each sensor are represented as a graph.
- 4. Select the sensor parameters to observe, as shown in Figure 5-39. The graph of the selected parameters appears.



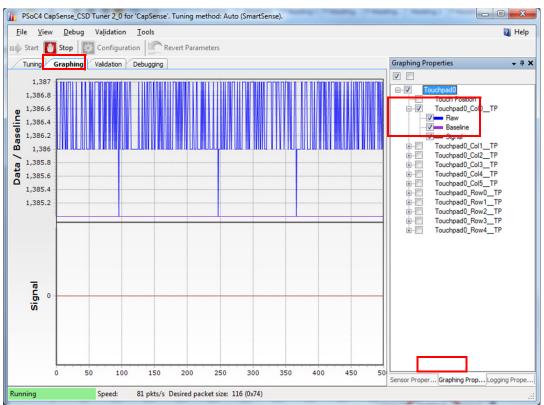


Figure 5-39. Sensor Parameter Graph

5. Touch a sensor or slider element and see the increase in Raw count and Signal, as shown in Figure 5-40.



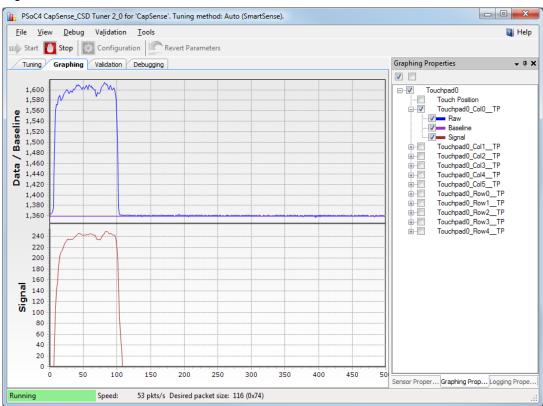


Figure 5-40. Raw Count Increase

5.5 Project: Color Palette

5.5.1 Project Overview

The project *CY8CKIT_040_Color_Palette.cyprj* demonstrates the capability of PSoC 4000 device to interface with a capacitive Trackpad and control an RGB LED based on the color selected by touching the sticker on top of the Trackpad. The sticker will also include a slider area (part of Trackpad), which will control the color brightness of the RGB LED. The project will demonstrate the proximity sensing capability of the device using a wire. The LED intensity control is done using software Precision Illumination Signal Modulator (PrISM). The project details are discussed in Firmware Details on page 82. Figure 5-41 shows the top design schematic of the project.



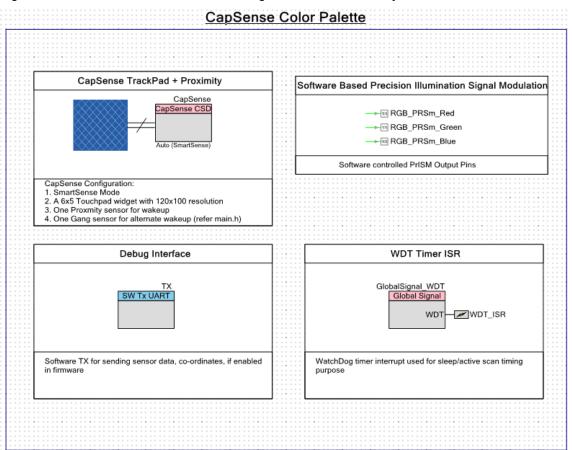


Figure 5-41. PSoC Creator Schematic Design of Color Palette Project

5.5.2 Project Description

5.5.2.1 PSoC Creator Component Configuration

CapSense:

The CapSense Component is configured in SmartSense Auto-tuning mode with one 6x5 touchpad, one dedicated proximity sensor, and a dummy/gang proximity sensor for the design with the parameters shown in Table 5-11.

Refer to the PSoC 4 CapSense Design Guide or the CapSense component datasheet for details on various widgets used and parameters.



Parameter	Tab Present	Value	Rationale
Tuning method		Auto(SmartSense)	Automatically adjust sensitivity for different system environments.
Threshold mode	General	Automatic	Enable run-time threshold calculation for 5:1 SNR.
Raw data noise filter		First Order IIR 1/8	Filter out noise/unwanted spikes in raw count. This setting can be tweaked based on requirement.
TouchPad0		-	Add and configure the touchpad as shown in Figure 5-42.
ProximitySensor 0	Widgets Config	-	Add and configure a proximity sensor as shown in Figure 5-42.
ProximitySensor 1		_	Add and configure a proximity sensor (Gang) as shown in Figure 5-42. Note that this sensor is a dummy sensor, which is ganged with the row ele- ments of the touchpad for wakeup.
Analog switch drive source		PRS-Auto	For reduced EMI emission and enhanced EMC immunity.
Sensor auto reset		Disabled	Not required in the design. Can be added if required by the application.
Low baseline reset		5	System dependent number. Can be configured as per the user needs.
Inactive sensor connection	Advanced	Ground	Make sure the proximity loop does not pick up any charge when not scanned.
Shield		Disabled	Not used in the design.
Guard sensor		Disabled	Not used in the design.
Cmod pre- charge		Precharge by Vref buffer	Vref is enough for precharging here, as there is only one sensor, Cmod voltage will not drop too low for a fast GPIO precharge.
		4	Select all the sensors by pressing and holding [CTRL] or [Shift] and clicking on all the touchpad sensors. When selected, enter the value in the Sensitivity field.
Sensitivity	Scan Order	1	Click ProximitySensor0 and set sensitivity as '1' for maximum range.
		5	Click ProximitySensor1 and set sensitivity as '5' for minimum resolution of scan during sleep scan for reduced power consumption.
Sensor ganging		-	Gang the touchpad row elements to ProximitySensor1 as shown in Figure 5-43.
Enable Tuner helper	Tuner Helper	Disabled/unchecked	No tuner used.

Table 5-11. CapSense Component Parameters



Name: CapSense General Widgets Config	2	Scan Order Advanced Tun	e Helper Built-in	4 ۵
🗄 Add touchpad 🛛 🔣 Remov	_	Rename		
The Component is configured in the	e Sm	artSense mode, so some widget p	roperties are not available.	
		Column Tuning		
		Column Finger threshold (FT)	100	
		Column Noise threshold	20	
Matrix buttons		Column Scan resolution	12 bits	
🗄 🔤 Touchpads		General		
Touchpad0		Number of sensor rows	5	
Proximity sensors		Number of sensor columns	6	
ProximitySensor0		Row API Resolution	100	
ProximitySensor1		Column API Resolution	120	
Generics		Position Noise Filter	Averaging	
		Row Tuning		
		Row Finger threshold (FT)	100	
		Row Noise threshold	20	
		Row Scan resolution	12 bits	

Figure 5-42. CapSense Touchpad Parameters

Name: Cap Sense General Widgets Confi Cap Add proximity sensor	g Scan Order Advance Remove Rename	d Tune Helper Built-in	4 ۵
The Component is configured in the Component is configured in the second	SmartSense mode, so some General Number of dedicated se Tuning	e widget properties are not available ensor elemen 1	e
Matrix buttons Touchpads Touchpad0 Proximity sensors Proximity Sensor0 Proximity Sensor1 Generics	Finger threshold (FT) Noise threshold Hysteresis Debounce Scan resolution	100 20 10 3 12 bits	
Datasheet		OK Apply	Cancel



Configure 'CapSense_CSD_P4' Name: CapSense General Widgets Config Add proximity sensor		e Helper Built-in 4 b
Buttons Linear sliders Radial sliders Matrix buttons Touchpads Touchpad0 Proximity Sensor0 Proximity Sensor1 Generics	General Number of dedicated sensor elem Tuning Finger threshold (FT) Noise threshold Hysteresis Debounce Scan resolution	en 0 100 20 10 3 12 bits
Datasheet	ОК	Apply Cancel

Figure 5-43. Scan Order Tab in CapSense Component Configure Window

onfigure '(CapSense_CSD_P4'			? <mark>></mark>
Name:	CapSense			
Gene	ral Widgets Config Scan Order	Advanced Tune He	lper Built-in	4 ۵
Scan slot	Sensor		Sense clock divider	Modulator clock divider
7	Touchpad0_Row1TP		2	2
8	Touchpad0_Row2TP		2	2
9	Touchpad0_Row3TP		2	2
10	Touchpad0_Row4TP		2	2
11	ProximitySensor0_PROX, ProximitySen	nsor0_0_PROX -	2	2
12	ProximitySensor1PROX, Touchpad0	_Row0TP, Touchpa 🔫	2	2
Sensor so Modulatio	Touchpad0_Col2_TP Touchpad0_Col3_TP Touchpad0_Col4_TP	an time: UNKNOWN DAC: 80 Sensi	itivity: 5 📩	•
Datas	Touchpad0_Col5_TP Touchpad0_Row0_TP Touchpad0_Row1_TP Touchpad0_Row2_TP Touchpad0_Row3_TP	ок	Apply	Cancel
	Touchpad0_Row4_TP ProximitySensor0_0_PROX			

TX (SW TX UART)

The software transmit TX is used to send out sensor data for debugging if enabled in firmware (refer to *main.h* in the project). The configuration for the component is shown in Figure 5-44. The SW TX can be sent over to PC using either an RS-232 connector (with a voltage level translator in between) or through USB-UART bridge available in CY8CKIT-040 PSoC 5 LP UART Bridge or CY3240 bridge configured as UART bridge, as documented in AN2397. The TX pin, if enabled, is configured in firmware through TX_PORT and TX_PIN macro available in *main.h*.

Note: By default, P3[0] is used for TX in firmware and P3[0] is hardwired in CY8CKIT-040 to the PSoC 5LP USB-UART bridge's RX line. So SWD debug cannot be used if P3[0] is used for TX and is enabled in firmware. Refer to the last step in Programming the Example Projects on page 45 for details on how to use SWD debug and TX in the same project.

Configure 'SW_Tx_UART'	8	x
Name: TX		
Basic Built-in		1 Þ
Parameter	Value	
BaudRate	115200	
PinAssignmentMethod	Dynamic	
Parameter Information		
Datasheet	OK Apply Cancel	

Figure 5-44. Software UART TX Component Parameters

RGB_PRSm_<Color> (Digital Output Pin)

To drive the software PrISM output to the respective <color> LEDs. It is a standard strong drive firmware controlled output pin.

GlobalSignal_WDT (Global Signal Reference with ISR)

This component is used to route the WDT ISR to an ISR Component. This ISR is then configured in firmware for generating periodic wakeup signal using WDT during Sleep_Scan mode.

5.5.2.2 Firmware Details

Firmware Structure

The color palette firmware is written in a modular format with different aspects of the functionality provided in separate functions, source, and header files. This enables users to understand the firmware structure better and to modify the firmware easily to meet the application requirements. The



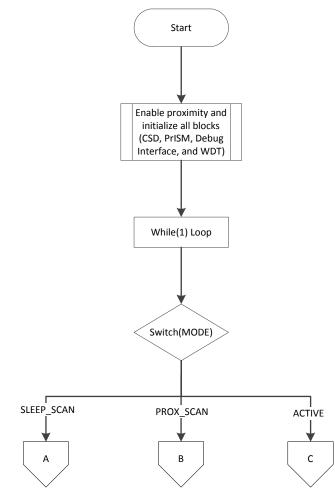
key aspects of the firmware can be modified to meet a different application requirement using various macros defined in the *main.h* file.

File Names	Purpose
RGB_PRSm.c, RGB_PRSm.h	These files contain the implementation of the software PRS modulator, which controls the LED intensity.
main.c	This file contains all the function definitions used in the firmware.
main.h	This file contains all the macro definitions, imported variables, and exported function decla- rations. Key aspects of the firmware can be modified by changing the macro values in this file.

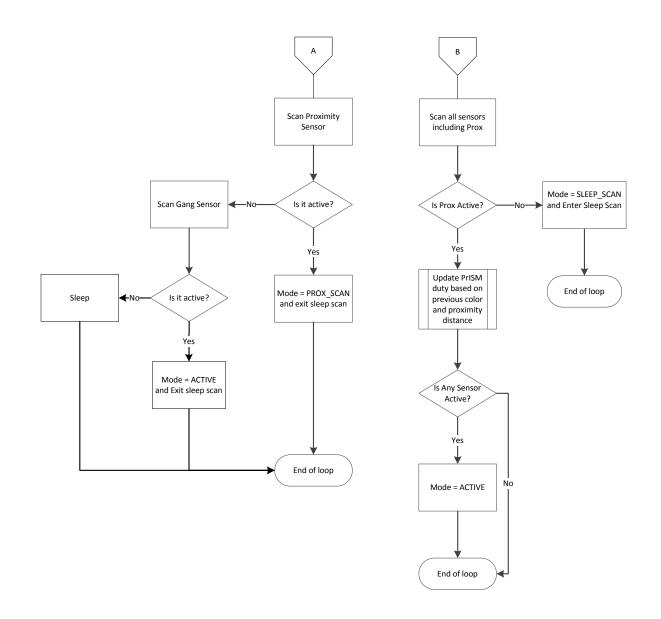
Firmware Flow Chart

Figure 5-45 shows the flow chart of code implemented in main.c.

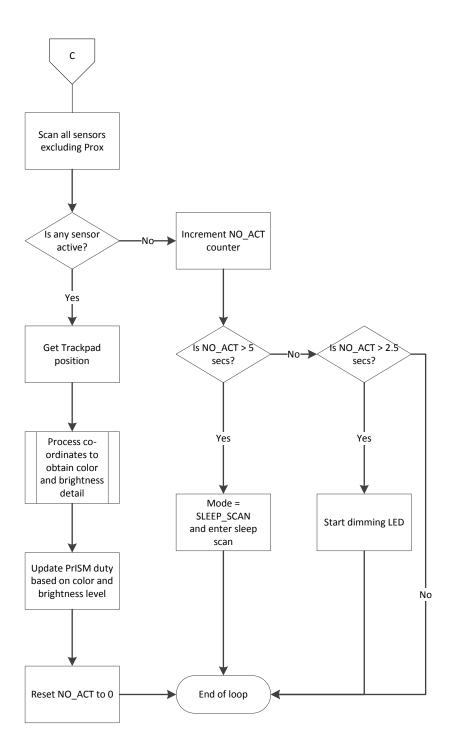
Figure 5-45. Color Palette Project Flow Chart













Firmware Implementation Details

The firmware consists of three different modes: active scanning (ACTIVE_SCAN), sleep scanning (SLEEP_SCAN), and proximity scanning (PROX_SCAN)

Active Scanning: Active scanning primarily performs two tasks:

- □ Scans the touchpad
- Updates the color based on touchpad activity

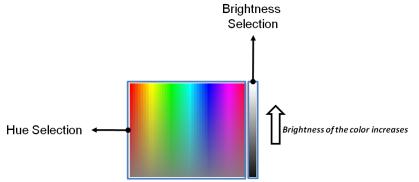
Scanning the touchpad sensors along with the proximity sensor, if enabled, in turn performs two subtasks:

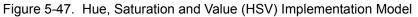
- Keeps the device in active scan mode
- Calculates the touch coordinate itself

Any activity on the touchpad or proximity sensor keeps the device in active scanning mode. If all activity ceases, the firmware will fade off the RGB color displayed and enter sleep scanning mode. The timeline at which the LED fades from last detected touch/proximity activity is defined by the macro LED_DIM_THRESHOLD. The LED_DIM_RATE macro defines the rate at which LED dimming is done. Refer to the *main.h* file for details on various macros used and their usage.

Figure 5-46 and Figure 5-47 show the two different color selection models implemented in the project. The two options are selectable using DO_SATURATION macro defined in *main.h*.

Figure 5-46. Hue and Brightness Control Implementation Model (default model)





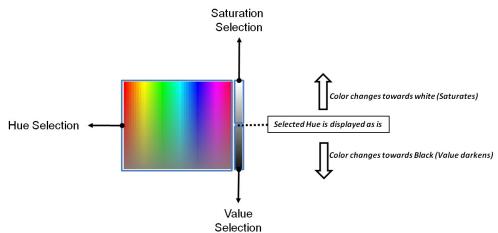




Figure 5-48 through Figure 5-51 explain the process of generating the intensity levels for the RGB LEDs. The color palette and the saturation slider are both part of the Trackpad area. The following steps summarize the flow.

1. The Trackpad coordinates are obtained as shown in the mapping in Figure 5-48. Figure 5-49 shows various macro definitions used in the firmware (main.h) with respect to the Trackpad/color palette area.

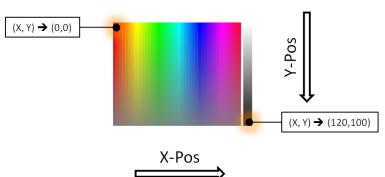
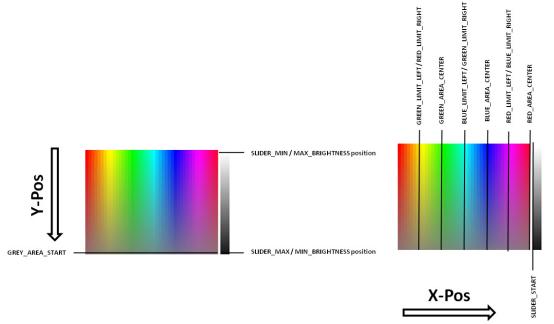


Figure 5-48. Trackpad Coordinate Selection

Figure 5-49. Macro Definitions Along X-Y Axis



2. Intensity levels of the RGB colors, before applying saturation, are calculated as shown in Figure 5-50. Along the X-Pos, the color palette is divided into six windows. In each window, one color is at maximum brightness, one is at minimum brightness, and one moves from maximum to minimum or vice versa. The macros shown in Figure 5-49 mark these window boundaries. <color>_AREA_CENTER marks the center of a <color> window on whose either side the <color> will be at maximum intensity. <color>_LIMIT_LEFT/<color>_LIMIT_RIGHT macros mark the edge of each <color>'s area inside which it is at maximum intensity.

Along Y-Pos, each color's intensity moves towards half-intensity level (50 percent) from the intensity level selected along X-Pos (x percent). The color levels are first calculated along the X-Pos using the window rule and X coordinate. The color thus obtained is then processed with the Y coordinate.



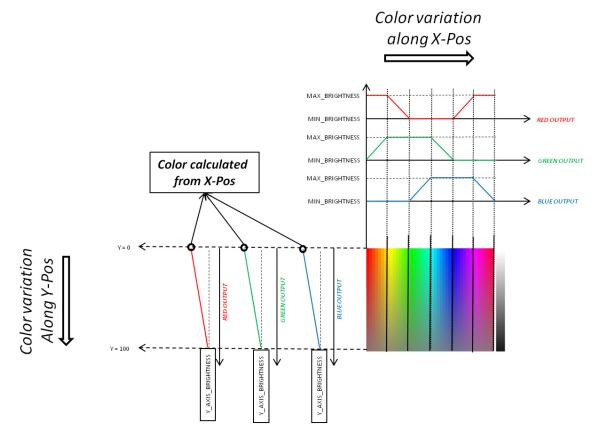


Figure 5-50. Intensity Calculation Based on X and Y in Color Palette

- 3. After the primary color intensities (from the color palette) are derived from step 2, either the brightness level or the saturation level selected by the slider is applied. The selection between brightness and saturation is done using DO_SATURATION macro defined in *main.h.*
 - a. Disabling/commenting out the DO_SATURATION macro applies brightness control on the color. Brightness control is a simple scaling of the three color intensities, derived at step 2, using slider output. Default brightness value at power on is 100% or maximum.
 - b. Enabling the DO_SATURATION macro applies saturation as explained in Figure 5-51. In saturation control, if the slider output is at half value, the intensities are retained. If it is less than half value, all the intensities move towards the minimum intensity (black/darker). If it is more than half value, all the intensities move towards the maximum intensity (white/brighter). Default saturation value at power ON is 50 percent or HALF_SATURATION value. The amount by which each color darkens or brightens is proportional to the slider position's relative difference from its center. See Figure 5-51.

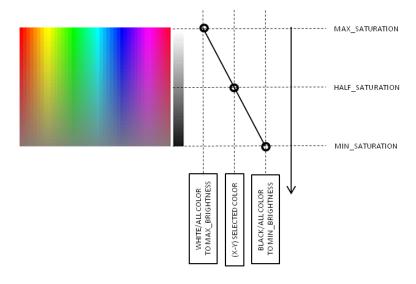


Figure 5-51. Saturation Calculation Based on Slider Output

Sleep-Scan Mode

In the sleep-scan mode, the firmware scans either the proximity sensor or the gang sensor (sensors in touchpad row elements ganged together) or both. This option is configurable during project compilation by commenting out the ENABLE_PROXIMITY or ENABLE_GANG macros. Keeping both will scan both the sensors in sleep, and disabling both will keep the device in active scanning mode always.

- □ If a proximity sensor is used in sleep-scan mode and the device detects any proximity activity on the sensor, the device will enter proximity scanning mode.
- □ If a gang sensor is used in sleep-scan mode and the device detects activity on the Trackpad during its scan, the device enters active scanning mode directly.

The rate at which the device scans in the sleep-scan mode is configurable by changing the macro SLEEP_TIMER_PERIOD defined in *main.h*.

Proximity Scanning Mode:

In proximity scanning mode, the device primarily scans the proximity sensor, and the touchpad sensors are scanned for activity. The previously selected RGB color is turned on with intensity proportional to the proximity signal strength. The device enters active scanning mode when a Trackpad element is touched. It stays in proximity scanning mode as long as there is some activity on the proximity sensor. If the hand moves out of proximity range, the device will return to sleep-scan mode.

Output Interfaces

Optional TX Interface

The firmware also features an optional UART TX interface, which can be enabled or disabled based on need. To enable or disable the TX interface, uncomment or comment TX_ENABLE macro present in *main.h.* The firmware also provides an option to select between two types of packets that are transmitted over the TX line. One packet contains the raw count, baseline, and signal data of all the sensors in the system in multichart format (refer to AN2397 for details on the multichart charting tool). Another type of packet sends out just the Trackpad coordinates. To select between these two types of packets, you can use the MINIMAL_TX macro. If the macro is commented out, then the first type of packet is transmitted. If the macro is present, then the second type of packet is sent.



Note: To view the MINIMAL_TX data, you can use BCP (as explained in UART Data Viewing on page 62) with *CapSense Color Palette (Minimal TX) - Commands.iic* and *CapSense Color Palette (Minimal TX) - Variables.ini* files available under ...\Firmware\PSoC 4\Bridge Files folder. AN2397's Multichart tool can be used to view the first type of packet because of the packet size.

Software PrISM for LED Intensity Control

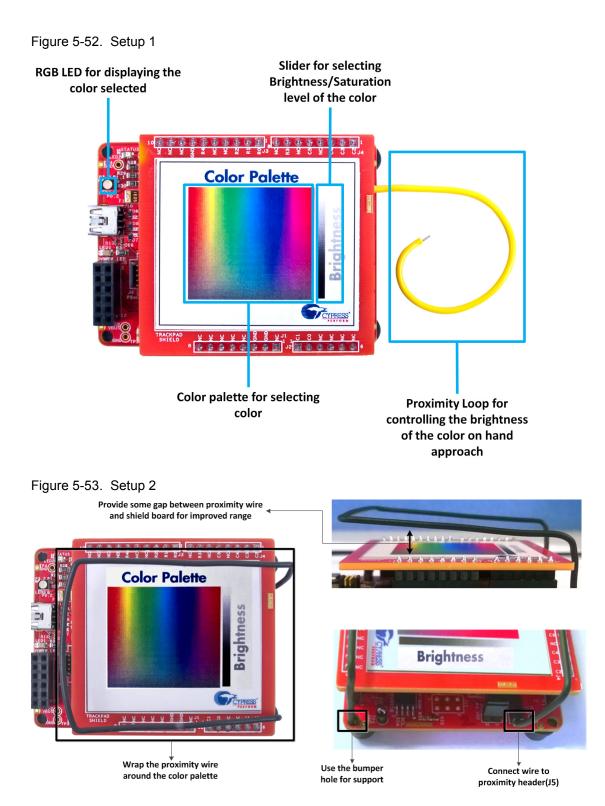
The firmware uses a software implemented 7-bit PRS for controlling the LED intensity through Precision Illumination Signal Modulation (PrISM). The implementation uses the SysTick timer available as part of Cortex-M0 CPU subsystem. The SysTick timer generates the interrupt where the PRS computations are performed to generate the Pseudo Random signal at the output of the LED pins. The implementation details can be found in *RGB_PRSm.h* and *RGB_PRSm.c* files available as part of the project. The polynomial used to generate the PRS is fixed to [7,6,5,2]. The period or repeat rate of the PRS is 127 counts. With SysTick timer generating an ISR at 20 kHz, this will result in a ~150 Hz output. Because the output is not a fixed pulse-width signal and is, a high-frequency signal (with a maximum frequency of 10 kHz and average of around 5 kHz), the signal easily gets filtered out avoiding any flickers usually noticed with PWM signals at this frequency.

5.5.2.3 Hardware Connections

The example requires the CapSense Trackpad shield connection, as explained in Project: CapSense Touchpad with I2C Tuner on page 68. The RGB LED and CMOD connections are hardwired on the board. Optionally, a proximity loop can be connected, as shown in setup 1 in Figure 5-52. To use the firmware efficiently, the proximity loop can be formed as a loop around the Trackpad, as shown in Figure 5-53. With setup 2, the LED turns on to show the previous color as the hand approaches the Trackpad.

Note: For the proximity loop, the wire shipped with the kit (4 inches in length) can be wound to form a loop of 1 to 2 inch diameter; the range obtained will be approximately the same as the loop diameter for a fast approaching hand. To obtain a higher range, use a longer wire/bigger loop.







Open CY8CKIT_040_Color_Palette.cydwr under the **Source** vertical tab in the **Workspace Explorer** and select the suitable pin.

Pin Name	Port Name
Trackpad_X0	P0_3
Trackpad_X1	P0_7
Trackpad_X2	P0_6
Trackpad_X3	P0_5
Trackpad_X4	P0_0
Trackpad_X5	P0_1
Trackpad_Y0	P1_4
Trackpad_Y1	P1_5
Trackpad_Y2	P1_6
Trackpad_Y3	P1_0
Trackpad_Y4	P1_7
CMOD	P0_4
Proximity	P2_0
Red LED	P3_2
Green LED	P1_1
Blue LED	P0_2
UART TX	P3_0 ²

Table 5-13. Pin Connection	Table	5-13.	Pin	Connection
----------------------------	-------	-------	-----	------------

Trackpad_X0 (Touchpad0_Col0_TP) to Trackpad_X5 (Touchpad0_Col5_TP) maps to COL5 to COL0 of the Trackpad so as to make the Trackpad x-axis left aligned.
 Selected in firmware (refer to main.h)



Alias	Name 🗠	Port		Pin	
Cmod	\CapSense:Cmod\	P0[4] EXTCLK	•	5	•
Touchpad0_Col0TP	\CapSense:Sns[0]\	P0[3]	•	4	•
Touchpad0_Col1TP	\CapSense:Sns[1]\	P0[7]	•	11	•
Touchpad0_Col2TP	\CapSense:Sns[2]\	P0[6]	•	10	•
Touchpad0_Col3TP	\CapSense:Sns[3]\	P0[5]	•	9	•
Touchpad0_Col4TP	\CapSense:Sns[4]\	P0[0]	•	1	•
Touchpad0_Col5TP	\CapSense:Sns[5]\	P0[1]	•	2	•
Touchpad0_Row0TP	\CapSense:Sns[6]\	P1[4]	•	16	•
Touchpad0_Row1TP	\CapSense:Sns[7]\	P1[5]	•	17	•
Touchpad0_Row2TP	\CapSense:Sns[8]\	P1[6] TCPWM0:N	•	18	•
Touchpad0_Row3TP	\CapSense:Sns[9]\	P1[0]	•	12	•
Touchpad0_Row4TP	\CapSense:Sns[10]\	P1[7] EXTCLK	•	19	•
ProximitySensor0_0_PROX	\CapSense:Sns[11]\	P2[0]	•	20	•
	RGB_PRSm_Blue	P0[2]	•	3	•
	RGB_PRSm_Green	P1[1] TCPWM0:P	•	13	•
	RGB_PRSm_Red	P3[2] TCPWM0:P	•	23	•

Figure 5-54.	Pin Selection for Color Palette Project
--------------	-----------------------------------------

5.5.3 Verify Output

Build and program the code example onto the device. The default color for the RGB LEDs is blue. Therefore, when you move over the proximity sensor before touching the Trackpad, the RGB LED turns blue. Follow these steps to verify the code.

1. Touch the color palette on the Trackpad, as shown in Figure 5-55. The color touched will be displayed in the RGB LEDs.

Note: Because the LED is bigger (compared to pixels in displays and paper), the individual color/ LED may be visible at some points. An extra diffuser such as a thin paper can be placed on top of the RGB LED, to see proper color mixing. In addition, the LEDs can exhibit different maximum brightness levels depending on the maximum current that can flow through them and the maximum luminosity they can generate (brightness = current × luminosity-to-current-ratio). For instance, red can be brighter than blue for a given series resistance, because of higher luminosity to current ratio and lower voltage drop across them, which results in higher current. For such cases, the brighter color's intensity can be limited to ensure proper mixing. An example for the same is also presented in the example project by limiting the intensity of the red LED to ~85 percent.

Note that when the brightness slider is set to minimum/zero brightness, touches on the color area are not displayed on the LED.



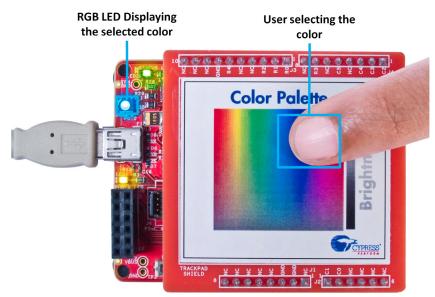


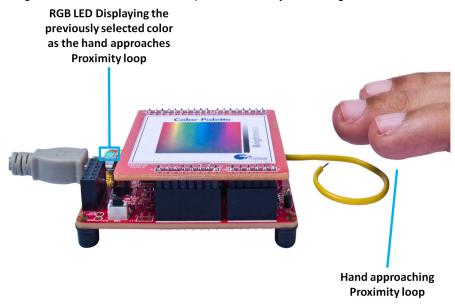
Figure 5-55. Color Palette Output in Active Scanning Mode

 Move through the slider area provided at the right corner of the Trackpad to adjust the brightness level of the color. Observe the color varying from Off to full brightness from one end to other.
 Note: When moving from the color selection area to the slider area, remove your finger from the

Trackpad and place it back on the slider for it to work normally. This is implemented to prevent accidental touches to the slider area when selecting color near the slider area.

- 3. Move away from the Trackpad and observe the color shown in the LED gradually ramp down after 2-3 seconds and go off after approximately 5 seconds.
- 4. When the LEDs are off, moving your hand towards the proximity loop will slowly ramp up the brightness of the LED as the hand enters the proximity range, and brightness will be at the maximum when it is near the Trackpad. The color on the LED will be the previously selected Trackpad color, as shown in Figure 5-56.

Figure 5-56. Color Palette Output in Proximity Scanning Mode





This section describes some of the advanced features available in the kit.

6.1 Using PSoC 5LP as a USB-I²C Bridge

6. Advanced Topics

The PSoC 5LP serves as a USB-I²C bridge, which can be used to communicate with the USB-I²C software running on the PC.

Note: Section Project: CapSense Touchpad with I2C Tuner on page 68 also uses the USB-I²C bridge available in the kit, but with the CapSense Tuner window.

The following steps describe how to use the USB-I 2 C bridge to communicate between the BCP and the PSoC 4.

1. Create a new project targeting the PSoC 4 device in PSoC Creator, as shown in Figure 6-1.

Figure 6-1. Create a New Project in PSoC Creator

Design Other		· · · · · · · · · · · · · · · · · · ·
Empty Templates		
Empty PSo	C 3 Design	Creates a PSoC 3, 8 bit, design project.
Empty PSo	C 4 Design	Creates a PSoC 4, 32 bit, design project.
Empty PSo	C 5LP Design	Creates a PSoC 5LP, 32 bit, design project.
PSoC 3 Starter Desi	gns	
ADC_DMA	VDAC	Shows how to transfer data from an ADC to a DAC using DMA with no CPU intervention.
DelSig_160	hannel	Shows a 16-channel, 12-bit Delta Sigma ADC in PSoC 3 sequenced in hardware; samples are transferred from ADC to SRAM using DMA - without processor intervention.
DelSig_I2C	м	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.
DelSig_I2C	S	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.
Name: USB-I	2C	
ocation: C:\Us	ers\msur\Documents\	VPSoC Creator\USB-I2C
Advanced		
Workspace:	Add to Current W	Inkspace
Workspace Name:	CY8CKIT_040_C	ap Sense_I2C
Device:	CY8C4014LQI-42	22
	Empty (11" x 8.5	")
Sheet Template:		
Sheet Template: Application Type	Normal	



2. Drag and drop an I^2C Component to the top design, as shown in Figure 6-2.

Figure 6-2.	I ² C Component in	Component Catalog
-------------	-------------------------------	--------------------------

Component Catalog (81 components)
Search for 🦓 🚺 🗘 📭
Cypress Off-Chip 4 b
Cypress Component Catalog
🖶 🐼 Analog
🕀 🔯 CapSense
🖨 🔯 Communications
🖻 🚾 I2C
EZI2C Slave (SCB mode) [v1.20]
R 12C (SCB mode) [v1.20]
Serial Communication Block (SCB) [v1.20]
Software Transmit UART [v1.20]
Software Fransmit OART [V1.20] Software Fransmit OART [V1.20]
ter too Dighai
⊕ too System
Image: Image
Open datasheet
I2C Communication Interface (SCB mode) I2C_1
120_1
Slave

3. To configure the I²C Component, double-click or right-click the I²C Component and select **Con**figure, as shown in Figure 6-3.

Figure 6-	-3. Open l ²	C Configuration	Wi
12C			
****	Cu <u>t</u>	Ctrl+X	
E	<u>С</u> ору	Ctrl+C	
12	<u>P</u> aste	Ctrl+V	
×	<u>D</u> elete	Del	
	Select <u>A</u> ll	Ctrl+A	
	Zoom	•	
	Shape	•	

Configure...

/indow

4. Configure the l^2C with the settings and click **OK**, as shown in Figure 6-4 and Figure 6-5.



Figure 6-4. I²C 'Configuration' Tab

Configure 'SCB_P4'	? <mark>×</mark>	
Name: I2C_1		
Configuration I2C Built-in	4 Þ	
Unconfigured SCB		
I2C		
© EZI2C		
○ SPI		
O UART		
The SPI and UART modes are not available for the selected device.		
Datasheet OK Apply	Cancel	

Figure 6-5. 'I²C' Tab

Configure 'SCB_P4'	2 2	×		
Name: I2C_1				
Configuration	I2C Built-in	4 ۵		
Mode:	Slave -			
Data rate (kbps):	100 Actual data rate (kbps): 93.75 			
Oversampling factor:	16 🔶 Low: 8 📩 High: 8 📩			
Clock from terminal	I			
Median filter				
	Bit 7 6 5 4 3 2 1	0		
Slave address (7-bits):	0x08 0 0 0 1 0 0 0	x		
Slave address mask:	0xFE 1 1 1 1 1 1	0		
Accept matching a	address in RX FIFO			
Enable wakeup from Sleep Mode				
Datasheet	OK Apply Can	el		

5. Select pin P1[2] for the I²C SCL and pin P1[3] for the I²C SDA in the Pins tab of <project.cydwr>, as shown in Figure 6-6.



Figure 6-6. Pin Selection

*USB-I2C.cydwr									
TCPWM0:P DEBUG	DEBUG			Alias	Name 🛛 🛆		Port		Pin
DE DE	DEBU				\I2C_1:scl\	P1[2]	SCB0:I2C:SCL	-	14
24 23 22	21 20 19				\I2C 1:sda\	P1[3]	SCB0:I2C:SDA	•	15
XRES P3[2]	P3[0]								
[0]C		P1[6] 18	TCPWM0:N						
ן(ז)		P1[5] 17							
J[2]		P1[4] 16							
J[3] CY8C4014	4LQI-422	P1[3] 15	\I2C_1:sda\						
D[4] 24-Q	≀FN	P1[2] 14	\I2C_1:scl\						
QQQ		P1[1] 13	TCPWM0:P						
۶ 8	10 P0[6] 11 P0[7] 12 P1[0]								
.3v 9	11								

- Place the code available in USB_I2C-main.c, which is attached to this PDF document, in your main.c project file. This code enables the PSoC 4 device to transmit and receive I²C data to and from the BCP application.
- 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 code onto the PSoC 4 through the PSoC 5LP programmer or MiniProg3.

Note: A warning may be displayed on the I²C input clock. This is because to generate a 100-kbps I²C clock, the block needs a 1.6-MHz signal, which cannot be derived from the default HFCLK setting of 12 MHz. To remove the warning, go to *<project_name>.cydwr* > **Clocks** and double-click **HFCLK**. Set the **IMO** to **32 MHz** and **HFCLK divider** to '**2**' in the window that appears (see Figure 6-7). This generates a 16-MHz HFCLK; using a divider of 10, the 1.6-MHz clock required for I²C block will be generated.



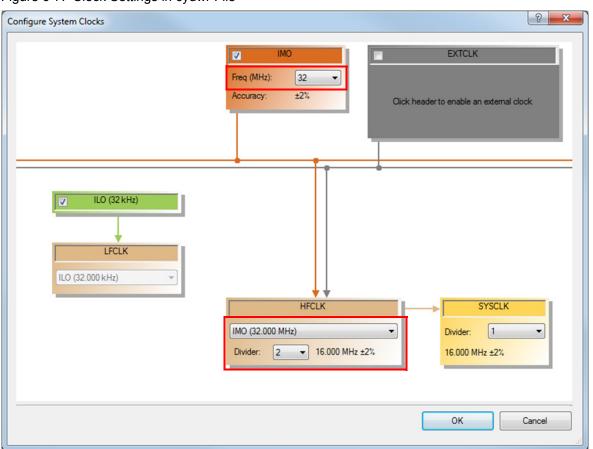


Figure 6-7. Clock Settings in cydwr File

- 8. Open BCP from Start > All Programs > Cypress > Bridge Control Panel <version number>.
- 9. Connect to KitProg/ under Connected I²C/SPI/RX8 Ports, as shown in Figure 6-8.



Figure 6-8.	Connect to	KitProg/ i	in BCP
-------------	------------	------------	--------

Diridge Control Panel					
<u>File Editor Chart Execute Tools Help</u>					
🔗 🛛 🚊 🕼 🛍 🛛 🛷 匡 🛛 🔤 🧱					
Editor Chart Table File					
	~				
4					
Opening Port	A				
Successfully Connected to KitProg/191607A0032C2400					
KitProg Version 2.07					
	-				
<	Þ				
Connected 12C/SPL/RX8 Pode:	Protocol				
Construction of the second all strings:	.0V () 12C				
Stop Wheelar Stop wheelar states and states	.5V BX8 (LABT)				
Scan period, ms: 0 €	.8V				
1:1 Syntax: OK Connected Powered Voltage: 5032 mV					

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

Figure 6-9. Open Protocol Configuration Window in BCP

Didge Control Panel	
File Editor Chart Execute Tools Help Image: Im	onfiguration F7
Editor Chart Table File I2C Bootloa	
Protocol Configuration	

11. To transfer data, type the command shown in Figure 6-10 and press [Enter] or click the Send button in the BCP. The log shows whether the transaction was successful. A "+" indication after each byte indicates that the transaction was successful and a "-" indicates that the transaction was a failure, as shown in Figure 6-10 and Figure 6-11.



🗱 Bridge Control Panel					
File Editor Chart Execute To	ols Help				
🚅 🖬 🚊 🖓 🖿 🛍 🗇 🧮 🔤	× 🖪 🕅				
Editor Chart Table File					
w 08 00 01 02 03 04 p					~
	nerates Stop condition I2C bus				
Data Bytes					
Slave Address					
"Write Data" Command					
<					Þ.
Opening Port					~
Successfully Connected	d to KitProg/0D0)F13CC013224	00		
KitProg Version 2.07 w 08+ 00+ 01+ 02+ 03+	04+ n				
	out p				
'+' Indicates Acknowledgement (AC	ск)				
4					
1		Connected I2C/SPI/RX	8 Ports:		
Reset Send	Send all strings:	KitProg/0D0F13CC013 COM3		Power +5.0V	Protocol
	Repeat count: 0 🚔	COM3 COM43		• +3.3V	SPI
	Scan period, ms:			● +2.5V ● +1.8V	RX8 (UART)
1:22 Syntax: OK	ok	Connected	Powered	Voltage: 5061 mV	.::





Diridge Control Panel	- • ×		
<u>Eile E</u> ditor <u>C</u> hart E <u>x</u> ecute <u>I</u> ools <u>H</u> elp			
❷ ■ ⑧ 圖 ● ◎ ◇ 三 歴 医 歴			
Editor Chart Table File			
w 07 00 01 02 03 04 p	*		
	-		
	Þ		
Opening Port Successfully Connected to KitProg/0D0F13CC01322400	^		
KitProg Version 2.07			
w $08+ 00+ 01+ 02+ 03+ 04+ p$ w $070 00- 01- 02- 03- 04- p$			
'' Indicates No Acknowledgment (NACK)			
4	• • •		
Connected I2C/SPI/RX8 Ports:	Protocol		
Common Send all strings: KtPppp/2000F1SCC01322400 Power Verset \$\$2.List \$\$Send \$\$5.0V	I2C		
Scap period, ms: 0 ↔ COM43	 SPI RX8 (UART) 		
	() the (overly		
1:22 Syntax: OK ok Connected Powered Voltage: 5061 mV	:		

12. From the BCP, read five bytes of data from the I²C slave device with slave address 0x08, as shown in Figure 6-12. The log shows whether the transaction was successful.



W Bridge Control Panel	- • ×		
<u>Eile Editor Chart Execute Tools Help</u>			
📽 🖩 🚊 🕲 📧 🖉 🧮 🗮 🧱			
Editor Chait Table File			
r 08 x x x x p	×		
Generates Stop condition			
on I2C bus			
No. of data bytes to read. x – Reserved symbol, which means			
that 1 byte of data should be read			
Slave Address			
"Read Data" Command	-		
4	•		
Opening Port	*		
Successfully Connected to KitProg/0D0F13CC01322400			
KitProg Version 2.07 w 08+ 00+ 01+ 02+ 03+ 04+ p			
w 07- <u>00-01-02-03-04-</u> p			
r 08+ 00+ 01+ 02+ 03+ 04+ p			
Data bytes read from the slave device			
4	*		
Connected I2C/SPI/RX8 Ports:			
Conflicted table of Protocontexts Power Kerrog/0D00F13CC01322400 Power +5.0V	Protocol I2C 		
COM3 COM3 COM3 COM43 COM	SPI		
Scan period, ms: 0	RX8 (UART)		
1:17 Syntax: OK ok Connected Powered Voltage: 5061mV			

Note: See Help Contents under Help in the BCP or press [F1] for details of the I²C commands.



6.2 Using FM24W256 F-RAM

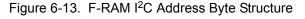
The PSoC 4000 Pioneer Kit has an onboard Ferroelectric RAM chip that can hold up to 32 KB of data. Cypress F-RAM products combine the nonvolatile data storage capability of ROM with the benefits of RAM, which include a high number of read and write cycles, high-speed read and write cycles, and low power consumption. F-RAM core memory and integrated products are ideal for applications that require high data integrity and ultra-low power consumption. These products target markets in automotive, industrial, enabling technologies, and networking. F-RAM inherently features high endurance, fast single-cycle, and symmetrical read/write speeds, along with low energy consumption, gamma radiation tolerance, and immunity to electromagnetic noise.

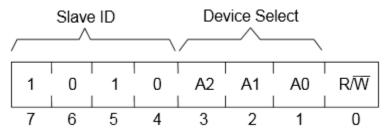
The F-RAM chip provides an I²C communication interface for data access. It is hardwired to the PSoC 4000 I²C lines (P1_2 and P1_3); the same lines are routed to the PSoC 5LP I²C lines as well. Because the F-RAM device is an I²C slave, it can be accessed or shared among various I²C masters on the same line. For more details on the F-RAM device, refer to the <u>device datasheet</u>.

This section describes a simple example on how to set the address of the F-RAM device, use it with an I^2C master (PSoC 4000) device, and share the same RAM with the BCP through the PSoC 5LP USB- I^2C bridge.

6.2.1 Address Selection

The slave address of the F-RAM device consists of two parts, as shown in Figure 6-13: slave ID and device select. Slave ID is an F-RAM family-specific ID located in the particular F-RAM device data-sheet. For the device used in CY8CKIT-040 (FM24W256-G), the slave ID is 1010b. The device select bits are set using the three physical pins A2-A0 in the device. The setting of these three pins in CY8CKIT-040 is controlled by resistors R19/R18 (A0), R17/R16 (A1), and R15/R14 (A2). See Cypress Ferroelectric RAM (F-RAM) on page 42 for details.





6.2.2 Write/Read Operation

The device's datasheet includes details on how to perform a write/read operation with the F-RAM. Figure 6-14 and Figure 6-15 provide a snapshot of the write/read packet structure as a quick reference.



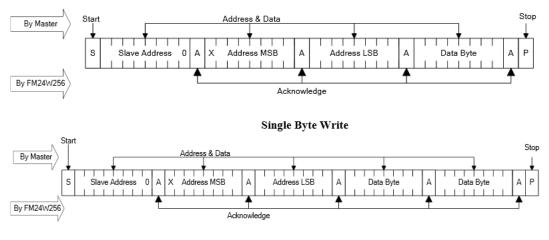
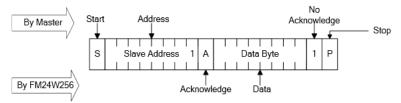
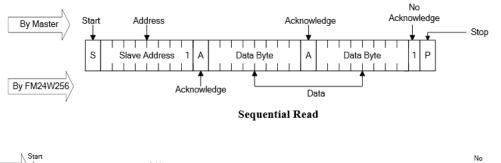


Figure 6-14. F-RAM Single/Multiple-Byte Write Packet Structure

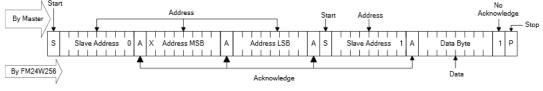
Multiple Byte Write







Current Address Read



Selective (Random) Read

In the previous figures, "Slave Address" denotes the address of the F-RAM slave device. [Address MSB: Address LSB] forms the 16-bit address of the memory location in the F-RAM to be accessed for read/write operation. The first two bytes following the slave address byte during a write operation constitute the initial memory address to be accessed. From there on, each byte accessed (read/ write) will increment this address by one and the count wraps around at the boundary (0x7FFF to



0x0000 for a 32 KB device). The value can be reset at any time by doing a write operation with the desired memory address.

6.2.3 Example Firmware

The following steps describe how to create a project with the PSoC 4000 family that uses the onboard F-RAM and shares it with another I^2C master (BCP through PSoC 5LP USB- I^2C bridge).

1. Open PSoC Creator 3.0 SP1 and create a new PSoC 4000 project, as shown in Figure 6-16.

Figure 6-16. Create a New PSoC 4000 Project in PSoC Creator

New Project		? ×			
Design Other		4 Þ			
Empty Templates		A			
Empty PSoC 3 D	Design	Creates a PSoC 3, 8 bit, design project.			
Empty PSoC 4 D)esign	Creates a PSoC 4, 32 bit, design project.			
Empty PSoC 5LF	P Design	Creates a PSoC 5LP, 32 bit, design project.			
 PSoC 3 Starter Designs 					
ADC_DMA_VDA	AC	Shows how to transfer data from an ADC to a DAC using DMA with no CPU intervention.			
DelSig_16Chann	nel	Shows a 16-channel, 12-bit Delta Sigma ADC in PSoC 3 sequenced in hardware; samples are transferred from ADC to SRAM using DMA - without processor intervention.			
DelSig_I2CM		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.			
DelSig_I2CS		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.			
Name: FRAM_Exa	Name: FRAM_Example				
Location: C:\Users\m	msur\Documents\PSc	oC Creator			
- Advanced					
Workspace:	Create New Workspa	ace 🔹			
Workspace Name: FRAM_Example					
Device: CY8C4014LQI-422		-			
Sheet Template:	Empty (11" x 8.5")	•			
Application Type Normal		•			
		OK Cancel			

 From the Components Catalog, place an I²C (SCB mode) Component in the TopDesign from Communication > I²C > I²C (SCB mode) [<version>] and configure it as I²C in the Configuration tab with the parameters shown in Figure 6-17.



Figure 6-17. I²C Master Configuration

Configure 'SCB_P4'	२ <mark>- × -</mark>
Name: I2C_M	
Configuration 1	I2C Built-in 4 D
Mode:	Multi-Master 👻
Data rate (kbps):	100 Actual data rate (kbps): 93.75
Oversampling factor:	16 😴 Low: 8 🜩 High: 8 🜩
Clock from terminal	
Median filter	
Slave address (7-bits): Slave address mask: Accept matching a Enable wakeup fro	
Datasheet	OK Apply Cancel

3. Select Pins 1[2] and 1[3] as the I²C pins in the **Pins** tab of the *.cydwr* file, as shown in Figure 6-18.

Figure 6-18. Pin Selection

Alias	Name 🗠	Port		Pin		L
	\I2C_M:scl\	P1[2] SCB0:I2C:SCL	•	14	•	
	\I2C_M:sda\	P1[3] SCB0:I2C:SDA	•	15	•	

- 4. Place the code available in *Example_FRAM-main.c,* which is attached to this PDF document, in the *main.c* file.
- Build the project by choosing Build > Build Project or pressing [Shift] F6]. After the project builds without any errors and warnings, program the device by pressing [Ctrl] F5] through the MiniProg3 or PSoC 5LP programmer in the kit.

Note: A warning may appear on the I²C input clock. This is because to generate a 100-kbps I²C clock, the block needs a 1.6-MHz signal, which cannot be derived from the default HFCLK setting of 12 MHz. To remove the warning, go to *<project_name>.cydwr* > **Clocks** and then double-click **HFCLK**. Set the **IMO** to **32 MHz** and **HFCLK** divider to '**2**' in the window that appears (see Figure 6-7). This generates a 16-MHz HFCLK; using a divider of 10, the 1.6-MHz clock required for I²C block will be generated.

 Open BCP and configure the I²C protocol as defined in Using PSoC 5LP as a USB-I2C Bridge on page 95.



- 7. In the command window, copy and paste the code from the *F-RAM_BCP_Commands.txt* file attached to this document.
- 8. By default the F-RAM device is configured with a 0x50 slave address. If the value has been changed as explained in Address Selection on page 103, then change the slave address in the command window (replace "50" with the slave address in hex format).
- Do a write to the F-RAM device by sending the 'W 50 00 00 01 02' line, which writes to memory location 0x0000 with the value 0x01 and memory location 0x0001 with 0x02. The command sent should be ACKed properly by the slave to make sure the transfer occurred properly, as shown Figure 6-19.

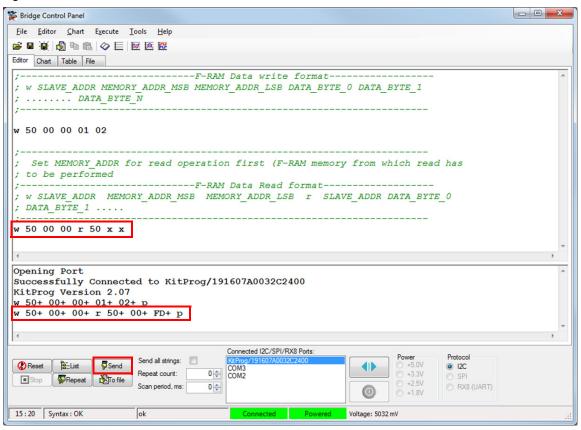
Figure 6-19. Send/Write Data to F-RAM

🗱 Bridge Control Panel					
<u>File Editor Chart Execute Iools Help</u>					
❷ ■ 潢 過 № № ◇ ⋿ 軽 孫 歴					
Editor Chart Table File					
;F-RAM Data write format	*				
; w SLAVE ADDR MEMORY ADDR MSB MEMORY ADDR LSB DATA BYTE 0 DATA BYTE 1 ; DATA BYTE N					
;					
w 50 00 01 02					
;					
; Set MEMORY ADDR for read operation first (F-RAM memory from which read has ; to be performed					
;F-RAM Data Read format					
; W SLAVE ADDR MEMORY ADDR MSB MEMORY ADDR LSB r SLAVE ADDR DATA BYTE 0					
; DATA_BYTE_1					
w 50 00 00 r 50 x x					
	-				
•	•				
Opening Port					
Successfully Connected to KitProg/191607A0032C2400 KitProg Version 2.07					
w 50+ 00+ 01+ 02+ p					
	-				
•	- F				
Connected I2C/SPI/RX8 Ports:					
Reset B: List Send all strings: COM2					
6:17 Syntax: OK ok Connected Powered Voltage: 5034 mV					

10. If the write transfer is successful, then check back the data using a read command to the same address by sending the second command line, as shown in Figure 6-20. The read command on the same locations will yield a '0' at 0x0000 (flag cleared) and 0xFD at 0x0001 (inverse of 0x02 sent). This shows that the PSoC 4 onboard accessed the bytes and modified them. On writing any byte to 0x0001, as explained in step 9, the inverse of the same will be calculated and stored back at the same location by PSoC 4 if the 0x0000 byte is set to '1'.



Figure 6-20. Read Data From F-RAM



6.3 Using PSoC 5LP as a USB-UART Bridge

The PSoC 5LP serves as a USB-UART bridge, which can communicate with the COM terminal software. This section explains how to use the PSoC 5LP's USB-UART bridge with an external device/ board (PSoC 4200 with CY8CKIT-001) with an example.

Note: This project explains how to use the USB-UART bridge of PSoC 5 LP for external UART lines. Project: CapSense Proximity and UART on page 52 can be used as reference for using the USB-UART bridge with the PSoC 4000 family featured on the board. The PSoC 4000 family supports only a software UART transmit line, which is explained in Project: CapSense Proximity and UART on page 52.

Users who have a Windows operating system that does not have HyperTerminal can use an alternative terminal software such as PuTTY.

1. Create a new PSoC 4 project in PSoC Creator, as shown in Figure 6-21. Select an appropriate location for your project and rename it as required.

ew Project		<u>୧</u> ×			
Design Other		4 Þ			
 Empty Templates 					
Empty PSoC 3	Design	Creates a PSoC 3, 8 bit, design project.			
Empty PSoC 4	Design	Creates a PSoC 4, 32 bit, design project.			
Empty PSoC 5	LP Design	Creates a PSoC 5LP, 32 bit, design project.			
 PSoC 3 Starter Design 	IS				
ADC_DMA_VE	DAC	Shows how to transfer data from an ADC to a DAC using DMA with no CPU intervention.			
DelSig_16Cha	nnel	Shows a 16-channel, 12-bit Delta Sigma ADC in PSoC 3 sequenced in hardware; samples are transferred from ADC to SRAM using DMA - without processor intervention.			
DelSig_I2CM		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.			
DelSig_I2CS		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.			
DelSig_SPIM		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over SPI.			
		Shows how to filtor an analog input all in hardware and requides all the DMA notion to transfer			
Name: USB-UA	RT				
Location: C:\Users	<pre>wmsur\Documents\PS</pre>	SoC Creator			
Advanced					
Workspace:	Create New Worksp	bace 🔹			
Workspace Name:	USB-UART				
Device:	Device: CY8C4245AXI-483 - (Default PSoC 4 Device)				
Sheet Template: Empty (11" x 8.5")					
Application Type	•				
		OK Cancel			

Figure 6-21. Create a New Project From PSoC Creator

2. Drag and drop a UART (SCB) component from the **Component Catalog** shown in Figure 6-22 to the top design.

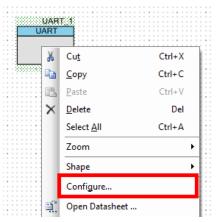


Component Catalog (142 components)
Search for 🏘 🚺 🔟 💶
Cypress Off-Chip 4 b
Cypress Component Catalog
🗄 🔯 Analog
🕀 🔯 CapSense
🖨 🔯 Communications
🗄 🔯 I2C
🕀 🚾 I2S
Serial Communication Block (SCB) [v1.20]
🗄 🔯 SMBus/PMBus Slave
Software Transmit UART [v1.20]
🕂 🐼 SPI
🕀 🔯 Digital
🕀 🔯 Display
🕀 🔯 Ports and Pins
🕀 🐼 System
⊞ 🐼 Thermal Management
Open datasheet
Universal Asynchronous Receiver Transmitter (SCB
mode)
UART_1 UART
Standard

Figure 6-22. UART Component Under Component Catalog

3. To configure the UART, double-click or right-click the UART Component and select **Configure**, as shown in Figure 6-23.





4. Configure the UART as shown in Figure 6-24, Figure 6-25, and Figure 6-26 and then click OK.



Figure 6-24. 'Configuration' Tab

Configure 'SCB_P4'	? <mark>x</mark>
Name: UART	
Configuration UART Basic UART Advanced Built-in	n ∢ Þ
Unconfigured SCB	
◎ I2C	
© EZI2C	
© SPI	
O UART	
Datasheet OK Apply	Cancel

Figure 6-25. 'UART Basic' Tab

Configure 'SCB_P4'		? X
Name: UART		
Configuration	UART Basic UART Advanced Built-in	4 ۵
Mode:	Standard 💌	<u>^</u>
Direction:	TX + RX 🔹	
Baud rate (bps):	9600 Actual baud rate (bps): 117647	
Data bits:	8 bits 🔻	E
Parity:	None	
Stop bits:	1 bit 🔹	
Oversampling:	12	
Clock from termin	al	
Median filter		-
Datasheet	OK Apply C	ancel



	Advanced Built-in 4
TV Dullel size. p	None Internal
TX buffer size: 8	External
Interrupt sources	RX FIFO not empty
TX FIFO not full	RX FIFO full
TX FIFO empty	RX FIFO overflow
TX FIFO overflow	RX FIFO underflow
TX FIFO underflow	RX frame error
TX lost arbitration	RX parity error
TX NACK	RX FIFO trigger: 7 👻
TX FIFO trigger: 0 💌	
Multiprocessor mode	RX FIFO drop
Address (hex): 2	On parity error
Mask (hex): FF	On frame error
Accept matching address in RX FIFO	

Figure 6-26. 'UART Advanced' Tab

5. Select P4[0] for UART RX and P4[1] for UART TX in the **Pins** tab of *<Project_Name>.cydwr*, as shown in Figure 6-27.

Figure 6-27. Pin Selection

RT.cydwr								- ⊲	⊳ ×
порожен порожен порожен порожен соримен	3.34		Alias	Name 🗸	Port		Pin		Lock
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	8			\UART:rx\	P4[0] SCB0:12C:SCL, SCB0:SPI:MOSI SCB0:UART:RX	•	20	•	V
P 19		33		\UART:tx\	P4[1] SCB0:I2C:SDA, SCB0:SPI:MISO SCB0:UART:TX	•	21	•	V
	XRES	32						_	_
	P0(7)	31 scersess, water							
	P0(6)	30 scelensus, artour							
	P0(5)	29 SCRISCISCA, SCRISPINSO SCRIVARTIX							
CY8C4245AXI-483	P0(4)	28 SCHISCHOL, SCHISPMOS SCHILARTRX							
44-TQFP	P0(3)	27							
44-TQFP	P0(2)	26 scsosmas							
	P0(1)	25 9090 97982							
	P0(0)	24 9090 97991							
	P4(3)	23 9090 97990							

- 6. Place the code available in *USB_UART-main.c,* which is attached to this PDF document, in your *main.c* project file. The code will echo any UART data received.
- Build the project by choosing Build > Build {Project Name} or pressing [Shift] [F6]. After the project is built without errors and warnings, program (by choosing Debug > Program) the project to PSoC 4 through MiniProg3.
- 8. Connect the RX line of the PSoC 4 to J8_10 and the TX line of the PSoC 4 to J8_9, as shown in Figure 6-28 and Figure 6-29.



Notes:

- Before connecting the RX line, remove R57 connecting P3[0] of the PSoC 4000 device to the PSoC 5LP RX line. This makes sure the PSoC 4000 device can be programmed/debugged while using the RX line for external bridge.
- The setup with CY8CKIT-001, CT8CKIT-038, and CY8CKIT-040 is provided for reference only on how to use the USB-UART bridge for connecting to an external UART interface.

Figure 6-28. UART Connection Between PSoC 4 and PSoC 5LP

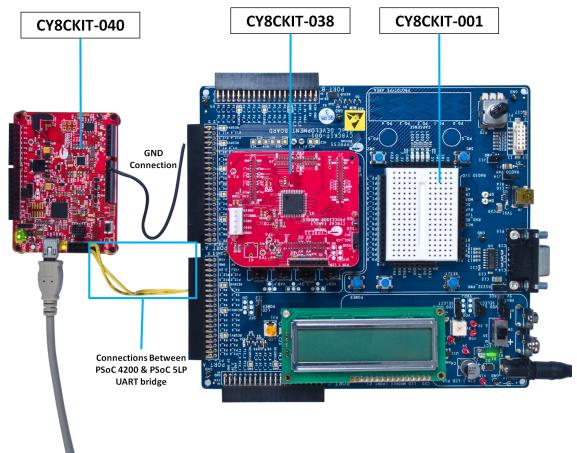
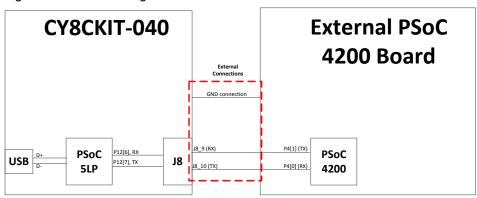


Figure 6-29. Block Diagram of UART Connection Between PSoC 4 and PSoC 5LP



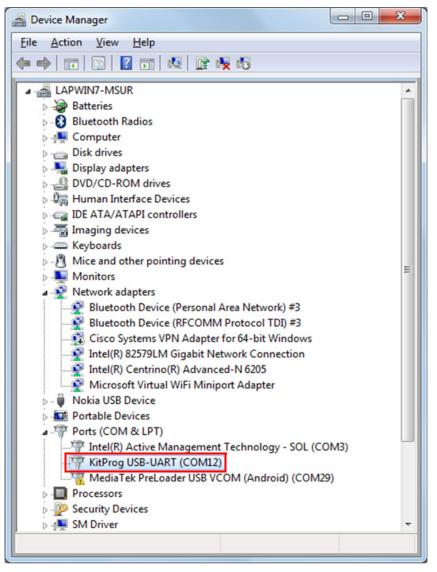


Note: UART RX and UART TX can be routed to any digital pin on PSoC 4 based on the configuration of the UART Component. An SCB implementation of the UART will route the RX and TX pins to either one of the following subsets: (P0[4], P0[5]) or (P3[0],P3[1]) or (P4[0],P4[1]).

To communicate with the PSoC 4 from the terminal software, follow this procedure:

 Connect USB Mini-B to J10. 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 6-30.

Figure 6-30. KitProg USB-UART in Device Manager



2. Open HyperTerminal, choose **File > New Connection**, enter a name for the new connection, and click **OK**, as shown in Figure 6-31.



Connection Description	? ×
New Connection	
Enter a name and choose an icon for the connection:	
Name:	
USB-UART communication	
Icon:	
< III	
•	P
· · · · · · · · · · · · · · · · · · ·	
ОК	Cancel

Figure 6-31. Open New Connection HyperTerminal

3. For PuTTY, double-click the PuTTY icon and select **Serial** under **Connection**, as shown in Figure 6-32.

Figure 6-32.	Open New Connection in	PuTTY
i igule 0-52.	Open New Connection in	IUIII

Reputity Configuration		×
Category: 	Select a serial line Serial line to connect to Configure the serial line Speed (baud)	Ig local serial lines COM12 9600 8
Behaviour Translation Selection Colours Onnection Data Proxy Telnet Rlogin SSH Senal	Data <u>b</u> its Stop bits <u>P</u> arity <u>F</u> low control	8 1 None V XON/XOFF V
About		Open <u>C</u> ancel

- 4. A new window opens, where you can select the communication port, as shown in Figure 6-33.
 - a. In HyperTerminal, select COMx (or the specific communication port that is assigned to the Kit-Prog USB-UART) in **Connect using** and click **OK**. This code example uses **COM12**.



b. In PuTTY, enter COMx in Serial line to connect to. This code example uses COM12.

Figure 6-33. Select Communication Port - HyperTerminal and PuTTY

Connect To			? ×		
USB-UA	RT communic	cation			
Enter details for the	he phone num	ber that you war	nt to dial:		
<u>C</u> ountry/region:	India (91)				
Ar <u>e</u> a code:	080				
Phone number:					
Connect using:	COM12				
		ОК	Cancel		
Real Putty Configu	uration				X
Category:			0.1		1 / 1 / 1 P
Session Logging		Select a seri	10.0	trolling	local serial lines
Terminal Keyboard		Seria <u>l</u> line to			COM12
Bell Features		Configure the	e serial line		
Window Appearance	ce	Speed (bau			9600
- Behaviour Translation		Data <u>b</u> its			8
Selection		Stop bits			1
Colours		<u>P</u> arity			None -
- Data - Proxy		<u>F</u> low contro	I		XON/XOFF -
Telnet Rlogin					
⊡ SSH Serial					
About				(Open <u>C</u> ancel

 In HyperTerminal, select Bits per second, Data bits, Parity, Stop bits, and Flow control under Port Settings and click OK, as shown in Figure 6-34. Make sure that the settings are identical to the UART settings configured for PSoC 4.

In PuTTY, select **Speed (baud)**, **Data bits**, **Stop bits**, **Parity**, and **Flow control** under **Configure the serial line** shown in Figure 6-33 (second image). Click **Session** and select **Serial** under **Connection type**, as shown in Figure 6-35. **Serial line** shows the communication port (COM12),



and Speed shows the baud rate selected. Click Open to start the communication.

Figure 6-34. Configure Communication Port in HyperTerminal

COM12 Properties		? 🔀
Port Settings		
<u>B</u> its per second:	9600	
<u>D</u> ata bits:	8	~
Parity:	None	~
<u>S</u> top bits:	1	~
Elow control:	Xon / Xoff	~
	<u>R</u> estore	Defaults
	K Cancel	

Figure 6-35. Select Communication Type in PuTTY

Reputity Configuration		×			
Category:					
- Session	Basic options for your PuTTY session				
Logging	Specify the destination you want to connect to				
Terminal Keyboard	Serial li <u>n</u> e	Speed			
Bell	COM12	9600			
Features	Connection type: ◯ Ra <u>w</u> ◯ <u>T</u> elnet ◯ Rlogin ◯ <u>S</u> SH				
Appearance Behaviour Translation Selection	Load, save or delete a stored session Sav <u>e</u> d Sessions				
Colours Connection Data Proxy Telnet Rogin H- SSH	Default Settings	Load Sa <u>v</u> e Delete			
tan Serial	Close window on exit: Always Never Only on	clean exit			
About	Open	Cancel			

6. Enable Echo typed characters locally under File > Properties > Settings > ASCII Setup to display the typed characters in HyperTerminal, as shown in Figure 6-36. In PuTTY, enable Force



on under **Terminal > Line discipline options** to display the typed characters in PuTTY, as shown in Figure 6-37.

Figure 6-36. Enable Echo of Typed Characters in HyperTerminal

ASCII Setup				
ASCII Sending				
Send line ends with line feeds				
Echo typed characters locally				
Line delay: 0 milliseconds.				
Character delay: 0 milliseconds.				
ASCII Receiving Append line feeds to incoming line ends Force incoming data to 7-bit ASCII				
✓ <u>W</u> rap lines that exceed terminal width				
OK Cancel				

Figure 6-37. Enable Echo of Typed Characters in PuTTY

Session Options controlling the terminal emulation Logging Set various terminal options Keyboard Auto wrap mode initially on Bell DEC Origin Mode initially on Features Implicit CB in every LF Window Implicit LE in every CR Appearance Implicit LE in every CR Behaviour Enable blinking text Translation Enable blinking text Selection Answerback to ^E: Colours PuTTY Line discipline options Force on Proxy Local echo: Rlogin Auto SSH Outo Auto Force on Porce off	Category:	
Terminal Set various terminal options Keyboard Auto wrap mode initially on Bell DEC Origin Mode initially on Features Implicit CR in every LF Window Implicit LE in every CR Behaviour Use background colour to erase screen Translation Enable blinking text Selection Answerback to ^E: Connection PuTTY Data Line discipline options Proxy Local echo: Rlogin Auto SSH Local line editing:		Options controlling the terminal emulation
Remote-controlled printing Printer to send ANSI printer output to: None (printing disabled)	Terminal Keyboard Bell Features Window Appearance Behaviour Translation Selection Colours Connection Data Proxy Telnet Rlogin SSH	Set various terminal options Auto wrap mode initially on DEC Origin Mode initially on Implicit CB in every LF Implicit LE in every CR V Use background colour to erase screen Enable blinking text Answerback to ^E: PuTTY Line discipline options Local echo: Auto Force off Local line editing: Auto Force on Force off Remote-controlled printing Printer to send ANSI printer output to:

7. The COM terminal software displays both the typed data and the echoed data from the PSoC 4 UART, as shown in Figure 6-38 and Figure 6-39.



USB-UART commu	inication - Hyper	Terminal						 X
ile <u>E</u> dit <u>V</u> iew <u>C</u> a		lp						
) 🖻 🗃 🖏 🗠 🎽) 🖬							
CY8CKIT-040 U	ISB-UART PF	SSooCC_						
onnected 0:00:36	Auto detect	9600 8-N-1	SCROLL	CAPS	NUM	Capture	Print echo	

Figure 6-38. Data Displayed on HyperTerminal

Figure 6-39. Data Displayed on PuTTY

Putty COM12 - Putty		- C X
CY8CKIT-040 USB-UART	PPSSooCC	*

6.4 Developing Applications for PSoC 5LP

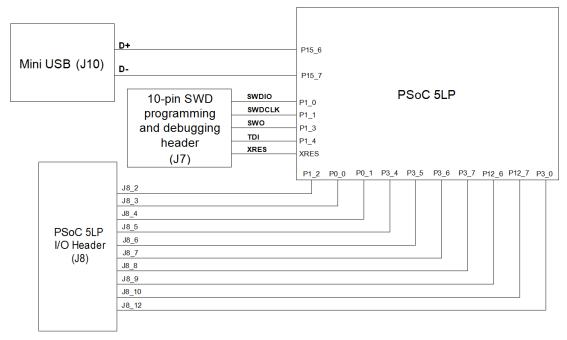
The PSoC 4000 Pioneer Kit has an onboard PSoC 5LP whose primary function is that of a programmer and a bridge. You can build either a normal project or a bootloadable project using the PSoC 5LP.

The PSoC 5LP connections on the board are summarized in Figure 6-40. J8 is the I/O connector (see A.2.2 PSoC 5LP GPIO Header (J8)). The USB (J10) is connected and used as the PC interface. However, you can still use this USB connection to create customized USB designs.



The programming header (J7) is meant for standalone programming. This header needs to be populated. See the "No Load Components" section in A.6 Bill of Materials.

Figure 6-40. PSoC 5LP Block Diagram



6.4.1 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. The bootloader hex file is available in the kit files or can be downloaded from the kit web page.

The hex files are included in the following kit installer directory, as illustrated in Figure 6-41:

```
<Install_Directory>\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version>\Firm-
ware\Programmer\KitProg Bootloader
```

Figure 6-41. KitProg Bootloader Hex File Location

Organize 👻 Inclu	ude in library 🔻	Share with 🔻	Burn Ne	w folder	•== •	
🔆 Favorites	 Name 	^		Date mo	dified	Туре
E Desktop	E Kitf	rog_Bootloader.elf		2/27/201	4 9:57 PM	ELF File
鷆 Downloads		prog_Bootloader.hex		2/27/201	4 9:57 PM	HEX File
🔛 Recent Places						
🐔 SkyDrive						
词 Libraries						
Documents						
🚽 Music						



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

 In PSoC Creator, choose New > Project > PSoC 5LP; click the expand button adjacent to Advanced and select the Device as CY8C5868LTI-LP039, as shown in Figure 6-42. Select the Application Type as Bootloadable from the drop-down list.

New Project					
Design Other		4 ۵			
 Empty Templates 		<u>^</u>			
Empty PSoC 3	Design	Creates a PSoC 3, 8 bit, design project.			
Empty PSoC 4 Design		Creates a PSoC 4, 32 bit, design project.			
		Creates a PSoC 5LP, 32 bit, design project.			
 PSoC 3 Starter Designs 	\$				
ADC_DMA_VD	AC	Shows how to transfer data from an ADC to a DAC using DMA with no CPU intervention.			
▶a DelSig_16Char	nnel	Shows a 16-channel, 12-bit Delta Sigma ADC in PSoC 3 sequenced in hardware; samples are transferred from ADC to SRAM using DMA - without processor intervention.			
DelSig_I2CM		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.			
DelSig_I2CS		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.			
DelSig_SPIM		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over SPI. Shows how to filter an analog input all in hardware, and provides all the DMA setup to transfer the data from the ADC directly to the Digital Filter Block, bypassing the processor.			
					HW_Fan_Control_with_Alert
PSoC 4 Starter Designs	3	•			
Name: Design 19					
Location: C:\Users\	vmsur\Documents\PS	CoC Creator			
Advanced					
Workspace:	Create New Worksp	vace 🔹			
Workspace Name:	Design 19				
Device:	CY8C5868LTI-LP03	9			
Sheet Template: Empty (11" x 8.5") Application Type Bootloadable					
		•			
		OK Cancel			

Figure 6-42. Create a New Project in PSoC Creator

2. Navigate to the schematic view and drag and drop a **Bootloadable** Component on the TopDesign.



Component Catalog (201 components)	-
Search for 🦓 🚺 🛄 📭	
Cypress Off-Chip	4 ۵
Cypress Component Catalog	-
🗄 🐼 Analog	
🖻 🔯 CapSense	
🗈 🔯 Communications	
🔁 🐼 Digital	
🕀 🔯 Display	
🕀 🔯 Filters	=
🗈 🐼 Ports and Pins	
🕀 🔯 Power Supervision	
E 🐼 System	
Boost Converter [v5.0]	
Bootloadable [v1.20]	
Bootloader [v1.20]	
Clock [v2.20]	
 Die Temperature [v2.0] DMA [v1.70] 	
EEPROM [v2.10]	
Emulated EEPROM [v1.10]	-
Open datasheet	
Provides bootloadable application functionality.	
Inst_N	
Bootloadable	

Figure 6-43. 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 6-44. Select the *KitProg_Bootloader.hex* and *KitProg_Bootloader.elf* files; click **Open**, as shown in Figure 6-45 and Figure 6-46 and then click **OK** on the Configuration window.

Figure 6-44. Configuration Window of Bootloadable Component

Configure 'Bootloadable'	2		x
Name: Bootloadable_1			
General Dependencies Built-in		٩	Þ
Bootloadable projects require a reference to the associated Bootloader project's HE files. The HEX files extension is *.hex. The ELF files extension depends on IDE and *.elf, *.out, *.axf, or other.		F	
Bootloader HEX file:		_	
	rowse		
Bootloader ELF file:			
E	Prowse		
Datasheet OK Apply	Cancel		



Select a Bootloader Hex File	-	X
C V V V V V V V V V V V V V V V V V V V	Programmer KitProg_Bootloader	✓ Search KitProg_Bootloa
Organize 🔻 New folder		ii • 🗌 📀
E Desktop	Name	Date modified Type
 Downloads Dropbox 	KitProg_Bootloader.hex	3/18/2013 6:38 HEX File
S Recent Places		
 Libraries Documents Music Pictures Videos 		
💐 Computer		
😻 Windows7_OS (C:		
🚷 Lenovo_Recovery 👻 🤞	e III	4
File <u>n</u> ame	: KitProg_Bootloader.hex	← Hex Files (*.hex) ← Open Cancel

Figure 6-45. Select KitProg Bootloader Hex File

Figure 6-46. Select KitProg Bootloader Elf File

Select a Bootloader Hex File					
G V V Kirmware V Pr	ogrammer 🕨 KitProg_Bootloader	- - i i j	Search KitProg_B	ootloader 🔎	
Organize 🔻 New folder			8==	• 🔳 🔞	
☆ Favorites ■ Desktop	Name KitProg_Bootloader.elf		Date modified 1/18/2013 1:07 AM	Type ELF File	
Downloads	E				
Cibraries					
 Music Pictures Videos 					
Computer					
Local Disk (C:) CY8C58 Family Processo		III		•	
File <u>n</u> ame:		•	Elf Files (*.elf, .axf, Open	.out) Cancel	



- 4. Develop your custom project.
- 5. The NVL setting of the Bootloadable project and the KitProg_Bootloader project must be the same. Figure 6-47 shows the *KitProg_Bootloader.cydwr* system settings.

Figure 6-47. KitProg Bootloader System Settings

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

- 6. Build the project in PSoC Creator by choosing Build > Build Project or pressing [Shift] [F6].
- 7. To download the project onto the PSoC 5LP device, open the Bootloader Host tool, which is available from PSoC Creator. Choose **Tools > Bootloader Host**, as shown in Figure 6-48.

Figure 6-48. Open Bootloader Host Tool from PSoC Creator

Bootloadable - PSoC Creator 3.0 [C:\\Bootloadable\Bootloadable.cydsn\TopDesign\TopDesign.cy				
<u>File Edit View Project Build Debug</u>	<u>T</u> ools <u>W</u> indow <u>H</u> elp			
🚼 🎦 着 📂 🖬 🖉 🖨 🔍 🗼 🛍 🛍	Install drivers for µVision			
🔚 🗝 🚵 🦃 💕 🞇 💂 Microsoft San	Datapath Config Tool U			
Workspace Explorer	DMA Wizard Design.cysch			
a -	Bootloader Host			
Workspace 'Bootloadable' (1 Projects)	Options			
🖻 🔁 Project 'Bootloadable' [CY8C5868LTI-LF	(Juni 1997)			

8. In the Bootloader Host tool, click **Filters** and add a filter to identify the USB device. Set **VID** as **0x04B4**, **PID** as **0xF13B**, and click **OK**, as shown in Figure 6-49.



	IT Beetleader Tieet T	
🛓 Bootloader Host		
File Actions Help		
🖆 🗼 BB 📎 🛞		
File: C:\Users\msur\Documents\PSoC Creator\	Bootloadable\Bootloadable.cydsn\	CortexM3\ARM_GCC_473\Debug\B
Ports: Filters	Port Configuration 12C	Port Information
S Intel(R) Active Management	I2C address: 0	KitProg Version 2.07
5 KitProg/191607A0032C2400	I2C Speed	
	anagement Technology - SOL (COI	M3) M3)
Ready		

Figure 6-49. Port 'Filters' Tab in Bootloader Host Tool

9. In the Bootloader Host tool, click the **Open File** button to browse to the location of the bootloadable file (*.cyacd), as shown in Figure 6-50.

Figure 6-50. Open Bootloadable File from Bootloader Host Tool

🛓 Bootloader Host		
<u>File Actions H</u> elp		
🖆 🔰 BB 📎 🔘		
Fle: C:\Jsers\ancy\Desktop\Bootloadable project	t\Bootloadable.cydsn\CortexM3\ARM_GCC_	441\Debug\Bootloadable.cyacd
Forts: Filters	Port Configuration USB Vo configuration necessary for this port.	Port Information VID: 04B4 PID: F13B
Program Button		
Open File Button		
Log:		
12:35:02 PM - Selected device: USB Human Interfact 12:35:02 PM - Selected device: USB Human Interfact 12:35:08 PM - Selected device: USB Human Interfact	e Device (04B4_F13B)	



10.Keep the reset switch (SW1) pressed and plug in the USB Mini-B connector. If the switch is pressed for more than 100 ms, the PSoC 5LP enters into bootloader. click the **Program** button in the Bootloader Host tool to program the device.

Figure 6-51. Select Bootloadable .cyacd File from Bootloader Ho

🛓 Open		×
Solution - Solution	adable.cydsn → CortexM3 → ARM_GCC_473 → Debug →	✓ ✓ Search Debug
Organize 👻 New fo	older	i 🕶 🖬 🔞
☆ Favorites ■ Desktop	Documents library Debug	Arrange by: Folder ▼
Downloads	Name	Date modified Type
E Recent Places	a 🔐 .deps	4/15/2014 8:13 PM File folder
Skybine	Bootloadable.cyacd	4/15/2014 8:13 PM CYACD File
🥽 Libraries		
Documents		
J Music		
Pictures		
Videos		
-		
🖳 Computer	• • • III	•
File	e <u>n</u> ame: Bootloadable.cyacd	■ Bootloader Files (*.cyacd) ■ Open Cancel

11. If bootload is successful, the log of the tool displays "Programming Finished Successfully"; otherwise, it displays "Failed" and a statement for the failure.

Notes:

- The PSoC 5LP pins are brought to the PSoC 5LP GPIO header (J8). These pins are selected to support high-performance analog and digital projects. See A.2.2 PSoC 5LP GPIO Header (J8) for pin information.
- Take care when allocating the PSoC 5LP pins for custom applications. For example, P2[0]-P2[4] are dedicated for programming the PSoC 4. Refer to A.1 CY8CKIT-040 Schematics before allocating the pins.
- When a custom bootloadable project is programmed onto the PSoC 5LP, the initial capability of the PSoC 5LP to act as a programmer, USB-UART bridge, or USB-I²C bridge is not available.
- The status LED does not function unless used by the custom project.

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

6.4.2 Building a Normal Project for PSoC 5LP

A normal project is a completely new project created for the PSoC 5LP device on the CY8CKIT-040. Here, the entire flash of the PSoC 5LP is programmed, overwriting all bootloader and programming code. To recover the programmer, reprogram the PSoC 5LP device with the factory-set *KitProg.hex* file, which is shipped with the kit installer.

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



<Install_Directory>\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version>\Firmware\Programmer\KitProg

This advanced functionality requires a MiniProg3 programmer, which is not included with this kit. The MiniProg3 can be purchased from www.cypress.com/go/CY8CKIT-002. To build a normal project for the PSoC 5LP, follow these steps:

 In PSoC Creator, choose New > Project > PSoC 5LP; click the expand button adjacent to Advanced and select Device as CY8C5868LTI-LP039; select Application Type as Normal from the drop-down list, as shown in Figure 6-52.

Figure 6-52. Create a New Project in PSoC Creator

w Project			8	X
Design	Other			۹ ۵
Empty Temp	lates			^
📃 Emp	ty PSoC 3	Design	Creates a PSoC 3, 8 bit, design project.	
📃 Emp	ty PSoC 4	Design	Creates a PSoC 4, 32 bit, design project.	
📃 Emp	ty PSoC 5	LP Design	Creates a PSoC 5LP, 32 bit, design project.	Ε
 PSoC 3 Star 	ter Designs	s		
Pa ADC	_DMA_VD	AC	Shows how to transfer data from an ADC to a DAC using DMA with no CPU intervention.	
Dels	Sig_16Char	nnel	Shows a 16-channel, 12-bit Delta Sigma ADC in PSoC 3 sequenced in hardware; samples are transferred from ADC to SRAM using DMA - without processor intervention.	
Pa DelS	Sig_I2CM		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.	_
Pa DelS	Sig_I2CS		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.	
Dels	Sig_SPIM		Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over SPI.	_
🖻 Filte	er_ADC_VE	DAC	Shows how to filter an analog input all in hardware, and provides all the DMA setup to transfer the data from the ADC directly to the Digital Filter Block, bypassing the processor.	
PSoC 4 Star		rol_with_Alert	Shows how performing fan control in hardware completely frees up the CPU.	
- 1000 4 0 tal	tor Design	3		Ψ.
Name:	Design20)		
Location:	C:\Users	\msur\Documents\	PSoC Creator]
Advanced			· · · · · · · · · · · · · · · · · · ·	
Workspace:		Create New Work	space	•
Workspace Na	me:	Design20		
Device:		CY8C5868LTI-LP	039 - (Last Used PSoC 5LP Device)	-
Sheet Template	e:	Empty (11" x 8.5"))	•
Application Typ	e	Normal		-
			OK Cancel	

- 2. Develop your custom project.
- 3. Build the project in PSoC Creator by choosing Build > Build Project or pressing [Shift] [F6].
- 4. Connect the 10-pin connector of MiniProg3 to the onboard 10-pin SWD debug and programming header J7 (which needs to be populated).
- To program the PSoC 5LP with PSoC Creator, choose Debug > Program or press [Ctrl] [F5]. The Programming window shows MiniProg3 and the selected device in the project under it (CY8C5868LTI-LP039).



6. Click on the device and click **Connect** to program.

Notes:

- The 10-pin SWD debug and programming header (J7) is not populated. See the "No Load Components" section of Bill of Materials on page 146 for details.
- The PSoC 5LP pins are brought to the PSoC 5LP GPIO header (J8). These pins are selected to support high-performance analog and digital projects. See PSoC 5LP GPIO Header (J8) on page 143 for pin information.
- Take care when allocating the PSoC 5LP pins for custom applications. For example, P2[0]-P2[4] are dedicated for programming the PSoC 4. Refer to CY8CKIT-040 Schematics on page 136 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 used by the custom project.

6.5 **PSoC 5LP Factory Program Restore Instructions**

The CY8CKIT-040 PSoC 4000 Pioneer Kit features a PSoC 5LP device that comes factory-programmed as the onboard programmer and debugger for the PSoC 4 device.

In addition to creating applications for the PSoC 4 device, you can also create custom applications for the PSoC 5LP device on this kit as explained in Developing Applications for PSoC 5LP on page 119. Reprogramming or bootloading the PSoC 5LP device with a new flash image will overwrite the factory program and forfeit the ability to use the PSoC 5LP device as a programmer/debugger for the PSoC 4 device. Follow these instructions to restore the factory program on the PSoC 5LP and enable the programmer/debugger functionality.

6.5.1 PSoC 5LP Programmed with a Bootloadable Application

If the PSoC 5LP is programmed with a bootloadable application, restore the factory program by using one of the following two methods.

6.5.1.1 Restore PSoC 5LP Factory Program Using PSoC Programmer

- 1. Launch PSoC Programmer 3.18 or later from **Start > Cypress > PSoC Programmer**.
- Configure the kit in Service mode. To do this, while holding down the reset button (SW1 Reset), plug the PSoC 4000 Pioneer Kit into the computer using the included USB cable (USB A to Mini-B). This puts the PSoC 5LP into service mode, which is indicated by the blinking green status LED.
- 3. The following message shown in Figure 6-53 appears in the PSoC Programmer results window: "KitProg Bootloader device is detected."



<u> </u>		
PSoC Programmer		x
File View Options Help		
🔁 - 🔪 💿 BB 🚺		
Port Selection Progra	ammer Utilities JTAG	
Prog	gramming Parameters	
File	Path: C:\Program Files (x86)\Cypress\CY8CKIT-040 PSoC 4 Pioneer Kit\1.0\Firmware\Programmer	\KitPr
	✓ III	P.
Proc	grammer:	
Proc	gramming Mode: 🔘 Reset 🔘 Power Cycle 🔘 Power Detect	
Veri	ification: On Off Connector: 5p 10p	
Device Family	Detection: On Off Clock Speed: 1.6 MHz	
	grammer Characteristics	\equiv
	iocol: O JTAG O SWD O ISSP O 12C Execution Time:	
Deutee	age: 0 5.0 V 0 3.3 V 0 2.5 V 0 1.8 V	
CY8C5868LTI-LP039 -	<u>Voltage:</u> NA	
Actions	Results	
Connected at 6:40:11 PM	KitProg bootloader device is detected	
	Please close all ports, then navigate to the Utilities tab and click the Upgrade Firmware button to recover Bridge	
	Select Port in the PortList, then try to connect	
Device set to CY8C5868LTI-LP039 at 6:38:29 PM	262144 FLASH bytes	E
Device Family set to CY8C5xxxLP at 6:38:29 PM	i de la constante d	
Active HEX file set at 6:38:28 PM	C:\Program Files (x86)\Cypress\CY8CKIT-042 PSoC 4 Pioneer Kit\1.0\Firmware\Programmer\KitProg\KitProg.hex	
	Users must be aware that the following PSoC device should not be powered o programmed at 5V. Doing so will cause damage to the device: CY8C89xxx	r
Session Started at		
For Help, press F1	Not Conne	cted 🔡

Figure 6-53. PSoC Programmer Results Window

4. Switch to the **Utilities** tab in PSoC Programmer and click the **Upgrade Firmware** button, as shown in Figure 6-54. Unplug all other PSoC programmers (such as MiniProg3 and DVKProg) from the PC before clicking the **Upgrade Firmware** button.

Figure 6-54. Upgrade Firmware

PSoC Programmer			- 0 X
File View Options Help			
🔁 - 🔪 💿 BB 🖉			
Port Selection Progra	mmer Utilities JTAG		
Device Family CY8C5xxLP Device CY8C5868LTI-LP039	Ograde Firmware Click to upgrade connected device's firmware Erase Block Click to erase user specific flash block		
Actions	Results		
	KitProg bootloader device is detected Please close all ports, then navigate to the Utilities tab a Upgrade Firmware button to recover Bridge	and clic	k the



5. After programming is completed, the following message appears: "Firmware Update Finished at <time>," as shown in Figure 6-55.

Figure 6-55. Firmware Update Complete

PSoC Programmer	
File View Options Help	
🖆 · 🗼 💿 BB 🖉 🗎 🗅 🕒 (8
Port Selection IProgrammer Utilities JTA	G
	ck to upgrade connected device's firmware ck to erase user specific flash block
CY8C5xxxLP v	
Device	
Actions	Results
Successfully Connected to KitProg/110E192D00232400 at 6:42:32 PM Opening Port at 6:42:29 PM	KitProg Version 2.02
Connected at 6:42:29 PM	KitProg/110E192D00232400
Disconnected at 6:42:28 PM	Bootloader device E
Firmware Update Finished at 6:42:27 PM	
	Succeeded
	Verifying
	Upgrading
The second second second second second second second	Initializing
Firmware Upgrade Started at 6:42:21 PM Firmware Upgrade Requested at 6:42:21 PM	
Connected at 6:40:11 PM	KitProg bootloader device is detected
	Please close all ports, then navigate to the Utilities tab and click the Upgrade Firmware button to recover Bridge
	Select Port in the PortList, then try to connect
For Help, press F1	PASS Powered Connected

6. The factory program is now successfully restored on the PSoC 5LP. It can be used as the programmer/debugger for the PSoC 4 device.

6.5.1.2 Restore PSoC 5LP Factory Program Using USB Host Tool

 Launch the Bootloader Host tool from Start > Cypress > PSoC Creator, as shown in Figure 6-56.

Using the File > Open menu, load the *Kit Prog.cyacd* file, which is installed with the kit software. The default location for this file is <Install_Directory>\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version>\Firmware\Programmer\KitProg\KitProg.cyacd.



Figure 6-56. Load KitProg.cyacd File

Bootloader Host	
<u>File Actions H</u> elp	
File: [C:\Program Files (x86)\Cypress\CY8CKIT-040 PSoC 4 Pioneer Kit\1.0\Firmware\Programmer\KitProg\KitProg.cyacd	
Ports: Filters Port Configuration UART Port Information	
Baud 9600	
Data Bits 8	
Stop Bits One	
Parity None	
Log:	
06:15:14 PM - Selected device: Communications Port (COM1) 06:15:14 PM - Selected device: Communications Port (COM1)	
10, 13, 14 PM - Selected device, communications Poil (COMT)	
Ready	
Search KitProg Programmer > KitProg V 47 Search KitProg P	
Organize ▼ New folder 🛛 🕄 🐨 🗍 🔞	
Favorites Name Date modified Type Size	
E Desktop KitProg.cyacd 4/18/2013 1:07 AM CYACD File 121 KB	
Downloads	
Recent Places	
Image: Second Places Image: Second Places Image: Secon	
Image: Second Places Image: Second Place	
Image: Second Places Image: Second Places Image: Secon	
Recent Places Google Drive Dropbox No-Zoolz Zone SkyDrive Libraries Apps	
Recent Places Google Drive Dropbox No-Zoolz Zone SkyDrive Libraries Apps Documents	
Recent Places Google Drive Dropbox No-Zoolz Zone SkyDrive	

- Configure the Pioneer Kit in Service mode. To do so, while holding down the reset button (SW1 Reset), plug the PSoC 4000 Pioneer Kit into the computer using the included USB cable (USB A to Mini-B). This puts the PSoC 5LP into service mode, which is indicated by the blinking green status LED.
- 3. In the Bootloader Host tool, set the filters for the USB devices with VID: 04B4 and PID: F13B. The **USB Human Interface Device** port appears in the Ports list. Click that port to select it.



Bootloader Host				
<u>File Actions H</u> elp				
🖆 💊 BB 📎 🔕				
File: C:\Program Files (x86)\Cypress\CY8CKIT-	040 PSoC 4 Pioneer Kit\1.0)\Firmware\Pro	grammer\KitProg\KitProg.cyacd	
Ports: Filters Communications Port (COM1) USB Human Interface Device.	Port Configuration No configuration necessa port.	USB *	Port Information VID: 04B4 PID: F13B	
Log: 06:15:14 PM - Selected device: Communications 06:15:14 PM - Selected device: Communications 06:21:45 PM - Selected device: USB Human Inte 06:25:53 PM - Selected device: Communications 06:25:58 PM - Selected device: Communications 06:25:58 PM - Selected device: USB Human Inte 06:26:09 PM - Selected device: USB Human Inte	Port (COM1) erface Device (04B4_F13B Port (COM1) erface Device (04B4_F13B Port (COM1))		
Ready				

Figure 6-57. Select USB Human Interface Device

- 4. Click the **Program** button (or menu item **Actions > Program**) to restore the factory program by bootloading it onto the PSoC 5LP.
- 5. After programming is completed, the following message appears "Programming Finished Successfully," as shown in Figure 6-58.

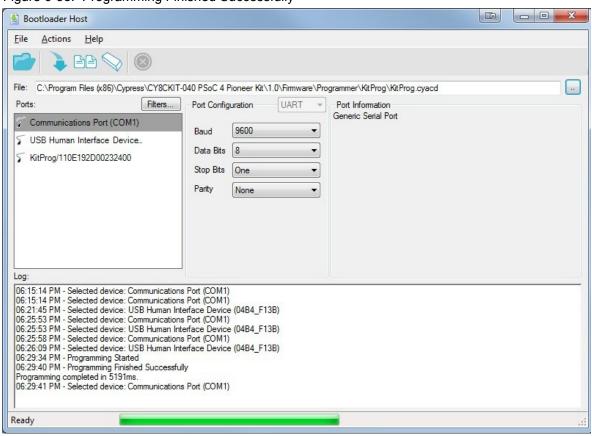


Figure 6-58. Programming Finished Successfully

6. The factory program is now successfully restored on the PSoC 5LP. You can now use it as the programmer/debugger for the PSoC 4 device.

6.5.2 PSoC 5LP Programmed with a Standard Application

If PSoC 5LP is programmed with a standard application, restore the factory program by using the following method.

- 1. Launch PSoC Programmer 3.18 or later from Start > Cypress > PSoC Programmer.
- 2. Use the File > Open menu to load the *KitProg.hex* factory program hex file, which is shipped with the kit. The default location for this file is <Install_Directory>\CY8CKIT-040 PSoC 4000 Pioneer Kit\<version>\Firmware\Programmer\KitProg.
- Connect a CY8CKIT-002 MiniProg3 (sold separately) to the computer. The 10-pin connector cable on the MiniProg3 plugs into the header [J7]. Note that the J7 header is unpopulated. For more details, see Bill of Materials on page 146.
- 4. Ensure that MiniProg3 is the selected port in PSoC Programmer and the 10-pin connector (10p option) is selected, as shown in Figure 6-59. If the board is not powered over USB, select the **Power Cycle** programming mode.

Figure 6-59. Select MiniProg3

PSoC Programmer			- • ×
File View Options H	elp		
🖆 · 🗼 🔘 BB			
Port Selection	Programmer Utilities JTAG		
MiniProg3/3209AA000002	Programming Parameters File Path: C:\Program Files (x86)\Cypress\CY8CKIT-040 PSoC 4 Pioneer Kitt1.0/Firmware\Programmer(KitProglKitProglKitProg hex Programmer: MiniProg3/3209AA000002 Programming Mode: © Reset © Power Detect		
	Verification: ● On Off Connector; ● 5p ● 10p AutoDetection; ● On Off Clock Speed: 1.6 MHz ▼		
Device Family CY8C500xLP Device CY8C5868LTI-LP039	AutoDetection: Image: Organity of the system Off Clock Speed: 1.6 MHz Image: Organity of the system Image: Organity of the system		
Actions	Results		
Successfully Connec MiniProg/3209A000 Opening Port at 6:5	002 at 6:58:42 PM MiniProg3 version 2.05 [3.08/2.05]		
For Help, press F1	FALL Not Por	rered	Connected

- 5. When ready, select the **Program** button (or **File > Program**) to program the PSoC 5LP device, as shown in Figure 6-60.
- 6. After programming is completed, the following message appears: "Program Finished at <time>."



Figure 6-60. Program Finished

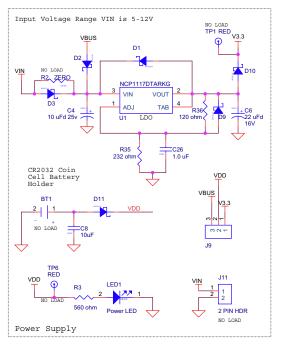
PSoC Programmer		
File View Options H	Help	
🖆 · 🔪 💿 BB		
Port Selection	Programmer Utilities JTAG	
MiniProg3/3209AA000002	Programming Parameters Elle Path: C:Program Files (x86))Cypress(CY8CKIT-040 PSoC 4 Pioneer Kit1.0)Firmware\Programmer/KitProg/KitProg.hex	
Program Button	Programmer. MiniProg3/3209AA000002 Programming Mode: © Reset @ Power Oycle Power Detect Verification: @ On Off Connector: © 5p @ 10p	
Device Family	AutoDetection: On Off <u>Clock Speed:</u> 1.6 MHz	
CY8C5xxxLP * Device CY8C5868LTI-LP039 *	Programmer Characteristics Status 19.5 seconds Protocol: JTAG SWD ISSP I2C Voltace: S.0.V 3.3.V 2.5.V 18.V Voltace: S.0.V 3.3.V 2.5.V 0.8.V	
Actions	Results	
Program Finished at	t 7:00:04 PM Programming Succeeded	
	Programming Succeeded Doing Checksum	
	Doing Frotest	
	Programming of Flash Succeeded	
	Programming of Flash Starting	
	Erag Succeeded	
Device set to CY8C5 PM	5868LTI-LP039 at 6:59:55 262144 FLASH bytes	
	to CY8C5xxxLP at 6:59:55	
	Automatically Detected Device: CY8C5868LTI-LP039	
Program Requested a	at 6:59:44 PM	
Successfully Connec MiniProg3/3209AA000		
Opening Port at 6:5	58:41 PM	
For Help, press F1	PASS Not Powered	Connected .

7. The factory program is now successfully restored on the PSoC 5LP. You can use it as the programmer/debugger for the PSoC 4 device.

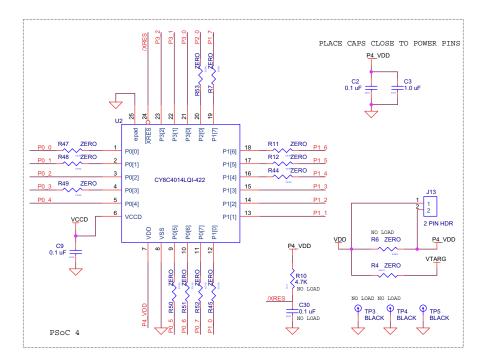
A. Appendix

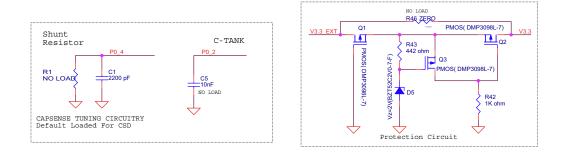




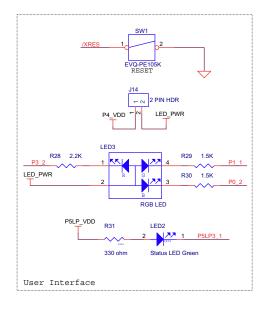


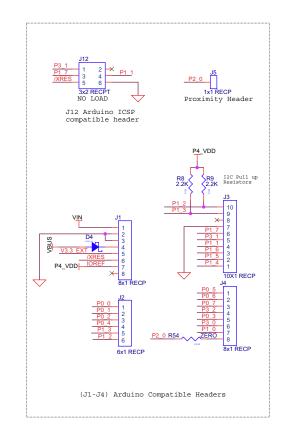


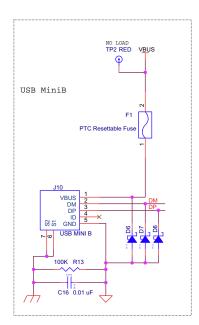


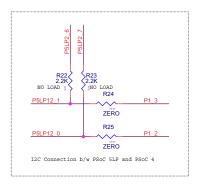




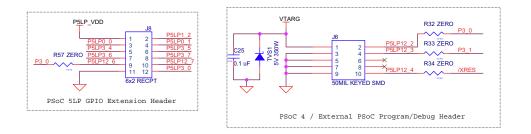


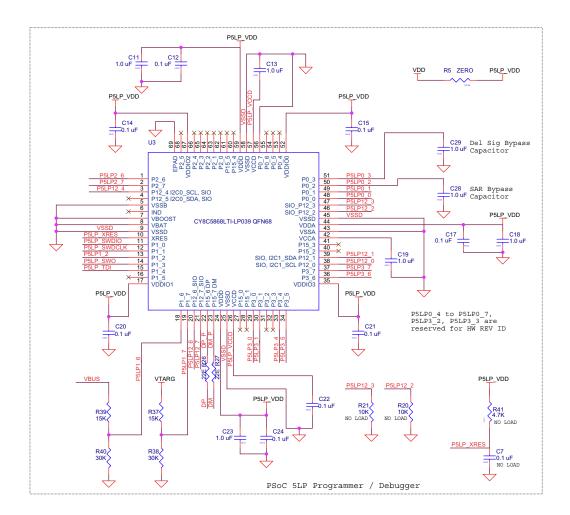


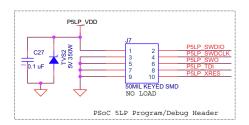




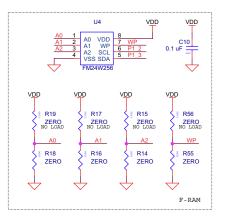




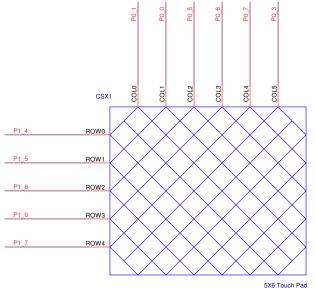


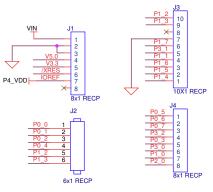






A.1.1 CapSense Touchpad Shield Board





(J1-J4) Arduino Compatible Headers

CapSense Touch Pad



A.2 Pin Assignment Table

This section provides the pin map of the headers and their usage.

A.2.1 Arduino Compatible Headers (J1, J2, J3, J4, and J12)

Power Connector (J1)						
Pin	Baseboard Signal	Trackpad Shield Signal				
J1_01	VIN	NC				
J1_02	GND	GND				
J1_03	GND	GND				
J1_04	V5.0	NC				
J1_05	V3.3	NC				
J1_06	RESET	NC				
J1_07	IOREF/P4_VDD	NC				
J1_08	NC	NC				

J2 Connector							
Pin	Baseboard Signal	Trackpad Shield Signal					
J2_01 (A0)	P0[0]	TRACK_COLUMN1					
J2_02 (A1)	P0[1]	TRACK_COLUMN0					
J2_03 (A2)	P0[2] (TCPWM_LINE/Blue LED, CTANK)	NC					
J2_04 (A3)	P0[4] (CMOD)	NC					
J2_05 (A4)	P1[3] (SDA)	NC					
J2_06 (A5)	P1[2] (SCL)	NC					

J3 Connector						
Pin	Baseboard Signal	Trackpad Shield Signal				
J3_01 (D8)	P1[4]	TRACK_ROW0				
J3_02 (D9)	P1[5]	TRACK_ROW1				
J3_03 (D10)	P1[6]	TRACK_ROW2				
J3_04 (D11)	P1[1] (TCPWM_LINE/Green LED)	NC				
J3_05 (D12)	P3[1] (SWDCK)	NC				
J3_06 (D13)	P1[7]	TRACK_ROW4				
J3_07	GND	GND				
J3_08	NC	NC				
J3_09	P1[3] (SDA)	NC				
J3_10	P1[2] (SCL)	NC				



J4 Connector							
Pin	Baseboard Signal	Trackpad Shield Signal					
J4_01 (D0)	P0[5]	TRACK_COLUMN2					
J4_02 (D1)	P0[6]	TRACK_COLUMN3					
J4_03 (D2)	P0[7]	TRACK_COLUMN4					
J4_04 (D3)	P3[2] (TCPWM_LINE, Red LED)	NC					
J4_05 (D4)	P0[3]	TRACK_COLUMN5					
J4_06 (D5)	P3[0] (SWDIO)	NC					
J4_07 (D6)	P1[0]	TRACK_ROW3					
J4_08 (D7)	P2[0] (PROX)	NC					

	J12					
Pin	Kit Signal	PSoC 4 Description				
J12_01	P3[1]	GPIO				
J12_02	NC	NC				
J12_03	P1[7]	GPIO				
J12_04	P1[1]	GPIO				
J12_05	/XRES	PSoC 4 RESET				
J12_06	GND	GND				



A.2.2 PSoC 5LP GPIO Header (J8)

J8 is a 2×6 header that connects PSoC 5LP pins to support GPIO controls for custom PSoC 5LP projects.

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

A.3 Program and Debug Headers

A.3.1 PSoC 4 Direct Program/Debug Header (J6)

	J6						
Pin	PSoC 5LP Signal	PSoC 4 Signal	Description	Pin	PSoC 5LP Signal	PSoC 4 Signal	Description
J6_01	VDD	VDD	VCC	J6_02	P12[2]	P3[0]	TMS/SWDIO
J6_03	GND	GND	GND	J6_04	P12[3]	P3[1]	TCLK/SWCLK
J6_05	GND	GND	GND	J6_06	NC	NC	NC
J6_07	GND	GND	GND	J6_08	NC	NC	NC
J6_09	GND	GND	GND	J6_10	P12[4]	XRES	RESET

A.3.2 PSoC 5LP Direct Program/Debug Header (J7)

	J7							
Pin	PSoC 5LP Signal	Description	Pin	PSoC 5LP Signal	Description			
J7_01	VDD	VCC	J7_02	P1[0]	TMS/SWDIO			
J7_03	GND	GND	J7_04	P1[1]	TCLK/SWCLK			
J7_05	GND	GND	J7_06	P1[3]	TDO/SWO			
J7_07	GND	GND	J7_08	P1[4]	TDI			
J7_09	GND	GND	J7_10	XRES	RESET			



A.4 Use of Zero-ohm Resistors and No Load

Unit	Resistor	Usage
Power supply	R2	Solder zero-ohm resistors to access voltage from VBUS (USB).
I ² C connection between PSoC 5LP and PSoC 4	R24 and R25	Unsolder the resistors to communicate with an external PSoC using the PSoC 5LP. Removing these will disable the PSoC 4 I ² C communication with the PSoC 5LP device.
PSoC 4/external PSoC program/ debug header	R32, R33, and R34	Unsolder the resistors to disconnect SWD lines from the PSoC 4. Use J6 to connect and program an external PSoC. Removing these will disable PSoC 4 programming by the PSoC 5LP device and through the J6 header.
Protection circuit	R46	Solder zero-ohm resistors to bypass the entire protection circuitry.
CapSense tuning circuitry	R1	Convert IDAC output to a voltage, or used as a bleed resistor to CMOD.
CapSense tuning circuitry/user inter- face	R30	Unsolder R30, which connects P0[2] to the blue LED to enable shield tank capacitor C5 on P0[2].
PSoC 4	R4, R6	Unsolder R4 to remove supply to VTARG and solder zero-ohm resistors R6 to supply P4_VDD with VDD instead of J13.
PSoC 5LP programmer/debugger	R11, R12, R14, R15, R16	For future use.
	R5	Unsolder the zero-ohm resistor to cut the VDD supply to PSoC 5LP.
	R7	For future use.
F-RAM	R14, R15, R16, R17, R18, and R19	Select the lower three bits of the F-RAM I^2C slave address. R14- R15 selects bit 2 (A2), R16-R17 selects bit 1(A1), and R18-R19 selects bit 0 (A0). The selected bits are OR'ed with the F-RAM fam- ily's I^2C address (0x50) to decide the slave address for the one on the board.
	R56 and R55	Solder a zero-ohm resistor for R56 to write-protect the entire F-RAM memory. R55 is not required to be populated as the WP pin is internally pulled down. When the WP pin is left floating or R55 is populated, write access to F-RAM is restored.
PSoC 5LP GPIO header	R57	Unsolder the zero-ohm resistor to disconnect P3[0] from the PSoC 5LP RX line and use P3[0] for PSoC 4 debug or PSoC 5LP as a USB-UART bridge for another device.



A.5 Error in Firmware/Status Indication in Status LED

	User Indication	Scenario	Action Required by user
1	LED blinks at a fast rate (ON Time = 0.25 s, OFF Time = 0.25 s)	Bootloadable file is corrupt	Bootload the *. <i>cyacd</i> file over the USB interface, which is shipped with PSoC Programmer using the Boot- loader Host GUI shipped with PSoC Creator. The files are located in the PSoC Programmer root installation directory.
2	LED blinks at a slow rate (ON Time = 1.5s, OFF Time = 1.5s)	Entered Bootloader by pressing the PSoC 4 Reset switch	Unplug power and plug it in again if you entered this mode by mistake; the LED gives the indication. If the mode entry was intentional, bootload the new *. <i>cyacd</i> file using the Bootloader Host tool shipped with PSoC Creator.
3	LED glows steadily	Programmer appli- cation is running successfully	USB is enumerated successfully and the programmer is up and running. The PSoC 4 device can now be pro- grammed any time using the onboard PSoC 5LP pro- grammer.

Note: LED status is not applicable when a custom project is running in PSoC 5LP.



A.6 Bill of Materials

A.6.1 CY8CKIT-040 Baseboard

No.	Qty	Reference	Value	Description	Manufacturer	Mfr Part Number
1				PCB, 68.58 mm x 53.34 mm, High Tg, ENIG finish, 4 layer, Color = RED, Silk = WHITE.	Cypress	
2	1	C1	2200 pFd	CAP CER 2200PF 50V 5% NP0 0805	Murata	GRM2165C1H222JA01D
3	13	C2,C9,C10,C12, C14,C15,C17,C2 0,C21,C22,C24, C25,C27	0.1 uFd	CAP .1UF 16V CERAMIC Y5V 0402	Panasonic - ECG	ECJ-0EF1C104Z
4	9	C3,C11,C13,C18 ,C19,C23,C26,C 28,C29	1.0 uFd	CAP CERAMIC 1.0UF 25V X5R 0603 10%	Taiyo Yuden	TMK107BJ105KA-T
5	1	C4	10 uF 25V	CAP TANT 10UF 25V 10% 1210	AVX Corporation	TPSB106K025R1800
6	1	C6	22 uF 16V	CAP TANT 22UF 16V 10% 1210	AVX Corporation	TPSB226K016R0600
7	1	C16	0.01 uFd	CAP 10000PF 16V CERAMIC 0402 SMD	Panasonic - ECG	ECJ-0EB1C103K
8	1	C8	10uFd	CAP CER 10UF 6.3V 20% X5R 0603	Samsung Electro- Mechanics America, Inc	CL10A106MQ8NNNC
9	7	D1,D2,D3,D4,D9 ,D10,D11	MBR05	DIODE SCHOTTKY 0.5A 20V SOD- 123	Fairchild Semiconduc- tor	MBR0520L
10	1	LED1	Power LED Amber	LED 595NM AMB DIFF 0805 SMD	Avago Technologies	HSMA-C170
11	1	D5	2V Zener	DIODE ZENER 2V 500MW SOD123	Diodes Inc	BZT52C2V0-7-F
12	3	D6, D7, D8	ESD diode	SUPPRESSOR ESD 5VDC 0603 SMD	Bourns Inc.	CG0603MLC-05LE
13	1	LED3	RGB LED	LED RED/GREEN/BLUE PLCC4 SMD	Cree, Inc.	CLV1A-FKB- CJ1M1F1BB7R4S3
14	1	LED2	Status LED Green	LED GREEN CLEAR 0805 SMD	Chicago Miniature	CMD17-21VGC/TR8
15	1	F1	FUSE	PTC Resettable Fuses 15Volts 100Amps	Bourns	MF-MSMF050-2
16	2	J1, J4	8x1 RECP	CONN HEADER FEMALE 8POS .1" GOLD	Sullins Connector Solutions	PPPC081LFBN-RC
17	1	J2	6x1 RECP	CONN HEADER FMAL 6POS.1" GOLD	Sullins Connector Solutions	PPPC061LFBN-RC
18	1	J3	10x1 RECP	CONN HEADER FMALE 10POS .1" GOLD	Sullins Connector Solutions	PPPC101LFBN-RC
19	1	J5	1X1 RECP	CONN RCPT 1POS .100" SNGL HORZ	Samtec Inc	BCS-101-L-S-HE
20	1	J6	50MIL KEYED SMD	CONN HEADER 10 PIN 50MIL KEYED SMD	Samtec	FTSH-105-01-L-DV-K
21	1	J8	6x2 RECP	CONN HEADER FMAL 12PS.1" DL GOLD	Sullins Connector Solutions	PPPC062LFBN-RC
22	1	Jð	3p_jumper	CONN HEADER VERT SGL 3POS GOLD	3M	961103-6404-AR
23	1	J10	USB MINI B	CONN USB RECEPTACLE 5POS RT ANG	Molex Inc	0548190519



No.	Qty	Reference	Value	Description	Manufacturer	Mfr Part Number
24	2	J13,J14	2p_jumper	CONN HEADER VERT SGL 2POS GOLD	3M	961102-6404-AR
25	3	Q1,Q2,Q3	PMOS	MOSFET P-CH 30V 3.8A SOT23-3	Diodes Inc	DMP3098L-7
26	1	R3	560 ohm	RES 560 OHM 1/8W 5% 0805 SMD	Panasonic - ECG	ERJ-6GEYJ561V
27	24	R4,R7,R11,R12, R14,R16,R18,R2 4,R25,R32,R33, R34,R44,R45,R4 7,R48,R49,R50, R51,R52,R53,R5 4,R55,R57	ZERO	RES 0.0 OHM 1/10W 0603 SMD	Panasonic-ECG	ERJ-3GEY0R00V
28	1	R5	ZERO	RES 0.0 OHM 1/8W 0805 SMD	Panasonic-ECG	ERJ-6GEY0R00V
29	4	R8,R9	2.2K	RES 2.2K OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ222V
30	1	R13	100K	RES 100K OHM 1/10W 5% 0402 SMD	Panasonic - ECG	ERJ-2GEJ104X
31	2	R26, R27	22E	RES 22 OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF22R0V
32	1	R28	2.2K	RES 2.2K OHM 1/8W 5% 0805 SMD	Panasonic - ECG	ERJ-6GEYJ222V
33	2	R29,R30	1.5K	RES 1.5K OHM 1/8W 5% 0805 SMD	Panasonic - ECG	ERJ-6GEYJ152V
34	1	R31	330 ohm	RES 330 OHM 1/8W 5% 0805 SMD	Panasonic - ECG	ERJ-6GEYJ331V
35	1	R35	232 ohm	RES 232 OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF2320V
36	1	R36	120 ohm	RES 120 OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF1200V
37	2	R37,R39	15K	RES 15K OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ153V
38	2	R38,R40	30K	RES 30K OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ303V
39	1	R42	1K	RES 1K OHM 1/8W 5% 0805 SMD	Panasonic - ECG	ERJ-6GEYJ102V
40	1	R43	442 ohm	RES 442 OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF4420V
41	1	SW1	SW PUSHBUT- TON	SWITCH TACTILE SPST-NO 0.05A 12V	Panasonic - ECG	EVQ-PE105K
42	1	TP5	BLACK	TEST POINT PC MINI .040"D Black	Keystone Electronics	5001
43	2	TVS1,TVS2	5V 350W	TVS UNIDIR 350W 5V SOD-323	Dioded Inc.	SD05-7
44	1	U1	NCP1117DTAR KG	NCP1117DTARKG	ON Semiconductor	NCP1117DTARKG
45	1	U2	PSoC 4 S0 (CY8C400)	24 QFN PSoC4 S0 target chip	Cypress Semiconduc- tor	CY8C4014LQI-422
46	1	U3	PSoC 5LP (CY8C5868LTI- LP039)	68QFN PSoC 5LP chip for USB debug channel and USB-Serial interface	Cypress Semiconduc- tor	CY8C5868LTI-LP039
47	1	U4	F-RAM	F-RAM with I ² C interface	Cypress Semiconduc- tor	FM24W256-G
No Lo	oad Co	mponents				
48	1	BT1	Coin Cell Bat- tery Holder	HOLDER CR2032 GOLD LEADS SMD	MPD	BU2032SM-BT-GTR
49	1	C5	10000 pFd	CAP CER 10000PF 50V 5% NP0 0805	Murata	GRM2195C1H103JA01D
50	2	C7,C30	0.1 uFd	CAP .1UF 16V CERAMIC Y5V 0402	Panasonic - ECG	ECJ-0EF1C104Z
51	1	J7	50MIL KEYED SMD	CONN HEADER 10 PIN 50MIL KEYED SMD	Samtec	FTSH-105-01-L-DV-K
52	1	J11	2 PIN HDR	CONN HEADER FEMALE 2POS .1" GOLD	Sullins Connector Solutions	PPPC021LFBN-RC
53	1	J12	3X2 RECP	CONN HEADER .100 DUAL STR 12POS	Sullins Connector Solutions	PBC06DFAN

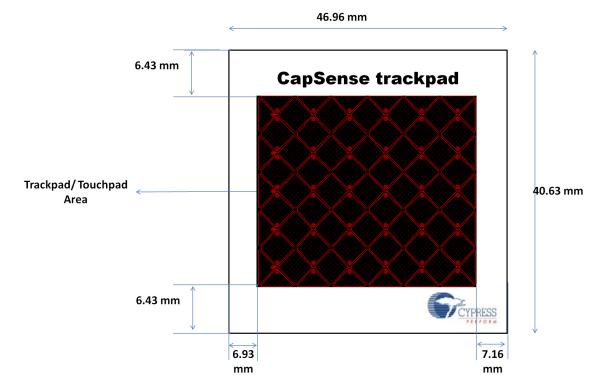


No.	Qty	Reference	Value	Description	Manufacturer	Mfr Part Number
54	7	R1,R2,R15,R17, R19,R46,R56	ZERO	RES 0.0 OHM 1/10W 0603 SMD	Panasonic-ECG	ERJ-3GEY0R00V
55	1	R6	ZERO	RES 0.0 OHM 1/8W 0805 SMD	Panasonic-ECG	ERJ-6GEY0R00V
56	2	R10,R41	4.7K	RES 4.7K OHM 1/10W 5% 0603 SMD	Panasonic-ECG	ERJ-3GEYJ472V
57	2	R20,R21	10K	RES 10K OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ103V
58	2	R22,R23	2.2K	RES 2.2K OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ222V
59	3	TP1,TP2,TP6	RED	TEST POINT PC MINI .040"D RED	Keystone Electronics	5000
60	2	TP3,TP4	BLACK	TEST POINT PC MINI .040"D Black	Keystone Electronics	5001
Insta	ll on B					
61	4	N/A	N/A	BUMPON CYLINDRICAL .312X.215 BLACK	ЗМ	SJ61A6
Spec	ial Jun					
62	1	19	Install jumper across pins 1 and 2	Rectangular Connectors MINI JUMPER GF 6.0MM CLOSE TYPE BLACK	Kobiconn	151-8010-E
63	1	J13	Install jumper across pins 1 and 2	Rectangular Connectors MINI JUMPER GF 6.0MM CLOSE TYPE BLACK	Kobiconn	151-8010-E
64	1	J14	Install jumper across pins 1 and 2	Rectangular Connectors MINI JUMPER GF 6.0MM CLOSE TYPE BLACK	Kobiconn	151-8010-E

A.6.2 CY8CKIT-040 Trackpad Shield Board

No.	Qty.	Reference	Value	Description	Manufacturer	Mfr Part Number
1				PCB, 53.34 mm x 53.34 mm, High Tg, ENIG finish, 2 layer, Color = RED, Silk = WHITE.	Cypress	
2	2	J1,J4	CON8	CONN HEADER 8POS .100 STR 30AU	FCI	68001-108HLF
3	1	J2	CON6	CONN HEADER 6POS .100 STR 30AU	FCI	68001-106HLF
4	1	J3	CON10	CONN HEADER 10POS .100 STR 30AU	FCI	68001-110HLF





A.7 Trackpad/Touchpad Sticker Details

A.8 Regulatory Compliance Information

The CY8CKIT-040 PSoC 4000 Pioneer Kit has been tested and verified to comply with the following electromagnetic compatibility (EMC) regulations:

- EN 55022:2010 Class A Emissions
- EN 55024:2010 Class A Immunity

Revision History



CY8CKIT-040 PSoC® 4000 Pioneer Kit Guide Revision History

Document Title: CY8CKIT-040 PSoC 4000 Pioneer Kit Guide							
Documen	Document Number: 001-91316						
Revision	Issue Date	Origin of Change	Description of Change				
**	04/21/2014	RKAD	New kit guide				