

# CY8CKIT-001

# PSoC<sup>®</sup> Development Kit Guide

Doc. # 001-48651 Rev. \*L November 16, 2012

Cypress Semiconductor 198 Champion Court San Jose, CA 95134-1709 Phone (USA): 800.858.1810 Phone (Intnl): 408.943.2600 http://www.cypress.com



#### Copyrights

© Cypress Semiconductor Corporation, 2009-2012. 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 Designer<sup>™</sup> and PSoC Creator<sup>™</sup> are trademarks and PSoC<sup>®</sup> and CapSense<sup>®</sup> are registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

#### **Flash Code Protection**

Cypress products meet the specifications contained in their particular Cypress PSoC Datasheets. Cypress believes that its family of PSoC 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



1. Inti	roduct	ion	7
	1.1 I	Kit Overview	7
	1.2 I	Kit Contents	7
	1.3 I	nstallation	9
		1.3.1 Before You Begin	9
		1.3.2 Prerequisites	9
		1.3.3 Installing PSoC 1 Development Software	9
		1.3.4 Installing PSoC 3 Development Software	
		1.3.5 Installing PSoC 5LP Development Software	.10
	1.4 I	PSoC Development Board	.11
		1.4.1 Default Switch and Jumper Settings	.11
	1.5 I	Kit Revision	.13
	1.6 /	Additional Resources	.14
		1.6.1 Beginner Resources	
		1.6.2 Engineers Looking for More	
		1.6.3 Learning from Peers	
		1.6.4 More Code Examples	
		Document Conventions	
	1.8 I	Document Revision History	.17
2 1 0	adina	My First PSoC Project	10
2. Loa	-	My First PSoC Project	19
2. Loa	2.1 I	My First PSoC 1 (CY8C28) Project	.20
2. Loa	2.1 I	My First PSoC 1 (CY8C28) Project 2.1.1 Loading My First PSoC 1 Project	.20 .20
2. Loa	2.1 I	My First PSoC 1 (CY8C28) Project 2.1.1 Loading My First PSoC 1 Project 2.1.2 Building My First PSoC 1 Project	.20 .20 .21
2. Loa	2.1	My First PSoC 1 (CY8C28) Project 2.1.1 Loading My First PSoC 1 Project 2.1.2 Building My First PSoC 1 Project 2.1.3 Programming My First PSoC 1 Project	.20 .20 .21 .22
2. Loa	2.1	My First PSoC 1 (CY8C28) Project 2.1.1 Loading My First PSoC 1 Project 2.1.2 Building My First PSoC 1 Project 2.1.3 Programming My First PSoC 1 Project 2.1.4 Running My First PSoC 1 Project	.20 .20 .21 .22 .23
2. Loa	2.1   2.2   2.2	My First PSoC 1 (CY8C28) Project 2.1.1 Loading My First PSoC 1 Project 2.1.2 Building My First PSoC 1 Project 2.1.3 Programming My First PSoC 1 Project 2.1.4 Running My First PSoC 1 Project My First PSoC 1 (CY8C29) Project	.20 .20 .21 .22 .23 .25
2. Loa	2.1 I 2.2 I 2.2 I	My First PSoC 1 (CY8C28) Project 2.1.1 Loading My First PSoC 1 Project 2.1.2 Building My First PSoC 1 Project 2.1.3 Programming My First PSoC 1 Project 2.1.4 Running My First PSoC 1 Project My First PSoC 1 (CY8C29) Project 2.2.1 Loading My First PSoC 1 Project	.20 .20 .21 .22 .23 .25 .25
2. Loa	2.1   2.2   2.2	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         Wy First PSoC 1 (CY8C29) Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project	.20 .20 .21 .22 .23 .25 .25 .26
2. Loa	2.1 I 2.2 I 2.2 I	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project	.20 .21 .22 .23 .25 .25 .26 .27
2. Loa	2.1 I 2.2 I	My First PSoC 1 (CY8C28) Project	.20 .20 .21 .22 .23 .25 .25 .26 .27 .28
2. Loa	2.1 [ 2.2 ] 2.3 ]	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Loading My First PSoC 1 Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         My First PSoC 3 (CY8C38) Project	.20 .20 .21 .22 .23 .25 .25 .26 .27 .28 .30
2. Loa	2.1 [ 2.2 ] 2.3 ]	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.5 PSoC 1 (CY8C29) Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.3 Programming My First PSoC 3 Project         2.2.4 Running My First PSoC 3 Project         2.3.1 Loading My First PSoC 3 Project	.20 .20 .21 .22 .23 .25 .25 .26 .27 .28 .30 .30
2. Loa	2.1 I 2.2 I 2.3 I	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 3 Project         2.3.1 Loading My First PSoC 3 Project         2.3.2 Building My First PSoC 3 Project	.20 .20 .21 .22 .23 .25 .25 .25 .26 .27 .28 .30 .30 .31
2. Loa	2.1 [ 2.2 ] 2.3 ]	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Loading My First PSoC 1 Project         2.1.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 3 Project         2.3.1 Loading My First PSoC 3 Project         2.3.2 Building My First PSoC 3 Project         2.3.3 Programming My First PSoC 3 Project	.20 .21 .22 .23 .25 .25 .26 .27 .28 .30 .30 .31 .32
2. Loa	2.1   2.2   2.3   2.3	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.5 Programming My First PSoC 3 Project         2.3.1 Loading My First PSoC 3 Project         2.3.2 Building My First PSoC 3 Project         2.3.3 Programming My First PSoC 3 Project         2.3.4 Running My First PSoC 3 Project         2.3.4 Running My First PSoC 3 Project	.20 .21 .22 .23 .25 .25 .26 .27 .28 .30 .31 .32 .33
2. Loa	2.1 [ 2.2 [ 2.3 [ 2.4 ]	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Loading My First PSoC 1 Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.3 Loading My First PSoC 1 Project         2.2.4 Running My First PSoC 3 Project         2.3.1 Loading My First PSoC 3 Project         2.3.2 Building My First PSoC 3 Project         2.3.3 Programming My First PSoC 3 Project         2.3.4 Running My First PSoC 3 Project         3.4 Running My First PSoC 3 Project	.20 .21 .22 .23 .25 .25 .26 .30 .30 .31 .32 .33 .34
2. Loa	2.1 [ 2.2 ] 2.3 ] 2.4 ]	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.3.1 Loading My First PSoC 3 Project         2.3.2 Building My First PSoC 3 Project         2.3.3 Programming My First PSoC 3 Project         2.3.4 Running My First PSoC 5 LP (CY8C58LP) Project         2.4.1 Loading my First PSoC 5LP Project	.20 .21 .22 .23 .25 .25 .25 .26 .27 .28 .30 .31 .32 .33 .34 .34
2. Loa	2.1 [ 2.2 ] 2.3 [ 2.4 ]	My First PSoC 1 (CY8C28) Project	.20 .21 .22 .23 .25 .25 .26 .27 .28 .30 .31 .32 .33 .34 .34 .35
2. Loa	2.1 [ 2.2 ] 2.3 ] 2.4 ]	My First PSoC 1 (CY8C28) Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.3 Programming My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.1.1 Loading My First PSoC 1 Project         2.1.2 Building My First PSoC 1 Project         2.1.4 Running My First PSoC 1 Project         2.2.1 Loading My First PSoC 1 Project         2.2.2 Building My First PSoC 1 Project         2.2.3 Programming My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.2.4 Running My First PSoC 1 Project         2.3.1 Loading My First PSoC 3 Project         2.3.2 Building My First PSoC 3 Project         2.3.3 Programming My First PSoC 3 Project         2.3.4 Running My First PSoC 5 LP (CY8C58LP) Project         2.4.1 Loading my First PSoC 5LP Project	.20 .21 .22 .23 .25 .25 .26 .27 .28 .30 .31 .32 .33 .34 .34 .35 .36



3. Sample	Projects	39
3.1	CY8C28 Family Processor Module Code Examples	39
	3.1.1 My First PSoC 1 (CY8C28) Project	
	3.1.2 ADC to LCD Project	
	3.1.3 ADC to UART with DAC	
	3.1.4 CapSense	
3.2	CY8C29 Family Processor Module Code Examples	75
	3.2.1 My First PSoC 1 (CY8C29) Project	
	3.2.2 ADC to LCD Project	
	3.2.3 ADC to LCD with DAC and UART	90
3.3	CY8C38 / CY8C58LP Family Processor Module Code Examples	
	3.3.1 My First PSoC 3 / PSoC 5LP Project	
	3.3.2 ADC to LCD Project	114
	3.3.3 ADC to UART with DAC	120
	3.3.4 USB HID	136
	3.3.5 CapSense	148
	3.3.6 SAR ADC (PSoC 5LP Only)	
Appendix	A. Board Specifications and Layout	171
A.1	PSoC Development Board	171
	A.1.1 Factory Default Configuration	171
	A.1.2 Power Supply Configuration Examples	172
	A.1.3 Prototyping Components	177
	A.1.4 LCD Module	179
	A.1.5 CapSense Elements	180
	A.1.6 Processor Module	
	A.1.7 Expansion Ports	183
A.2	Schematics	
	A.2.1 CY8CKIT-001 PSoC Development Board	187
	A.2.2 CY8C28 Family Processor Module	188
	A.2.3 CY8C29 Family Processor Module	
	A.2.4 CY8C38 Family Processor Module	
	A.2.5 CY8C58LP Family Processor Module	191
	A.2.6 Enabling Boost Component in PSoC 3 and PSoC 5LP Processor Mo	dules192
A.3	Bill of Materials	192
	A.3.1 CY8CKIT-001 PSoC Development Board	
	A.3.2 CY8C28 Family Processor Module	194
	A.3.3 CY8C29 Family Processor Module	
	A.3.4 CY8C38 Family Processor Module	196
	A.3.5 CY8C58LP Family Processor Module	197
Appendix	B. MiniProg3	199
B.1	MiniProg3 LEDs	
B.2	Programming in Power Cycle Mode	199
B.3	Interface Pin Assignment Table	199
B.4	Protection Circuitry	200
B.5	Level Translation	200
Appendix	C. MiniProg3 Technical Description	201
C.1	Interfaces	
0.1		
0.1	C.1.1 ISSP C.1.2 JTAG	



	C.1.3 SWD/SWV	
	C.1.4 I2C <sup>™</sup>	
	Connectors	
	C.2.1 5-Pin Connector	
	C.2.2 10-Pin Connector	
C.3	Power	

# Appendix D. PSoC Creator DWR

205

Contents





# 1.1 Kit Overview

The CY8CKIT-001 PSoC<sup>®</sup> Development Kit provides a common development platform where you can prototype and evaluate different solutions using either the PSoC 1, PSoC 3, or PSoC 5LP architectures. See table below which highlights the main difference between the three families of PSoC. This guide gives you a practical understanding of PSoC technology. The kit also includes several code examples with step-by-step instructions to enable you to easily develop PSoC solutions. This kit includes PSoC CY8C28, CY8C38, and CY8C58LP family processor modules.

Feature	PSoC 1	PSoC 3	PSoC 5
CORE	8-bit M8C 4 MIPS	8-bit 8051 33 MIPS	ARM Cortex-M3 100 DMIPS
FLASH	4 KB – 32 KB	16 KB <b>–</b> 64 KB	32 KB – 256 KB
SRAM	256B – 2 KB	2 KB – 8 KB	16 KB – 64 KB
EEPROM	-	512B – 2 KB	512B – 2 KB
POWER	Active: 2 mA Sleep: 3 uA	Active: 1.2 mA Sleep: 1 uA Hibernate: 200 nA	Active: 2 mA Sleep: 2 uA Hibernate: 300 nA
ADC	6- to 14-bit ΔΣ	12- to 20-bit ΔΣ	12- to 20-bit ΔΣ 2x 12-bit SAR (1 Msps)
DAC	6- to 8-bit	8- to 12-bit	8- to 12-bit
VOLTAGE REFERENCE	± 1.53% Vref Accuracy	± 0.1% Vref Accuracy	± 0.1% ∀ref Accuracy
DIGITAL BLOCKS	Up to 16 Digital Blocks	Up to 24 UDBs PLD-based	Up to 24 UDBs PLD-based
CONNECTIVITY	UART, I2C, SPI, USB	UART, I2C, SPI, USB, CAN, LIN, I2S	UART, I2C, SPI, USB, CAN, LIN, I2S
I/Os	Up to 64	Up to 72	Up to 72

# 1.2 Kit Contents

The CY8CKIT-001 PSoC Development Kit includes:

- PSoC development board
- PSoC CY8C28 family processor module
- PSoC CY8C38 family processor module
- PSoC CY8C58LP family processor module
- MiniProg3 programmer and debug tool
- USB cable<sup>1</sup>



- 12-V power supply adapter
- Wire pack
- Printed documentation
  - Quick start guide
  - □ Schematic and pinout of PSoC development board design
- PSoC 1 software DVD (contents are installed in \PSoC Development Kit CY8C28):
  - □ PSoC Designer<sup>™</sup> IDE
  - PSoC Programmer software
  - CY8C28 datasheets
  - Kit release notes
  - □ Software release notes
  - **D** Code example files, firmware, and documentation
- PSoC 3 and PSoC 5LP software DVD (contents are installed in the \CY8CKIT-009A folder for PSoC 3 module kit and \CY8C58LP Family Processor Module folder for PSoC 5LP module kit):
  - □ PSoC Creator<sup>™</sup> IDE
  - PSoC Programmer software
  - CY8C38 datasheet
  - CY8C58LP datasheet
  - □ Kit release notes
  - □ Software release notes
  - □ Code example files, firmware, and documentation

<sup>1.</sup> Any USB certified cable up to 2 meters in length can be used with the DVK.



# 1.3 Installation

Everything you need to use the PSoC Development Kit is included; you only need to install the software for the processor module you plan to use.

**Note** CY8CKIT-008 CY8C29 family processor module is not part of this kit; you can purchase this module from <a href="http://www.cypress.com">http://www.cypress.com</a>.

### 1.3.1 Before You Begin

All Cypress software installations require administrator privileges, but this is not required to run the installed software.

Shut down any currently running Cypress software.

Disconnect any ICE-Cube or MiniProg devices from your computer.

### 1.3.2 Prerequisites

PSoC Creator and PSoC Designer both use Microsoft .NET Framework, Adobe Acrobat Reader, and a Windows Installer. If .NET Framework and Windows Installer are not on your computer, the installation automatically installs them. If you do not have Adobe Acrobat Reader, download and install it from the Adobe website.

### 1.3.3 Installing PSoC 1 Development Software

To use the CY8C28 or CY8C29 family processor module (PSoC 1), you need:

- PSoC Designer 5.0 SP6 or higher
- PSoC Programmer 3.12.3 or later

If PSoC Designer 5.0 is currently installed, uninstall it. Click Start  $\rightarrow$  Control Panel  $\rightarrow$  Add or Remove Programs.

Insert the PSoC 1 Software DVD; using the menu, select Install Software for PSoC 1.

After installation, user guides and key documents are located in the  $\Documentation$  subdirectory of the PSoC Designer installation directory.

### 1.3.4 Installing PSoC 3 Development Software

To use the CY8C38 family processor module (PSoC 3), you need:

- PSoC Creator 1.0 Production or later
- PSoC Programmer 3.12.3 or later
- PSoC Development Kit example files

Insert the PSoC 3 or PSoC 5LP software DVD; in the menu, select **Install Software for PSoC 3**. This option installs all three required software packages. The installers for PSoC Programmer and PSoC Creator automatically start before the kit examples are installed.

For each installation, select **Typical** on the **Installation Type** page.

PSoC Creator uses the DP8051 Keil 8.16 compiler to build PSoC 3 applications. This compiler is included on the DVD; if the installer does not detect the compiler, you will be prompted to install it.

**Note** The Keil compiler is distributed with a free license. You must activate this license within 30 days of installation. When the Cypress software installation is complete, and you run PSoC Creator, activate the compiler license from **Help**  $\rightarrow$  **Register**  $\rightarrow$  **Keil**.



Important for Win7 and Vista users: Rename the \*\_tools.ini file in <Install\_Directory>:\ PSoC Creator\<version>\PSoC Creator\import\keil\pk51\<version> to "*tools.ini*" for the Keil registration to be successful.

After installing PSoC Creator and PSoC Programmer, refer to the documentation as needed:

- **PSoC Creator**  $\rightarrow$  Help  $\rightarrow$  Topics  $\rightarrow$  Getting Started
- **Programmer**  $\rightarrow$  **Documentation**  $\rightarrow$  **User Guide**

Other documents included with this release are located in the  $\Documentation$  subdirectory of the PSoC Creator installation directory. The default location is:

<Install\_Directory>:\PSoC Creator\<version>\PSoC Creator\Documentation

You can access this directory from within PSoC Creator under Help  $\rightarrow$  Documentation. Documents include (but are not limited to):

- PSoC Creator Component Author Guide (component\_author\_guide.pdf)
- Warp Verilog Reference Guide (warp\_verilog\_reference.pdf)
- Customization API Reference (customizer\_api.chm)

**Note** After the installation is complete, the kit contents are available at the following location: <Install\_Directory>:\CY8CKIT-009A\<version>

### 1.3.5 Installing PSoC 5LP Development Software

To use the CY8C58LP family processor module (PSoC 5LP), you need:

- PSoC Creator 2.1 or later
- PSoC Programmer 3.16 or later
- PSoC Development Kit example files

Insert the PSoC 3 or PSoC 5LP Software DVD; in the menu, select **Install Software for PSoC 5LP**. This option installs all three required software packages. The installers for PSoC Programmer and PSoC Creator automatically start before the kit examples are installed.

For each installation, select Typical on the Installation Type page.

PSoC Creator uses the GNU GCC 4.4.1 compiler to build PSoC 5LP applications.

After installing PSoC Creator and PSoC Programmer, refer to the documentation as needed:

- **PSoC Creator**  $\rightarrow$  Help  $\rightarrow$  Topics  $\rightarrow$  Getting Started
- $\blacksquare \quad Programmer \rightarrow Documentation \rightarrow User \ Guide$

Other documents included with this release are located in the \Documentation subdirectory of the PSoC Creator installation directory. The default location is:

<Install\_Directory>:\PSoC Creator\<version>\PSoC Creator\Documentation

You can access this directory from within PSoC Creator under Help  $\rightarrow$  Documentation. Documents include (but are not limited to):

- PSoC Creator Component Author Guide (component\_author\_guide.pdf)
- Warp Verilog Reference Guide (warp\_verilog\_reference.pdf)
- Customization API Reference (customizer\_api.chm)

**Note** After the installation is complete, the kit contents are available at the following location: <Install\_Directory>:\CY8C58LP Family Processor Module\<version>



# 1.4 **PSoC Development Board**

The CY8CKIT-001 PSoC Development Board is designed to aid hardware, firmware, and software developers in building their own systems around Cypress's PSoC devices. The flexibility to configure the power domains is one of the foremost features of this board. Input power to the board is from one of two sources:

- 12 V 1-A power supply adapter
- 9-V alkaline battery (not included)

Note Do not apply more than 15 V as input voltage. Also, do not power the board from test points.

This full-featured board incorporates three onboard linear regulators that power peripherals and PSoC processor modules at voltages between 1.7 V and 5.0 V. These regulators include a fixed 5 V 1-A linear regulator, a fixed 3.3 V 300-mA linear regulator, and a 1.5 V to 3.3 V for 3.3-V supply and 1.5 V to 5 V for 5-V supply adjustable regulator. The board also provides the ability to separate the PSoC core VDD rail into two separate rails, analog and digital. In addition, the board is able to separate the I/O VDD rails, giving the flexibility to power the I/O ports at different voltages.

The board is equipped with a 2x16 alphanumeric LCD module capable of 1.8 V to 5.0 V I/O. In addition, there is a mini-B full-speed USB interface and a female DB9 serial communications interface. Also included is a 12-pin wireless radio module interface, which can be used to develop CyFi<sup>™</sup> low-power RF or other embedded RF solutions with this kit. The board also has a prototyping area containing a small breadboard, complete with I/O port sockets nearby, multipurpose LEDs, mechanical push buttons, and a multipurpose variable resistor. In addition, three capacitive sensing elements (two buttons and a five segment slider) are included on the board to allow the evaluation of CapSense<sup>®</sup> applications.

The board has four general-purpose I/O (GPIO) expansion slots, allowing the I/O to expand to external boards.

The board is protected against reverse voltage and overvoltage on the 5-V and 3.3-V lines on the expansion slots. See Protection Circuit on page 185 for more information.

The board is designed with modularity in mind and, as a result, supports removable processor modules. This allows you to plug different PSoC processor modules into the board based upon the desired features of both 8-bit and 32-bit PSoC devices.

#### Note

- The PSoC device may get hot or damaged if many I/O pins are configured as strong drive with initial state HIGH and grounded externally using wires.
- The PSoC device may get hot or damaged if many I/O pins are configured as strong drive with initial state LOW and connected to Vcc externally using wires.

### 1.4.1 Default Switch and Jumper Settings

Jumpers on the CY8CKIT-001 PSoC development board have a default setting to operate at 3.3 V. For default configuration, each of the jumpers must be set according to these instructions.

**Note** All CY8C28 and CY8C29 family processor module code examples are configured for 5 V. Configure the board to 5 V, before creating the code examples.



SW3 - VDD Select. Default Position: 3.3 V (down position)



J8 - 5 V Source. Default Position: VREG (upper two pins)



J7, J6 - VDD Digital, VDD Analog. Default Position: VDD (upper two pins, both headers)



J12 - LCD Power. Default Position: ON (lower two pins)



J2-J5 - VDDIO Power Select. Default Position: VDD (upper left two pins)



J10 - RS-232 Power (Serial Communications). Default Position: Installed





### J14 - Radio Power. Default Position: Installed



### J11 - Variable Resistor Power. Default Position: Installed



### 1.4.1.1 AC/DC Adaptor Specifications

Use adaptors with the following specifications:

- Input voltage: 100 to 240 VAC, 50 Hz to 60 Hz, 1 A
- Output voltage: 12 VDC, 1 A
- Power output: 12 W
- Polarization: Positive center
- Certification: CE certified

Some recommended part numbers include EPSA120100U-P5P-EJ (CUI Inc.) and LTE12W-S2 (Li Tone Electronics Co. Ltd).

### 1.4.1.2 Battery Specifications

Use batteries with the following specifications:

- Battery type: 9 V
- Output voltage: 9 VDC
- Type: Non-rechargeable alkaline consumer batteries
- RoHS status: RoHS compliant
- Lead free status: Pb-free

Some recommended part numbers include 6LR61XWA/1SB (Panasonic), MN1604 (Duracell), and 6LR61 (Energizer).

# 1.5 Kit Revision

To know the kit revision, look for the white sticker on the bottom left on the back of the kit box. If the revision reads CY8CKIT-001C Rev \*\*, then congratulations, you own the latest version.

To upgrade CY8CKIT-001B to CY8CKIT-001C, the main DVK and kit DVD must be updated. Purchase the latest development board and download the latest DVD ISO image at http:// www.cypress.com/go/CY8CKIT-001.

To upgrade CY8CKIT-001A to CY8CKIT-001C, besides the upgrades stated above, you need to update the PSoC 3 processor module and kit DVD. Purchase the latest processor module at <a href="http://www.cypress.com/go/CY8CKIT-009">http://www.cypress.com/go/CY8CKIT-009</a> and download the latest DVD ISO image at <a href="http://www.cypress.com/go/CY8CKIT-001">http://www.cypress.com/go/CY8CKIT-009</a> and download the latest DVD ISO image at <a href="http://www.cypress.com/go/CY8CKIT-001">http://www.cypress.com/go/CY8CKIT-009</a> and download the latest DVD ISO image at <a href="http://www.cypress.com/go/CY8CKIT-001">http://www.cypress.com/go/CY8CKIT-009</a> and download the latest DVD ISO image at <a href="http://www.cypress.com/go/CY8CKIT-001">http://www.cypress.com/go/CY8CKIT-001</a>.

To upgrade CY8CKIT-001 to CY8CKIT-001C, besides the upgrades stated above, you need to purchase the latest PSoC 5LP process module at http://www.cypress.com/go/CY8CKIT-010.



# 1.6 Additional Resources

Visit http://www.cypress.com/go/training for additional learning resources in the form of datasheets, technical reference manual, and application notes.

## 1.6.1 Beginner Resources

AN54181 - PSoC 3 - Getting Started with a PSoC 3 Design Project

**PSoC Designer Training** 

**PSoC Designer FAQ** 

**PSoC Creator Training** 

## 1.6.2 Engineers Looking for More

AN54460 - PSoC 3 and PSoC 5 Interrupts

AN52705 - PSoC 3 and PSoC 5 - Getting Started with DMA

AN52701 - PSoC 3 - How to Enable CAN Bus Communication

AN54439 - PSoC 3 and PSoC 5 External Crystal Oscillators

AN52927 - PSoC 3: Segment LCD Direct Drive

Cypress continually strives to provide the best support. Click here to view a growing list of application notes for PSoC 3 and PSoC 5LP.

### 1.6.3 Learning from Peers

Cypress Developer Community Forums

### 1.6.4 More Code Examples

PSoC Creator provides a host of example projects that makes the code development very fast and easy. To access these example projects, click on the **Find Example Project...** under **Example and Kits** section in **Start Page** of PSoC Creator or by navigating to **File**  $\rightarrow$  **Open-Example Project...** 



Start Page	
PSoC <sup>®</sup> Creator <sup>™</sup>	
Recent Projects	^
Create New Project Open Existing Project	
Getting Started	
PSoC Creator Start Page Quick Start Guide Intro to PSoC Intro to PSoC Creator PSoC Creator Training Help Tutorials Getting Started With PSoC 3 Getting Started With PSoC 5	
Examples and Kits	
Find Example Project	

Eile	Edit	⊻iew	<u>D</u> ebug	Project	Buil	d <u>T</u> ools	<u>W</u> indow	He
	<u>N</u> ew			•		X	50	_ :
	<u>O</u> pen			•	â	<u>P</u> roject/W	/orkspace	
	Add			×.	2	<u>F</u> ile	Ctrl+O	
	⊆lose		Ct	rl+F4		Example (	Project	

The Find Example project has various filters that help you locate the most relevant project you are looking for.

PSoC Creator provides several Starter Designs. These designs highlight features that are unique to PSoC devices. They allow you to create a design with various components and code is also provided, instead of creating a new empty design. To use a starter design for your project, navigate to **File**  $\rightarrow$  **New**  $\rightarrow$  **Project** and select the design required.



lew Project		? 🛛
Design	Other	4 Þ
Empty Tem	plates	^
Empl	y PSoC 3 Design Empty PSoC 5 Design Empty PSoC 5LP De	esign
PSoC 3 Sta	arter Designs	
ADC.	_DMA_VDAC DelSig_16Channel DelSig_12CM	
DelS	ig_I2CS DelSig_SPIM DelSig_SPIM DelSig_SPIM	
P3 HWI	Fan Control with Alert	
_	arter Designs	
	_DMA_VDAC DelSig_12CM DelSig_12CS	
-	3, 8 bit, design project.	~
Name:	Design02	
Location	C:\Documents and Settings\sash\Desktop\a	
Advanced		
		OK Cancel

**Note:** The example projects and starter designs are designed for CY8CKIT-001 PSoC Development Kit. However, these projects can be converted for use with CY8CKIT-030 PSoC 3 Development Kit or CY8CKIT-050 PSoC 5 Development Kit by following the procedure in the knowledge base article Migrating project from CY8CKIT-001 to CY8CKIT-030 or CY8CKIT-050.

Apart from the example projects and starter designs that are available within PSoC Creator, Cypress continuously strives to provide the best support. Click here to view a growing list of application notes for PSoC 3 and PSoC 5.

# 1.7 Document Conventions

These conventions are used throughout this guide.

Convention	Usage
Courier New Size 12	Displays file locations and source code: C:\cd\icc\.
Italics	Displays file names and reference documentation: sourcefile.hex
[bracketed, bold]	Displays keyboard commands in procedures: [Enter] or [Ctrl] [C]

Table 1-1. Documentation Conventions



Convention	Usage
$\textbf{Bold} \rightarrow \textbf{With} \rightarrow \textbf{Arrows}$	Represents menu paths, user entered text: <b>File</b> $\rightarrow$ <b>New Project</b> $\rightarrow$ <b>Clone</b>
Bold	Displays commands and selections, and icon names in procedures: Click the <b>Debugger</b> icon, and then click <b>Next</b> .
Note	Displays functionality unique to PSoC Designer, PSoC Creator, or the PSoC device.
WARNING:	Displays cautions that are important to the subject.

Table 1-1. Documentation Conventions (continued)

# 1.8 Document Revision History

	Document Title: CY8CKIT-001 PSoC® Development Kit Guide Document Number: 001-48651		
Revision	Issue Date	Origin of Change	Description of Change
**	6/23/09	AESA	New Guide
*A	7/22/09	AESA	CDT based updates
*В	11/19/09	AESA	CDT based updates
*C	05/21/10	AESA	Updated with PSoC 5LP.
*D	01/05/11	RKAD	Updated images. Updated PSoC Creator and PSoC Programmer versions
*E	02/10/11	RKAD	Updated images. Added Kit Revision section
*F	12/16/11	RKAD	Content updates throughout the document.
*G	12/30/11	RKAD	Updated installation directory path. Added Figure 2-4 and Figure 2-10. Added note on Keil compilers in section 1.3.4.
*H	01/13/12	RKAD	Added note on USB cable in section 1.2 - Kit Contents. Appended to note in section A.1.4 - LCD Module
*	01/18/12	RKAD	Minor ECN to include attachments in pdf. No content updates made.
*J	05/03/12	SASH	Added the Additional Resources section
*K	07/04/12	SASH	Added Appendix D for PSoC Creator DWR. Updated images for PSoC Creator version 2.1.
*L	11/08/2012	SASH	Added A.1.7.4 Protection Circuit. Updated 1.5 Kit Revision. Updated figures in Chapter 3.

Introduction



# 2. Loading My First PSoC Project



The CY8CKIT-001 PSoC Development Kit supports projects across the PSoC 1, PSoC 3, and PSoC 5 architectures. This section walks you through the high-level design process for opening, building, programming, and running your first PSoC project using this kit.

Before beginning, follow each of these steps to make certain that your software and hardware environments are properly configured and ready for these projects:

- Install PSoC Designer using the steps listed in Installing PSoC 1 Development Software on page 9.
- 2. Install PSoC Creator using the steps listed in Installing PSoC 3 Development Software on page 9.
- 3. Connect the MiniProg3 into your PC using the supplied USB cable. When you connect the MiniProg3, Microsoft Windows<sup>®</sup> may indicate that it has found new hardware. All required drivers are installed as part of the PSoC Programmer installation process; however, if Windows opens the driver installation dialog boxes, accept the defaults and allow Windows to automatically find the appropriate driver.
- 4. Close any open PSoC Creator or PSoC Designer applications and projects.
- 5. Configure the PSoC development board (jumper settings and switches) in its default configuration, as described in Default Switch and Jumper Settings on page 11.
- Use the PSoC CY8C28 family processor module or PSoC CY8C29 family processor module for the PSoC 1 version of your first PSoC project (My First PSoC 1 (CY8C28) Project on page 20 or My First PSoC 1 (CY8C29) Project on page 25).
- Use the PSoC CY8C38 family processor module for the PSoC 3 version of your first PSoC project (My First PSoC 3 (CY8C38) Project on page 30).
- Use the PSoC CY8C58LP family processor module for the PSoC 5LP version of your first PSoC project (My First PSoC 5LP (CY8C58LP) Project on page 34).
- 9. For a PSoC 1 project, use the ISSP header on the PSoC CY8C28 family processor module or PSoC CY8C29 family processor module and connect the MiniProg3 ISSP port.
- 10. For a PSoC 3 or PSoC 5LP project, use the JTAG ribbon cable. Connect the ribbon cable to the MiniProg3 and the CY8C38 family processor module or CY8C58LP family processor module into the header labeled PROG on the processor module.

**Note** The MiniProg3 should not be "hot plugged" into processor modules that are attached to the PSoC development board. In other words, do not plug the ribbon cable of the MiniProg3 into the processor module while code is actively running on the module. Doing so may cause the PSoC device to unintentionally reset. Power down the PSoC development board and module by unplugging the power supply from the development board before attaching the MiniProg3 device to the module board. When the ribbon cable is attached to the module board, power the system by plugging in the power supply to the PSoC development board. This will avoid any undesirable PSoC device resets.

11. Power the PSoC development board using the 12-V AC power supply adapter.



# 2.1 My First PSoC 1 (CY8C28) Project

This is a simple PSoC 1 project using a pulse width modulator (PWM) peripheral inside PSoC, and software to control the blinking rates of two different LED outputs. For this project, be sure you have the PSoC CY8C28 family processor module inserted into the PSoC development board and the appropriate software installed. This section walks you through the steps to open, build, and program a project.

# 2.1.1 Loading My First PSoC 1 Project

- 1. Open PSoC Designer.
- 2. In the Start Page, navigate to File  $\rightarrow$  Open Project/Workspace
- 3. Navigate to the project directory: <Install\_Directory>:\PSoC Development Kit CY8C28\<version>\Firmware\CY8C28.
- 4. Open the folder Ex1\_LED\_with\_PWM.
- 5. Double-click **Ex1\_LED\_with\_PWM.app**.
- 6. The project opens in the Chip Editor view. All project files are in the Workspace Explorer.

Figure 2-1. Chip Editor View





# 2.1.2 Building My First PSoC 1 Project

1. Select Build  $\rightarrow$  Generate/Build 'Ex1\_LED\_with\_PWM' Project.

### Figure 2-2. Build Project

Eile Edit Yiew P	roject Interconnect	Build	d <u>D</u> ebug P <u>r</u> ogram <u>T</u> ools <u>W</u> indow <u>H</u> elp
1 🗋 🖬 🖉 .	S II S A A	<b></b>	Generate/Build 'Ex1_LED_with_PWM' Project
Global Resources - ex1_l			Generate/Build <u>A</u> ll Projects S
Power Setting [Vcc /	Sy 5.0V / 24MHz	~	Generate Configuration Files/Build 'Ex1_LED_with_PWM' Project
CPU_Clock	SysClk/2	-	aginerate coningeration mesybelia exi_teb_man_i with hojett
32K_Select	Internal		Generate Configuration Files for 'Ex1_LED_with_PWM' Project
PLL_Mode	Disable		Generate Configuration Files for All Projects
Sleep Timer Period	1.95ms		denerate configuration his for him fojects
VC1= SysClk/N	16	配	Compile
VC2= VC1/N	16		Build 'Ex1_LED_with_PWM' Project
VC3 Source	VC2		Faid ext_res_mail. Hit tober

2. PSoC Designer builds the project and displays comments in the **Output** window. When you see the message that the project is built with 0 errors and 0 warnings, you are ready to program the device.

Figure 2-3. Output Window

Show output from:	Build	~		
LMM info: area 'ex1 ROM 4% full. 728 b RAM 0% full. 5 byte idata dump at output	led with pwm RAM	M.idata	M page () ;).	



# 2.1.3 Programming My First PSoC 1 Project

Figure 2-4. Connect MiniProg3 to J5 on CY8C28 Family Processor Module



### Programming using PSoC Designer

- 1. Open **Program Part** from within PSoC Designer by selecting **Program**  $\rightarrow$  **Program Part**.
- 2. In the Program Part window, ensure that MiniProg3 is selected in the Port Selection box.
- 3. In the Program Part window, set Acquire Mode to Reset.
- 4. In the **Program Part** window, set **Verification** to **On**. This ensures that downloaded checksum matches the actual checksum.
- 5. In the **Program Part** window, click the program arrow to program the device.

### Programming using PSoC Programmer

- In PSoC Programmer, set AutoDetection to On to enable the software to automatically detect and configure for the target device family and device. If PSoC Programmer is properly configured, AutoDetection reports a device family of 28xxx.
   Note Make sure ISSP protocol is selected.
- 7. Wait until programming is completed, to continue.

**Note** For debugging purposes, the CY8C28 family processor module is designed to accommodate the use of the CY3215-DK In-Circuit Emulator (ICE-Cube). When using the ICE-Cube debugger, make certain that PSoC Designer is configured so that the ICE-Cube does not provide power to the processor module. Within the PSoC Designer application, select **Project**  $\rightarrow$  **Settings** and select **Debugger** from the tree. Make sure that **External only** is selected under the **Pod Power Source** section and select **Execute Program** from the **Debug** menu to start debugging.

Connect the processor module to the CY3215-DK ICE-Cube, as shown in Figure 2-5.





Figure 2-5. ICE-Cube Connected to CY8C28 (PSoC 1) Processor Module

# 2.1.4 Running My First PSoC 1 Project

- 1. Connect P1[6] to LED1 and P1[7] to LED2. Verify that LED1 and LED2 are blinking based on the project's use of the PWM and software. Now that the PSoC 1 device is programmed, reset the PSoC development board by pressing and releasing the reset switch (SW4).
- 2. LED1 blinks approximately once every second and LED2 blinks about three times a second.





Figure 2-6. Connect P1[6] to LED1 and P1[7] to LED2

3. For more details regarding this project, see the detailed project instructions in My First PSoC 1 (CY8C28) Project on page 39.



# 2.2 My First PSoC 1 (CY8C29) Project

This is a simple PSoC 1 project using a PWM peripheral inside PSoC, and software to control the blinking rates two different LED outputs. For this project, be sure you have the PSoC CY8C29 family processor module inserted into the PSoC development board and the appropriate software installed. This section walks you through the steps to open, build, and program a project.

# 2.2.1 Loading My First PSoC 1 Project

- 1. Open PSoC Designer.
- 2. In the Start Page, navigate to File  $\rightarrow$  Open Project/Workspace.
- 3. Navigate to the project directory: C:\Cypress\CY8CKIT-001\CY8C29 Projects.
- 4. Open the folder Example\_My\_First\_PSoC\_Project.
- 5. Double-click Example\_My\_First\_PSoC\_Project.app.
- 6. The project opens in the Chip Editor view. All project files are in the Workspace Explorer.

### Figure 2-7. Chip Editor View





#### 2.2.2 Building My First PSoC 1 Project

### 1. Select Build → Generate/Build 'Example\_My\_First\_PSoC\_Project' Project.

Figure 2-8. Build	Project			
Example_My_First_P5oC_Project - P5oC Designer 5.0				
<u>File E</u> dit <u>V</u> iew <u>P</u> rojec	t <u>I</u> nterconnect	Build	d <u>D</u> ebug P <u>r</u> ogram <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🎦 🚅 🔙 🥥 📜 😭	1	₩	Generate/Build 'Example_My_First_PSoC_Project' Project	F6
Global Resources - example_my_first_psoc_proje			Generate/Build <u>All</u> Projects	Shift+F6
Power Setting [Vcc / 5.0V / 24MHz		æ	Generate Configuration Files/Buildle_My_First_PSoC_Proje	ct' Ribojæft6
CPU_Clock SysClk/2		_		
32K_Select Internal			Generate Configuration Files for 'le_My_First_PSoC_Project'	Project
PLL_Mode Disable			Generate Configuration Files for All Projects	
Sleep_Timer 512_Hz				
VC1= SysClk/N 16		÷.	Compile	Ctrl+F7
VC2=VC1/N 16			Build 'Example My First PSoC Project' Project	F7
VC3 Source VC2				
VC3 Divider 256		<u>R</u> ebuild 'Example_My_First_PSoC_Project' Project		
SysClk Source Inte	rnal		Clean 'Example_My_First_PSoC_Project' Project	
SysClk*2 Disable Yes	:			
Analog Power SC	Off/Ref Low		Show Last Build Report for 'Example_My_First_PSoC_Project' I	Project
Ref Mux (Vd	d/2)+/-(Vdd/2)		Proce and Processing	

2. PSoC Designer builds the project and displays comments in the Output window. When you see the message that the project is built with 0 errors and 0 warnings, you are ready to program the device.







# 2.2.3 Programming My First PSoC 1 Project

Figure 2-10. Connect MiniProg3 to J5 on CY8C29 Family Processor Module



- 1. Open PSoC Programmer from within PSoC Designer by selecting **Program** → **PSoC Programmer**.
- 2. In PSoC Programmer, make sure that **MiniProg3** is selected in the **Port Selection** box.
- 3. In PSoC Programmer, set Programming Mode to Reset.
- 4. In PSoC Programmer, set **Verification** to **On** so that the software verifies that the downloaded program's checksum matches the actual checksum of the flash memory after programming. This is a precautionary check to verify that there is no data corruption during programming.
- 5. In PSoC Programmer, set **AutoDetection** to **On** to enable the software to automatically detect and configure for the target device family and device. If PSoC Programmer is properly configured, AutoDetection reports a device family of 29x66 and device of CY8C29466. **Note** Make sure ISSP protocol is selected.
- 6. With these settings configured, click **Program** to program your PSoC 1 device.
- 7. Wait until programming is complete before continuing.

**Note** For debugging purposes, the CY8C29 family processor module is designed to accommodate the use of the CY3215-DK In-Circuit Emulator (ICE-Cube). When using the ICE-Cube debugger, make certain that PSoC Designer is configured so that the ICE-Cube does not provide power to the processor module. Within the PSoC Designer application, select **Project**  $\rightarrow$  **Settings** and select **Debugger** from the tree. Make sure that **External only** is selected under the **Pod Power Source** section and select **Execute Program** from the **Debug** menu to start debugging.

Connect the processor module to the CY3215-DK ICE-Cube, as shown in Figure 2-11.

Connect USB cable between ICE-Cube and PC. Also Connect 12 V power supply to ICE-Cube separately.





## Figure 2-11. ICE-Cube Connected to CY8C29 (PSoC 1) Processor Module

- Running My First PSoC 1 Project 2.2.4
  - 1. Connect P0[7] to LED1 and P1[7] to LED2. Verify that LED1 and LED2 are blinking based on the project's use of the PWM and software. Now that the PSoC 1 device is programmed, reset the PSoC development board by pressing and releasing the reset switch (SW4).
  - 2. LED1 blinks approximately once every second and LED2 blinks about three times a second.





Figure 2-12. Connect P0[7] to LED1 and P1[7] to LED2

3. For more details regarding this project, see the detailed project instructions in My First PSoC 1 (CY8C29) Project on page 75.



# 2.3 My First PSoC 3 (CY8C38) Project

This is a PSoC 3 project using a PWM peripheral programmed from inside the PSoC 3 device to control the blinking rates of two different LED outputs. For this project, insert the PSoC CY8C38 family processor module in the PSoC development board and install the appropriate software. This section shows you the steps to open, build, and program a project.

## 2.3.1 Loading My First PSoC 3 Project

- 1. Open PSoC Creator.
- 2. In the Start Page, under Examples and Kits expand Kits.
- 3. Under Kits, expand CY8CKIT-009A 2.1.
- 4. Click Ex1\_LED\_with\_PWM.cywrk to open the project.

Figure 2-13. Kits List

Recent Projects	
Ex1_LED_with_PWM.cywrk     Ex1_LED_with_PWM.cywrk     Ex2_ADC_to_LCD.cywrk     Ex3_ADC_to_LCD.cywrk     Ex3_ADC_to_UART_with_DAC.cywrk     Ex5_CapSense.cywrk	
Create New Project Open Existing Project	
Getting Started	
PSoC Creator Start Page Quick Start Guide Intro to PSoC Intro to PSoC Creator PSoC Creator Training Help Tutorials Getting Started With PSoC 3 Getting Started With PSoC 5	Y
Examples and Kits	
Find Example Project → Kits Ø ← CY8CKIT-009A 2.1 Ø Ex1_LED_with_PWM.cywrk Ø Ex2_ADC_to_LCD.cywrk Ø Ex3_ADC_to_UART_with_DAC.cywrk Ø Ex4_USB_HID.cywrk Ø Ex5_CapSense.cywrk	
Product Information	
PSoC Creator PSoC Programmer PSoC 3 PSoC 5	



- 5. Select the directory to store the project.
- 6. After the project opens, you can see the project files in Workspace Explorer (see Figure 2-14).

Figure 2-14. Workspace Explorer

Workspace Explorer 🗸 🗸 🗸	×
ā. 🥷	
🔯 Workspace 'Ex1_LED_with_PWM' (1 Projects)	
E Project 'Ex1_LED_with_PWM' [CY8C3866AXI-040]	S
- 📓 TopDesign.cysch	Source
- 🔐 Ex1_LED_with_PWM.cydwr	8
😑 🧰 Header Files	0
h device.h	iomp
🖻 🧰 Source Files	Components
main.c	onts
	Res

- 2.3.2 Building My First PSoC 3 Project
  - 1. Select **Build**  $\rightarrow$  **Build Ex1\_LED\_with\_PWM**.

Figure 2-15. Build Window



2. PSoC Creator builds the project and displays the comments in the **Output** window. When you see the message "Build Succeeded", you are ready to program the device.



Figure 2-16. Output Window



# 2.3.3 Programming My First PSoC 3 Project

Figure 2-17. Connect MiniProg3 to J5 on CY8C38 Family Processor Module





1. If this is your first time running PSoC Creator, follow these steps to configure the MiniProg3 device for these PSoC development kit projects. If these configurations are set, skip to the next step and begin programming.

**Note** VTARG of the MiniProg3 is wired exclusively to VDDIO1 of the chip on the PSoC CY8C38 family processor module. Because of this, you cannot perform power cycle mode programming.

- □ From the Tools menu in PSoC Creator, click Options. The Options window opens.
- $\hfill \label{eq:constraint}$  In the Options window, select Program/Debug  $\rightarrow$  Port Configuration  $\rightarrow$  MiniProg3 from the list.
  - Set Power to 3.3 V
  - Set Active Protocol to SWD
  - Set Connector to 10 Pin
  - Set Acquire Mode to Reset
  - Set Clock Speed to 3.2 MHz
  - Click OK.
- From the Debug menu, select Select Debug Target. The Select Debug Target dialog box opens.
- □ Expand the tree under MiniProg3 and click Port Acquire.
- □ Select the appropriate device and click **Connect**.
- □ Click **Close**.
- 2. In PSoC Creator, from the **Debug** menu, click **Program**.
- 3. The PSoC Creator status bar indicates that the device is programming.
- 4. Wait until programming is complete before continuing.

## 2.3.4 Running My First PSoC 3 Project

- 1. Unplug the development board, switch SW3 to 3.3 V and then reapply power to the board.
- Connect P1[6] to LED1 and P1[7] to LED2. Verify that LED1 and LED2 are blinking based on the project's use of the PWMs.
- 3. LED1 blinks approximately once every second and LED2 blinks about three times a second.

Figure 2-18. Connect P1[6] to LED1 and P1[7] to LED2



4. For more details regarding this project, review the detailed project instructions in My First PSoC 3 / PSoC 5LP Project on page 106.



# 2.4 My First PSoC 5LP (CY8C58LP) Project

This project uses a PWM peripheral programmed from inside PSoC 5LP to control the blinking rates of two different LED outputs. For this project, insert the PSoC CY8C58LP family processor module in the PSoC development board and install the appropriate software. This section shows the steps to open, build, and program a project.

- 2.4.1 Loading my First PSoC 5LP Project
  - 1. Open PSoC Creator.
  - 2. In the Start Page, under Examples and Kits expand Kits.
  - 3. Under Kits, expand CY8CKIT-010LP 2.1.
  - 4. Click Ex1\_LED\_with\_PWM.cywrk to open the project.

Figure 2-19. Kits List

Examples and Kits
Find Example Project
🖃 Kits 🧭
E CY8CKIT-010LP 2.1
Ex1_LED_with_PWM.cywrk
Ex2_ADC_to_LCD.cywrk
Ex3_ADC_to_UART_with_DAC.cywrk
Ex4_USB_HID.cywrk
Ex5_CapSense.cywrk
Ex6_SAR_to_UART_with_DAC.cywrk
PSoC MFi Expansion Board Kit 2.0
PSoC3 MFi Example Project CY8CKIT-001.cywrk

- 5. Select the directory to store the project.
- 6. After the project opens, you can see the project files in Workspace Explorer.

Figure 2-20. Workspace Explorer

Workspace Explorer (1 project) 🗸 🗸	×
· 🔒 🕿	_
Workspace 'Ex1_LED_with_PWM' (1 Projects)	$\overline{}$
🗄 🔁 Project 'Ex1_LED_with_PWM' [CY8C5868AXI-LP035]	S)
	Source
Ex1_LED_with_PWM.cydwr	8
🗄 🗀 Header Files	0
h device.h	Components
🖻 🗀 Source Files	ne l
main.c	ints
	5
	)ata
	Datasheets
	ŝ
	R
	Results
	S.



# 2.4.2 Building My First PSoC 5LP Project

1. Select Build  $\rightarrow$  Build Ex1\_LED\_with\_PWM.

### Figure 2-21. Build Window



2. PSoC Creator builds the project and displays the comments in the Output window. When you see the message "Build Succeeded", you are ready to program the device.

Figure 2-22. Output Window

Output	- u :
5how output from: All	<ul> <li></li> <li></li> </ul>
arm-none-eabi-gcc.exe arm-none-eabi-gcc.exe arm-none-eabi-gcc.exe arm-none-eabi-gcc.exe arm-none-eabi-ar.exe: arm-none-eabi-ar.exe: arm-none-eabi-gcc.exe arm-none-eabi-gcc.exe arm-none-eabi-objcopy cyhextool -o.\ARM_GCC Flash used: 3992 of 20 SRAM used: 296 of 6553	



# 2.4.3 Programming My First PSoC 5LP Project

Figure 2-23. Connect MiniProg3 to J5 on CY8C58LP Family Processor Module



1. If this is your first time running PSoC Creator, follow these steps to configure the MiniProg3 device for these PSoC development kit projects. If these configurations are set, skip to the next step and begin programming.

**Note** VTARG of the MiniProg3 is wired exclusively to VDDIO1 of the chip on the PSoC CY8C58LP family processor module. Because of this, you cannot perform power cycle mode programming.

From the **Tools** menu in PSoC Creator, click **Options**.

In the Options window, select  $\textbf{Program/Debug} \rightarrow \textbf{Port Configuration} \rightarrow \textbf{MiniProg3}$  from the list.

- Set Power to 3.3 V
- Set Active Protocol to SWD
- Set Connector to 10 Pin
- Set Acquire Mode to Reset
- Set Clock Speed to 3.2 MHz
- Click OK

From the **Debug** menu, select **Select Debug Target**. Expand the tree under **MiniProg3** and click **Port Acquire**.

Select the appropriate device and click **Connect**.

Click Close.

- 2. In PSoC Creator, from the **Debug** menu, click **Program**.
- 3. The PSoC Creator status bar indicates that the device is programming.


4. Wait until programming is complete before continuing.

## 2.4.4 Running My First PSoC 5LP Project

- 1. Unplug the development board, switch SW3 to 3.3 V and then reapply power to the board.
- 2. Connect P1[6] to LED1 and P1[7] to LED2. Verify that LED1 and LED2 are blinking based on the project's use of the PWMs.
- 3. LED1 blinks approximately once every second and LED 2 blinks about three times a second.

Figure 2-24. Connect P1[6] to LED1 and P1[7] to LED2



Loading My First PSoC Project



# 3. Sample Projects



This chapter shows you how to create the sample projects included with this kit.

Read these precautions before you create code examples:

All CY8C28 and CY8C29 family processor module code examples are configured for 5 V. All CY8C38 and CY8C58LP family processor module code examples are configured for 3.3 V. Close any open project in PSoC Creator before loading or creating a code example.

When working with code examples, use the 12-V power supply adapter.

Remove power before changing board jumpers for each code example. Reapply power after you place jumpers on the breadboard.

When you complete each project make certain to save the project.

## 3.1 CY8C28 Family Processor Module Code Examples

3.1.1 My First PSoC 1 (CY8C28) Project

#### 3.1.1.1 Creating My First PSoC 1 (CY8C28) Project

- 1. Open PSoC Designer.
- 2. To create a new project, click **File**  $\rightarrow$  **New Project**. The **New Project** window opens.
- 3. In the New Project window, select Chip-Level Project. Name the project Ex1\_LED\_with\_PWM.
- 4. In the Location field, click Browse and navigate to the appropriate directory.

Figure 3-1. New Project Window

New Project		? 🛛
Project types:		
Creates an empty p	roject, that supports User Module selection and placement.	
<u>N</u> ame:	Ex1_LED_with_PWM	· · · · · · · · · · · · · · · · · · ·
Location:	C.\	<u>B</u> rowse
Workspace na <u>m</u> e:	Ex1_LED_with_PWM Create directory for workspace	
		<u>C</u> ancel

5. Click OK. The Select Project Type window opens.



Figure 3-2. Select Project Type Window

Path:	cject:	Browse
	O Use the same target device O Select target device	Clear Path
Select T	arget Device	
Device	CYSC28645-24LTXI View Cat	alog
	Generate 'Main' file using:	
	⊙ C	
	O Assembler	

- 6. Under Select Target Device, click View Catalog.
- 7. The **Device Catalog** window opens. Click on the **PSoC** tab and scroll down to the **CY8C28XXX** section.
- 8. In this section, click the CY8C28645-24LTXI device; click Select.

Figure 3-3. Device Catalog Window

	Part Number	Analog Blocks	Digital Blocks	Flash	RAM	IO Count	Supply Voltage	SMP	Tem
	4*			<b>▲</b> ▼	A.		47		
	Click here to Remove All Filters	al 💌	al 🕑	all 🗸	al 🗸	all 💙	all 💌	al 🗸	all
	CY8C27443-245XI	12	8	16K	256	24	3.0 th 5.25	YFS	inr 🔥
	CY8C27543-24AX	12	8	16K	256	40	3.0 to 5.25	YES	Inc
	CV8C27643-24PVXI	12	8	16K	256	44	3.0 to 5.25	YES	Inc
	CY8C27643-24LFXI	12	8	16K	256	44	3.0 to 5.25	YES	inc
	E 🔁 CY8C28XXX (Datast	eet) (Heip Me C	hoose a Part)						
ľ	CY8C28403-24PVXI	0	12	16K	1024	24	3.0 to 5.25	Yes	Inc
	CY8C28413-24PVXI	0 + ^4	12	16K	1024	24	3.0 to 5.25	Yes	inc
ľ	CY8C28513 24AXI	0 + *1	12	16K	1024	10	3.0 to 5.25	Yes	inc
	CY8C28623-24LTXI	6	12	16K	1024	44	3.0 to 5.25	Yes	Inc
	CY8C28433-24PVXI	6 + ^4	12	16K	1024	24	3.0 to 5.25	Yes	Inc
ľ	CY8C28533-244X	6 + *4	12	16K	1024	40	3.0 to 5.25	Yes	Inc
	CV8C28243-24PVXI	12	12	16K	1024	16	3.0 to 5.25	Yes	inc
	CY8C28643-24LTXI	12	12	16K	1024	44	3.0 to 5.25	Yes	Inc
	CY8C28445-24PVXI	12 + 14	12	16K	1024	24	3.0 to 5.25	Yes	inc
	CY8C28545-24AX	12 + *4	12	16K	1024	40	3.0 to 5.25	Yes	Inc
ſ	CY8C28645-24LTXI	12 + 14	12	16K	1024	44	3.0 to 5.25	Yes	Inc
	CY8C28452-24PVXI	12 + *4	8	16K	1024	24	3.0 to 5.25	Yes	Inc
	Extended CY8C28X	X (Datasheet) (	Help Me Choos	e a Part)					
	CY8C28403-12PVXQ	0	12	16K	1024	24	4.75 to 5.25	Yes	Ext
	CY8C28413-12PVXQ	0 + *4	12	16K	1024	24	4.75 to 5.25	Yes	Ext
	CY8C28513-12AXQ	0 + *4	12	16K	1024	40	4.75 to 5.25	Yes	Ext

Under Generate 'Main' File Using:, select C; then, click OK.
 By default, the project opens in Chip view.

#### Figure 3-4. Default View.

	led_with_pwm + 🕈 🗙	Start P	age / ex1_led_with_pwm [Chip]
Power Setting [Vcc /		(41)	7 00 07 0E 0 7 900 07 0
CPU_Clock	SysClk/0	200	
32K_Select	Internal		
PLL_Mode	Disable		
Sleep Timer Period	1.95ms	and the second	
VE1 = SysClk/N	1		
VC2= VC1/N	1		
VE3 Source	SysClk/1	Per.U.	
VC3 Divider	1		
SysCik Source	Internal	20.0	нтр
SysClk*2 Disable	No		
Analog Power	SC On/Ref Low		4
Ref Mux	(Vdd/2)+/-BandGap		FORCH FORCH FREEZ FREEZES
AGindBypass	Disable 😽 😽	24,01 (44,0)	
	vinternal clocks (V1, V2, V3, and CPU clo		
operties	v + x		
			Pack Pack Proz. Pack

11. In the User Modules window, expand the PWMs folder.

Figure 3-5. User Modules Window



12. In this folder, right-click on **PWM8** and select **Place**. The user module (UM) is placed in the first available digital block.



Figure 3-6. Place User Module PWM8



13.Click the placed PWM8\_1 UM; the **Properties** window opens on the left side of the screen. Configure the PWM with the settings shown in the following figure. If the **Properties** window does not appear, click **View** → **Properties Window**.

Figure 3-7. Properties Window

Name	PWM8_1			
User Module	PWM8			
Version	2.5			
Clock	VC3			
Enable	High			
CompareOut	Row_0_Output_3			
TerminalCountOut	None			
Period	100			
PulseWidth	50			
CompareType	Less Than Or Equal			
InterruptType	Terminal Count			
ClockSync	Sync to SysClk			
InvertEnable	Normal			

14.Next, route the PWM **CompareOut** signal to P1[7]. The first step is to configure the lookup table (LUT) on **Row\_0\_Output3**.



Figure 3-8. Route PWM8 CompareOut Signal to P1[7]



15. Double-click the LUT, the Digital Interconnect window opens.

16. In this window, enable Row\_0\_Output\_3\_Drive\_3 to connect to GlobalOutOdd\_7.

Figure 3-9. Digital Interconnect Window



17.Click Close.

18. Click on GlobalOutOdd\_7. In the window that appears, configure Pin for Port\_1\_7.



Figure 3-10. Configure Pin for Port_1_7	
	Port_1_4
	Port_1_5
	Port_1_6
BlopaiDutOdd_7	Port_1_7
Pin Port_1_7  Interconnect	Port_2_0
	Port_2_1
	Port_2_2
OK Cancel	Port_2_3
	2.2.1

- 19.Click **OK** to continue.
- 20. In the User Modules window, expand the Misc Digital folder. In this folder, right-click LED and select Place; this adds the UM to the project. This UM does not use digital or analog blocks. It appears in Workspace Explorer → Ex1\_LED\_with\_PWM[CY8C28] → Ex1\_LED\_with\_PWM[Chip] → Loadable Configurations → Ex1\_LED\_with\_PWM 2 User Modules.

Figure 3-11. Workspace Explorer



21. Click the LED\_1 UM and navigate to the Properties window. Configure the LED for P1[6].

Figure 3-12. Properties Window

operties - LED_1		- 4 ×
Name	LED_1	
User Module	LED	
Version	1.2	
Port	Port_1	
Pin	Port_1_6	
Drive	Active High	

22. Configure the **Global Resources** window to match the following figure.



Power Setting [Vcc / Sy	5.0V / 24MHz	~		
CPU_Clock	SysClk/2			
32K_Select	Internal			
PLL_Mode	Disable			
Sleep Timer Period	1.95ms			
VC1= SysClk/N	16			
VC2=VC1/N	16			
VC3 Source	VC2			
VC3 Divider	256			
SysClk Source	Internal			
SysClk*2 Disable	Yes			
Analog Power	SC Off/Ref Low			
Ref Mux	(Vdd/2)+/-(Vdd/2)			
AGndBypass	Disable			
Op-Amp Bias	Low			
SwitchModePump	OFF			
Trip Voltage [LVD (SMP)	4.81V (5.00V)			
LVDThrottleBack	Disable			
Watchdog Enable	Disable			

#### Figure 3-13. Global Resources Window

23. Open the existing *main.c* file in Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C28\_main\_Ex1.c* file, which is available within the attachments feature of this PDF document.

Figure 3-14. Workspace Explorer



24. Save the project.

25. To build the project, click Build  $\rightarrow$  Generate/Build 'Ex1\_LED\_with\_PWM' Project.

26. Disconnect power to the board.

- 27. Configure the DVK board SW3 to 5 V.
- 28. Configure the DVK breadboard using the included jumper wires:
  - P1[6] to LED1

P1[7] to LED2

- 29. Reapply power to the board.
- 30.Use PSoC Designer as described in Programming My First PSoC 1 Project on page 22 to program the device.
- 31. Reset the DVK and observe the blinking LEDs.
- 32. Save and close the project.



#### 3.1.1.2 main.c

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded *CY8C28\_main\_Ex1.c* file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
#include <m8c.h>
                  /* Part specific constants and macros */
#include "PSoCAPI.h"
                  /* PSoC API definitions for all User Modules */
* Function Name: main
* Summary:
* The main function initializes the PWM and starts the PWM clock which will
* blink LED1. Then the main loop is entered which delays enough for LED2 to
* blink at a quicker rate than LED1.
* Parameters:
* void
* Return:
* void
void main(void)
{
                /* Variable used for delay */
  WORD i;
  LED_1_Start();
                /* Enable Software controlled LED */
  /* The following loop controls the software LED connected to P1.7 */
  while(1)
  /* Delay time depends on compiler optimization levels and CPU clock */
  for (i = 0; i < 60000; i++);// Gives approximately 450 msec delay with Image-
Craft
  // and 170 msec with HiTech
  #ifdef HI TECH C
  for (i = 0; i < 60000; i++);// Give some more delay if HiTech compiler is used.
  for (i = 0; i < 40000; i++);
  #else
  #endif
  /* Switch the state of Software LED (on or off) */
  LED_1_Invert();
  } /* End of while(1) */
} /* End of main */
/* [] END OF FILE */
```



## 3.1.2 ADC to LCD Project

This project demonstrates a 9-bit delta-sigma analog-to-digital converter (ADC) by measuring the voltage of the potentiometer center tap wiper and displaying the result on the LCD. Connect the voltage potentiometer (VR) to the ADC input P0[1]. The program reads the 9-bit ADC result and prints it to the LCD.

#### 3.1.2.1 Creating ADC to LCD Project

- 1. Follow steps 1 to 10 in the section Creating My First PSoC 1 (CY8C28) Project on page 39; change the project name to Ex2\_ADC\_to\_LCD.
- In the User Modules window, expand the ADCs folder and right-click DelSigPlus; select Place. A window opens with multiple options for the DelSigPlus UM. Here, the DS1128 configuration is used. Scroll down in the window to verify that this is the case.



Figure 3-15. Multiple User Module Window

- 3. Click OK.
- 4. Verify that the DelSigPlus\_1 UM is placed in ASC10.
- 5. In the User Modules window, expand the Amplifiers window. Right-click PGA and select Place. Ensure that the PGA is placed in ACC00.







- 6. In the User Modules window, expand Misc Digital; right-click LCD and click Place.
- 7. Click **PGA\_1** and configure the properties to match this figure.

Figure 3-17. PGA\_1 Properties

Name	PGA_1
User Module	PGA
Version	3.2
Gain	1.000
Input	AnalogColumnMUXBusSwitch_0
Reference	AGND
AnalogBus	Disable

8. Click **DelSigPlus\_1** and configure the properties to match this figure.

Name	DelSigPlus_1	
User Module	DelSigPlus	
Version	1.0	
DataFormat	Unsigned	
ClockPhase	Normal	
PosInput	ACC00	
NegInput	ACC00	
NegInputGain	Disconnected	



9. Click **LCD\_1** and configure the properties to match this figure.

#### Figure 3-19. LCD\_1 Properties

perties - LCD_1		<b>-</b> ₽ ×
Name	LCD_1	
User Module	LCD	
Version	1.5	
LCDPort	Port_2	
BarGraph	Disable	

10. Configure the **Global Resources** to match the following figure.

#### Figure 3-20. Global Resources

Global Resources - ex2_adc	_to_lcd	🚽 🕂 🗙
Power Setting [Vcc / Sy	5.0V / 24MHz	~
CPU_Clock	SysClk/2	
32K_Select	Internal	
PLL_Mode	Disable	
Sleep Timer Period	1.95ms	
VC1= SysClk/N	12	
VC2= VC1/N	16	
VC3 Source	VC2	
VC3 Divider	256	
SysClk Source	Internal	
SysClk*2 Disable	Yes	
Analog Power	SC On/Ref High	
Ref Mux	(2 BandGap)+/-BandGap	
AGndBypass	Disable	
Op-Amp Bias	High	
SwitchModePump	OFF	
Trip Voltage (LVD (SMP)	4.81V (5.00V)	
LVDThrottleBack	Disable	
Watchdog Enable	Disable	

11. Ensure that **AnalogColumn\_InputMUX\_0** is connected to **Port\_0\_1**. If it is not configured for this port, double-click the mux and choose **Port\_0\_1**.





Figure 3-21. AnalogColumn\_InputMUX\_0 Connected to Port\_0\_1

12. Ensure that AnalogColumn\_Clock\_0 is connected to VC1. If it is not, double-click the mux and choose VC1.

> GA 1 GAIN

Reference



AGND

Figure 3-22. AnalogColumn\_Clock\_0 Connected to VC1



- 13.Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded CY8C28\_main\_Ex2.c file, which is available within the attachments feature of this PDF document.
- 14. Save the project.
- 15. To build the project, click **Build**  $\rightarrow$  **Generate/Build** 'Ex2\_ADC\_to\_LCD' Project.
- 16. Disconnect power to the board.
- 17. Configure the DVK SW3 to 5 V.
- 18. Configure the DVK breadboard using the included jumper wires:

P0[1] to VR

Figure 3-23. Connect P0[1] to VR



- 19. Reapply power to the board.
- 20.Use PSoC Designer as described in Programming My First PSoC 1 Project on page 22 to program the device.
- 21. After programming the device, press the reset button and vary the potentiometer (R20) to see the results on the LCD.



**Note** The ADC output values may not reach full range due to potentiometer and ADC limitations. ADC values may fluctuate several counts due to system noise, and if the potentiometer voltage is at the edge of an ADC count.

22. Save and close the project.

#### 3.1.2.2 main.c

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded *CY8C28\_main\_Ex2.c* file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
/* part specific constants and macros */
#include <m8c.h>
#include "PSoCAPI.h"
                   /* PSoC API definitions for all User Modules */
/* LCD specific */
#define ROW_0 0 /* LCD row 0
                            */
#define ROW_1 1 /* LCD row 1
                            */
#define COLUMN_0 0 /* LCD column 0 */
#define COLUMN_9 9 /* LCD column 9 */
* Function Name: main
*
* Summary:
  The main function initializes both the ADC and LCD, starts and waits for an
*
  ADC conversion, then it displays the raw counts to the LCD.
* Parameters:
 void
* Return:
 void
void main(void)
{
   WORD adcResult; /* Holds the integer ADC result */
   /* Initialize the PGA used to buffer input from the potentiometer (VR) on
     P0.1 to the ADC */
   PGA_1_Start(PGA_1_HIGHPOWER);
   DelSigPlus_1_Start(DelSigPlus_1_HIGHPOWER); /* Initialize the ADC */
   LCD_1_Start();
                               /* Initialize the LCD */
   LCD_1_Position(ROW_0, COLUMN_0); /* Set the LCD to (Row=0,Column=0) */
   LCD_1_PrCString("V Count: ");
   DelSigPlus_1_StartAD(); /* Start gathering conversions from the ADC */
   M8C_EnableGInt;
                  /* Enable Global interrupts */
   /* This loop waits for a valid ADC result, and displays it on the LCD */
```



```
while (1)
{
    /* Is there ADC data? */
    if(DelSigPlus_1_fIsDataAvailable())
    {
        /* Store result from ADC */
        adcResult = DelSigPlus_1_wGetDataClearFlag();
        LCD_1_Position(ROW_0, COLUMN_9); /* Set LCD to (Row=0,Column=9) */
        LCD_1_PrHexInt(adcResult); /* Print ADC result on LCD */
     }
    /* End of while(1) */
} /* End of main */
```

/\* [] END OF FILE \*/

## 3.1.3 ADC to UART with DAC

This project demonstrates sine wave generation by using a 6-bit digital-to-analog converter (DAC). The sine wave period is based on the current value of the ADC. The firmware reads the voltage output by the DVK board potentiometer and displays the raw counts on the DVK board character LCD display similar to those shown in the previous project. A 6-bit DAC outputs a table generated sine wave at a frequency proportional to the ADC count. The frequency outputs to an oscilloscope. A 38400 Baud UART outputs the current ADC count as ASCII formatted into a hexadecimal number.

#### 3.1.3.1 Creating ADC to UART with DAC Project

- 1. Follow steps 1 to 10 in the section Creating My First PSoC 1 (CY8C28) Project on page 39; change the project name to Ex3\_ADC\_to\_UART\_with\_DAC.
- In the User Modules window expand the ADCs folder and right-click DelSigPlus; select Place. A window opens with multiple options for the DelSigPlus UM. Here, the DS1128 configuration is used. Scroll down in the window to verify that this is the case.
- 3. Click OK.
- 4. Verify that the UM is placed in **ASC10**.
- 5. In the User Modules window, expand the Amplifiers window. Right-click PGA and select Place. Ensure that the PGA is placed in ACC00.







- 6. In the User Modules window, expand Misc Digital, right-click LCD, and select Place.
- 7. In the User Modules window, expand Counters, right-click Counter16, and select Place.
- 8. In the User Modules window, expand Digital Comm, right-click TX8, and select Place.
- 9. In the **User Modules** window, expand **DACs**, right-click **DAC6**, and select **Place**. User module is placed in ASD20 analog block by default. Drag and drop it to ASC21 block.
- 10. Move the UMs so that they match the configuration shown in Figure 3-25.





Figure 3-25. Configure User Modules

11. Click on **DelSigPlus\_1** and configure it to match this figure.

Figure 3-26. DelSigPlus\_1 Properties

Name	DelSigPlus_1	
User Module	DelSigPlus	
Version	1.0	
DataFormat	Unsigned	
ClockPhase	Normal	
PosInput	ACC00	
NegInput	ACC00	
NegInputGain	Disconnected	



#### 12.Click **PGA\_1** and configure it to match this figure.

#### Figure 3-27. PGA\_1 Properties

Name	PGA_1
User Module	PGA
Version	3.2
Gain	1.000
Input	AnalogColumnMUXBusSwitch_0
Reference	AGND
AnalogBus	Disable

13. Click **DAC6\_1** and configure it to match this figure.

## Figure 3-28. DAC6\_1 Properties

Name	DAC6 1	
User Module	DAC6	
Version	4.3	
AnalogBus	AnalogOutBus_1	
ClockPhase	Normal	
DataFormat	OffsetBinary	

14. Click **LCD\_1** and configure it to match this figure.

#### Figure 3-29. LCD\_1 Properties

perties - LCD_1		- 4 ×
Name	LCD_1	
User Module	LCD	
Version	1.5	
LCDPort	Port_2	
BarGraph	Disable	

15. Click on **Counter16\_1** and configure it to match this figure.

Figure 3-30. Counter16\_1 Properties

Name	Counter16_1
User Module	Counter16
Version	2.5
Clock	VC2
Enable	High
CompareOut	None
TerminalCountOut	None
Period	0
CompareValue	0
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

16.Click **TX8\_1** and configure it to match this figure.

Figure 3-31. TX8\_1 Properties

Name	TX8 1
User Module	TX8
Version	3.3
Clock	VC3
Output	Row_0_Output_2
TX Interrupt Mode	TXComplete
ClockSync	Sync to SysClk
Data Clock Out	None

17.Click RO0[2] LUT, enable Row\_0\_Output\_2\_Drive\_2 to connect GlobalOutOdd\_2.



Figure 3-32. Digital Interconnect Window

18. Click GlobalOutOdd\_2. In the window that appears, configure Pin for Port\_1\_2.



Figure 3-33. Configure Pin for Port\_1\_2



19.Click **OK** to continue.20.Click **AnalogOutBuf\_1** and configure it for **Port\_0\_5**.

Figure 3-34. Configure AnalogOutBuf\_1



21. Verify that **AnalogColumn\_InputMUX\_0** is connected to **Port\_0\_1**. If it is not configured for this port, double-click the mux and choose **Port\_0\_1**.





Figure 3-35. AnalogColumn\_InputMUX\_0 Connection



22. Verify that AnalogColumn\_Clock\_0 and AnalogColumn\_Clock\_1 are connected to VC2. If it is not, double-click the mux and chose VC2.

Figure 3-36. AnalogColumn\_Clock\_0 Connection





23. Configure Global Resources to match the following figure.

Figure 3-37. Configure Global Resources

G	obal Resources - ex3_adc		- 4 ×
	Power Setting [Vcc / Sy		~
	CPU_Clock	SysClk/2	
	32K_Select	Internal	
	PLL_Mode	Disable	
	Sleep Timer Period	1.95ms	
	VC1= SysClk/N	16	
	VC2=VC1/N	6	
	VC3 Source	SysClk/1	
	VC3 Divider	78	
	SysClk Source	Internal	
	SysClk*2 Disable	No	
	Analog Power	SC On/Ref High	
	Ref Mux	(2 BandGap)+/-BandGap	
	AGndBypass	Disable	
	Op-Amp Bias	High	
	SwitchModePump	OFF	
	Trip Voltage (LVD (SMP)	4.81V (5.00V)	
	LVDThrottleBack	Disable	
	Watchdog Enable	Disable	

- 24. Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C28\_main\_Ex3.c* file, which is available within the attachments feature of this PDF document.
- 25. Save the project.
- 26. To generate the project, click Build  $\rightarrow$  Generate/Build 'Ex3\_ADC\_to\_UART\_with\_DAC' Project.
- 27.Open your *boot.tpl* file in the project folder **Files** → **Open File**. Select **All Files** for **Files of the type:**.
- 28. Select *boot.tpl* in the list of files and click **Open**.
- 29.Find the line '@INTERRUPT\_9' (for PSoC Block DBC01) and replace that line with: ljmp\_Counter16\_C\_ISR
- 30. Save the project.
- 31. To build the project, click Build  $\rightarrow$  Build 'Ex3\_ADC\_to\_UART\_with\_DAC' Project.
- 32. Disconnect power to the board.
- 33. Configure the DVK SW3 to 5 V.
- 34. Configure the DVK breadboard using the included jumper wires as follows:

P0[1] to VR

P1[2] to TX

P0[5] to Oscilloscope

**Note** An LED (P0[5] to LED1) by nature does not accurately show the changes in frequency the best way to see this is to use a Oscilloscope (P0[5] to Oscilloscope).





Figure 3-38. Connect P0[1] to VR, P1[2] to TX, and P0[5] to LED1

- 35.Connect a serial cable to the PC and the DVK board.
- 36.On the DVK board, verify that RS232\_PWR(J10) is jumpered to ON.
- 37. Reapply power to the board.
- 38.Use a terminal application such as TeraTerm or HyperTerminal with these setup parameters. Baud Rate: 38400
  - Data: 8-bit
  - Parity: none
  - Stop: 1bit
  - Flow Control: none
- 39.Use PSoC Designer as described in Programming My First PSoC 1 Project on page 22 to program the device.
  - After programming the device, press Reset and vary the pot to see the result on the LCD as well as in the terminal application. View the DAC output on a scope or with an LED.
  - **Note** The ADC output values may not reach full range due to potentiometer and ADC limitations. ADC values may fluctuate several counts due to system noise, and if the potentiometer voltage is at the edge of an ADC count.
- 40. Save and close the project.



#### 3.1.3.2 main.c

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded CY8C28\_main\_Ex3.c file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
/* part specific constants and macros */
#include <m8c.h>
#include "PSoCAPI.h"
                    /* PSoC API definitions for all User Modules */
/* Counter16 Interrupt Handler */
#pragma interrupt_handler Counter16_C_ISR
/* LCD specific */
#define ROW_0 0 /* LCD row 0
                               */
#define ROW 1 1 /* LCD row 1
                               */
#define COLUMN_0 0 /* LCD column 0 */
#define COLUMN_9 9 /* LCD column 9 */
const BYTE sinTable[]=
{
  0\,,\quad 0\,,\quad 1\,,\quad 2\,,\quad 3\,,\quad 4\,,\quad 6\,,\quad 7\,,\ 10\,,\ 12\,,\ 14\,,\ 17\,,\ 20\,,\ 23\,,\ 26\,,\ 29\,,
 31, 33, 36, 39, 41, 44, 46, 49, 51, 53, 55, 56, 58, 59, 59, 60,
 60, 60, 59, 59, 58, 56, 55, 53, 51, 49, 47, 44, 42, 39, 36, 33,
 31, 28, 25, 22, 19, 16, 13, 11, 9, 7, 5, 3, 2, 1, 0, 0
};
BYTE tablePos = 0;
* Function Name: main
*
* Summary:
   The main function initializes the ADC, PGA, LCD, Counter, DAC and UART.
   In the main loop, it continuously checks for an ADC conversion. If there is
*
   one then it displays the ADC raw count to the LCD, transmits the raw count
*
   serially, and updates the Counter16 period (based on the raw count) for the
*
   DAC output.
*
* Parameters:
 void
* Return:
 void
void main(void)
{
   /* Variable for holding ADC result, and updating counter period */
   WORD adcResult;
   Counter16_1_Start();
                          /* Enable the counter used for DAC update rate */
   Counter16_1_EnableInt(); /* Enable DAC update interrupt */
```



```
/* Start the TX8 UM with no parity (baud rate = 38400) */
   TX8_1_Start(TX8_1_PARITY_NONE);
   /* Enable to PGA to buffer signal from VR to ADC */
   PGA_1_Start(PGA_1_HIGHPOWER);
   DAC6_1_Start(DAC6_1_HIGHPOWER);
                                    /* Start the DAC */
   DelSigPlus_1_Start(DelSigPlus_1_HIGHPOWER); /* Start the ADC */
                                    /* Start reading values on the ADC */
   DelSigPlus_1_StartAD();
   LCD_1_Start();
                                     /* Start the character LCD */
   LCD_1_Position(ROW_0, COLUMN_0); /* Set the LCD to (Row=0,Column=0) */
   LCD_1_PrCString("V Count: ");
   M8C EnableGInt;
                                    /* Enable Global Interrupts */
   while(1)
   {
       /* Step 1: Get BYTE data from the ADC
          Step 2: Write BYTE data from ADC to the counter to
                 change the DAC update rate
          Step 3: Move the LCD cursor back to the beginning and display new
                 ADC data
          Step 4: Write ADC data out the TX port, and then send a return
       */
       /* Is new data available from the ADC? */
       if (DelSigPlus_1_fIsDataAvailable())
       {
           adcResult = DelSigPlus_1_wGetDataClearFlag(); /* Get new ADC data */
           /* Change DAC update rate counter */
           Counter16_1_WritePeriod((adcResult << 4) + 200);</pre>
           LCD_1_Position(ROW_0, COLUMN_9); /* Move LCD (row=0,column=0) */
           LCD_1_PrHexInt(adcResult); /* Print ADC result to LCD */
TX8_1_PutSHexInt(adcResult); /* Write LCD result to TX8 -> PC */
                                          /* Write return character to TX8 */
           TX8_1_PutCRLF();
       }
   } /* End of while(1) */
} /* End of Main */
* Function Name: Counter16_C_ISR
* Summary:
*
   This is the interrupt service routine for the Counter16 usermodule written
   in C. The boot.tpl has been modified to jump to this ISR every terminal
   count. The related #pragma above is necessary for the boot.asm file to jump
   to it. Every time a terminal count is reached the DAC will get the next
   value from the sinTable.
* Parameters:
* void
* Return:
* void
```

+



/\* [] END OF FILE \*/

#### 3.1.4 CapSense

This project demonstrates CapSense. The firmware displays the CapSense button presses on the LCD (row 1) and associated LEDs. It also displays the CapSense slider position on the LCD (row 2).

Note that this project uses IDAC. But if you are using an external Rb with CSD, then populate R15 (connected to P3[1]). Rb can range from 2 k to 10 k. See the CapSense user module datasheet for more information on using Rb.

#### 3.1.4.1 Creating CapSense Project

- Follow steps 1 to 10 in the section Creating My First PSoC 1 (CY8C28) Project on page 39; change the project name to Ex4\_CapSense.
- 2. In the User Modules window, expand the **Cap Sensors** folder. Right-click **CSD** and select **Place**. A window appears with the option to use the default configuration.

	Default Configuration	
	e the default configuration ( IDAC Configuration + First Order Sigma Delta \$16 ) and close the wizard?	
	⊙ Yes	
	O No, I want to make a custom selection	
ncel		ОК

Figure 3-39. Select Multi User Module Window

- 3. Select **Yes** and click **OK**.
- 4. Right-click the CSD user module in the workspace explorer and select CSD Wizard.



Figure 3-40. Select CSD Wizard

	<b>→</b> 4 Þ <b>×</b>	Workspace Explorer	<del>↓</del> # X
	MAU           MAU	Ex4_CapS	CapSense' (1 project) ense [CY8C28645-24LTXI] pSense [Chip] dable Configurations ex4_capsense - 1 User Modules CSD_1 Next Allowed Placement Place
	No.4 No.4 No.4 No.4 No.4 No.4 No.4 No.4		Unplace Rename Delete Select Color > Datasheet
2227 <b>1</b> 00227	NUC	User Modules User Modu De Co ADCs Amplifi	Properties Selection Options CSD Wizard

5. The CapSense Wizard window opens.





- 6. In the CapSense Wizard window, under the Global Settings tab, set the # of buttons to '2'.
- 7. Select P0[7] as the Modulator Capacitor Pin.



CapSense Wizard	
Global Settings Sensors Settings Buttons 2 Siders 1 Radial Silders 0 Analog Bus Both Modulator Capacitor F P0[7]	SW0 SW1
Modulator Capacitor Pin Modulator Capacitor Pin Chip Pin Assignment View Table Pin Assignment View	(c)
P2[3]         1         C         P36         F         P26         P26	
P3(1)         7         30         P3(1)           P3(5)         9         28         P3(8)           P3(5)         9         28         P3(8)           P3(1)         10         27         P3(4)           P3(1)         11         26         P3(2)           P3(1)         12         26         P3(2)           T3 14 15 16 17 18 19 20 21 22 23 24         P3(0)         P3(0)           E         E         E         E         E         E         E           E <td>Legend for Chip: Unavailable pins</td>	Legend for Chip: Unavailable pins
Total Sensors: 7   Switches: 2   Sliders: 1   Radial Sliders: 0	Locked pins     Available pins     Assigned pins     OK     Cancel

Figure 3-42. CapSense Wizard Place Buttons

- 8. Click and hold SW0 and drag it to P0[5].
- 9. Click and hold SW1 and drag it to P0[6].







- 10. Repeat for each slider sensor and corresponding pin.
  - S0[0] to P0[0] S0[1] to P0[1] S0[2] to P0[2] S0[3] to P0[3] S0[4] to P0[4]
- 11. Select the Sensors Settings tab.
- 12.Set the Resolution to 80.

Figure 3-44. Sensors Settings Tab

Configure 'cy_pins'		? 🔀
Name: LED1		
Pins Mapping Bu	ilt-in	4 Þ
Number of Pins: 1	× 🗗 🕈 🖊 🕅 💥	
[All Pins]	Type General Input Output	
└⊠ LED1_0	Analog Preview:	
	Digital Input	
	HW Connection	
	🗹 Digital Output	
	HW Connection	
	Output Enable	
	Bidirectional	
	Show Annotation Terminal	
	]	
Datasheet	OK Apply	Cancel

13.Click **OK**.

14. In the User Modules window, expand Misc Digital, right-click LCD, and select Place.
15. In the User Modules window, expand Misc Digital, right-click LED, and select Place.
16. In the User Modules window, expand Misc Digital, right-click LED, and select Place.
17. Click CSD\_1 and configure it to match this figure.



Figure 3-45. CSD\_1 Properties

Configure 'cy_pins'		? 🛛
Name: LED1		
Pins Mapping Bui	t-in	4 Þ
Number of Pins: 1		
(Al Fins) └─⊠ LED1_0	Type General Input	Output Initial State: Low (0) V Minimum Supply Voltage:

18. Click **LCD\_1** and configure it to match this figure.

#### Figure 3-46. LCD\_1 Properties

operties - LCD_1		- 4 ×
Name	LCD_1	
User Module	LCD	
Version	1.5	
LCDPort	Port_2	
BarGraph	Enable	

19. Click **LED\_1** and configure it to match this figure.

## Figure 3-47. LED\_1 Properties

Properties - LED_1		<b>→</b> ₽ ×
Name	LED_1	
User Module	LED	
Version	1.2	
Port	Port_1	
Pin	Port_1_6	
Drive	Active High	

20. Click **LED\_2** and configure it to match this figure.

#### Figure 3-48. LED\_2 Properties

Properties - LED_2	<b>~</b> 4	×
Name	LED_2	
User Module	LED	
Version	1.2	
Port	Port_1	
Pin	Port_1_7	
Drive	Active High	

21. Configure Global Resources to match the following figure.



Figure 3-49. Configure Global Resources



- 22. Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C28\_main\_Ex4.c* file, which is available within the attachments feature of this PDF document.
- 23. Save the project.
- 24. To generate and build the project, click  $\textbf{Build} \rightarrow \textbf{Generate/Build 'Ex4_CapSense' Project}.$
- 25. Disconnect power to the board.
- 26. Configure the DVK board SW3 to 5 V.
- 27. Configure the DVK breadboard using the included jumper wires:

P1[6] to LED1

P1[7] to LED2

- 28. Ensure that P0[1], P0[5], and P0[7] are disconnected.
- 29. Reapply power to the board.
- 30.Use PSoC Designer as described in Programming My First PSoC 1 Project on page 22 to program the device.
- 31. Reset the DVK. An LED lights up when either CapSense button is pushed. If B1 (P0[5]) is pushed, it also displays "Button1" in the top row of the LCD display. Similarly, if B2 (P0[6]) is pushed, it displays "Button2" in the top row of the LCD display. The bottom row of the LCD displays the slider position with a horizontal bargraph.
- 32. Save and close the project.



#### 3.1.4.2 main.c

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded CY8C28\_main\_Ex4.c file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
#include <m8c.h> /* part specific constants and macros */
#include "PSoCAPI.h" /* PSoC API definitions 5
                    /* PSoC API definitions for all User Modules */
/* LCD specific */
#define ROW_0 0 /* LCD row 0
                               */
              1 /* LCD row 1
#define ROW 1
                               * /
#define COLUMN_0 0 /* LCD column 0 */
#define NUM_CHARACTERS 16 /* Number of characters on LCD */
/* For clearing a row of the LCD*/
#define CLEAR ROW STR
                     .....
/* Button 1 only string for row 0 of the LCD */
#define BUTTON_1_STR "Button1 "
/* Button 2 only string for row 0 of the LCD */
#define BUTTON 2 STR
                   "
                              Button2"
/* Button 1 and 2 string for row 0 of the LCD */
#define BUTTON_1_2_STR "Button1 Button2"
/* Default string for button row of the LCD */
#define DEFAULT_ROW_0_STR "Touch Buttons "
/* Default string for slider row of the LCD */
#define DEFAULT ROW 1 STR "Touch The Slider"
/* CapSense specific */
#define SLIDER_RESOLUTION 80
#define SCANSENSOR_BTN_B1 0
#define SCANSENSOR_BTN_B2 1
void UpdateButtonState(BYTE sensor_1, BYTE sensor_2);
void UpdateSliderPosition(BYTE value);
* Function Name: main
* Summary:
* The main function initializes CapSense and the LCD. Then it continuously
* scans all CapSense sensors (slider sensors and buttons), gets the state of
* the buttons and slider and updates the LCD with the current state.
* Parameters:
 void
* Return:
* void
void main(void)
{
```



```
BYTE pos;
              /* Slider Position */
   BYTE stateB_1; /* Button1 State */
   BYTE stateB_2; /* Button2 State */
   M8C_EnableGInt; /* Enable Global Interrupts */
   /* LCD Initialization */
   LCD_1_Start();
   /* For Bargraph display on LCD */
   LCD_1_InitBG(LCD_1_SOLID_BG);
   /* LED1 Initialization */
   LED_1_Start();
   /* LED2 Initialization */
   LED 2 Start();
   /* CapSense Initialization */
   CSD 1 Start();
   /* Initialize the baselines by scanning all sensors and getting the initial
      raw data values */
   CSD 1 InitializeBaselines();
   /* Load finger thresholds set in user module parameters */
   CSD_1_SetDefaultFingerThresholds();
   while(1)
   {
       /* Scan each CapSense sensor and update their raw data value */
       CSD_1_ScanAllSensors();
       /* Update baselines for each sensor */
       CSD_1_UpdateAllBaselines();
       /* Update state to active/inactive for each button sensor */
       stateB_1 = CSD_1_bIsSensorActive(SCANSENSOR_BTN_B1);
       stateB_2 = CSD_1_bIsSensorActive(SCANSENSOR_BTN_B2);
       /* Get Linear Slider Position */
       pos = CSD_1_wGetCentroidPos(1);
       /* Update LCD and LED's with current Button and Linear Slider states */
       UpdateButtonState(stateB_1, stateB_2);
       UpdateSliderPosition(pos);
   }
* Function Name: UpdateButtonState
* Summary:
  Updates the LCD screen with the current button state by displaying which
* button is being touched on row 0. LED's are also updated according to button
  state.
* Parameters:
* sensor_1: Button state for B1
* sensor_2: Button state for B2
* Return:
```

}


```
void
*
*
void UpdateButtonState(BYTE sensor_1, BYTE sensor_2)
{
   LCD_1_Position(ROW_0,COLUMN_0);
   /* Check the state of the buttons and update the LCD and LEDs */
   if (sensor_1 && sensor_2)
      /* Display both Button strings on LCD if both button sensors are active */
      LCD_1_PrCString(BUTTON_1_2_STR);
       /* Both LED's are on in this state */
      LED_1_On();
      LED_2_On();
   }
   else if (sensor_1 || sensor_2)
   {
       if (sensor_1)
       {
          /* Display Button 1 state on LCD and LED1 */
          LCD_1_PrCString(BUTTON_1_STR);
          LED_1_On();
          /* Button 2 is not active */
          LED_2_Off();
       }
      else // sensor_2
       {
          /* Display Button 2 state on LCD and LED2 */
          LCD_1_PrCString(BUTTON_2_STR);
          LED_2_On(); /* Turn on LED2 */
          LED_1_Off(); /* Turn off the LED1 */
      }
   }
   else
   {
       /* Display default string on LCD and set LED's to off */
      LCD_1_PrCString(DEFAULT_ROW_0_STR);
      /* Set both LED's off in this state */
      LED_1_Off();
      LED_2_Off();
   }
}
* Function Name: UpdateSliderPosition
*
* Summary:
 Updates the LCD screen with the current slider position by displaying the
*
 horizontal bargraph.
* Parameters:
*
 value: Centroid position from CapSense slider.
*
* Return:
*
 void
```



```
void UpdateSliderPosition(BYTE value)
{
   /* The slider position is 0xFF if there is no finger present on the slider */
   if (value > SLIDER_RESOLUTION)
   {
      /* Clear old slider position (2nd row of LCD) */
      LCD_1_Position(ROW_1, COLUMN_0);
      LCD_1_PrCString(DEFAULT_ROW_1_STR);
   }
   else
   {
      /* Update the bargraph with the current finger position */
      LCD_1_DrawBG(ROW_1, COLUMN_0, NUM_CHARACTERS, value + 1);
   }
}
/* [] END OF FILE */
```



# 3.2 CY8C29 Family Processor Module Code Examples

## 3.2.1 My First PSoC 1 (CY8C29) Project

## 3.2.1.1 Creating My First PSoC 1 (CY8C29) Project

- 1. Open PSoC Designer
- 2. To create a new project, click **File**  $\rightarrow$  **New Project**. The **New Project** window opens.
- 3. In the **New Project window**, select the **Chip-Level Project**. Name the project **Example\_My\_First\_PSoC\_Project**.
- 4. In the Location field, click Browse and navigate to the appropriate directory.

Figure 3-50. New Project Window

New Project	<u>?</u> ]	×
Project types:		
Chip-level S Project	ystem-level Project	
This is a classic v4	4.x PSoC Designer project, selecting and placing user modules.	
<u>N</u> ame:	Example_My_First_PSoC_Project	
Location:	C:\ <u>B</u> rowse	
Workspace na <u>m</u> e:	Example_My_First_PSoC_Project Create directory for workspace	
	<u> </u>	

5. Click OK. The Select Project Type window opens.



Path:			<u>B</u> rowse
	$\mathbf{C}$ Use the same target device	C Select target device	Cl <u>e</u> ar Path
Select i	Target Device		1
<u>D</u> evice	CY8C29866-24AXI		
	Generate 'Main' file using:		
	C Assembler		
	_		

Figure 3-51. Select Project Type Window

- 6. In this window, under Select Target Device, click View Catalog.
- 7. The **Device Catalog** window opens. Click on the **PSoC** tab, and scroll down to the **CY8C29466**, **CY8C29566**,... section.
- 8. For this project, click any device in this section and then click **Select**.

Figure 3-52. Device Catalog Window

Part Number	Analog Blocks	Digital Blocks	Flash	RAM	IO Count	Supply Voltage	SMP	т
A <b>T</b>	A.4	A <b>T</b>	<b>**</b>	A	A.	<b>**</b>	A	
Click here to Remove All Filters	all 🔽	all 🔽	all 💌		all 🔽	all 💌	all 💌	all "
CY8C28445-24PVXI	12 + *4	12	16K	1024	24	3.0 to 5.25	Yes	In
CY8C28545-24AXI	12 + *4	12	16K	1024	40	3.0 to 5.25	Yes	In
CY8C28645-24LTXI	12 + *4	12	16K	1024	44	3.0 to 5.25	Yes	Ir
CY8C29466-24PVXI	12	16	32K	2K	24	3.0 to 5.25	YES	Ir
CY8C29466, CY8C29 CY8C29466-24PXI	12	16	32K	2K	24	3.0 to 5.25	YES	Ir
CY8C29466-24PVXI	12	16	32K 32K	2K 2K	24	3.0 to 5.25 3.0 to 5.25	YES	
CY8C29466-24SXI					24 40			lr -
	12	16	32K	2K		3.0 to 5.25	YES	lr
CY8C29666-24PVXI	12	16	32K	2K	44	3.0 to 5.25	YES	Ir
CY8C29666-24LFXI	12	16	32K	2K		3.0 to 5.25	YES	lr
CY8C29866-24AXI	12	16	32K	2K	64	3.0 to 5.25	YES	l Ir
🖃 🚖 СҮ8СНР102В, СҮ8СН	IP102E (Datashe	et) (Help Me Cl	hoose a Part	)				
CY8CNP102B-AXI	12	16	32K	2K	33	3.0 to 3.6	YES	l
CY8CNP102E-AXI	12	16	32K	2K	33	4.75 to 5.25	YES	1

9. Under Generate 'Main' File Using:, select C, then click OK.



10.By default, the project opens in Chip view.

#### Figure 3-53. Default View.



11. In the User Modules window, expand the PWMs folder.



Figure 3-54. User Modules Window

12. In this folder, right-click on **PWM8** and select **Place**. The User Module (UM) is placed in the first available digital block.





Figure 3-55. Place User Module PWM8

13. Double-click the placed PWM8\_1 UM; the **Properties** window opens on the left side of the screen. Configure the PWM with the settings as in the following figure. If the **Properties** window does not appear, click **View** → **Properties Window**.

Name	PWM8_1
User Module	PWM8
Version	2.5
Clock	VC3
Enable	High
CompareOut	Row_0_Output_3
TerminalCountOut	None
Period	100
PulseWidth	50
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InvertEnable	Normal

Figure 3-56. Properties Window

G00

0 7



14.Next, route the PWM CompareOut signal to P0[7]. The first step is to configure the lookup table (LUT) on Row\_0\_Output3.







15. Double-click the LUT, the **Digital Interconnect** window opens.

16. In this window, enable Row\_0\_Output\_3\_Drive\_1 to connect to GlobalOutEven\_7.

Figure 3-58. Digital Interconnect Window



## 17.Click Close.

18. Click on GlobalOutEven\_7. In the window that appears, configure Pin for Port\_0\_7.



Figure 3-59.	Configure	Pin for	Port	0	7
i igui e e ee.	Configure	1 111 101	1 011		_'

Port_0_7           Port_1_0           Port_1_1           Port_1_2           Port_1_3           Port_1_6           Port_1_7           Port_1_6           Port_2_0           Port_2_1           Port_2_3           Port_2_4           Port_2_6           Port_2_7			1
Global DutEven_1       Port_1_1         Port_1_3       Port_1_4         Port_1_5       Port_1_6         Port_2_7       Port_2_0         Interconnect       Port_2_1         QK       Cancel         QK       Cancel		Port_0_7	Σ
GlobalOutEven_1       Port_1_2         Port_1_3       Port_1_4         Port_1_5       Port_1_6         Port_2_0       Port_2_0         Port_2_1       Port_2_2         Port_2_3       Port_2_4         Port_2_6       Port_2_6		Port_1_0	>
GlobalDutEven_1       Port_1_3         Port_1_4       Port_1_5         Port_1_6       Port_1_7         Interconnect       Port_2_1         Port_2_3       Port_2_3         Port_2_5       Port_2_8		Port_1_1	>
GidbalOutEven_       Port_1_4         Port_1_5       Port_1_6         Port_1_7       Port_2_0         Port_2_1       Port_2_1         Port_2_3       Port_2_4         Port_2_5       Port_2_6		Port_1_2	>
Port_1_4           Port_1_5           Port_1_6           Port_1_7           Interconnect           Port_2_1           Port_2_2           Port_2_3           Port_2_5           Port_2_8		Port_1_3	>
Pin         Port_0_7         Port_1.5           Interconnect         Port_2.0         Port_2.1           Port_2.2         Port_2.3         Port_2.3           Port_2.5         Port_2.6         Port_2.6		Port_1_4	>
Pin         Port_0_7         Port_1_6           Interconnect         Port_1_7         Port_2_0           Port_2_1         Port_2_1           Port_2_3         Port_2_3           Port_2_5         Port_2_6		Port_1_5	>
Interconnect         Port_1_7           Port_2_0         Port_2_1           Port_2_2         Port_2_2           Port_2_3         Port_2_4           Port_2_5         Port_2_8		Port_1_6	$\geq$
Port_2_0           Port_2_1           Port_2_2           Port_2_3           Port_2_4           Port_2_5           Port_2_8		Port_1_7	
Opt         Cancel           Port_2_2         Port_2_3           Port_2_4         Port_2_4           Port_2_5         Port_2_8		Port_2_0	$\geq$
Port_2_3           Port_2_4           Port_2_5           Port_2_6	_	Port_2_1	$\geq$
DK         Cancel         Port_2_4           Port_2_5         Port_2_6	_	Port_2_2	
<u> </u>	_	Port_2_3	$\geq$
UN Lancel Port_2_6	_	Port_2_4	$\geq$
Port_2_6	OK Cancel	Port_2_5	
Port_2_7		Port_2_6	>
		Port_2_7	>

- 19. Click **OK** to continue.
- 20. In the User Modules window expand the Misc Digital folder. In this folder, right-click the LED and select Place; this adds the UM to the project. This UM does not use digital or analog blocks. It appears in Workspace Explorer → Example\_My\_First\_PSoC\_Project[CY8C29] → Example\_My\_First\_PSoC\_Project[Chip] → Loadable Configurations → example\_my\_first\_psoc\_project - 2 User Modules.

Figure 3-60. Workspace Explorer



21.Double-click the LED\_1 UM and navigate to the **Properties** window. Configure the LED for **Port\_1\_7.** 

#### Figure 3-61. Properties Window

Name	LED_1	
User Module	LED	
Version	1.2	
Port	Port_1	
Pin	Port_1_7	
Drive	Active High	



22. Configure the Global Resources window to match the following figure.

Figure 3-62.	Global	Resources	Window
--------------	--------	-----------	--------

obal Resources - exa	mple_my_first_psoc_project 📃 🕨	×
Power Setting [Vcc / S	5.0V / 24MHz	•
CPU_Clock	SysClk/2	
32K_Select	Internal	
PLL_Mode	Disable	
Sleep_Timer	512_Hz	
VC1= SysClk/N	16	
VC2= VC1/N	16	
VC3 Source	VC2	
VC3 Divider	256	
SysClk Source	Internal	
SysClk*2 Disable	Yes	
Analog Power	SC Off/Ref Low	
Ref Mux	(Vdd/2)+/-(Vdd/2)	
AGndBypass	Disable	
Op-Amp Bias	Low	
A_Buff_Power	Low	
SwitchModePump	OFF	
Trip Voltage [LVD (SMF	4.81V (5.00V)	
LVDThrottleBack	Disable	
Watchdog Enable	Disable	

23.Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C29\_main\_Ex1.c* file, which is available within the attachments feature of this PDF document.

Figure 3-63. Workspace Explorer



- 24. Save the project.
- 25. To build the project, click Build  $\rightarrow$  Generate/Build 'Example\_My\_First\_PSoC\_Project' Project.
- 26. Disconnect power to the board.
- 27. Configure the DVK board SW3 to 5 V.
- 28. Configure the DVK breadboard using the included jumper wires:

P0[7] to LED1

P1[7] to LED2

- 29. Reapply power to the board.
- 30.Use PSoC Designer as described in Programming My First PSoC 1 Project on page 27 to program the device.
- 31. Reset the DVK, and observe the blinking LEDs.
- 32. Save and close the project.



## 3.2.1.2 main.c

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded *CY8C29\_main\_Ex1.c* file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
* File Name: main.c
* Description:
* This file provides source code for My First PSoC Project example. The
* firmware blinks one LED at about 3.6 Hz with a PWM, and blinks another LED
*
 with a software timing loop.
*
   PWM Settings:
*
    Input Clock = VC3 //VC3 = 24 MHz/16/16/256 = 366.2 Hz
*
    Enable
                  = Hiqh
    Enable = Hign
CompareOut = ROW_0_Output_3
    TerminalCountOut = None
    Period
             = 100 Output period = (Period+1)*(1/Input Clock) = 101/
366.2 = .275 sec
    or 3.6 Hz
*
    PulseWidth
                = 50
    CompareType
                 = Less Than Or Equal
    InterruptType = Terminal Count
                 = Sync to SysClk
    ClockSync
    InvertEnable
                  = Normal
*****
/
#include <m8c.h> // part specific constants and macros
#include "PSoCAPI.h" // PSoC API definitions for all User Modules
unsigned int i;
              // Variable used for delay
void main(void)
{
   PWM8_1_Start();// Turn on the PWM to blink LED on P0.7
  LED_1_Start();// Enable Software controlled LED
  // The following loop controls the software LED connected to P1.7
  while(1)
  {
    for (i=0;i<60000;i++){} //Length of delay depends on compiler and CPU clock
    LED_1_Invert(); //Switch the state of Software LED, if on turn it off,
     //if off turn it on
  } //End of while(1)
}//End of main
```



## 3.2.2 ADC to LCD Project

This project demonstrates a 9-bit Delta-Sigma ADC by measuring the voltage of the potentiometer center tap wiper and displaying the result on the LCD. Connect the voltage potentiometer (VR) to the ADC input P0[1]. The program reads the 9-bit ADC result and prints it to the LCD.

## 3.2.2.1 Creating ADC to LCD Project

- 1. Follow steps 1 to 10 in the section Creating My First PSoC 1 (CY8C29) Project on page 75; change the project name to **Example\_ADC\_to\_LCD**.
- In the User Modules window, expand the ADCs folder; right-click DelSig and select Place. A window opens with multiple options for the DelSig UM. Scroll down, if necessary, and select the DS1128 configuration. Click OK.



Figure 3-64. Select Multi User Module Window

- 3. Click OK.
- 4. Verify that the **DelSig\_1** UM is placed in **ASC10**.
- 5. In the User Modules window, expand the Amplifiers window. Right-click PGA and select Place. Ensure that the PGA is placed in ACB00.







- 6. In the User Modules window, expand Misc Digital; right-click LCD and select Place.
- 7. Double-click **PGA\_1** and configure the properties to match this figure.

Figure 3-66.	PGA_1	Properties
--------------	-------	------------

Properties - PGA_1	1 ×
Name	PGA_1
User Module	PGA
Version	3.2
Gain	1.000
Input	AnalogColumn_InputMUX_0
Reference	AGND
AnalogBus	Disable

8. Double-click **DelSig\_1** and configure the properties to match this figure.

Figure 3-67. DelSig\_1 Properties

Name	DelSig_1
User Module	DelSig
Version	1.2
DataFormat	Unsigned
Data Clock	VC1
ClockPhase	Normal
PosInput	ACB00
NegInput	ACB00
NegInputGain	Disconnected
PWM Output	None
PulseWidth	1

9. Double-click **LCD\_1** and configure the properties to match this figure.



## Figure 3-68. LCD\_1 Properties

Name	LCD_1	
User Module	LCD	
Version	1.5	
LCDPort	Port_2	
BarGraph	Disable	

10. Configure the **Global Resources** to match the following figure.

Figure 3-69. Global Resources Properties

bal Resources - exa Power Setting [Vcc / S	
CPU_Clock	SysClk/2
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	12
VC2= VC1/N	16
VC3 Source	VC2
VC3 Divider	256
SysClk Source	Internal
SysClk*2 Disable	Yes
Analog Power	SC On/Ref High
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	High
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMF	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

11. Ensure that **AnalogColumn\_InputMUX\_0** is connected to **Port\_0\_1**. If it is not configured for this port, double-click the mux and choose **Port\_0\_1**.





Figure 3-70. AnalogColumn\_InputMUX\_0 is Connected to Port\_0\_1

12.Ensure that AnalogColumn\_Clock\_0, is connected to VC1. If it is not, double-click the mux and chose VC1.





- 13.Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C29\_main\_Ex2.c* file, which is available within the attachments feature of this PDF document.
- 14. Save the project.
- 15. To build the project, click **Build**  $\rightarrow$  **Generate/Build** 'Example\_ADC\_to\_LCD' Project.



- 16.Disconnect power to the board.
- 17. Configure the DVK SW3 to 5 V.

18. Configure the DVK breadboard using the included jumper wires:

P0[1] to VR





- 19. Reapply power to the board.
- 20.Use PSoC Designer as described in Programming My First PSoC 1 Project on page 27 to program the device.
- 21. After programming the device, press the reset button and vary the potentiometer (R20) to see the results on the LCD.

**Note** The ADC output values may not reach full range due to potentiometer and ADC limitations. ADC values may fluctuate several counts due to system noise, and if the potentiometer voltage is at the edge of an ADC count.

22. Save and close the project.



#### 3.2.2.2 main.c

- 1. Open the existing *main.c* file within Workspace Explorer.
- 2. Replace the existing *main.c* content with the content of the embedded *CY8C29\_main\_Ex2.c* file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
* File Name: main.c
* Description:
This file provides source code for the ADC to LCD code example. The
 firmware takes a voltage output from a potentiometer and displays the raw
 counts on an LCD.
*
  PGA Settings: (The PGA buffers the potentiometer voltage on P0.1 into the ADC)
*
*
   Gain
         = 1
*
   Input = AnalogColumn_InputMUX_0 (P0.1)
   Reference = AGND
   AnalogBus = Disable
*
  LCD Settings:
   LCDPort = Port_2
   BarGraph = Disable
DelSig Settings:
*
 The ADC can read full range values from 0-5 V, if the Ref Mux setting is
selected
* as (Vdd/2)+/-(Vdd/2) and Vdd = 5 V. The ADC is configured for a resolution of
9 bits,
 this is achieved by selecting the appropriate configuration when placing the
UM.
*
*
   DataFormat = Unsigned
                // VC1 = 24MHz/12 = 2MHz
   DataClock = VC1
   ClockPhase = Normal
*
   PosInput = ACB00 (PGA_1)
           = ACB00 *Note, this parameter is unused
   NegInput
   NegInputGain = Disconnected
   PWM Output = None
   PulseWidth = 1
                *Note, this parameter is unused
#include <m8c.h>
              // part specific constants and macros
#include "PSoCAPI.h" // PSoC API definitions for all User Modules
unsigned int wADCResult; // Holds the integer ADC result
void main(void)
{
```



```
PGA 1 Start(PGA 1 HIGHPOWER);//Initialize the PGA, PGA used to buffer input
from the VR on P0.1 to the ADC
   DelSig_1_Start(DelSig_1_HIGHPOWER); //Initialize the ADC
   LCD_1_Start(); //Initialize the LCD
   LCD_1_Position(0,0);//Set the LCD to (Row=0,Column=0)
   LCD_1_PrCString("V Count: ");
   DelSig_1_StartAD();//Start gathering conversions from the ADC
   M8C_EnableGInt; //Enable Global interrupts
   //This loop waits for a valid ADC result, and then displays it on the LCD
   while (1)
   {
      while (!(DelSig_1_fIsDataAvailable()));//Wait for ADC data to be ready
      wADCResult=DelSig_1_wGetDataClearFlag();//Store result from ADC
      LCD_1_Position(0,9);
                            //Set LCD to (Row=0,Column=9)
      LCD_1_PrHexInt(wADCResult);//Print ADC result on LCD
   }
```

```
}
```



## 3.2.3 ADC to LCD with DAC and UART

This project demonstrates sine wave generation by using a 6-bit DAC. The sine wave period is based on the current value of the ADC. The firmware reads the voltage output by the DVK board potentiometer and displays the raw counts on the DVK board character LCD display similar to those shown in the previous project. A 6-bit DAC outputs a table generated sine wave at a frequency proportional to the ADC count. The frequency is in the approximate range of 15 Hz to 350 Hz and outputs to port to observe on scope. A 38400 Baud UART outputs the current ADC count as ASCII formatted into a hexadecimal number.

## 3.2.3.1 Creating ADC to LCD with DAC and UART Project

- 1. Follow steps 1 to 10 in the section Creating My First PSoC 1 (CY8C29) Project on page 75; change the project name to Example\_ADC\_to\_LCD\_with\_DAC\_and\_UART.
- 2. In the **User Modules** window, expand the **ADCs** folder; right-click **DelSig** and select **Place**. A window opens with multiple options for the DelSig UM. Scroll down, if necessary, and select the **DS232** configuration. Click **OK**.



Figure 3-73. Select Multi User Module Window

3. Click **OK**.



- 4. Verify that the UM is placed in **ASC10**.
- 5. In the **User Modules** window, expand the **Amplifiers** window. Right-click **PGA** and select **Place**. Ensure that the **PGA** is placed in **ACB00**.



- 6. In the User Modules window, expand Misc Digital, right-click LCD, and select Place.
- 7. In the **User Modules** window, expand **Counters**, right-click **Counter8**, and select **Place**. Complete this step twice to place two **Counter8**s.
- 8. In the User Modules window, expand Digital Comm, right-click TX8, and select Place.
- 9. In the User Modules window, expand DACs, right-click DAC6, and select Place.
- 10. Move the UMs so that they match the configuration shown in Figure 3-75 on page 92.



Figure 3-75. Configure User Modules





11. Double-click **DelSig\_1** and configure it to match this figure.

Figure 3-76. DelSig\_1 Properties

operties - DelSig_1	
Name	DelSig_1
User Module	DelSig
Version	1.2
DataFormat	Unsigned
Data Clock	VC2
ClockPhase	Normal
PosInput	ACB00
NegInput	ACB00
NegInputGain	Disconnected
PWM Output	None
PulseWidth	1

12.Double-click **PGA\_1** and configure it to match this figure.

Figure 3-77. PGA\_1 Properties

Name	PGA_1
User Module	PGA
Version	3.2
Gain	1.000
Input	AnalogColumn_InputMUX_0
Reference	AGND
AnalogBus	Disable

13. Double-click **DAC6\_1** and configure it to match this figure.

## Figure 3-78. DAC6\_1 Properties

operties - DAC6_1	
Name	DAC6_1
User Module	DAC6
Version	4.3
AnalogBus	AnalogOutBus_1
ClockPhase	Normal
DataFormat	OffsetBinary

14. Double-click **LCD\_1** and configure it to match this figure.

## Figure 3-79. LCD\_1 Properties

roperties - LCD_1	
Name	LCD_1
User Module	LCD
Version	1.5
LCDPort	Port_2
BarGraph	Enable



## 15. Double-click **Counter8\_1** and configure it to match this figure.

Figure 3-80. Counter8\_1 Properties

operties - Counter8_1		
Name	Counter8_1	
User Module	Counter8	
Version	2.5	
Clock	VC2	
ClockSync	Sync to SysClk	
Enable	High	
CompareOut	None	
TerminalCountOut	None	
Period	255	
CompareValue	0	
CompareType	Less Than Or Equal	
InterruptType	Terminal Count	
InvertEnable	Normal	

16.Double-click **Counter8\_2** and configure it to match this figure.

Figure 3-81.	Counter8	2 Properties
riguic o or.	Obunitoro_	

operties - Counter8_2		
Name	Counter8_2	
User Module	Counter8	
Version	2.5	
Clock	VC3	
ClockSync	Sync to SysClk	
Enable	High	
CompareOut	None	
TerminalCountOut	Row_2_Output_1	
Period	38	
CompareValue	0	
CompareType	Less Than Or Equal	
InterruptType	Terminal Count	
InvertEnable	Normal	

17. Double-click **TX8\_1** and configure it to match this figure.

Figure 3-82. TX8\_1 Properties

roperties - TX8_1	×
Name	T×8_1
User Module	TX8
Version	3.3
Clock	Row_2_Output_1
Output	Row_2_Output_0
TX Interrupt Mode	TXComplete
ClockSync	Sync to SysClk
Data Clock Out	None



18. Double-click **RO2[0] LUT**, enable **Row\_2\_Output\_0\_Drive\_1** to connect **GlobalOutEven\_4**. Figure 3-83. Digital Interconnect Window





19. Double-click **GlobalOutEven\_4**. In the window that appears, configure **Pin** for **Port\_0\_4**. Figure 3-84. Configure Pin for Port\_0\_4

1 I I I I I I I I I I I I I I I I I I I	
BC0	Port_0_3
	Port_0_4
DBB00 DBB01 DCB02 DCB0	
DelSig_1 PWM	Port_0_6
	Port_0_7
PWH Output_	Port_1_0
D. O. M. O.	Pot_12
R00[0] R00[1]	
R00[2]	
R00[2]	
, RI1[0]	Pot_17
RI1[1]	Pot_2_0
Ri1[2]	Pot_2_1
RI1[3]	
BC1	
2 DBB10 DBB11 DCB12 DCB1	
Counter8_1	
True CNTR8	Pot_2,7
CompanyDu	Port_3_0
Printes CantOdPin	Port_0_4
Interconnect	Port 3,2
R01[0]	Port_3_3
R01[1] R01[2]	Pot_3_4
R01[3]	Port_3_5
	Port_3_6 Port_3_7
RI2[0]	Pot_4_0
RI2[1]	Pot_4_1
RI2[2]	OK Cancel Port4.2
RI2[3]	
BC2	Port 4.4
TXE	
DBB20 DBB21 3 DCB22 DCB2	23 Port 4,6
Counter8_2 TX8	
	Ded 5.4
TerrivaCourtOut	Pot.5.2
000/01	Port.5.3
R02[0] R02[1]	
R02[1]	Port.5.5
R02[3]	Port.5.6
• A	

20.Click **OK** to continue.



21.Click AnalogOutBuf\_1 and configure it for Port\_0\_5.

Figure 3-85. Configure AnalogOutBuf\_1



22. Verify that **AnalogColumn\_InputMUX\_0** is connected to **Port\_0\_1**. If it is not configured for this port, double-click the mux and choose **Port\_0\_1**.







23. Verify that AnalogColumn\_Clock\_0 is connected to VC2. If it is not, double-click the mux and chose VC2.





24. Configure **Global Resources** to match the following figure.

lobal Resources - exan	nple_adc_to_lcd_with_dac_an 🗷
Power Setting [Vcc / Sy	5.0V / 24MHz
CPU_Clock	SysClk/2
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	6
VC3 Source	SysClk/1
VC3 Divider	2
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref High
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	High
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage (LVD (SMP)	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

- 25. Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C29\_main\_Ex3.c* file, which is available within the attachments feature of this PDF document.
- 26. Save the project.
- 27. To generate the project, click Build  $\rightarrow$  Generate/Build 'Example\_ADC\_to\_LCD\_with\_DAC\_and\_UART' Project.
- 28.Open your *Counter8\_1INT.asm* file in **Files** → **lib** → **Library Source Files**. Copy the code found in the *Counter8\_1INT.asm* file in PDF attachment.
- 29. Save the project.
- 30. To build the project, click Build → Build 'Example\_ADC\_to\_LCD\_with\_DAC\_and\_UART' Project.

Note If prompted to reload an out of date file, select Yes.

- 31. Disconnect power to the board.
- 32.Configure the DVK SW3 to 5 V.
- 33. Configure the DVK breadboard using the included jumper wires as follows:

P0[1] to VR

P0[4] to TX

P0[5] to Scope

**Note** An LED (P0[5] to LED1) by nature does not accurately show the changes in frequency; the best way to see this is to use a Scope(P0[5] to Scope).





Figure 3-89. Connect P0[1] to VR, P0[4] to TX, and P0[5] to LED1

- 34.Connect a serial cable to the PC and the DVK board.
- 35.On the DVK board, verify that RS232\_PWR(J10) is jumpered to ON.
- 36. Reapply power to the board.
- 37.Use a terminal application such as TeraTerm or HyperTerminal with these setup parameters. Baud Rate: 38400
  - Data: 8-bit
  - Parity: none
  - Stop: 1 bit
  - Flow Control: none
- 38.Use PSoC Designer as described in Programming My First PSoC 1 Project on page 27 to program the device.
  - After programming the device, press Reset and vary the potentiometer to see the result on the LCD as well as in the terminal application. View the DAC output on a scope or with an LED.
  - **Note** The ADC output values may not reach full range due to potentiometer and ADC limitations. ADC values may fluctuate several counts due to system noise, and if the potentiometer voltage is at the edge of an ADC count.
- 39. Save and close the project.



#### 3.2.3.2 main.c

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded CY8C29\_main\_Ex3.c file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
* File Name: main.c
* Description:
 This file provides source code for the ADC to LCD with DAC and UART example
* project. The firmware takes a voltage output from a potentiometer and
* displays the ADC raw count on an LCD. The raw count is also transmitted
* serially. The raw count also determines the clock divider value of the clock
* driving the DAC update rate.
*
  PGA_1 Settings: (The PGA buffers the potentiometer voltage on P0.1 into the ADC)
*
   Gain
          = 1
*
   Input
          = AnalogColumn InputMUX 0 (P0.1)
   Reference = AGND
   AnalogBus = Disable
*
*
  LCD 1 Settings:
   LCDPort = Port 2
*
   BarGraph = Disable
DelSig 1 Settings:
  The ADC can read full range values from 0-5 V, if the Ref Mux setting is
selected
* as (Vdd/2)+/-(Vdd/2) and Vdd = 5 V. The ADC is configured for a resolution of
8 bits,
*
  this is achieved by selecting the appropriate configuration when placing the
UM.
*
   DataFormat = Unsigned
*
   DataClock = VC2 //VC2 = 24MHz/16/16 = 250kHz
   ClockPhase = Normal
   PosInput
            = ACB00 (PGA 1)
*
   NegInput
            = ACB00
                   *Note this parameter is not used
   NegInputGain = Disconnected
   PWM Output = None
*
   PulseWidth
            = N/A *Note this parameter is not used
Counter8 1 Settings:
  The Counter8_1 controls the update rate of the DAC. The DAC is updated during
ever
```



\* TerminalCount ISR. The frequency of the TerminalCount ISR is determined by the Counter Input Clock divided by the (Period value +1). The Period Value of the \* counter \* is changed by the ADC reading. Thus the frequency of the TerminalCount ISR can range from 125kHz (Period Value=1) to 977Hz (Period Value = 255) \* = VC2 // VC2 = 24MHz/16/16 = 250kHz Clock \* ClockSync = Sync to SysClk Enable = High CompareOut = None \* TerminalCountOut = None = 255 \*Note this parameter is updated in the main loop Period CompareValue = 0 \*Note this parameter is not used CompareType = Less Than or Equal \* \* \* InterruptType = Terminal Count InvertEnable = Normal \* Counter8 2 Settings: \* The Counter8\_1 provides a clock to the TX8 UM to achieved a desired baud rate. \* For this project the desired baud rate is 38400. The TX8 UM derives the baud rate \* by dividing its input clock by 8. Thus the input clock to the TX8 needs to be around 307.2 kHz to achieve a baud rate of 38400. The Counter8\_1 UM provides this clock by dividing VC3 (12MHz) by 39 to get 307.7 kHz. \* \* Clock = VC3 //VC3 = 24MHz/2 = 12MHz = Sync to SysClk \* ClockSync = High Enable CompareOut = None TerminalCountOut = Row\_2\_Output\_1 Period = 38 CompareValue = 0\*Note this parameter is not used CompareType = Less Than or Equal \* \* InterruptType = Terminal Count InvertEnable = Normal TX8\_1 Settings: \* The TX8 UM provides serial communication of the ADC data to another device or PC. The TX8 UM send data out at a baud rate of 38400. This baud rate is derived \* by dividing the UM's input clock by 8. \* = Row\_2\_Output\_1 (From Counter8\_1) \* Clock \* Output = Row\_2\_Output\_0 \* Tx Interrupt Mode = TXComplete ClockSync = Sync to SysClk Data Clock Out = None \* DAC6 Settings:



```
* The DAC6 outputs a sine wave on P0.5. The shape of the sine wave is determined
  by a 64 element lookup table found in SINtable.asm. The update rate of the DAC6
*
   is determined by the Counter8 terminal count ISR. The frequency of the DAC out-
put
* equals the Counter8 Terminal Count frequency divided by 64 (the number of ele-
ments in the table).
*
*
     AnalogBus = AnalogOutBus_1
*
     ClockPhase = Normal
     DataFormat = OffsetBinary
#include <m8c.h>
                       // part specific constants and macros
#include "PSoCAPI.h"
                      // PSoC API definitions for all User Modules
const BYTE SINtable[]=
{
  31, 33, 36, 39, 41, 44, 46, 49, 51, 53, 55, 56, 58, 59, 59,
  60, 60, 60, 59, 59, 58, 56, 55, 53, 51, 49, 47, 44, 42, 39,
  36, 33, 31, 28, 25, 22, 19, 16, 13, 11, 9, 7, 5, 3, 2, 1, 0,
  0, 0, 0, 1, 2, 3, 4, 6, 7, 10, 12, 14, 17, 20, 23, 26, 29
};
BYTE bADCvalue;//Variable for holding ADC result, and updating counter period
void main(void)
{
   Counter8_1_Start();//Enable the counter used for DAC update rate
   Counter8_1_EnableInt();//Enable DAC update interrupt
   Counter8_2_Start();//Enable counter for TX8 clock rate divider
   TX8_1_Start(TX8_1_PARITY_NONE);//Start the TX8 UM with no parity (baud rate =
38400)
   PGA_1_Start(PGA_1_HIGHPOWER);//Enable to PGA to buffer signal from VR to ADC
   DAC6_1_Start(DAC6_1_HIGHPOWER);//Start the DAC
   DelSig_1_Start(DelSig_1_HIGHPOWER);//Start the ADC
   DelSig_1_StartAD();//Start reading values on the ADC
   LCD_1_Start(); //Start the character LCD
   M8C_EnableGInt; // Enable Global Interrupts
   while(1)
   {
        /* Step 1: Get BYTE data from the ADC
           Setp 2: Write BYTE data from ADC to the counter to change the DAC
update rate
           Step 3: Move the LCD cursor back to the beginning and display new ADC
data
           Setp 4: Write ADC data out the TX port, and then send a return
           * /
     if (DelSig_1_fIsDataAvailable())//Is new data available from the ADC?
      {
        bADCvalue = DelSig_1_bGetDataClearFlag(); // Get new data from ADC
        Counter8_1_WritePeriod(bADCvalue); // Update DAC update rate counter
        LCD_1_Position(0,0); // Move LCD (row=0,column=0)
        LCD_1_PrHexByte(bADCvalue); // Print ADC result to LCD
```



```
TX8_1_PutSHexByte(bADCvalue); // Write LCD result out TX8 to PC
TX8_1_PutCRLF(); // Send a return character
}
} //end of while(1)
```

} //End of Main

#### 3.2.3.3 Counter8\_1INT.asm

- 1. Open your *Counter8\_1INT.asm* file in **Files**  $\rightarrow$  **lib**  $\rightarrow$  **Library Source Files**.
- 2. Replace the existing *Counter8\_1INT.asm* content with the content of the embedded file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

```
;; FILENAME: Counter8_1INT.asm
  Version: 2.5, Updated on 2009/3/31 at 12:2:49
;;
;;
  Generated by PSoC Designer 5.0.423.0
;;
;; DESCRIPTION: Counter8 Interrupt Service Routine
;;------
;; Copyright (c) Cypress MicroSystems 2000-2004. All Rights Reserved.
include "m8c.inc"
include "memory.inc"
include "Counter8_1.inc"
; ______
; Global Symbols
;------
export _Counter8_1_ISR
AREA InterruptRAM (RAM, REL, CON)
;@PSoC_UserCode_INIT@ (Do not change this line.)
;_____
; Insert your custom declarations below this banner
;-------
export bTablePos// Stores last table position index
export _bTablePos
;-----
; Includes
;------
```



```
;-----
; Constant Definitions
;-----
;-----
; Variable Allocation
;------
area bss(RAM)
bTablePos:blk 1
bTablePos:
;_____
; Insert your custom declarations above this banner
;-----
;@PSoC_UserCode_END@ (Do not change this line.)
AREA UserModules (ROM, REL)
;------
; FUNCTION NAME: _Counter8_1_ISR
;
; DESCRIPTION: Unless modified, this implements only a null handler stub.
;------
;
Counter8 1 ISR:
  ;@PSoC_UserCode_BODY@ (Do not change this line.)
  ; Insert your custom code below this banner
  NOTE: interrupt service routines must preserve
  ;
   the values of the A and X CPU registers.
  ;
  push A
  push X
  dec [bTablePos] ;Go to the next element in the table
  mov A, [bTablePos]
  jnz SINlookup ; If we are at the end go back to the beginning
  mov [bTablePos], 64
SINlookup:
       _SINtable;Get the value in the SINtable pointed to by [bTablePos]
  index
      DAC6_1_WriteBlind; Write value from SINtable (stored in A) to the DAC
  lcall
  рор Х
  pop A
  ; Insert your custom code above this banner
  ;______
  ;@PSoC_UserCode_END@ (Do not change this line.)
  reti
; end of file Counter8_1INT.asm
```



# 3.3 CY8C38 / CY8C58LP Family Processor Module Code Examples

## 3.3.1 My First PSoC 3 / PSoC 5LP Project

This project demonstrates basic hardware and software functionality with the PSoC 3 or PSoC 5LP device. It flashes two LEDs independently, one using hardware, the other with software. The hardware LED uses a hardware enabled digital port and a PWM to generate a duty cycle and flash the LED. The software LED uses a software enabled digital port and a simple delay in the *main.c* to flash the LED at a known rate.

This code example uses these components:

DigitaL Output Pin (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Digital Output Pin) PWM (Component Catalog  $\rightarrow$  Digital Functions  $\rightarrow$  PWM) Clock (Component Catalog  $\rightarrow$  System  $\rightarrow$  Clock) Logic Low (Component Catalog  $\rightarrow$  Digital  $\rightarrow$  Logic  $\rightarrow$  Logic Low) Logic High (Component Catalog  $\rightarrow$  Digital  $\rightarrow$  Logic  $\rightarrow$  Logic High)

- 3.3.1.1 Creating My First PSoC 3 / PSoC 5LP Project
  - 1. Open PSoC Creator.
  - 2. Create a new project by clicking Create New Project... in the Start Page of PSoC Creator.
  - In the New Project window, select the Empty PSoC3 Design template for a PSoC 3 design, or Empty PSoC5 Design template for a PSoC 5LP design and name the project Ex1\_LED\_with\_PWM.
  - 4. In the **Location** field, type the path where you want to save the project, or click the \_\_\_\_ button and navigate to the appropriate directory.



Figure 3-90. New Project Window

New Project	6			? 🛛
Design	Other			4 ۵
Empty Ten	nplates			<u>^</u>
Emp	ty PSoC 3 Design	Empty PSoC 5 Design	Empty PSoC 5LP Design	
PSoC 3 St	arter Designs			=
	_DMA_VDAC	DelSig_16Channel		
DelS	Sig_12CS		Filter_ADC_VDAC	
Ръ нм	Fan Control with Alert			
PSoC 5 St	arter Designs			
PB ADC	C_DMA_VDAC	DelSig_12CM		~
Creates a PSoC	C 3, 8 bit, design project.			
Name:	Design01			
Location:	C:\Sashi\02_MyWo	k\Q412\Kit-010 LP\Firmware\Ex1_LED_v	with_PWM	
+ Advanced	<u></u>			
			ОК	Cancel

5. By default, the design window opens *TopDesign.cysch*. This is the project's schematic entry file within PSoC Creator.

Figure 3-91. Ex1\_LED\_with\_PWM

Ex1_LED_with_PWM - PSoC Creator 2	2.1	[C:\\Ex1_LED_with_PWM\Ex1_LED_with_PWM.cydsn\TopDesign\TopDesign.cysch]					
Eile Edit <u>V</u> iew <u>D</u> ebug <u>Project</u> <u>Build</u> <u>Iools</u> <u>Window</u> <u>H</u> elp							
👔 🎦 📸 🗃 🛃 🧔 👃 🖄 🗅 🖄 🔸 🖃 🤍 🖕 Debug 🔹 🖕 29% 🔹 🔍 🤤							
遡→送沙賞  荸荠。, Microsoft Sans Serif - 10 - B / U    ■ 書 目 △ - 2 - 3 - , 沙唱唱   4 - 4 - 4 公田市 ,							
Workspace Explorer (1 project) 🚽 🕂	×	Start Page main.c Ex1_LED_with_PWM.cydwr <b>*TopDesign.cysch</b>					
			2 🖬 🖪 🙆 🕅				
Workspace 'Ex1_LED_with_PWM' (1		٦.	離 緯 💠 🔿				
Project 'Ex1_LED_with_PWN TopDesign.cysch	S)		Annotation Concept Cypress 4 b				
Port_LED_with_PWM.cydwr         Pier         Header Files         Imain.c	71	□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Cypress Component Catalog  Cypress Component Catalog  Cypress Component Catalog  Cypress Component Sigma ADC [v1.30]  Cypress Component Sigma ADC [v1.30]  Analog MuX  Analog MuX  Analog MuX  Analog MuX  Component Preview  Datasheet				
	- 1	Output - + + X Notice List Output					
Ready 0 Errors 0 Warnings 0 Notes							



- 3.3.1.2 Placing and Configuring PWM
  - 1. Drag and drop the **PWM** component (**Component Catalog**  $\rightarrow$  **Digital**  $\rightarrow$  **Functions**  $\rightarrow$  **PWM**) toworkspace.
  - 2. Double-click the **PWM\_1** component in the schematic to open the configuration window.
  - 3. Configure the PWM as follows:

Configure Tab Name: PWM\_1 Resolution: 8-Bit PWM Mode: One Output Period: 100 CMP Value 1: 50 CMP Value Type 1: Less or Equal

Figure 3-92. PWM Component Configuration

Configure 'PWM'		? 🛛
Name: <u>PWM_1</u>		
Configure	Advanced Built-in	4 ۵
period 🗕 🕂 100 —	0 #4-100	-0+1
pwm		
Implementation:	Fixed Function     O     UDB	
Resolution:		
PWM Mode:	One Output	~
Period:	100 (Max Period = 1.01s	
CMP Value 1:	50	
CMP Type 1:	Less or Equal	
Dead Band:	Disabled 2	\$
Datasheet	OK Apply Car	ncel

Advanced Tab

Enable Mode: Hardware Only Interrupt On Terminal Count Event: Select


# Figure 3-93. PWM Component Advanced Tab Configuration

Configure 'PWM'		?	X
Name: PWM_1			
Configure Adv	anced Built-in	٩	⊳
Enable Mode:	Hardware Only	~	
Run Mode:	Continuous	~	
Trigger Mode:	None	~	
Kill Mode:	Disabled 🔽 1	*	
Capture Mode:	None	~	
	None Interrupt On Terminal Count Event Interrupt On Compare 1 Event Interrupt On Compare 2 Event Interrupt On Kill Event		
Data Sheet	OK Apply Canc	el	

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

#### 3.3.1.3 Placing and Configuring Digital Output Pin Hardware

- 1. Drag and drop the **Digital Output Pin** component (**Component Catalog** → **Ports and Pins** → **Digital Output Pin**).
- 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
- 3. Configure the digital output pin:
  - Type Tab

Name: LED1

Select Digital Output check box

Select HW Connection check box

Figure 3-94. LED1 Component Configuration

Configure 'cy_pins'			? 🗙
Name: LED1			
Pins Mapping Re	set Built-in		۹ ۵
Number of Pins: 1	K 🖾 🕈 🗣 📓 🐰		
[All Fins]	Type General Input	Output	
└⊠ LED1_0	🔲 Analog	Preview:	
	🔲 Digital Input		
	HW Connection		
	🗹 Digital Output		
	HW Connection		
	🔲 Output Enable		
	Bidirectional		
	Show Annotation Terminal		
	]		
Datasheet	ОК	Apply Canc	el



### General Tab

Drive Mode: Strong Drive

Leave the remaining parameters as default

Figure 3-95. Pins - LED1 Component Configuration

Configure 'cy_pins'		? 🛛
Name: LED1		
Pins Mapping Re	eset Built-in	٩ ۵
Number of Pins: 1	× 🗗 🛊 🕴 🐰	
[All Fins]	Type General Input	Output
└──⊠ LED1_0	Drive Mode	Initial State:
	Strong Drive	🛛 🔽 🖌 Low (0)
		Minimum Supply Voltage:
Datasheet	ОК	Apply Cancel

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

- 3.3.1.4 Placing and Configuring Software Digital Output Pin
  - 1. Drag and drop the **Digital Output Pin** component (**Component Catalog** → **Ports and Pins** → **Digital Output Pin**).
  - 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
  - 3. Configure the digital output pin:

Type Tab

Name: LED2





Configure 'cy_pins'		? 🗙
Name: LED2		
Pins Mapping Re	set Built-in	4 Þ
Number of Pins: 1	< 🗗 🕈 🔰 🕅 💥	
[All Fins]	Type General Input Output	
└──⊠ LED2_0	Analog Preview:	
	Digital Input	
	HW Connection	
	Digital Output	
	HW Connection	
	Output Enable	
	Bidirectional	
	Show Annotation Terminal	
Datasheet	ОК Арріу С.	ancel

#### General Tab

Drive Mode: Strong Drive

Leave the remaining parameters as default

Figure 3-97. Pins - LED2 Component Configuration

Configure 'cy_pins'		? 🛛
Name: LED2		
Pins Mapping Re	set Built-in	4 ۵
Number of Pins: 1		
[All Pins]	Type General Input O	utput
└⊠ LED2_0	Drive Mode	Initial State:
	Strong Drive	🖌 🛛 Low (0) 🛛 🔽
		Minimum Supply Voltage:
Datasheet	ОК Аррі	Cancel

3.3.1.5 Connecting the Components Together

- 1. Using the Wire Tool 🛄, connect **pwm** (in the PWM component) to hardware connection point of LED1.
- 2. Connect a Logic High component (Component Catalog  $\rightarrow$  Digital  $\rightarrow$  Logic  $\rightarrow$  Logic High) to the enable on the PWM.



- 3. Connect a Logic Low component (Component Catalog → Digital → Logic → Logic Low) to the reset on the PWM.
- 4. Connect a Clock component (Component Catalog  $\rightarrow$  System  $\rightarrow$  Clock) to the clock on the PWM.
- 5. Double-click the **Clock\_1** component to configure.
- 6. Configure the clock:

Configure Clock Tab Name: Clock\_1

Source: ILO (1.000 kHz)

Select Divider and set the value as 10

Leave the remaining parameters as default

Figure 3-98. Clock Component Configuration

Configure 'cy	_clock'		? 🗙
Name: Clo	:k_1		
Configur	E Clock Advance	d Built-in	4 ⊳
Clock Type:	<ol> <li>New</li> </ol>	O Existing	
Source:	ILO <i>(1.000 kHz)</i>		*
Specify:	Frequency		
	<ol> <li>Divider</li> </ol>	10 🗢	
Uses Cloc Source Clo Name: ILI Enabled: Frequenc; Accuracy	) Yes ự: 1.000 kHz ∽50, +100	: Yes 'start on reset'. The setting can be changed in the Design '	wide
Datasheet		OK Apply Cance	

7. When complete, the schematic looks similar to Figure 3-99.

Figure 3-99. Connected Components



## 3.3.1.6 Configuring Pins

- 1. From the **Workspace Explorer**, double-click the *Ex1\_LED\_with\_PWM.cydwr* file (see Figure 3-100).
- 2. Click the Pins tab.
- 3. Select pin P1[6] for LED1.
- 4. Select pin P1[7] for LED2.



#### Figure 3-100. Pin Assignments



# 3.3.1.7 Creating main.c File

- 1. Open the existing *main.c* file within Workspace Explorer.
- Replace the existing *main.c* content with the content of the embedded CY8C38\_main\_Ex1.C for PSoC 3 module and CY8C58LP\_main\_Ex1.C for PSoC 5 module file, which is available within the attachments feature of this PDF document.

### Note

To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

- 3. From the Build menu, select Build Ex1\_LED\_with\_PWM.
- 4. PSoC Creator builds the project and displays the comments in the **Output** dialog box. When you see the message "Build Succeeded", the build is complete.
- 3.3.1.8 Configuring and Programming PSoC Development Board
  - 1. Disconnect power to the board.
  - 2. Configure the DVK SW3 to 3.3 V.
  - 3. Configure the following on the PSoC development board's prototyping area using the included jumper wires:
    - P1[6] to LED1
    - P1[7] to LED2
  - 4. Apply power to the board.
  - 5. Use PSoC Creator as described in Programming My First PSoC 3 Project on page 32 or Programming My First PSoC 5LP Project on page 36 to program the device.
  - After programming the device, press the **Reset** button on the PSoC development board. The PWM causes the LED1 to blink at approximately 1 Hz due to PSoC Creator's PWM component and LED2 blinks at a faster rate using a software timing loop to toggle the LED.
  - 7. Save and close the project.



# 3.3.2 ADC to LCD Project

This project demonstrates the Delta-Sigma ADC by measuring the voltage of the potentiometer on the board and displays the result on the character LCD of the PSoC development board.

The ADC is clocked by the internal clock of 3 MHz and the sampling rate is set to 10,000 sps. Connect the voltage potentiometer (labeled "VR" on the PSoC development board) to the ADC input (programmed to P0[7] for this example). The program reads the ADC result and prints it to the LCD.

The instructions that follow assume that you have completed My First PSoC 3 / PSoC 5LP Project and therefore have a basic understanding of the PSoC Creator software environment.

This code example uses these components:

Delta Sigma ADC (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  ADC  $\rightarrow$  Delta Sigma ADC) Character LCD (Component Catalog  $\rightarrow$  Display  $\rightarrow$  Character LCD) Analog Pin (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Analog Pin)

# 3.3.2.1 Creating ADC to LCD Project

- 1. Open PSoC Creator.
- 2. Create a new project by clicking Create New Project... in the Start Page of PSoC Creator.
- In the New Project window, select the Empty PSoC3 Design template for a PSoC 3 design, or Empty PSoC5 Design template for a PSoC 5LP design and name the project Ex2\_ADC\_to\_LCD.
- 4. In the **Location** field, type the path where you want to save the project, or click .... and navigate to the appropriate directory.
- 5. By default, the design window opens *TopDesign.cysch*. This is the project's schematic entry file within PSoC Creator.
- 3.3.2.2 Placing and Configuring Delta Sigma ADC
  - 1. Drag and drop the Delta Sigma ADC component (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  ADC  $\rightarrow$  Delta Sigma ADC).
  - 2. Double-click the ADC\_DelSig\_1 component in the schematic to open the configuration window.
  - 3. Configure the Delta Sigma ADC as follows:

## Configure Tab

Name: ADC\_DelSig\_1

Conversion Mode: Continuous

# Configs: 1

Resolution: 8

Conversion Rate: 10000

Input Range: Vssa to Vdda

Buffer Gain: 1

Reference: Internal Vref

Clock Source: Internal



Configure 'ADC_DelSig'	28
Name: ADC_DelSig_1 Configure Built-in	
Config 1       Common         Comment :       Default Config         Config Name :       CFG1         ADC_DelSig_1_CFG1         Sampling         Conversion Mode       2 · Continuous         # Configs       1 ©         Resolution       8 <	
Input Options Input Mode O Differential O Single Input Range Vssa to Vdda Buffer Gain 1 V Buffer Mode Rail to Rail	
Reference Vref Internal Vdda/4	
Datasheet	OK Apply Cancel

### Figure 3-101. ADC Component Configuration

- 3.3.2.3 Placing and Configuring an Analog Pin
  - 1. Drag and drop the analog pin component (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Analog Pin).
  - 2. Double-click on the **Pin\_1** component in the schematic to open the configuration window.
  - 3. Configure the analog pin as follows:

Type Tab

Name: POT

Select Analog check box only





Configure 'cy_pins'		? 🛛
Name: POT Pins Mapping R4	eset Built-in	4 Þ
Number of Pins: 1	X 🛛 🕇 🕈 🗮 🐰	
[All Rins] └──⊠ POT_O		Output  review:
Datasheet	ОК Ар	pply Cancel

### General Tab

Drive Mode: High Impedance Analog

Leave the remaining parameters as default

Figure 3-103. Select High Impedance Analog Drive Mode

Configure 'cy_pins'		? 🔀
Name: POT		
	set Built-in	4 ⊳
Number of Pins: 1		
[All Fins]	Type General Input Ou	Itput
⊠ POT_0	Drive Mode	Initial State:
	High Impedance Analog 🛛 🗸	🖌 Low (0) 🔽
	25 - 21 25 - 2	Minimum Supply Voltage:
Datasheet	ОК Арру	Cancel

# 3.3.2.4 Placing and Configuring Character LCD

- 1. Drag and drop the character LCD component (Component Catalog  $\rightarrow$  Display  $\rightarrow$  Character LCD)
- 2. Double-click the LCD\_Char\_1 component in the schematic to open the configuration window.
- 3. Configure the character LCD:

Name: LCD\_Char\_1 LCD Custom Character Set: None Include ASCII to Number Conversion Routines: check box



For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Figure 3-104. Configure LCD\_Char\_1

lame: LCD_Char_1		
General Built-in		4 ۵
Parameters	Custom Character Editor	
LCD Custom Character Set None		
🔿 Vertical Bargraph		
🔿 Horizontal Bargraph		
O User Defined		
✓ Include Number to ASCII Conversion Routines		

# 3.3.2.5 Connecting the Components Together

- 1. Using the Wire Tool 1, connect **POT** to **ADC\_DelSig (ADC\_DelSig\_1)**.
- 2. When complete, the schematic looks similar to Figure 3-105.

Figure 3-105. Connected Components



# 3.3.2.6 Configuring Pins

- 1. From the **Workspace Explorer**, double-click the *Ex2\_ADC\_to\_LCD.cydwr* file.
- 2. Click the Pins tab.
- 3. Select pins P2[6:0] for LCD\_Char\_1.
- 4. Select pin P0[7] for POT.



#### Figure 3-106. Pins Assignments

	ΨX	Start Page Ex2_ADC_to_LCD.cydwr TopDesign.cysch					
· · · · · · · · · · · · · · · · · · ·	-	5 5 5 5 5 5	Alias	s Name /	Port		Pin
Workspace 'Ex2_ADC_to_LCD' (1 Projects)	$\mathbf{n}$	4007	11	, Mano	- un		
🗄 🔁 Project 'Ex2_ADC_to_LCD' [CY8C5868AX	(ي 10	10 more 11 more 12 more		\LCD_Char_l:LCDPort[6:0])	P2[6:0] Trace, Trace, Trace, Trace	~	9599,12
TopDesign.cysch	ŝ					-	
Ex2_ADC_to_LCD.cydwr	Ŕ			POT	PO[7] IDAC:HI	~	79
🖃 🧀 Header Files	$\square$	linku:					
h device.h	Components						
	- DQ						
🖻 🧰 Source Files	ne	LCD_Cher_1LCDPerpt 1 929 Inten 19860 00 1 LCD_Cher_1LCDPerpt 2 929 Inten 0 photo-	~				
main.c	nts –	2 P2(1) Times Crysteps P2(2)					
		P1201 05005L CpVreprint P001 12     P1202 0505A CpVreprint P001 17					
	2						
	tas	1 May					
	Datasheets						
	8	wab wat 🦛					
	20		~				
	Results						
	lts.						
		17 PQI 01 02					
		19 POD 00 00 00 00 00 00 00 00 00 00 00 00 00					
		COLLO 21 PHU SHOCK JTACTCK XTAL JOHN MISS					
		Image: Section 2011         Sectio					
		PIR I Monitor 🛔 🛔 Coverant PAR 🖬					
		1000 S S S S S S S S S S S S S S S S S S					
		2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2					
			<				
< >	⊼ ⊽	📝 Pins 🕅 Analog 🕒 Clocks 💉 Interrupts 👫 DMA 👳 System 📓	Directive	es 📄 Flash Security			

# 3.3.2.7 Creating main.c File

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- Replace the existing *main.c* content with the content of the embedded CY8C38\_main\_Ex2.C for PSoC 3 module and CY8C58LP\_main\_Ex2.C for PSoC 5 module file, which is available within the attachments feature of this PDF document.

### Note

To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

3. From the **Build** menu, select **Build Ex2\_ADC\_to\_LCD**. PSoC Creator builds the project and displays the comments in the **Output** dialog box. When you see the message "Build Succeeded", the build is complete.

# 3.3.2.8 Configuring and Programming the PSoC Development Board

- 1. Disconnect power to the board.
- 2. Configure the DVK SW3 to 3.3 V.



- Using the jumper wires included, configure the PSoC development board's prototyping. P0[7] to VR
- 4. Verify that VR\_PWR (J11) is jumpered to **ON**.
- 5. Apply power to the board.
- 6. Use PSoC Creator as described in Programming My First PSoC 3 Project on page 32 or Programming My First PSoC 5LP Project on page 36 to program the device.
- 7. After programming the device, press the **Reset** button on the PSoC development board to see the output of the ADC displayed on the LCD. Turning the potentiometer results in the LCD value changing.



**Note** The ADC output values may not reach full range due to potentiometer and ADC limitations. ADC values may fluctuate several counts due to system noise, and if the potentiometer voltage is at the edge of an ADC count.

8. Save and close the project.



# 3.3.3 ADC to UART with DAC

This project demonstrates sine wave generation by using an 8-bit DAC and DMA. The sine wave period is based on the current value of the ADC value of the potentiometer.

The firmware reads the voltage output by the DVK board potentiometer and displays the raw counts on the DVK board character LCD display similar to that shown in the previous project. An 8-bit DAC outputs a table generated sine wave to an LED using DMA at a frequency proportional to the ADC count. A 9600 Baud 8N1 UART outputs the current ADC count as ASCII formatted into a hexadecimal number.

The following instructions assume that you have completed My First PSoC Project and ADC to LCD Project and therefore have a basic understanding of the PSoC Creator software environment.

This code example uses the following components:

Delta Sigma ADC (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  ADC  $\rightarrow$  Delta Sigma ADC) Voltage DAC (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  DAC  $\rightarrow$  Voltage DAC) Opamp (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  Amplifiers  $\rightarrow$  Opamp) DMA (Component Catalog  $\rightarrow$  System  $\rightarrow$  DMA) Character LCD (Component Catalog  $\rightarrow$  Display  $\rightarrow$  Character LCD) UART (Component Catalog  $\rightarrow$  Communications  $\rightarrow$  UART) Analog Pin (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Analog Pin) Digital Output Pin (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Digital Output Pin) Clock (Component Catalog  $\rightarrow$  System  $\rightarrow$  Clock) Logic Low (Component Catalog  $\rightarrow$  Digital  $\rightarrow$  Logic  $\rightarrow$  Logic Low)

# 3.3.3.1 Creating ADC to UART with DAC Project

- 1. Open PSoC Creator.
- 2. Create a new project by clicking Create New Project... in the Start Page of PSoC Creator.
- In the New Project window, select the Empty PSoC3 Design template for a PSoC 3 design, or Empty PSoC5 Design template for a PSoC 5LP design and name the project Ex3\_ADC\_to\_UART\_with\_DAC.
- 4. In the **Location** field, type the path where you want to save the project, or click .... and navigate to the appropriate directory.
- 5. By default, the design window opens *TopDesign.cysch*. This is the project's schematic entry file within PSoC Creator.

## 3.3.3.2 Configuring Clock for ADC to UART with DAC Project

1. Open the Ex3\_ADC\_to\_UART\_with\_DAC.cydwr file from Workspace Explorer. See figure below

Figure 3-107. Workspace Explorer



2. Select the Clocks tab and click on Edit Clock....

### Figure 3-108. Edit Clock

1	Start Page       TopDesign.cysch       Ex3_ADC_tDAC.cydwr										
Туре 🛆	Name	Domain		Desired Frequenc		Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	T
System	USB_CLK	DIGITAL		48.000	MHz	? MH	z ±0	-	1		IMOx2
System	Digital Signal	DIGITAL		2	MHz	? MH	z ±0	-	0		
System	XTAL 32kHz	DIGITAL		32.768	kHz	? MDH	z ±0	-	0		
System	ILO	DIGITAL		2	MHz	1.000 kH	z -50, +100	-	0		
System	XTAL	DIGITAL		24.000	MHz	24.000 MH	z ±6.001	-	0	<b>V</b>	
System	IMO	DIGITAL		24.000	MHz	24.000 MH	z ±6.001	-	0		XTAL
System	BUS_CLK (CPU)	DIGITAL		2	MHz	33.000 MH	z ±6.001	-	1	<b>V</b>	MASTER_CLK
System	MASTER_CLK	DIGITAL		2	MHz	33.000 MH	z ±6.001	-	l	<b>V</b>	PLL_OUT
System	PLL_OUT	DIGITAL		33.000	MHz	33.000 MH	z ±6.001	-	0		IMO
Local	Clock_2	DIGITAL	*	76.900	kHz	76.923 kH	z ±6.001	-	312	<b>v</b>	XTAL
Local	ADC_DelSig_1_theACLK	ANALOG	*	160.000	kHz	160.194 kH	z ±6.001	-	206	<b>v</b>	MASTER_CLK
Local	ADC_DelSig_1_Ext_CP_Clk	DIGITAL	*	1.000	MHz	1.000 MH	z ±6.001	-	33	<b>v</b>	MASTER_CLK
Local	Clock_1	DIGITAL	*	3.000	MHz	3.000 MH	z ±6.001	-	8	<b>~</b>	IMO
	Pins M Analog Clocks I Interrupts DMA V System Directives      Flash Security										

3. In the 'Configure System Clocks' window, enable and configure the XTAL to **24 MHz** with accuracy as **± 6%**.

Figure 3-109. Configure XTAL

XTAL Configuration		? 🗙
Freq: 24 MHz V Accuracy - 6 + 6 % V	Enable fault	natic gain control recovery reference levels
	Feedback:	3 🖌
	Watchdog:	3
	ОК	Cancel

- 4. Set the IMO source to XTAL.
- 5. Select PLL source to IMO (24.000 MHz) and the desired output value to 33 MHz.
- 6. Select PLL\_OUT (33.000 MHz) as the source clock to Master Clock.
- 7. Set ILO to KHz and select OK.





Figure 3-110. Configure System Clocks

Now, go back to TopDesign.cysch to place and connect the components required for the project.

## 3.3.3.3 Placing and Configuring Delta Sigma ADC

- 1. Drag and drop the Delta Sigma ADC component (Component Catalog → Analog → ADC → Delta Sigma ADC)
- 2. Double-click the **ADC\_DelSig\_1** component in the schematic to open the configuration window.
- 3. Configure the Delta Sigma ADC as follows:

#### Configure Tab

Name: ADC\_DelSig\_1 Conversion Mode: Continuous # Configs: 1 Resolution: 8 Conversion Rate: 10000 Input Range: Vssa to Vdda Buffer Gain: 1 Reference: Internal Vref Clock Source: Internal



Configure 'ADC_DelSig'       ?         Name:       ADC_DelSig.1         Configure Bult-in       4         Config 1 Common       4         Config 1 Common       4         Config Name :       CFG1         ADC_DelSig_1_CFG1       Sampling         Conversion Mode       2 · Continuous         # Configs 1       # Configs 1         Resolution       8       bits         Conversion Rate       10000       \$ SPS Range [ 8000 · 137500 SPS ]	
Configure       Built-in       4         Config 1       Common         Comment :       Default Config         Config Name :       CFG1         ADC_DelSig_1_CFG1         Sampling         Conversion Mode       2 - Continuous         # Configs       1          Resolution       8          bits	2 🔀
Configure       Built-in       4         Config 1       Common         Comment :       Default Config         Config Name :       CFG1         ADC_DelSig_1_CFG1         Sampling         Conversion Mode       2 - Continuous         # Configs       1          Resolution       8          bits	
Config 1       Common         Comment :       Default Config         Config Name :       CFG1         ADC_DelSig_1_CFG1         Sampling         Conversion Mode       2 · Continuous         W       # Configs         Resolution       8         bits	٩ ٧
Comment :       Default Config         Config Name :       CFG1         ADC_DelSig_1_CFG1         Sampling         Conversion Mode       2 · Continuous         # Configs       1 >         Resolution       8        bits	
Config Name :     CFG1     ADC_DelSig_1_CFG1       Sampling	
Sampling Conversion Mode 2 · Continuous V # Configs 1 2 Resolution 8 V bits	
Conversion Mode     2 - Continuous     Image: Configs       Resolution     8     bits	
Resolution 8 bits	
Conversion Rate 10000 🗢 SPS Range [ 8000 - 137500 SPS ]	
Clock Frequency 160.000 kHz	
Input Options	
Input Mode O Differential O Single	
Input Range Vssa to Vdda	
Buffer Gain 1 🗸 Buffer Mode Rail to Rail	
Reference	
Vref Internal V/dda/4	
Datasheet OK Apply Cancel	OK Apply Cancel

Figure 3-111. Delta Sigma ADC Component Configuration

- 3.3.3.4 Placing and Configuring an Analog Pin
  - 1. Drag and drop the Analog Pin component (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Analog Pin)
  - 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
  - 3. Configure the analog pin:

Type Tab

Name: POT

Select Analog check box only







### General Tab

Drive Mode: High Impedance Analog

Leave the remaining parameters as default

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

## 3.3.3.5 Placing and Configuring Character LCD

- 1. Drag and drop the character LCD component (Component Catalog  $\rightarrow$  Display  $\rightarrow$  Character LCD)
- 2. Double-click the LCD\_Char\_1 component in the schematic to open the configuration window.
- 3. Configure the character LCD:

Name: LCD\_Char\_1

LCD Custom Character Set: None

#### Include ASCII to Number Conversion Routines: check box



General Built-in	4
Parameters	Custom Character Editor
LCD Custom Character Set	
O Vertical Bargraph	
🔘 Horizontal Bargraph	
O User Defined	
Include Number to ASCII Conversion Routines	

Figure 3-113. Character LCD Component Configuration

- 3.3.3.6 Placing and Configuring Voltage DAC
  - Drag and drop the Voltage DAC component (Component Catalog → Analog → DAC → Voltage DAC)
  - 2. Double-click the **VDAC8\_1** component in the schematic to open the configuration window.
  - 3. Configure the VDAC:
    - Basic Tab

Name: VDAC8\_1

Data\_Source: CPU or DMA (Data Bus)

Strobe\_Mode: Register Write

VDAC\_Range: 0 - 4.080V (16mV/bit)

VDAC\_Speed: Slow Speed

Value\_mV: 1600

#### Value\_8 bit hex: 64



Figure 3-114. Voltage DAC Component Configuration

Configure 'VDAC8'		? 🛛
Name: VDAC8_1 Configure Built-in		4 Þ
VDAC Range ○ 0 - 1.020 V (4 mV / bit) ⊙ 0 - 4.080 V (16 mV / bit)	<ul> <li>Speed</li> <li>Slow Speed</li> <li>→ High Speed</li> </ul>	
Value mV: 1600 8 bit Hex: 64	Data Source O DAC Bus O CPU or DMA (Data Bus)	
Note: Changing any value field recalculates the other	Strobe Mode O External O Register Write	
Datasheet	ОК Арріу	Cancel

# 3.3.3.7 Placing and Configuring Opamp

- 1. Drag and drop the Opamp component (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  Amplifiers  $\rightarrow$  Opamp)
- 2. Double-click the **Opamp\_1** component in the schematic to open the configuration window.
- 3. Configure the Opamp:

Basic Tab

Name: Opamp\_1

Mode: Follower

Power: High Power





Configure 'Op	Amp'	? 🗙
Name: Opa	mp_1	
Configure	Built-in	4 ۵
Mode	Follower	~
Power	High Power	~
Data Sheet	OK Apply	Cancel
Data Sheet	ОК Арріу	Cancel

- 3.3.3.8 Placing and Configuring Analog Pin
  - 1. Drag and drop the analog pin component (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Analog Pin)
  - 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
  - 3. Configure the analog pin:

Type Tab

Name: LED

Select Analog check box only

Figure 3-116. LED Component Configuration

Configure 'cy_pins'		? 🛛
Name:		
	eset Built-in	4 Þ
Number of Pins: 1	× 🗗 🕈 🕴 🐰	
[All Pins]	Type General	Input Output
⊠ LED_0	🗹 Analog	Preview:
	🔲 Digital Input	
	✓ HW Connection	
	🔲 Digital Output	
	✓ HW Connection	
	🗌 Output Enable	
	Bidirectional	
	Show Annotation Term	ninal
Datasheet	ОК	Apply Cancel

# General Tab

**Drive Mode**: High Impedance Analog Leave the remaining parameters as default



For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Figure 3-117. LED Component Configuration



# 3.3.3.9 Placing and Configuring UART

- 1. Drag and drop the UART component (Component Catalog  $\rightarrow$  Communications  $\rightarrow$  UART)
- 2. Double-click the **UART\_1** component in the schematic to open the configuration window.
- 3. Configure the UART:
  - Advanced Tab

Clock: External

Figure 3-118. Advanced Tab

Configure 'UART'		? 🛛
Name: UART_1		
Configure Advanced Built-in Clock Selection:		4 Þ
<ul> <li>Internal Clock</li> <li>External Clock</li> </ul>	UNKNOWN SOURCE FREQ	
Interrupts RX - On Byte Received	TX - On TX Complete	
RX - On Parity Error	TX - On FIFO Empty	
RX - On Stop Error	🔲 TX - On FIFO Full	
RX - On Break	🔲 TX - On FIFO Not Full	_
RX - On Overrun Error		
RX -On Address Match		
RX - On Address Detect		
RX Address Configuration	Buffer Sizes:	_ ⊻
Datasheet OK	Apply Can	cel

Configure Tab Name: UART\_1 Mode: TxOnly Leave the remaining parameters to default

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Configu	ire 'UART'		?	X
Name:	UART_1			
-	onfigure Advance	ed Built-in	٩	⊳
Mod	e: ) Full UART (TX + F	XX) 🔿 RX Only		
C	) Half Duplex	<ul> <li>TX Only</li> </ul>		
	Bits per second:	1/8 Input Clock Frequency		
	Data bits:	8	·	
	Parity Type:	None	•	
		API control enabled		
	Stop bits:	1	·	
	Flow Control:	None	·	
Da	atasheet	OK Apply Cano	el	

Figure 3-119. UART Component Configuration

3.3.3.10 Placing and Configuring Digital Output Pin

- 1. Drag and drop the **Digital Output Pin** component (**Component Catalog** → **Ports and Pins** → **Digital Output Pin**)
- 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
- 3. Configure the digital output pin:

Type Tab

Name: TX\_OUT

Select HW Connection check box

Figure 3-120. TX\_OUT Component Configuration

Configure 'cy_pins'		? 🛛
Name: TX_OUT		
	set Built-in	4 Þ
Number of Pins: 1		
[All Pins]	Type General Input	Output
⊠ TX_OUT_0	🗌 Analog	Preview:
	🔲 Digital Input	
	HW Connection	
	🗹 Digital Output	
	✓ HW Connection	
	🔲 Output Enable	
	Bidirectional	
	Show Annotation Terminal	
Datasheet	ОК	Apply Cancel



### General Tab

Under Drive Mode: Strong Drive

Leave the remaining parameters as default

Figure 3-121. Pins - TX\_OUT Component Configuration

Configure 'cy_pins'		? 🛛
Name: TX_OUT		
Pins Mapping Re	set Built-in	4 ۵
Number of Pins: 1		
[All Pins]	Type General Input	Output
└──⊠ TX_OUT_O	Drive Mode	Initial State:
	Strong Drive	🖌 Low (0)
		Minimum Supply Voltage:
Datasheet	ОК Ар	Cancel

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

# 3.3.3.11 Placing and Configuring Clock for UART

- 1. Connect a clock component (**Component Catalog**  $\rightarrow$  **System**  $\rightarrow$  **Clock**) to the UART clock.
- 2. Double-click the **Clock** component.
- 3. Configure the clock:

Name: Clock\_2

Source: XTAL (24.000 MHz)

Desired Frequency: 76.9 kHz

Leave the remaining parameters at default.

**Note** The desired frequency of the clock is 76.9 kHz because this value should be eight times the required baud rate: 76900/8 = 9612.5, which is approximately 9600 (required baud rate).



# Figure 3-122. Configure UART Clock

Configure 'cy_clock'	? 🗙
Name: Clock_2	
Configure Clock Advanced Built-in	4 Þ
Clock Type: 💿 New 🔘 Existing	
Source: XTAL (24.000 MHz)	~
Specify: 💿 Frequency 76.9 kHz 💌	
O Divider	
Summary API Generated: Yes Uses Clock Tree Resource: Yes Source Clock Info Name: XTAL Enabled: Yes Frequency: 24,000 MHz Accuracy: ±6.001	
Datasheet OK Apply Car	ncel

Figure 3-123. UART Configure Window After Assigning Clock Source - Configure Tab

Configure 'UART'		? 🗙
Name: UART_1		
Configure Advar	Built-in	٩ ۵
O Full UART (TX +	RX) 🔘 RX Only	
🚫 Half Duplex	<ul> <li>TX Only</li> </ul>	
Bits per second:	1/8 Input Clock Frequency (9.615385 KBaud)	
Data bits:	8	/
Parity Type:	None	/
	API control enabled	
Stop bits:	1	*
Flow Control:	None	/
Datasheet	OK Apply Can	cel



Figure 3-124. UART Configure Window After Assigning Clock Source - Advanced Tab

Configure 'UART'		?
Name: UART_1		
Configure Advanced Built-in		۹ ۵
Clock Selection: O Internal Clock	SOURCE FREQ = 76.923 kHz	7
Interrupts		
RX - On Byte Received	📃 TX - On TX Complete	
RX - On Parity Error	🔲 TX - On FIFO Empty	
RX - On Stop Error	🔲 TX - On FIFO Full	
RX - On Break	🔲 TX - On FIFO Not Full	
RX - On Overrun Error		
RX -On Address Match		
RX - On Address Detect		
RX Address Configuration	Buffer Sizes:	-
Datasheet OK	Apply Can	el

# 3.3.3.12 Placing and Configuring DMA

- 1. Drag and drop the DMA component (**Component Catalog**  $\rightarrow$  **System**  $\rightarrow$  **DMA**)
- 2. Double-click the **DMA\_1** component in the schematic to open the configuration window.
- 3. Configure the DMA:

#### Basic Tab

Name: DMA\_1

Hardware Request: Rising Edge

Leave the remaining parameters as default

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Figure 3-125. DMA Component Configuration

Basic Built-in	1	4
Hardware Request:	Rising Edge	
Hardware Termination:	Disabled	



# 3.3.3.13 Connecting the Components Together

- 1. Connect a Logic Low component (Component Catalog  $\rightarrow$  Digital  $\rightarrow$  Logic  $\rightarrow$  Logic Low) to the reset of the UART
- 2. Connect a Clock component (**Component Catalog**  $\rightarrow$  **System**  $\rightarrow$  **Clock**) to the **drq** of the DMA.
- 3. Double-click the **Clock** component to configure.
- 4. Configure the clock:

Configure Clock Tab Name: Clock\_1 Source: IMO (24.000 MHz) Desired Frequency: 3 MHz Leave the remaining parameters set to their default values

Figure 3-126. Clock Component Configuration

Configure 'cy_clock'	? 🗙
Name: Clock_1	
Configure Clock Advanced Built-in	4 Þ
Clock Type: 💿 New 🔿 Existing	
Source: IMO (24.000 MHz)	*
Specify: 📀 Frequency 3 MHz 💌	
O Divider	
Summary API Generated: Yes Uses Clock Tree Resource: Yes Source Clock Info Name: IMO Enabled: Yes Frequency: 24.000 MHz Accuracy: ±6.001	<
Datasheet OK Apply Cancel	<b>.</b>

- 5. Using the Wire Tool 📜, connect **tx** (in the UART component) to HW connection of the TX\_OUT digital output pin (TX\_OUT).
- 6. Using the Wire Tool 1, connect VDAC8 (VDAC8\_1) to Opamp (Opamp\_1).
- 7. Using the Wire Tool 1, connect **POT** to **ADC\_DelSig (ADC\_DelSig\_1)**.
- 8. Right-click the LED analog pin, select the **Shape** menu option and then **Flip Horizontal**. This allows the LED pin to line up with the Opamp output.
- 9. When complete, the schematic looks similar to Figure 3-127.







# 3.3.3.14 Configuring Pins

- 1. From the **Workspace Explorer**, double-click the *Ex3\_ADC\_to\_UART\_with\_DAC.cydwr* file.
- 2. Click the Pins tab.
- 3. Select pins P2[6:0] for LCD\_Char\_1
- 4. Select pin P0[7] for POT
- 5. Select pin P1[6] for LED
- 6. Select pin P1[2] for TX\_OUT

Figure 3-128. Pin Assignments





# 3.3.3.15 Creating main.c File

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- Replace the existing main.c content with the content of the embedded CY8C38\_main\_Ex3.C for PSoC 3 module and CY8C58LP\_main\_Ex3.C for PSoC 5 module file, which is available within the attachments feature of this PDF document.

Note

To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

3. From the **Build** menu, select **Build Ex3\_ADC\_to\_UART\_with\_DAC**. PSoC Creator builds the project and displays the comments in the **Output** dialog box. When you see the message "Build Succeeded", the build is complete.

## 3.3.3.16 Configuring and Programming the PSoC Development Board

- 1. Disconnect power to the board.
- 2. Configure the DVK SW3 to 5 V.
- Using the jumper wires included, configure the PSoC development board's prototyping area to: P0[7] to VR

P1[2] to TX

P1[6] to LED1

- 4. Verify that VR\_PWR (J11) is jumpered to **ON**.
- 5. Verify that RS232\_PWR (J10) is jumpered to **ON**.
- 6. Connect a serial cable from the PSoC development board to a PC.
- 7. Apply power to the board.
- 8. Install a terminal application such as TeraTerm or HyperTerminal with these setup parameters:

Baud Rate: 9600 Data: 8-bit Parity: none Stop: 1-bit Flow Control: none



COM1 Properties 🔹 💽 🔀
Port Settings
Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None
Restore Defaults
OK Cancel Apply

- 9. Use PSoC Creator as described in Programming My First PSoC 3 Project on page 32 or Programming My First PSoC 5LP Project on page 36 to program the device.
- 10. After programming the device, press the **Reset** button on the PSoC development board to see the output of the ADC displayed on the LCD and in the terminal application. LED1 is a sine wave output whose period is based on the ADC. Turning the potentiometer results in the LCD and observed terminal value change.

**Note** ADC values may fluctuate several counts due to system noise, and if the potentiometer voltage is at the edge of an ADC count.

11. Save and close the project.

## 3.3.4 USB HID

This project demonstrates a simple HID keyboard. The firmware begins enabling global interrupts, setting up the button (SW), and initializing USB for 3 V operation. The firmware, after allowing the HID device to enumerate, continuously checks for a button press to see if it needs to send the keyboard key sequences for the Cypress website. When you press the button, LED1 also toggles.

## 3.3.4.1 Creating USB HID Project

- 1. Open PSoC Creator.
- 2. Create a new project by clicking on Create New Project... in the Start Page of PSoC Creator.
- 3. In the **New Project** window, select the **Empty PSoC3 Design** template for a PSoC 3 design, or **Empty PSoC5 Design** template for a PSoC 5LP design and name the project **Ex4\_USB\_HID**.
- 4. In the **Location** field, type the path where you want to save the project, or click .... and navigate to the appropriate directory.
- 5. By default, the design window opens *TopDesign.cysch*. This is the project's schematic entry file within PSoC Creator.

#### 3.3.4.2 Placing and Configuring USBFS

1. Drag and drop a USBFS component from the Components Catalog  $\rightarrow$  Communication  $\rightarrow$  USBFS to the workspace.



- 2. Double-click the USBFS\_1 component.
- 3. Select the **HID Descriptor** tab.

Figure 3-130	<b>USBFS</b> Component Configuration
1 iguie 0 100.	CODI C Component Comiguration

Configure 'USBFS'		? 🛛
Name: USBFS_1		
Device Descriptor String Descriptor HID Descriptor	r Audio Descriptor MIDI Descriptor CDC Descriptor Advanced B	Built-in 4 Þ
🛖 Add Report 🛛 🐥 📕	💼 HID Items List	
IID Report Descriptors	USAGE USAGE_PAGE USAGE_MINIMUM USAGE_MAXIMUM DESIGNATOR_INDEX DESIGNATOR_MINIMUM DESIGNATOR_MAXIMUM STRING_INDEX STRING_MINIMUM STRING_MAXIMUM COLLECTION	
	Item Value (USAGE)	
		0x00
	Pointer	0x01
	Mouse	0x02
	Joystick	0x04
	Game Pad	0x05
	Keyboard	0x06
	Keypad	0x07
	X	0x30
	Add	0.21
Datasheet		Cancel

4. Click 👼 to import a report.





Open a Templa	te File					? 🔀
Look in:	🗀 template		~	00	• 🖽 🍯	
My Recent Documents	3ButtonMouse 5ButtonJoyst	ick.hid.xml				
My Documents						
My Computer						
	File name:	keyboard.hid.xml			~	Open
My Network	Files of type:	HID Template Files (*.hid.)	(ml)		*	Cancel

Figure 3-132. HID Descriptor Configuration





### 5. Select the String Descriptor tab.

Figure 3-133. String Descriptor Configuration

Configure 'USBFS'	2 🛛
Configure 'USBFS' Name: USBFS_1 Device Descriptor String Descriptor HID Descriptor Add String X Government And String 'Cypress Semiconductor' A String ''PSoC Development Kit'' Special Strings	r Audio Descriptor MIDI Descriptor CDC Descriptor Advanced Built-in 4 N Value PSoC Development Kit
Include Serial Number String Include MS OS String Descriptor Datasheet	OK Apply Cancel

- 6. Select String Descriptors in the left window.
- 7. Click Add String.
- 8. Click Add String a second time to add a total of two strings.
- 9. Click the **String** that shows up at the top in the left window.
- 10. Type Cypress Semiconductor in the Value field.
- 11. Click the **String** that shows up at the bottom in the left window.
- 12. Type **PSoC Development Kit** in the **Value** field.
- 13. Select the **Device Descriptor** tab.
- 14. Select Device Descriptor
- 15. Set the **Product ID** to **F11E**.
- 16. Set the Manufacturing String to Cypress Semiconductor.
- 17. Set the Product String to PSoC Development Kit.





- 18. Select Configuration Descriptor.
- 19. Set Device Power to Self Powered.
- 20. Set Max Power to 100 mA.

Figure 3-135. Configuration Descriptor Configuration

Configure 'USBFS'					?
Name: USBFS_1  Device Descriptor String Descriptor HID Descriptor	Audio Descriptor	MIDI Descriptor	CDC Descriptor	Advanced Built-in	40
Add Interface  Add Interface A	Configuration Attribut Configuration String: Max Power (mÅ): Device Power: Remote Wakeup:		× * *		

- 21. Select Alternate Setting 0.
- 22. Set Class to HID.

Figure 3-136. Interface Descriptor Configuration

Configure 'USBFS'	
Name: USBFS_1  Device Descriptor String Descriptor HID Descriptor	Audio Descriptor MIDI Descriptor CDC Descriptor Advanced Built-in 4 b
Add Endpoint X & F - F -	Interface Attributes Interface String: Interface Number: O Alternate Settings: O Class: HID  Subclass: No subclass



- 23. Select HID Class Descriptor.
- 24. Set HID Report to Keyboard w/LED Feature Report.
- 25. Select Endpoint Descriptor.
- 26. Set Direction to IN and Transfer Type to INT.

### 3.3.4.3 Placing and Configuring Software Digital Input Pin

- 1. Drag and drop a **Digital Input Pin** component (**Component Catalog** → **Ports and Pins** → **Digital Input Pin**)
- 2. Configure as follows.

### Type Tab

□ Name: Button

Figure 3-137. SW Digital Input Pin Configuration

Configure 'cy_pins'			? 🗙
Name: Button			
Pins Mapping R	eset Built-in		4 ⊳
Number of Pins: 1	X 🗗 🕈 ¥ 🗮 💥		
[All Pins]	Type General Input	t Output	
E Button_0	Analog	Preview:	
	🗹 Digital Input		
	HW Connection		
	🔲 Digital Output		
	HW Connection		
	Output Enable		
	Bidirectional		
	Show Annotation Terminal		
Datasheet	ОК	Apply Cano	cel .

#### General Tab

- Drive Mode: Resistive Pull up
- □ Initial State: Low (0)
- □ Leave the remaining parameters as default



Figure 3-138.	Pins - SW	Digital Inn	ut Pin	Configuration
i iyule 5-150.	F III 5 - 5 V V	Digital Inp	ULFIII	Configuration

Configure 'cy_pins'		?
Name: Button		
Pins Mapping Number of Pins: 1	Reset Built-in	4 ۵
[All Fins] ≟⊠ Button_0	Type General Input Drive Mode Resistive Pull Up	Output Initial State: Low (0)
Data Sheet		pply Cancel

- 3. Click **OK**.
- 3.3.4.4 Placing and Configuring LED
  - Drag and drop a Digital Output Pin component (Component Catalog → Ports and Pins → Digital Output Pin)
  - 2. Configure as follows:

Type Tab Name: LED

Figure 3-139. LED Component Configuration

Configure 'cy_pins'		? 🛛
Name: ED Pins Mapping Re	set Built-in	4 ۵
Number of Pins: 1		
[All Pros] └──⊠ LED_0	Type     General     Input       Analog     Digital Input     HW Connection       Digital Output     HW Connection       Output Enable     Bidirectional       Show Annotation Terminal	Output Preview:
Datasheet	ОК	Apply Cancel

### General Tab

Drive Mode: Strong Drive

Leave the remaining parameters as default



Configure 'cy_pins'		? 🛛
Name: LED		
Pins Mapping Re	set Built-in	4 Þ
Number of Pins: 1		
[All Pins]	Type General Input Ou Drive Mode	Itput
	Strong Drive	Initial State:
		Minimum Supply Voltage:
Datasheet	ОК Аррі	Cancel

Figure 3-140. Pins - LED Component Configuration

- 3.3.4.5 Configuring Clocks for CY8C38 Family Processor Module
  - 1. From the **Workspace Explorer**, open the *Ex4\_USB\_HID.cydwr* window and select the **Clocks** tab.
  - 2. Click on Edit Clock to open the Configure System Clocks window.
  - 3. Click on **IMO** from the listed rows and set Osc: 24.000 MHz
  - 4. Click the PLL Clock block and select IMO (48.000 MHz) for the Input.
  - 5. Set **Desired:** to 48 MHz for the PLL clock
  - 6. Enable the USB clock.
  - 7. Set the ILO clock to 100 kHz





Figure 3-141. Configure System Clocks

## 8. Click **OK**.

- 3.3.4.6 Configuring Clocks for CY8C58LP Family Processor Module
  - 1. From the **Workspace Explorer**, open the *Ex4\_USB\_HID.cydwr* window and select the **Clocks** tab.
  - 2. Click on Edit Clock to open the Configure System Clocks window.
  - Enable and configure XTAL to 24 MHz frequency.
     Note A 24-MHz crystal is installed on the board.
  - 4. Select the IMO source as XTAL.
  - 5. Enable the USB block and select IMOx2 48.000 MHz as input source.
  - 6. Set ILO at 100 KHz.
  - 7. In the PLL block, set the desired frequency as 33 MHz.
  - 8. For Master Clock, select PLL\_OUT (33 MHz) as input with Divider as 1.




Figure 3-142. Configure System Clock

Figure 3-143. XTAL Configuration

XTAL Configuration	? 🛛
Freq: 24 MHz V Accuracy • 6E-0 + 6E-0 % V	<ul> <li>Automatic gain control</li> <li>Enable fault recovery</li> </ul>
	OK Cancel

# 3.3.4.7 Configuring Pins

- 1. From the Workspace Explorer, double-click the *Ex4\_USB\_HID.cywrk* file.
- 2. Click the Pins tab.
- 3. Select and assign the pins as follows:

Assign USBFS\_dp to P15[6] Assign USBFS\_dm to P15[7] Assign LED to P1[6] Assign Button to P1[5]



Figure 3-144. Pin Assignments

Workspace Explorer (1 project) 🗸 🦷 🗙	Start Page main.c TopDesign.cysch Ex4_USB_HID.cydwr	
🖫 👦		
Workspace 'Ex4_USB_HID' (1 Projects)	Alias Name / Port	Pin
E Project 'Ex4_USB_HID' [CY8C5868AXI-LPC y	• * * * * * * * * * * * * * * * * * * *	36
TopDesign cysch	\USBF5_1:Dm\ 'I15(7) SWD:UK, USB:D- \VUSBF5 1:Dp\ P15(6) SWD:UD, USB:D+ \V	35
Ex4_USB_HID.cydwr	T ran how were to the second s	
device.h	2 PQR Table 0 PQR 2	
Source Files	Image (1)         Compare (0)	27
Gampon Gampon		
N N		
Dat		
Datasheets		
20 N		
T		
Results		
ज		
	104.0 Tel #1 805.11638 3 14.5 14 14 14 14 14 14 14 14 14 14 14 14 14	
	12 PPM HG2 DC100+PPM 12	
	المعلم العربي المعلم المعلم العرب المعلم العرب المعلم العرب المعلم العرب المعلم العرب المعلم العرب المعلم العرب 1 معلم المعلم العرب المعلم العرب المعلم العرب ال	
	and the set of the set	
	00000000000000000000000000000000000000	
	control of the second s	
	n n n n n n n n n n n n n n n n n n n	
A		
<	M Analog 🕘 Clocks 💉 Interrupts 🔡 DMA 😻 System 🖺 Directives 📄 Flash Security	

### 3.3.4.8 Creating main.c File

- 1. Open the existing *main.c* file within Workspace Explorer.
- Replace the existing main.c content with the content of the embedded CY8C38\_main\_Ex4.C for PSoC 3 module and CY8C58LP\_main\_Ex4.C for PSoC 5 module file, which is available within the attachments feature of this PDF document.

#### Note

To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

3. From the **Build** menu, select **Build Ex4\_USB\_HID**. PSoC Creator builds the project and displays the comments in the **Output** dialog box. When you see the message "Build Succeeded", the build is complete.

### 3.3.4.9 Configuring and Programming the PSoC Development Board

**Note** Due to the nature of the PSoC development board, powering the system from USB 'VBUS' can potentially reset other USB devices on the same hub. See the Appendix A, section titled Setting a 3.3-V Supply from VBUS on page 174.

- 1. Disconnect power to the board.
- 2. Configure the DVK SW3 to 5 V.
- 3. Configure the DVK breadboard using the jumper wires.

P1[5] to SW1

P1[6] to LED1

- 4. Connect the USB cable to the PC and to the USB port (J9)
- 5. Reapply power to the board.
- 6. Use PSoC Creator as described in section Programming My First PSoC 3 Project on page 32 or Programming My First PSoC 5LP Project on page 36 to program the device.
- 7. After programming the device, press Reset.
- 8. The PSoC development board is detected as a HID keyboard device. Wait until the device gets completely installed.



9. When button SW1 is pressed, the Windows Run window opens and the keyboard key sequence for the Cypress website is sent to open the Cypress website. When you press the button, LED1 also toggles.

10. Save and close the project.

**Note** The power setting of USB can be configured to either 3 V or 5 V mode in the firmware in the USBFS\_Start API. If the USB is configured for 3 V operation in firmware, ensure that the power switch (SW3) on the development kit is set to 3.3 V operation for the device to be detected (enumerated) on the PC.



# 3.3.5 CapSense

This project demonstrates CapSense. The firmware begins by initializing the LCD and CapSense components. In the main loop it scans the two buttons for activity. If there is a signal from either or both buttons, the corresponding LED lights up.

# 3.3.5.1 Creating CapSense Project

- 1. Open PSoC Creator.
- 2. Create a new project by clicking Create New Project... in the Start Page of PSoC Creator.
- 3. In the **New Project** window, select the **Empty PSoC3 Design** template for a PSoC 3 design, or **Empty PSoC5 Design** template for a PSoC 5LP design and name the project **Ex5\_CapSense**.
- 4. In the **Location** field, type the path where you want to save the project, or click \_\_\_\_\_ and navigate to the appropriate directory.
- 5. By default, the design window opens *TopDesign.cysch*. This is the project's schematic entry file within PSoC Creator.

### 3.3.5.2 Placing and Configuring CapSense

- 1. Drag and drop a CapSense component from the **Component Catalog**  $\rightarrow$  **CapSense**  $\rightarrow$  **CapSense\_CSD** to the workspace.
- 2. Double-click the **CapSense\_1** component
- 3. Configure CapSense as follows:

### General Tab

### Name: CapSense

Set parameters as shown in the following figure

Figure 3-145. CapSense Component Configuration

Configure 'CapSense_CSD'		?×
Name: CapSense		
General Widgets Config	Scan Order Advanced Tune Helper Built-in	۹ ۵
旑 Load Settings 🛛 🚽 Save Setting	35	
Tuning method	Auto (SmartSense)	
Number of channels	1 (default)	
Raw Data Noise Filter	None	
Water proofing and detection		
Clock Settings		
Scan Clock	12 MHz	
Important: Properties on A	dvanced tab were changed to reflect Auto Sense mode features.	
		144

- 4. Select the Widget Config Tab.
- 5. Add two buttons by clicking on Add Button. Leave the button parameters as default.



ne: CapSense			
	fig Scan Order Advanced Tune He	lper Built-in	4
Add Button Remove	Rename Sense mode so some widget properties are r	ot available	
Buttons	Tuning		
Button0	Finger Threshold (FT)	100	
Button1	Noise Threshold	20	
- Linear Sliders	Hysteresis	10	
LinearSlider0	Debounce	1	
- Radial Sliders	Scan Resolution	10 bits (default)	
Matrix Buttons			
Touch Pads			
- 🔄 Proximity Sensors			
Proximity Sensors			
- 🔄 Proximity Sensors			
Proximity Sensors			
Proximity Sensors			

Figure 3-146. Buttons - CapSense Component Configuration

- 6. Select Linear Slider and click on Add Linear Slider.
- 7. Change API resolution parameter to 80.

Figure 3-147. Slider Configuration

General Widgets Co	nfig Scan Order Advanced Tune Hel	per Built-in	
Add Linear Slider 🔣 Re			
and the second	ito Sense mode so some widget properties are no	ot available.	
Buttons	🗆 General		
Button0	Number of Sensor Elements	5	
Button1	API Resolution	80	
🛃 Linear Sliders	Diplexing	Non diplexed (default)	
🗘 LinearSlider0	Position Noise Filter	None	
🛃 Radial Sliders	🗆 Tuning		
Matrix Buttons	Finger Threshold (FT)	100	
Touch Pads	Noise Threshold	20	
Proximity Sensors	Scan Resolution	10 bits (default)	
Generics			

8. Select Tuner Helper tab and uncheck the Enable Tune Helper box.





Enable -	Tune Helper	nfig 🖌 Scan Orde		Tune Helper	1.11
	me for Ezl2C co	mponent: EZI.	20		
	en Ezl2C con se properties	ponent custon	nizer and		
Sub-addres:		- 16 💽 bit			

- 3.3.5.3 Placing and Configuring Character LCD
  - 1. Drag and drop a character LCD component from the **Component Catalog**  $\rightarrow$  **Display**  $\rightarrow$  **Character LCD** to the workspace.
  - 2. Double-click the LCD\_Char\_1 component.
  - 3. Set the parameter LCD Custom Character Set to Horizontal Bargraph.
  - 4. Select Include ASCII to Number Conversion Routines.
  - 5. Click OK

Figure 3-149. Horizontal Bargraph Configuration

Configure 'CharLCD'	? 🔀
Name: LCD_Char_1	
General Built-in	4 Þ
Parameters	Custom Character Editor
LCD Custom Character Set O None	
🔿 Vertical Bargraph	
<ul> <li>Horizontal Bargraph</li> </ul>	
O User Defined	
✓ Include Number to ASCII Conversion Routines	
Data Sheet OK	Apply Cancel



# 3.3.5.4 Placing and Configuring Digital Output Pin

- 1. Drag and drop two **Digital Output Pin** components from the **Component Catalog** → **Ports and Pins** → **Digital Output Pin** to the workspace.
- 2. Configure the two **Digital Port** components for LED1 and LED2.

Type Tab

Name: LED1

Figure 3-150. LED Configuration

Configure 'cy_pins'		? 🛛
Name: EDI Pins Mapping R Number of Pins: 1	eset Built-in	٩ ٩
[All Pins] └─⊠ LED1_0	Type       General       Input         Analog       Digital Input         HW Connection         Uigital Output         HW Connection         Output Enable         Bidirectional         Show Annotation Terminal	Output Preview:
Datasheet	ОК	Apply Cancel

General Tab

- Drive Mode: Strong Drive
- □ Leave the remaining parameters as default

Figure 3-151. Pins - LED Configuration

Configure 'cy_pins'		? 🛛
Name: LED1 Pins Mapping Rese Number of Pins: 1 (All Pins) LED1_0	et Built-in Prove General Input Outp Drive Mode Strong Drive	ut Initial State: Low (0) ♥ Minimum Supply Voltage:
Datasheet	OK Apply	Cancel

- 3. Click OK.
- 4. Configure LED2 similar to LED1.



# 3.3.5.5 Configuring Pins

- 1. From the Workspace Explorer, double-click the *Ex5\_CapSesne.cywrk* file.
- 2. Click the Pins tab
- 3. Select and assign the pins as follows:
  - Cmod to P2[7] for CY8C38 Family Processor Module and Cmod to P15[5] for CY8C58LP Family Processor Module
  - □ B1 to P0[5]
  - □ B2 to P0[6]
  - □ Position\_e0 to P0[0]
  - Position\_e1 to P0[1]
  - Position\_e2 to P0[2]
  - Position\_e3 to P0[3]
  - Position\_e4 to P0[4]
  - LED1 to P1[6]
  - □ LED2 to P1[7]
  - □ LCD\_Char\_1 to P2[0] to P2[6] (Drag it to P2[0] and PSoC Creator assigns the pin correctly.)

Figure 3-152. Pin Assignment for CY8C38 Family Processor Module

Workspace Explorer (1 project) 🔹 🕂 🗘	Start Page Ex5_Cap5ense.cydwr					
🖫 🕿 🔟 Workspace 'Ex5_CapSense' (1 Projects)		Alias	Name /	Port		
🖻 🔁 Project 'Ex5_CapSense' [CY8C38664X1-041 🖉		Cmod_CH0	\CapSense:CmodCH0\	P2[7]	~	3
- 📓 TopDesign.cysch - 🇀 Linker	10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Button0BTN	\CapSense:PortCH0[0]\	PO[5] OpAmp-	~	77
C Eve Carlo and a		Button1BTN	\CapSense:PortCH0[1]\	PO[6] IDAC:HI	~	78
Ex5_CapSense.cyre		LinearSlider0_e0LS	\CapSense:PortCH0[2]\	P0[0] OpAmp:out	~	71
Header Files     Meader Files     Meder Files	KD2(bac)(100*011))         TOP         Valid         To           KD2(bac)(100*011))         TOP         Valid         To           KD2(bac)(100*011))         TOP         Valid         To	LinearSlider0_e1LS	\CapSense:PortCH0[3]\	PO[1] OpAmp:out	~	72
E Constantia Constanti	Captenschaft H: 1 191 [Captenschaft H: 2 191	LinearSlider0_e2LS	\CapSense:PortCH0[4]\	PO[2] OpAmp+	~	73
i main.c	Paper (C230).     Paper (	LinearSlider0_e3LS	\CapSense:PortCH0[5]\	PO[3] OpAmp-, DSM:ExtVref	~	74
C main.c	1 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1	LinearSlider0_e4LS	\CapSense:PortCH0[6]\	PO[4] OpAmp+	~	76
			\LCD_Char_1:LCDPort[6:0]\	P2[6:0]	~	9599
Repuis	CYSC3860,XX1-040     m     m     CYSC3860,XX1-040     m     m     constant     CYSC3860,XX1-040     m     constant		LED1	P1[6]	~	27
, and the second s	CY8C3866AXI-040 a 100-TQFP		LED2	P1[7]	~	28
		<				
<	M Analog 🕑 Clocks 💉 Interrupts 👫 DMA 🦻 System	Directives 📃 📔 Flash Se	curity			



Workspace Explorer (1 project) 🗸 🗸 🗙	Start Page TopDesign.cysch Ex5 CapSense.cydwr			
립 📲 회 Workspace 'Ex5_CapSense' (1 Projects)		Alias	Name /	Port
Project 'Ex5_CapSense' [CY8C58684X4LP or 'TopDesign.cysch Project 'Ex5_CapSense.cydwr	A many and a	Cmod_CH0	\CapSense:CmodCH0\	P2[7] Trace
TopDesign.cysch     Ev5_CapSense.cudwr	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Button0BTN	\CapSense:PortCH0[0]\	PO[5] OpAmp-
🗆 🗁 Header Eilee	• 3555555555555555555555555555555555555	Button1BTN	\CapSense:PortCH0[1]\	PO[6] IDAC:HI
b device.h		LinearSlider0_e0LS	\CapSense:PortCH0[2]\	P0[0] OpAmp:out
redue mes     la device.h     Concerning     Concerning     Concerning     Concerning     Concerning	10230a_V0290111 [] PR http://www.state/	LinearSlider0_e1LS	\CapSense:PortCH0[3]\	P0[1] OpAmp:out
interio (a	Copfmac/Curl/R11         Priji tao         (v/cy-rrti)         D optmac/curl/R11()           4         Priji         Copfmac/curl/R11()         Optmac/curl/R11()	LinearSlider0_e2LS	\CapSense:PortCH0[4]\	P0[2] OpAmp+
Data	PTR (2320).     PTR (2320	LinearSlider0_e3LS	\CapSense:PortCH0[5]\	P0[3] OpAmp-, DSM:ExtVref
Detaineets Results		LinearSlider0_e4LS	\CapSense:PortCH0[6]\	P0[4] 0pAmp+
ă.			\LCD Char 1:LCDPort[6:0]\	P2[6:0] Trace, Trace, Trace, Trace
8			LED1	P1[6]
and the second s	CYCSSEAVCLP015		LKD2	P1[7]
		<		<u>д</u>
	🕼 Pins 🕅 Analog 🕒 Clocks 💉 Interrupts 🔡 DMA 🏾 察 System 🖺 Di	rectives 🔪 🦲 Flash Sec	curity	

Figure 3-153. Pin Assignment for CY8C58LP Family Processor Module

# 3.3.5.6 Creating main.c File

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded CY8C38\_main\_Ex5.C for PSoC 3 module and CY8C58LP\_main\_Ex5.C for PSoC 5 module file, which is available within the attachments feature of this PDF document.

#### Note

To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

- From the Build menu, select Build CapSense. PSoC Creator builds the project and displays the comments in the Output dialog box. When you see the message "Build Succeeded", the build is complete.
- 3.3.5.7 Configuring and Programming the PSoC Development Board
  - 1. Disconnect power to the board.
  - 2. Configure the DVK SW3 to 3.3 V.
  - 3. Using the jumper wires, configure the PSoC development board's prototyping area:
    - □ P1[6] to LED1
    - □ P1[7] to LED2
  - 4. Use PSoC Creator as described in Programming My First PSoC 3 Project on page 32 or Programming My First PSoC 5LP Project on page 36 to program the device.
  - 5. After programming the device, press Reset.
  - 6. When running the project, an LED lights up when either CapSense button is pushed. If B1 (P0[5]) is pushed, it also displays "Button1" in the top row of the LCD display. Similarly, if B2 (P0[6]) is pushed, it displays "Button2" in the top row of the LCD display. The bottom row of the LCD displays the slider position with a horizontal bargraph.
  - 7. Save and close the project.



# 3.3.6 SAR ADC (PSoC 5LP Only)

This project demonstrates sine wave generation by using an 8-bit DAC and DMA. The sine wave period is based on the current value of the ADC value of the potentiometer.

The firmware reads the voltage output by the DVK board potentiometer and displays the raw counts on the DVK board character LCD display similar to that shown in the previous project. An 8-bit DAC outputs a table generated sine wave to an LED using DMA at a frequency proportional to the ADC count. A 9600 Baud 8N1 UART outputs the current ADC count as ASCII formatted into a hexadecimal number.

The following instructions assume that you have completed My First PSoC Project and ADC to LCD Project and therefore have a basic understanding of the PSoC Creator software environment.

This code example uses the following components:

- SAR ADC (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  ADC  $\rightarrow$  SAR ADC)
- Voltage DAC (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  DAC  $\rightarrow$  Voltage DAC)
- Opamp (Component Catalog → Analog → Amplifiers → Opamp)
- DMA (Component Catalog → System → DMA)
- Character LCD (Component Catalog → Display → Character LCD)
- UART (Component Catalog → Communications → UART)
- Analog Pin (Component Catalog → Ports and Pins → Analog Pin)
- Digital Output Pin (Component Catalog → Ports and Pins → Digital Output Pin)
- Clock (Component Catalog → System → Clock)
- Logic Low (Component Catalog → Digital → Logic → Logic Low)
- Logic High (Component Catalog  $\rightarrow$  Digital  $\rightarrow$  Logic  $\rightarrow$  Logic High)

# 3.3.6.1 Creating ADC to UART with DAC Project

- 1. Open PSoC Creator.
- 2. Create a new project by clicking Create New Project... in the Start Page of PSoC Creator.
- 3. In the **New Project** window, select the **Empty PSoC5 Design** template and name the project **Ex6\_SAR\_to\_UART\_with\_DAC**.
- 4. In the **Location** field, type the path where you want to save the project, or click .... and navigate to the appropriate directory.
- 5. By default, the design window opens *TopDesign.cysch*. This is the project's schematic entry file within PSoC Creator.

# 3.3.6.2 Configuring Clock for ADC to UART with DAC Project

1. Open the Ex3\_ADC\_to\_UART\_with\_DAC.cydwr file from Workspace Explorer. See figure below

Figure 3-154. Workspace Explorer



2. Select the Clocks tab and click on Edit Clock....

DIGITAL 32.	.000         HHz         ? HHz           ?         HHz         ? HHz           .768         kHz         ? MHz           .000         HHz         ? MHz	±0 ±0 ±0	-	1 0 0		IMDx2
DIGITAL 32. DIGITAL 33.	.768 kHz ? MHz	±0				
DIGITAL 33.			-	0		
	.000 MHz ? MHz					
DIGITAL		±0	-	0		
	? MHz 1.000 kHz	-50, +100	-	0		
DIGITAL з.	.000 MHz 3.000 MHz	±l	-	0		
DIGITAL	? MHz 24.000 MHz	±l	-	1		MASTER_CLK
DIGITAL	? MHz 24.000 MHz	±l	-	1		PLL_OUT
DIGITAL 24.	.000 MHz 24.000 MHz	±l	-	0		IMO
DIGITAL 🔽 76.	.900 kHz 76.923 kHz	±1	-	312	<b>V</b>	MASTER_CLK
CLK ANALOG 🔽 1.	.800 MHz 1.846 MHz	±l	-	13		Auto: MASTER_CLK
DIGITAL 🔽 з.	.000 MHz 3.000 MHz	±l	-	1	<b>V</b>	IMO
	DIGITAL 24 DIGITAL 24 DIGITAL V 76 CLK ANALOG V 1	DIGITAL         ? NHz         24.000 NHz           DIGITAL         24.000 NHz         24.000 NHz           DIGITAL         ?         76.900 kHz         76.923 kHz           CLK         ANALOG         ?         1.800 NHz         1.846 NHz	DIGITAL         ? HHz         24.000 HHz         ±1           DIGITAL         24.000 HHz         24.000 HHz         ±1           DIGITAL         ?         76.900 kHz         76.923 kHz         ±1           CLK         ANALOG         ?         1.800 HHz         1.846 HHz         ±1	DIGITAL         ? HHz         24.000 HHz         41            DIGITAL         24.000 HHz         24.000 HHz         41            DIGITAL         76.900 kHz         76.923 kHz         41            CLK         ANALOG         1.800 HHz         1.846 HHz         41	DIGITAL         ? NHz         24.000 MHz         11         -         11           DIGITAL         24.000 MHz         24.000 MHz         11         -         10           DIGITAL         76.900 MHz         76.923 MHz         11         -         312           CLK         ANALOG         1.800 MHz         1.846 MHz         11         -         313	DIGITAL         ? MHz         24.000 MHz         1         -         1         I           DIGITAL         24.000 MHz         24.000 MHz         1         -         0         I           DIGITAL         24.000 MHz         24.000 MHz         1         1         -         0         I           DIGITAL         Image: Comparison of the state of the s

### Figure 3-155. Edit Clock

3. In the 'Configure System Clocks' window, enable and configure the XTAL to **24 MHz** with accuracy as  $\pm 6\%$ .

Figure 3-156. Configure XTAL

XTAL Configuration			? 🗙
Freq: 24 MHz V Accuracy - 6 + 6 % V	Enable autor Enable fault	recovery	
	Feedback:	3	~
	Watchdog:	3	~
	ОК	]	ancel

- 4. Set the IMO source to XTAL.
- 5. Select PLL source to IMO (24.000 MHz) and the desired output value to 33 MHz.
- 6. Select PLL\_OUT (33.000 MHz) as the source clock to Master Clock.
- 7. Set ILO to KHz and select OK.





Figure 3-157. Configure System Clocks

Now, go back to TopDesign.cysch to place and connect the components required for the project.

# 3.3.6.3 Placing and Configuring SAR ADC

- 1. Drag and drop the SAR ADC component (**Component Catalog**  $\rightarrow$  **Analog**  $\rightarrow$  **ADC**  $\rightarrow$  **SAR ADC**)
- 2. Double-click the **ADC\_SAR\_1** component in the schematic to open the configuration window.
- 3. Configure the SAR ADC as follows:

### Configure Tab

- □ Name: ADC\_SAR\_1
- Power: High Power
- Resolution: 12
- □ Conversion Rate: 100000
- **Clock Frequency**: 1800
- Input Range: Vssa to Vdda (Single Ended)
- Reference: Internal Vref
- □ Voltage Reference: 1.0240
- Sample Mode: Free Running
- Clock Source: Internal



Configure 'ADC_SAR'	? 🗙
Name:       ADC_SAR         Configure       Built-in         Modes       Resolution (bits):       12         Conversion rate (SPS):       100000       Triggered         Clock frequency (kHz):       1900       Clock source         Clock frequency (kHz):       1900       External         Input       Input range:       Vssa to Vdda (Single Ended)       V	
Reference:     Internal Vref       Voltage reference:     2.5000       Volts (Vdda/2)	
Datasheet OK Apply	Cancel

Figure 3-158. SAR ADC Component Configuration

- 3.3.6.4 Placing and Configuring an Analog Pin
  - 1. Drag and drop the Analog Pin component (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Analog Pin)
  - 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
  - 3. Configure the analog pin:
    - Type Tab
    - Name: POT
    - □ Select Analog check box only

Figure 3-159. POT Component Configuration

	-
٩	Þ
	_
	_
	,
-	
	_
!	

### General Tab

- Drive Mode: High Impedance Analog
- □ Leave the remaining parameters as default



# 3.3.6.5 Placing and Configuring Character LCD

- 1. Drag and drop the Character LCD component (Component Catalog  $\rightarrow$  Display  $\rightarrow$  Character LCD)
- 2. Double-click the LCD\_Char\_1 component in the schematic to open the configuration window.
- 3. Configure the Character LCD:
  - Name: LCD\_Char\_1
  - D LCD Custom Character Set: None
  - **Include ASCII to Number Conversion Routines**: check box

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Figure 3-160. Character LCD Component Configuration

General Built-in Parameters	4
LCD Custom Character Set None	
🔿 Vertical Bargraph	
🔿 Horizontal Bargraph	
O User Defined	
Include Number to ASCII Conversion Routines	

- 3.3.6.6 Placing and Configuring Voltage DAC
  - 1. Drag and drop the Voltage DAC component (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  DAC  $\rightarrow$  Voltage DAC)
  - 2. Double-click the VDAC8\_1 component in the schematic to open the configuration window.
  - 3. Configure the VDAC:

Basic Tab

- □ Name: VDAC8\_1
- Data\_Source: CPU or DMA (Data Bus)
- □ Initial\_Value: 1600
- Value\_mv: Register Write
- Value\_8 bit hex: 64
- D VDAC\_Range: 0 4.080 V (16 mV/bit)
- VDAC\_Speed: Slow Speed



Configure 'VDAC8'		? 🛛
Name: VDAC8_1		
Configure Built-in		4 ۵
VDAC		
Range	Speed	
○ 0 - 1.020 V (4 mV / bit)	Slow Speed	
⊙ 0 - 4.080 V (16 mV / bit)	○ High Speed	
Value mV: 1600 8 bit Hex: 64	Data Source O DAC Bus O CPU or DMA (Data Bus)	
Note: Changing any value field recalculates the other	Strobe Mode O External O Register Write	
Datasheet	ОК Арріу	Cancel

Figure 3-161. Voltage DAC Component Configuration

- 3.3.6.7 Placing and Configuring Opamp
  - 1. Drag and drop the Opamp component (Component Catalog  $\rightarrow$  Analog  $\rightarrow$  Amplifiers  $\rightarrow$  Opamp)
  - 2. Double-click the **Opamp\_1** component in the schematic to open the configuration window.
  - 3. Configure the Opamp:

Basic Tab

- Name: Opamp\_1
- Mode: Follower
- **Power**: High Power

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Figure 3-162. Opamp Component Configuration

Configure 'Op	ump'	? >
Name: Opa	np_1	
Configure	Built-in	4 ۵
Mode	Follower	~
Power	High Power	~
Data Sheet		Apply Cancel



- 3.3.6.8 Placing and Configuring Analog Pin
  - 1. Drag and drop the analog pin component (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Analog Pin)
  - 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
  - 3. Configure the analog pin:
    - **Type** Tab
    - Name: LED
    - Select Analog check box only

Figure 3-163. LED Component Configuration

Configure 'cy_pins'		? 🛛
Name:		
	eset Built-in	٩ ४
Number of Pins: 1		
[All Pins]	Type General Input	Output
⊠ LED_0	🗹 Analog	Preview:
	🔲 Digital Input	
	HW Connection	
	🔲 Digital Output	
	✓ HW Connection	
	Output Enable	
	Bidirectional	
	Show Annotation Terminal	
	,	
Datasheet	ОК	Apply Cancel

### General Tab

- Drive Mode: High Impedance Analog
- □ Leave the remaining parameters as default



Pins Mapping	Reset Built-in	4 Þ
[All Pins]	Type General Input Drive Mode High Impedance Analog	Output Initial State: Low (0)
	PR-054 PS	Minimum Supply Voltage

### Figure 3-164. LED Component Configuration

# 3.3.6.9 Placing and Configuring UART

- 1. Drag and drop the UART component (**Component Catalog**  $\rightarrow$  **Communications**  $\rightarrow$  **UART**)
- 2. Double-click the **UART\_1** component in the schematic to open the configuration window.
- 3. Configure the UART:

# Advanced Tab

Clock: External

Figure 3-165. Advanced Tab

Configure 'UART'		? 🗙
Name: UART_1		
Configure Advanced Built-in		4 ⊳
Clock Selection:		<u> </u>
Internal Clock	UNKNOWN SOURCE FREQ	
Interrupts		
RX - On Byte Received	📃 TX - On TX Complete	
RX - On Parity Error	🔲 TX - On FIFO Empty	
🔲 RX - On Stop Error	🔲 TX - On FIFO Full	
🔲 RX - On Break	🔲 TX - On FIFO Not Full	
RX - On Overrun Error		
📃 RX -On Address Match		
RX - On Address Detect		
RX Address Configuration	Buffer Sizes:	-
Datasheet OK	Apply Can	cel

### Configure Tab

- □ Name: UART\_1
- □ **Mode**: TxOnly
- □ Leave the remaining parameters to default



For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Figure 3-166. UART Component Configuration

Configur	re 'UART'		?	×
Name:	UART_1			
-	nfigure Advar	ced Built-in	٩	Þ
Mode	: Full UART (TX + I	RX) 🔿 RX Only		
0	Half Duplex	⊙ TX Only		
1	Bits per second:	1/8 Input Clock Frequency		
	Data bits:	8	]	
	Parity Type:	None		
		API control enabled		
	Stop bits:	1	]	
	Flow Control:	None	]	
Dat	asheet	OK Apply Cance	el	]

- 3.3.6.10 Placing and Configuring Digital Output Pin
  - 1. Drag and drop the Digital Output Pin component (Component Catalog  $\rightarrow$  Ports and Pins  $\rightarrow$  Digital Output Pin)
  - 2. Double-click the **Pin\_1** component in the schematic to open the configuration window.
  - 3. Configure the digital output pin:

Type Tab

- Name: TX\_OUT
- □ Select **HW Connection** check box

Figure 3-167. TX\_OUT Component Configuration

Configure 'cy_pins'		? 🛛
Name: TX_OUT		
Pins Mapping Re	set Built-in	4 ۵
Number of Pins: 1		
[All Fins]	Type General Input	Output
⊠ TX_OUT_0	🔲 Analog	Preview:
	🔲 Digital Input	
	HW Connection	
	🗹 Digital Output	
	HW Connection	
	🔲 Output Enable	
	Bidirectional	
	Show Annotation Terminal	
J	]	
Datasheet	ОК	Apply Cancel



### General Tab

- Under **Drive Mode**: Strong Drive
- □ Leave the remaining parameters as default

Figure 3-168. Pins - TX\_OUT Component Configuration

Configure 'cy_pins'		? 🔀
Name: TX_OUT		
Pins Mapping Re	eset Built-in	4 ⊳
Number of Pins: 1	× 🔊 🕈 🖊 🕺	
[All Pins]	Type General Input	Output
└⊠ тх_оит_о	Drive Mode	Initial State:
	Strong Drive	🛛 🔽 Low (0) 🔽
		Minimum Supply Voltage:
Datasheet	ОК	Apply Cancel

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

- 3.3.6.11 Placing and Configuring Clock for UART
  - 4. Connect a clock component (Component Catalog  $\rightarrow$  System  $\rightarrow$  Clock) to the UART clock.
  - 5. Double-click the **Clock** component.
  - 6. Configure the clock:
    - Name: Clock\_2
    - □ Source: XTAL (24.000 MHz)
    - Desired Frequency: 76.9 kHz
    - □ Leave the remaining parameters at default

**Note** The desired frequency of the clock is 76.9 kHz because this value should be eight times the required baud rate: 76900/8 = 9612.5, which is approximately 9600 (required baud rate).



Figure 3-169. Configure UART Clock

Configure 'cy	_clock'	? 🔀
Name: Clo	ck_2	
Configur	e Clock Advanced Built-in	۹ ۵
Clock Type:	New O Existing	
Source:	XTAL (24.000 MHz)	*
Specify:	● Frequency 76.9 kHz ▼	
	O Divider	
Uses Cloo Source Clo Name: X Enabled:	TAL Yes y: 24.000 MHz	<
, Datasheet	Car	ncel

Figure 3-170. UART Configure Window After Assigning Clock Source - Configure Tab

Configure 'UART'		? 🛛
Name: UART_1		
Configure Advan	iced Built-in	4 ۵
Mode: O Full UART (TX + I	RX) ORX Only	
O Half Duplex	• TX Only	
Bits per second:	1/8 Input Clock Frequency (9.615385 KBaud)	
Data bits:	8	~
Parity Type:	None	~
	API control enabled	
Stop bits:	1	~
Flow Control:	None	~
<u></u>		
Datasheet	OK Apply	Cancel



Configure 'UART'		? 🛛
Name: UART_1		
Configure Advanced Built-in		4 ۵
Clock Selection: O Internal Clock  ③ External Clock	SOURCE FREQ = 76.923 kHz	
Interrupts		
RX - On Byte Received	🔲 TX - On TX Complete	
RX - On Parity Error	🔲 TX - On FIFO Empty	
RX - On Stop Error	🔲 TX - On FIFO Full	
RX - On Break	🔲 TX - On FIFO Not Full	
RX - On Overrun Error		
RX -On Address Match		
RX - On Address Detect		
RX Address Configuration	Buffer Sizes:	- 🗹
Datasheet OK	Apply Can	cel

Figure 3-171. UART Configure Window After Assigning Clock Source - Advanced Tab

3.3.6.12 Placing and Configuring DMA

- 1. Drag and drop the DMA component (Component Catalog  $\rightarrow$  System  $\rightarrow$  DMA)
- 2. Double-click the **DMA\_1** component in the schematic to open the configuration window.
- 3. Configure the DMA: **Basic** Tab
  - Name: DMA\_1
  - Hardware Request: Rising Edge
  - □ Leave the remaining parameters to default

For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

Figure 3-172. DMA Component Configuration

Configure 'cy_dma'		? 🗙
Name: DMA_1		
Basic Built-in		4 ۵
Hardware Request:	Rising Edge	~
Hardware Termination:	Disabled	<ul> <li></li> <li></li> </ul>
Data Sheet	OK Apply	Cancel



- 3.3.6.13 Connecting the Components Together
  - Connect a Logic Low component (Component Catalog → Digital → Logic → Logic Low) to the reset of the UART
  - 2. Connect a Clock component (Component Catalog  $\rightarrow$  System  $\rightarrow$  Clock) to the clock of the DMA.
  - 3. Double-click the **Clock** component to configure.
  - 4. Configure the clock:

Configure Clock Tab

- □ Name: Clock\_1
- **Source**: IMO (24.000 MHz)
- Desired Frequency: 3 MHz
- □ Leave the remaining parameters set to their default values

Figure 3-173. Clock Component Configuration

Configure 'cy_clock'	? 🗙
Name: Clock_1	
Configure Clock Advanced Built-in	4 ⊳
Clock Type: 💿 New 🔿 Existing	
Source: IMO (24.000 MHz)	*
Specify: 💿 Frequency 3 MHz 🗸	
Summary         API Generated: Yes         Uses Clock Tree Resource: Yes         Source Clock Info         Name: IMO         Enabled: Yes         Frequency: 24.000 MHz         Accuracy: ±6.001	<
Datasheet OK Apply Can	icel

- 5. Using the Wire Tool 1, connect **tx** (in the UART component) to HW connection of the TX\_OUT digital output pin (TX\_OUT).
- 6. Using the Wire Tool 1, connect VDAC8 (VDAC8\_1) to Opamp (Opamp\_1).
- 7. Using the Wire Tool 1, connect **POT** to **ADC\_SAR (ADC\_SAR\_1)**.
- 8. Right-click the LED analog pin, select the **Shape** menu option and then **Flip Horizontal**. This allows the LED pin to line up with the Opamp output.
- 9. When complete, the schematic looks similar to Figure 3-174.







# 3.3.6.14 Configuring Pins

- 1. From the **Workspace Explorer**, open the *Ex6\_SAR\_to\_UART\_with\_DAC.cydwr* file.
- 2. Click the Pins tab.
- 3. Select pins P2[6:0] for LCD\_Char\_1.
- 4. Select pin P0[7] for POT.
- 5. Select pin P1[6] for LED.
- 6. Select pin P1[2] for TX\_OUT.

Figure 3-175. Pin Assignments





# 3.3.6.15 Creating main.c File

- 1. Open the existing *main.c* file within **Workspace Explorer**.
- 2. Replace the existing *main.c* content with the content of the embedded *CY8C58LP\_main\_Ex6.c* file, which is available within the attachments feature of this PDF document.

**Note** To access the embedded attachments feature in the PDF, click on the paper clip icon located in the lower left corner of the Adobe Reader application.

- 3. From the **Build** menu, select **Build Ex6\_SAR\_to\_UART\_with\_DAC**. PSoC Creator builds the project and displays the comments in the **Output** dialog box. When you see the message "Build Succeeded", the build is complete.
- 3.3.6.16 Configuring and Programming the PSoC Development Board
  - 1. Disconnect power to the board.
  - 2. Configure the DVK SW3 to 5 V.
  - 3. Using the jumper wires included, configure the PSoC Development Board's prototyping area to:
    - □ P0[7] to VR
    - □ P1[2] to TX
    - □ P1[6] to LED1
  - 4. Verify that VR\_PWR (J11) is jumpered to ON.
  - 5. Verify that RS232\_PWR (J10) is jumpered to ON.
  - 6. Connect a serial cable from the PSoC development board to a PC.
  - 7. Apply power to the board.
  - 8. Install a terminal application such as TeraTerm or HyperTerminal with these setup parameters:
    - Baud Rate: 9600
    - Data: 8-bit
    - Parity: none
    - □ Stop: 1-bit
    - □ Flow Control: none



# Figure 3-176. HyperTerminal Settings

COM1 Properties	? 🛛
Port Settings	
Bits per second:	9600
Data bits:	8
Parity:	None
Stop bits:	1
Flow control:	None
	Restore Defaults
	K Cancel Apply

- 9. Use PSoC Creator as described in Programming My First PSoC 5LP Project on page 36 to program the device.
- 10. After programming the device, press the **Reset** button on the PSoC Development Board to see the output of the ADC displayed on the LCD and in the terminal application. LED1 is a sine wave output whose period is based on the ADC. Turning the potentiometer results in the LCD and observed terminal value change.
- 11. Save and close the project.

Sample Projects



# Appendix A. Board Specifications and Layout



This appendix gives detailed specifications of the PSoC Development Kit board components

# A.1 PSoC Development Board

A.1.1 Factory Default Configuration

### A.1.1.1 Power Supply

The board has several power nets. Following are the definitions of the different power nets.

VIN (9 V or 12 V) - This is the input power before it is fed to any of the regulators. A 9-V to 12-V power supply adapter or a 9-V battery is used as the source.

VREG (5 V) - This is fed by VIN and is the output of the onboard 5-V regulator. VREG can be selected as the main 5-V source by using the J8 header.

VBUS (5 V) - This is power derived from the USB interface via a USB host. VBUS can be selected as the main 5-V source by using the J8 header.

VDD (3.3 V or 5 V) - This is fed by VREG, VBUS, or the onboard 3.3-V regulator. VDD can be chosen either to be 3.3 V or 5 V by the simple positioning of the VDD select switch.

VADJ (1.5 V to 3.3 V for 3.3-V supply and 1.5 V to 5 V for 5-V supply) - This is fed by VDD and is the output of the onboard adjustable regulator. It is mainly used when the PSoC core must be powered at lower voltages. An adjustable resistor R11 is used for adjusting the voltage.

VDD DIG - This is power derived from either VDD or VADJ. It is used to power the PSoC core. The source for VDD DIG can be chosen as VDD or VADJ using the J7 header.

VDD ANLG - This is power derived from either VDD or VADJ. It is mainly used to separate the analog power from the digital power. The source for VDD ANLG can be chosen as VDD or VADJ using the J6 header.

VDDIO - This is power derived from either VDD or VADJ. It is used to power digital I/O on the PSoC device. There are four sections of GPIO, which can be powered to 5 V, 3.3 V, or VADJ using four headers. It enables you to power the PSoC GPIOs at different voltages.



# A.1.2 Power Supply Configuration Examples

# A.1.2.1 Setting a 5-V Supply from VREG

- 1. Place the jumper on J8 header to select VREG as the source.
- 2. Move the VDD select switch to select the 5 V.
- 3. Place the jumper on J6 header to select VDD as source for VDD ANLG.
- 4. Place the jumper on J7 header to select VDD as source for VDD DIG.

Figure A-1. Setting a 5-V Supply from VREG



# A.1.2.2 Setting a 3.3-V Supply from VREG

- 1. Place the jumper on J8 header to select VREG as the source.
- 2. Move the VDD select switch to select 3.3 V.
- 3. Place the jumper on J6 header to select VDD as source for VDD ANLG.
- 4. Place the jumper on J7 header to select VDD as source for VDD DIG.

Figure A-2. Setting a 3.3-V Supply from VREG





# A.1.2.3 Setting VDD ANLG as VADJ and VDD DIG as VDD for VDD = 3.3 V

- 1. Place the jumper on J8 header to select VREG as the source.
- 2. Move the VDD select switch to select 3.3 V.
- 3. Place the jumper on J6 header to select VADJ as source for VDD ANLG.
- 4. Place the jumper on J7 header to select VDD as source for VDD DIG.

Figure A-3. Setting VDD ANLG as VADJ and VDD DIG as VDD for VDD = 3.3 V



This helps to separate the analog supply from the digital supply and VDD.

# A.1.2.4 Setting VDD DIG as VADJ and VDD ANLG as VDD for VDD = 3.3 V

- 1. Place the jumper on J8 header to select VREG as the source.
- 2. Move the VDD select switch to select 3.3 V.
- 3. Place the jumper on J6 header to select VDD as source for VDD ANLG.
- 4. Place the jumper on J7 header to select VADJ as source for VDD DIG.

Figure A-4. Setting VDD DIG as VADJ and VDD ANLG as VDD for VDD = 3.3 V



This helps to separate the digital supply from the analog supply and VDD.



# A.1.2.5 Setting a 5-V Supply from VBUS

- 1. Place the jumper on J8 header to select VBUS as the source.
- 2. Move the VDD select switch to select the 5 V.
- 3. Place the jumper on J6 header to select VDD as source for VDD ANLG.
- 4. Place the jumper on J7 header to select VDD as source for VDD DIG.

### Figure A-5. Setting a 5-V Supply from VBUS



# A.1.2.6 Setting a 3.3-V Supply from VBUS

Due to the nature of the PSoC development board, powering the system from USB 'VBUS' can potentially reset other USB devices on the same hub.

By design, the PSoC development board is capable of drawing more than 500 mA of current during normal operation, which exceeds USB bus power limits. Additionally, the development board exceeds inrush current limits due to 'VBUS' capacitance greater than 10 uF. As a result, plugging the PSoC development board into a USB hub can potentially cause other devices on the same hub to reset due to excessive inrush currents. Take care when powering the PSoC development board from 'VBUS'. It is good practice to plug the board into a host root hub, or a "self-powered" external hub when doing USB development. Bus powered applications done outside the realm of the PSoC development board should comply with the USB specification for inrush current limits and recommended bulk capacitance on 'VBUS'. See the Universal Serial Bus Specification Revision 2.0 for more details.

- 1. Place the jumper on J8 header to select VBUS as the source.
- 2. Move the VDD select switch to select 3.3 V.
- 3. Place the jumper on J6 header to select VDD as source for VDD ANLG.
- 4. Place the jumper on J7 header to select VDD as source for VDD DIG.



Figure A-6. Setting a 3.3-V supply from VBUS



You can measure current from VREG, VBUS, VDD ANLG, VDD DIG and VDDIOs by removing the jumpers and connecting the meter across the respective header.

A.1.2.7 J1 - DC Power Jack

Figure A-7. DC Power Jack



Use a 12 V/1 A power supply adapter when powering from the barrel power jack. This input power is VIN.

A.1.2.8 9-V Battery Terminals

Figure A-8. Battery Terminals



Use a 9-V alkaline battery to connect to the 9-V battery terminals. This input power is VIN.

A.1.2.9 J8 - 5-V Source

This header allows you to select the 5 V source from either the onboard 5 V regulator (VREG) or from the USB 5 V rail (VBUS).

### A.1.2.10 VDD Select Switch

This switch allows you to select either 5 V or 3.3 V. VDD feeds VDD DIG, VDD ANLG, and VDDIO.



# A.1.2.11 J7 - VDD DIG Select

This header allows you to select the PSoC core source power. To power the PSoC core at either 5 V or 3.3 V (based on the position of the VDD select switch), place the jumper on the upper two pins. To power the PSoC core at lower voltages (1.7 V to 4.95 V), place the jumper on the lower two pins. When the jumper is on the lower two pins, you must adjust R11 to tune the adjustable regulator to output the desired voltage.

# A.1.2.12 J6 - VDD ANLG Select

To separate the analog power from the digital power, you can position the jumper on the upper two pins to source analog power at 5 V or 3.3 V (based on the position of the VDD select switch), or on the lower two pins to source analog power at lower voltages (1.5 V to 3.3 V for 3.3-V supply and 1.5 V to 5 V for 5-V supply).

### A.1.2.13 R11 - Adjustable Regulator Variable Resistor

This adjustable resistor is used to tune the VADJ voltage. Turning this variable resistor swings the VADJ voltage between 1.6 V and 3.29 V when the VDD select switch is in the 3.3 V position. When the VDD select switch is in the 5 V position, turning this variable resistor swings the VADJ voltage between 1.7 V and 4.95 V.



# A.1.3 Prototyping Components

# A.1.3.1 Prototyping Area

**Note** CY8C38 family modules have a 2200-pF capacitor connected between P2[7] and ground. CY8C58LP family modules have a 2200-pF capacitor connected between P15[5] and ground. These configurations provide an external modulator capacitor for CapSense designs. To use P2[7] on CY8C38 family modules or P15[5] on CY8C58LP family modules for anything other than CapSense, it is recommended that C18 on these modules be removed, to avoid disrupting digital or analog signals on this I/O pin.





# A.1.3.2 P15 - DB9 Serial Communications Port

This is a standard female DB9 serial communications connector. Four signals are brought from the RS-232 transceiver to receptacle P16. These signals are Rx, Tx, Clear To Send, and Request To Send. To connect these signals to the PSoC I/O pins, use wires to jumper from P16 to P19, where sockets for ports zero and one are available.

Table A-1. Connector Pin Assignments - RS-232 (DTE) Serial Communications Socket

Pin Number	P15
1	(Empty)
2	ТХ
3	RX
4	(Empty)
5	GND
6	(Empty)
7	CTS
8	RTS
9	(Empty)

# A.1.3.3 J10 - Serial Port Power

Header J10 must be connected to use the serial communications port. Placing a jumper on J10 provides VDD power to the RS-232 transceiver. This power can be either 3.3 V or 5 V, depending on the position of the VDD select switch.

### A.1.3.4 J9 - Full Speed USB Port

The board has a mini-B full speed USB connector. There are also two test points for the differential pair signals D- and D+. These signals are routed to the processor module socket P1, pins 6 and 8 respectively. The power net VBUS is brought into the board through this interface.

# A.1.3.5 P17 - Artaflex WirelessUSB LP Radio Module Receptacle

Receptacle P17 is used specifically for the Artaflex AWP24S WirelessUSB module. Eight signals are routed from this receptacle to P12 receptacle. These signals are four serial peripheral interface (SPI) signals MISO (master-in-slave-out), MOSI (master-out-slave-in), nSS (slave select), SCK (serial clock), an IRQ (interrupt request) and RD\_RESET (radio reset). The other two signals are radio transmit and receive signals.

Note These I/O signals must not be greater than 3.3 V.

Pin Number	P17
1	GND
2	V3_3
3	IRQ
4	RD_RESET
5	MOSI
6	nSS
7	SCK
8	MISO

Table A-2. Connector Pin Assignments - Wireless Radio Module Socket



Table A-2. Connector Pin Assignments - Wireless Radio Module Socket (continued)

Pin Number	P17
9	GND
10	(Empty)
11	TxPA
12	RxPA

### A.1.3.6 J14 - Wireless Radio Module Power

Header J14 must be connected to use the Artaflex radio module. Placing a jumper on J14 provides 3.3 V power to the P17 module socket. This power is drawn directly from the 3.3 V regulator.

### A.1.3.7 R20 - Multipurpose Variable Resistor

The board is equipped with a 10 k $\Omega$  thumbwheel variable resistor referenced to ground. The high side of the resistor is tied to jumper J11. The wiper is tied to a receptacle pin on P14.

A.1.3.8 J11 - Variable Resistor Power

Header J11 must be connected to use the variable resistor. Placing a jumper on J11 provides VDD ANLG power to the high side of the resistor.

### A.1.3.9 SW1, SW2, SW5, and SW6 - Multipurpose Push Button Switches

The board has four multipurpose mechanical push buttons, SW1, SW2, SW3, and SW6, that are referenced to ground. The other sides of the switches are tied to receptacle pins on P14 and P9. The switches follow an inverted logic as they connect ground to receptacle pins on P14 or P9 when pressed.

Note SW3 is VDD SELECT, SW4 is RESET switch.

# A.1.4 LCD Module



The board has a 2×16 alpha-numeric LCD. I/Os of the module are connected to port two of the PSoC device and are routed to the processor module socket P2. This LCD is rated for 5 V. However, the I/Os have a level translator inline so that signaling may be as low as 1.8 V and still be recognized by the LCD. The header J12 must be connected for the LCD module to be powered; otherwise, it removes power from the level translator. If the LCD module is removed, the receptacle pins of P18 can be used as port 2.

**Note** You can connect R40 (0E resistor, no load part) or short the pads given for R40 to switch the LCD back-light on. The current consumption of the LCD with backlight is around 70 mA; this should be considered when you budget the power supply of the design. You may use the backlight LCD from Lumex Inc (part number: LCM-S01602DSF/A).



Pin Number	P18
1	GND
2	VCC_LCD
3	VO
4	RS
5	R/nW
6	EN
7	D0
8	D1
9	D2
10	D3
11	D4
12	D5
13	D6
14	D7
15	BACKLT LED ANODE
16	BACKLT LED CATHODE

### Table A-3. Connector Pin Assignments - LCD Module Socket

# A.1.4.1 R31 - LCD Contrast Adjustment

The board is equipped with an LCD contrast adjustment resistor R31. Turning the wiper counterclockwise increases the contrast, while turning the wiper clockwise decreases the contrast.

# A.1.4.2 J12 - LCD Module Power

Power for the LCD module is provided through header J12. Placing a jumper on the upper two pins shorts the VCC pin of the module to ground. Placing the jumper on the lower two pins provides 5 V to the VCC pin of the module. This 5 V power is taken directly from the onboard 5 V regulator.

# A.1.5 CapSense Elements

The prototyping area has three capacitive sensing elements. There are two CapSense buttons connected directly to port zero pins. In addition, there is a five-segment CapSense slider also connected directly to port zero. Series resistors are placed on these port zero I/Os and should be loaded with appropriate values. A value of  $0\Omega$  is used for general-purpose CapSense applications, but a value of 560  $\Omega$  should be used to achieve best performance. The board is loaded with  $0\Omega$  series resistors by default. The presence of CapSense elements does not affect the general purpose use of port zero pins.

# A.1.6 Processor Module

# A.1.6.1 J2, J3, J4, and J5 - VDDIO Select

These four headers allow you to power the PSoC GPIOs at different voltages. For instance, some of the I/O may be powered at 5 V, some at 3.3 V, and some at 1.8 V. There are four blocks of GPIO, each having its own source power. Each VDDIO header provides power to specific GPIOs and is selectable from VDD, 3.3 V, or VADJ. For details on which GPIOs are powered by which VDDIO header, see the datasheet for the PSoC device used with this board.
For example, VDDIO\_0 is configured to VDD, VDDIO\_1 is configured to 3.3 V, and VDDIO\_2 is configured to VADJ by placing the jumpers in the respective positions, as shown in Figure A-9.





## A.1.6.2 SW4 - Processor Reset Button

The board has a push button switch that resets the PSoC device attached to the processor module. One side of the switch is tied to the XRES pin of the processor module socket. The other end of the switch is tied to the HW\_RESET pin of the processor module socket. This allows the module designer to tie the HW\_RESET line either high or low, depending on which direction the processor reset is active.

**Note** PSoC 1 devices are active-high reset. Therefore, a light pull-down resistor may be necessary on the XRES pin of designs with these devices to avoid unintentional device resets. PSoC 3 and PSoC 5LP devices are active-low reset. Therefore, a light pull-up resistor may be necessary on the XRES pin of designs with these devices to avoid unintentional device resets.

#### A.1.6.3 U8 - External MHz Oscillator

The board supports the use of an external high frequency 8-pin PDIP oscillator. The speed of the oscillator supported is dependent on the specifications of the PSoC device used. The output of this oscillator is routed to P15[4] on receptacle P2 and TP62 near P2 of the DVK board.

#### A.1.6.4 P1, P2, P3, and P4 - Processor Module Receptacles

Processor modules provide modularity to this board. Sockets P1 to P4 are used to connect a processor module to the board. All supported GPIOs (including special I/Os), along with VDD DIG, VDD ANLG, 5 V, 3.3 V, VBUS, and VBAT (only connected to a surface mount pad on the board) are connected to these receptacles. In addition, each of the VDDIO power pins are connected to these receptacles. The full speed USB D+ and D– signals are also connected to one of the sockets. Processor reset is connected to P1. Any "no connect" pins are brought out to surface mount test pads.

Pin Number	P1 (West)	P2 (North)	P3 (East)	P4 (South)		
1	GND	GND	GND	GND		
2	VDDD	GND	GND	P7[7]		
3	V5_0	P6[1]	P12[2]	NC7		
4	GND	P6[0]	P12[3]	NC8		
5	VBAT	P6[3]	P8[0]	NC5		
6	DM	P6[2]	P8[1]	NC6		
7	V3_3	P15[5]	P4[0]	NC3		

Table A-4.	Connector Pin	Assignments -	Processor Module Sockets	
100107111		7 10019111101110		



Pin Number	P1 (West)	P2 (North)	P3 (East)	P4 (South)
8	DP	P15[4]	P4[1]	NC4
9	VBUS	P9[2]	P8[2]	P7[6]
10	VDDIO1	P9[0]	P8[3]	P7[5]
11	P5[6]	P2[1]	P0[0]	P12[0]
12	P5[7]	P2[0]	P0[1]	P12[1]
13	P5[4]	P2[3]	P0[2]	P3[6]
14	P5[5]	P2[2]	P0[3]	P3[7]
15	P12[6]	VDDIO2	VDDIO0	P7[4]
16	P12[7]	P9[3]	VDDA	VDDIO3
17	P1[6]	P2[5]	P0[4]	P3[4]
18	P1[7]	P2[4]	P0[5]	P3[5]
19	P1[4]	P2[7]	P0[6]	P3[2]
20	P1[5]	P2[6]	P0[7]	P3[3]
21	HW_RESET	P9[4]	P8[4]	P3[0]
22	P1[3]	P9[5]	P8[5]	P3[1]
23	P1[1]	P12[5]	P8[6]	P7[2]
24	P1[2]	P12[4]	P8[7]	P7[3]
25	P1[0]	P9[6]	P4[2]	(Empty)
26	P5[3]	P9[7]	P4[3]	(Empty)
27	P5[2]	P6[5]	P4[4]	P7[0]
28	P5[1]	P6[4]	P4[5]	P7[1]
29	P5[0]	P6[7]	P4[6]	NC1
30	XRES	P6[6]	P4[7]	NC2
31	GND	GND	GND	GND
32	GND	GND	P9[1]	GND

Table A-4. Connector Pin Assignments - Processor Module Sockets (continued)

Figure A-10. Mechanical Layout Details for Processor Module Connector





## A.1.7 Expansion Ports



The board accommodates I/O expandability. Around the upper, lower, and right sides of the board are 0.100-inch pitch, dual row right angle receptacles, each having at least three full 8-bit ports (one has four full ports). Each also has four special I/O pins available. Three of the ports have power and ground pins as well. The fourth is simply I/O and ground exclusively. These sockets can be used to join the processor module I/Os with external I/Os through the use of daughter boards.

Pin Number	P5 (PORT B)	P6 (PORT A')	P7 (PORT A)	P8 (PORT C)
1	P1[7]	P6[7]	P3[7]	P9[7]
2	P1[6]	P6[6]	P3[6]	P9[6]
3	P1[5]	P6[5]	P3[5]	P9[5]
4	P1[4]	P6[4]	P3[4]	P9[4]
5	P1[3]	P6[3]	P3[3]	P9[3]
6	P1[2]	P6[2]	P3[2]	P9[2]
7	P1[1]	P6[1]	P3[1]	P9[1]
8	P1[0]	P6[0]	P3[0]	P9[0]
9	GND	GND	GND	GND
10	RESRV3	RESRV8	RESRV11	RESRV14

Table A-5	Connector Pin Assignments - Expansion Port Sockets



Pin Number	P5 (PORT B)	P6 (PORT A')	P7 (PORT A)	P8 (PORT C)
11	P2[7]	P2[7]	P5[7]	P8[7]
12	P2[6]	P2[6]	P5[6]	P8[6]
13	P2[5]	P2[5]	P5[5]	P8[5]
14	P2[4]	P2[4]	P5[4]	P8[4]
15	P2[3]	P2[3]	P5[3]	P8[3]
16	P2[2]	P2[2]	P5[2]	P8[2]
17	P2[1]	P2[1]	P5[1]	P8[1]
18	P2[0]	P2[0]	P5[0]	P8[0]
19	GND	GND	GND	GND
20	RESRV2	RESRV7	RESRV10	RESRV13
21	P0[7]	P0[7]	P4[7]	P7[7]
22	P0[6]	P0[6]	P4[6]	P7[6]
23	P0[5]	P0[5]	P4[5]	P7[5]
24	P0[4]	P0[4]	P4[4]	P7[4]
25	P0[3]	P0[3]	P4[3]	P7[3]
26	P0[2]	P0[2]	P4[2]	P7[2]
27	P0[1]	P0[1]	P4[1]	P7[1]
28	P0[0]	P0[0]	P4[0]	P7[0]
29	GND	GND	GND	GND
30	RESRV1	RESRV6	RESRV9	RESRV12
31	P12[3]	P7[7]	P12[3]	P12[3]
32	P12[2]	P7[6]	P12[2]	P12[2]
33	P12[1]	P7[5]	P12[1]	P12[1]
34	P12[0]	P7[4]	P12[0]	P12[0]
35	V3_3	P7[3]	V3_3	V3_3
36	VADJ	P7[2]	VADJ	VADJ
37	GND	P7[1]	GND	GND
38	V5_0	P7[0]	V5_0	V5_0
39	VIN	GND	VIN	VIN
40	GND	RESRV5	GND	GND
41	x	P12[5]	x	x
42	x	P12[4]	x	x
43	x	P12[7]	x	x
44	x	P12[6]	x	x
45	x	GND	x	x
46	x	RESRV4	x	x

 Table A-5.
 Connector Pin Assignments - Expansion Port Sockets (continued)

#### A.1.7.1 Expansion Ports A and A'

Expansion port A can be used as I/O ports with three full 8-bit ports: port3, port4, and port5. It has four special I/Os as well as ground and voltage pins. It can be used to join processor module I/Os port3, port4, and port5 with external I/Os through the use of daughter boards.



Expansion port A' can be used as I/O ports with four full 8-bit ports: port0, port2, port6, and port7. It has four special I/Os as well as ground pins. It has no voltage pins. It can be used to join processor module I/Os port0, port2, port6, and port7 with external I/Os through the use of daughter boards.

The main use of port A' is that it can be used together with port A to join processor module I/Os port0, port2, port3, port4, port5, port6, and port7 with external I/Os through the use of daughter boards.

## A.1.7.2 Expansion Port B

Expansion port B can be used as I/O ports with three full 8-bit ports: port0, port1, and port2. It has four special I/Os as well as ground and voltage pins. It can be used to join processor module I/Os port0, port1, and port2 with external I/Os through the use of daughter boards. It is mainly used in devices with fewer I/Os.

#### A.1.7.3 Expansion Port C

Expansion port C can be used as I/O ports with three full 8-bit ports: port7, port8, and port9. It has four special I/Os as well as ground and voltage pins. It can be used to join processor module I/Os port7, port8, and port9 with external I/Os through the use of daughter boards. It is used for devices with a high I/O count.

#### A.1.7.4 Protection Circuit

The protection circuit consists of two P-channel MOSFET on the power line allowing the power/current to flow from input to output depending on the voltages applied at the external board connectors. This circuit protects the board from voltages above 5.5 V (over voltage) and reversing the power and ground terminals (reverse voltage) while powering the board from external connectors. Figure A-11 and Figure A-12 are protection circuits placed between EBK and the on-board components on the 5-V and 3.3-V line.



Figure A-11. Schematic for Protection Circuit on 5-V Power Line





#### Figure A-12. Schematic for Protection Circuit on 3.3-V Power Line

## **Functional Description of Circuit**

The protection circuit will protect from a maximum over-voltage or reverse-voltage of 12 Volts. The cut-off voltage on the 5-V line is 5.7 V and on the 3.3-V line is 3.6 V. This means, if you apply more than this voltage level from the external board connector side, the p-MOS Q5 will turn off, thus protecting the PSoC and other on-board components. The current consumption of these protection circuits is less than 6 mA.

When voltage from the external connector is between 1.8 V and 3.3 V, the p-MOS Q4 conducts. Because the voltage across D9 and R4 is the same, the p-MOS Q6 conducts, allowing voltage supply to the DVK.

When the external power supply exceeds 3.3 V, the p-MOS Q5 starts conducting. This eventually turns off p-MOS Q6 at 3.6 V, protecting the DVK from over-voltage.

When a reverse voltage is applied across the protection circuit from the external connector side, Q4 P-MOS will turn off, thus protecting the components on the board from reverse voltage.

If you intend to use the regulator power supply from the board to power the external modules, both the P-MOS Q4 and Q5 will always be on, allowing the flow of current with a maximum of 22 mV drop across the circuit when the current consumed by the external module is 150 mA.

**Note** The working of protection circuit on the 3.3-V line and 5-V line is as described above. For the purpose of explanation, the annotation of 3.3-V protection circuitry (Figure A-12) is used.



# A.2 Schematics

# A.2.1 CY8CKIT-001 PSoC Development Board













# A.2.3 CY8C29 Family Processor Module



# A.2.4 CY8C38 Family Processor Module







# A.2.5 CY8C58LP Family Processor Module



## A.2.6 Enabling Boost Component in PSoC 3 and PSoC 5LP Processor Modules

To enable the boost convertor functionality, make the following hardware changes on the board.

- Populate resistors R6, R7, R14, R15, R16, R17, and R38 with 0-Ω resistors.
- Ensure that R37 and R43 are removed.

The input power supply to the boost convertor must be provided through the Vbat. After making these changes, you can make a boost convertor based design by making the appropriate configurations in the project.

## A.3 Bill of Materials

## A.3.1 CY8CKIT-001 PSoC Development Board

ltem	tem Qty Reference		Reference Description M		Mfr Part Number	
1	1	N/A	Schematic	N/A	REF-14647 REV *C	
2	1	N/A	Assembly Drawing	N/A	121R-46100 REV *C	
3	1	N/A	Fab Drawing	Fab Drawing N/A		
4	1	N/A	Assembly Adhesive Label	N/A	121R-46100 REV *C	
5	1	РСВ	PRINTED CIRCUIT BOARD	Cypress Semiconductor	PDCR-9461 REV *B	
6	1	BH1	BATTERY HOLDER 9V Male PC MT	Keystone Electronics	593	
7	1	BH2	BATTERY HOLDER 9V Female PC MT	Keystone Electronics	594	
8	4	C1,C3,C9,C30	CAP ELECT 10UF 25V VS SMD size B	Panasonic - ECG	EEE-1EA100WR	
9	1	C2	CAP ELECT 330UF 25V FK SMD	Panasonic - ECG	EEE-FK1E331P	
10	2	C4,C25	CAP CERAMIC 1.0UF 25V X5R 0603 10%	Taiyo Yuden	TMK107BJ105KA-T	
11	1	C10	CAP 10000PF 16V CERAMIC X7R 0402	Yageo America	04022R103K7B20D	
12	10	C11,C13,C14,C15 ,C16,C21,C22,C2 6,C27,C28	CAP .10UF 16V CERAMIC X7R 0603	Kemet	C0603C104J4RACTU	
13	2	C12,C29	CAP 10UF 16V CERAMIC X5R 1210	Panasonic - ECG	ECJ-4YB1C106K	
14	1	C17	CAP .33UF 16V CERAMIC X7R 0805	Panasonic - ECG	ECJ-2YB1C334K	
15	2	C20,C23	CAP .1UF 50V CERAMIC X7R 0805	Panasonic - ECG	ECJ-2YB1H104K	
16	1	C24	CAP 33nF 50V CERAMIC X8R 0603	TDK Corporation	C1608X8R1H333K	
17	1	D1	LED GREEN CLEAR 1206 SMD	Chicago Miniature Lamp, Inc	CMD15-21VGC/TR8	
18	1	D2	DIODE SCHOTTKY 40V 1.5A SMA	Vishay IR	10MQ040NTRPBF	
19	4	D3,D4,D5,D6	LED HI EFF RED CLEAR 1206 SMD	Chicago Miniature Lamp, Inc	CMD15-21VRC/TR8	
20	1	D7	DIODE ZENER 3.6V 500MW SOD123	ON Semiconductor	MMSZ4685T1G	
21	1	D8	DIODE ZENER 4.3V 1W SOD-106	Rohm Semiconductor	PTZTE254.3B	
22	1	D9	DIODE ZENER 2V 500MW SOD-123	Diodes Inc	BZT52C2V0-7-F	
23	1	F1	FUSE RESETTABLE .10A 30V HLD SMD	Bourns	MF-USMF010-2	
24	1	J1	CONN JACK POWER 2.1mm PCB RA	CUI	PJ-102A	
25	4	J2,J3,J4,J5	CONN HEADER 6POS .100 STR 15AU	FCI	67996-206HLF	
26	4	J6,J7,J8,J12	CONN HEADR BRKWAY .100 03POS STR	Tyco Electronics/Amp	9-146280-0-03	
27	1	J9	CONN USB MINI B SMT RIGHT ANGLE	Тусо	1734035-2	
28	3	J10,J11,J14	CONN HEADR BRKWAY .100 02POS STR	Tyco Electronics	9-146280-0-02	
29	4	P1,P2,P3,P4	CONN FMALE 32POS DL .050 SMT GOLD	Samtec	RSM-116-02-S-D-LC	
30	3	P5,P7,P8	CONN FMALE 40POS DL .100 R/A GOLD	Sullins Electronics Corp.	PPPC202LJBN-RC	
31	1	P6	CONN FMALE 46POS DL .100 R/A GOLD	Sullins Electronics Corp.	PPPC232LJBN-RC	
32	1	P11	SOLDERLESS BREADBOARD 1.8x1.35	3M	923273-I	
33	2	P12,P9	CONN RECT 8POS .100 VERT	3M	929850-01-08-RA	
34	1	P14	CONN RECT 12POS .100 VERT	3M	929850-01-12-RA	



ltem	Qty Reference		Description	Manufacturer	Mfr Part Number	
35	1	P15	CONN D-SUB RCPT R/A 9POS 30GOLD	AMP Division of TYCO	5747844-4	
36	1	P16	CONN RECEPT 4POS .100 VERT GOLD	3M	929850-01-04-RA	
37	1	P17	CONN RECEPT 12POS 2mm SMD TIN	Hirose Electric Co. LTD.	DF11Z-12DS-2V(20)	
38	1	P18	CONN REC .100 14POS for LCM-S01602DSR/A	3M	929850-01-14-RA	
39	1	P19	CONN RECT 17POS .100 VERT	3M	929850-01-17-RA	
40	6	Q1,Q2,Q3,Q4,Q5, Q6	MOSFET P-CH 30V 3.8A SOT23-3	Diodes Inc	DMP3098L-7	
41	12	R1,R3,R5,R21,R2 2,R23,R24,R25,R 26,R28,R29,R30	RES 1.0K OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ102V	
42	1	R2	RES 220 OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF2200V	
13	1	R4	RES 442 OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF4420V	
14	1	R10	RES 120 OHM 1/10W 5% 0603 SMD	Panasonic-ECG	ERJ-3GEYJ121V	
15	1	R11	TRIMPOT 500 OHM 6mm SQ SMD	Bourns Inc.	3361P-1-501GLF	
46	1	R14	RES 100K OHM 1/16W 5% 0402 SMD	Panasonic - ECG	ERJ-2GEJ104X	
17	1	R15	RES 200 OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ201V	
48	9	R16,R18,R41,R4 2,R43,R44,R45,R 46,R47	RES ZERO OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V	
49	2	R17,R19	RES 100 OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ101V	
50	1	R20	POT 10K OHM 1/8W CARB VERTICAL	CTS Electrocomponents	296UD103B1N	
51	1	R31	POT 10K CARBON LAYDOWN (103)	Panasonic - ECG	EVN-D8AA03B14	
52	7	R32,R33,R34,R3 5,R36,R37,R38	RES 6.04K OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF6041V	
53	1	R39	RES 200K OHM 1/10W 5% 0603 SMD	Panasonic-ECG	ERJ-3GEYJ204V	
54	1	R48	RES 100K OHM 1/10W 1% 0603 SMD	Yageo	RC0603FR-07100KL	
55	5	SW1,SW2,SW4,S W5,SW6	SWITCH TACT 6mm MOM 150GF	Omron	B3F-1022	
56	1	SW3	SWITCH SLIDE MINI SPDT PCMNT SLV	APEM Components	GH36P010001	
57	5	TP1,TP2,TP3,TP 4,TP14	TEST POINT 43 HOLE 65 PLATED RED	Keystone Electronics	5000	
58	4	TP18,TP19,TP56, TP58	TEST POINT 43 HOLE 65 PLATED WHITE	Keystone Electronics	5002	
59	9	TP30,TP32,TP33, TP34,TP35,TP36, TP37,TP38,TP39	TEST POINT 43 HOLE 65 PLATED BLACK	Keystone Electronics	5001	
60	1	TVS1	TVS 5.0 VOLT 350 WATT SOD-323	Semtech	SD05.TCT	
61	1	U2	IC REG LDO 1.0A 5.0V TO-252	Diodes Inc	AP1117D50L-13	
62	1	U4	IC REG LDO 0.3A ADJ 8MSOP	National Semiconductor	LP3982IMM-ADJ/NOPB	
63	1	U5	IC SINGLE USB PORT TVS SOT-23-6	Texas Instruments	SN65220DBV	
64	1	U6	IC LINE DRVR/RCVR RS232 16-SOIC	Texas Insturments	SN65C3232ED	
65	1	U7	IC XLATR 8BIT LV 20-TSSOP	NXP Semiconductors	GTL2003PW	
6	1	U8	IC SOCKET 8PIN MS TIN/TIN .300	Mill-Max Manufacturing	110-44-308-41-001000	
67	1	U11	IC REG LDO 300mA 3.3V SOT89R	Diodes Inc	AP130-33YRL-13	
68	1	NA	5V LCD Module 16POS w/14 pin header installed	Lumex	LCM-S01602DSR/A	
69	5	NA	BUMPER WHITE .500X.23 SQUARE	Richco Plastics Co.	RBS-3R	
70	11	NA	SHUNT GOLD W/HANDLE, BLACK	Kobiconn	151-8030-E	
		STALL				
71	2	R27,R49	RES NO LOAD 0603 SMD	NA	NA	
72	2	R50,R51	RES 0.0 OHM 1/10W 5% 0805 SMD	Panasonic - ECG	ERJ-6GEY0R00V	
73	1	R40	RES NO LOAD 0805 SMD	NA	NA	



# A.3.2 CY8C28 Family Processor Module

ltem	Qty	Reference	Description	Manufacturer	Mfr Part Number
1	1	PCB	PRINTED CIRCUIT BOARD	Cypress Semiconductor	PDC-09547 REV **
2	1	C1	CAP CER 10UF 16V X5R 0805	Murata Electronics North America	GRM21BR61C106KE15L
3	2	C2,C6	CAP .10UF 16V CERAMIC X7R 0603	Kemet	C0603C104J4RACTU
4	1	C10	CAP 100PF 50V CERAMIC 0402 SMD	Panasonic - ECG	ECJ-0EC1H101J
5	1	C16	CAP CER 2200PF 50V 5% C0G 0603	Murata	GRM1885C1H222JA01D
6	4	J1,J2,J3,J4	CONN MALE 32POS DL .050 TH SHRD GOLD	Centronic Precision Elec- tronic Co.	HHLHS32GB1
7	1	J5	CONN HEADER 5POS 0.1 VERT KEYED	Molex	22-23-2051
8	1	P1	HDR VERT 20POS HIROSE	Hirose	DF12-5.0-20DP-0.5V-81
9	4	R1,R3,R5,R 7	RES 56 OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ560V
10	5	R2,R4,R6,R 8,R11	RES 1.0K OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ102V
11	1	R9	RES ZERO OHM 1/16W 0402 SMD	Panasonic - ECG	ERJ-2GE0R00X
12	4	R10,R16, R17,R18	RES ZERO OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V
13	3	TP1,TP3, TP5	TEST POINT 43 HOLE 65 PLATED RED	Keystone Electronics	5000
14	2	TP2,TP4	TEST POINT 43 HOLE 65 PLATED BLACK	Keystone Electronics	5001
15	2	NA	SMT Spacer/nut	PEM	SMTSO-440-8ET
16	1	U1	IC, 56 PIN SSOP OCD	Cypress Semiconductor	CY8C28000-24PVXI
17	1	LABEL1	PCA # Label		120-09547-0 REV **
No Lo	ad Co	mponents	÷	-	-
18	3	C12,C13, C14	CAP NO LOAD 0805	NA	NA
19	1	C15	CAP 0603 NO LOAD	NA	NA
20	4	R12,R13, R14,R15	RES NO LOAD 0603 SMD	NA	NA
21	1	D1	DIODE SCHOTTKY 40V 1.0A SOT23-3	Zetex	ZHCS1000TA
22	1	L1	INDUCTOR FIXED SMD 2.2uH 10%	Panasonic-ECG	ELJ-FC2R2KF
23	1	Y1	CRYSTAL 32.768 kHz CYL 12.5PF	Citizen America Corpo- ration	CFS206 32.768KDZF-UB
24	2	TP6,TP7	TEST POINT 43 HOLE 65 PLATED WHITE	Keystone Electronics	5002



# A.3.3 CY8C29 Family Processor Module

ltem	Qty	Reference	Description	Manufacturer	Mfr Part Number
1	1	C1	CAP CER 10UF 16V X5R 0805	Murata Electronics North America	GRM21BR61C106KE15L
2	4	C2,C3,C4,C 6	CAP .10UF 16V CERAMIC X7R 0603	Kemet	C0603C104J4RACTU
3	1	C10	CAP 100PF 50V CERAMIC 0402 SMD	Panasonic - ECG	ECJ-0EC1H101J
4	4	J1,J2,J3, J4	CONN MALE 32POS DL .050 TH SHRD GOLD	Centronic Precision Elec- tronic Co.	HHLHS32GB1
5	1	J5	CONN HEADER 5POS 0.1 VERT KEYED	Molex	22-23-2051
6	1	P1	RECP VERT 20POS HIROSE	Hirose	DF12-5.0-20DP-0.5V-81
7	4	R1,R3,R5,R 7	RES 56 OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ560V
8	5	R2,R4,R6,R 8,R11	RES 1.0K OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ102V
9	1	R9	RES ZERO OHM 1/16W 0402 SMD	Panasonic - ECG	ERJ-2GE0R00X
10	3	R10,R12, R13	RES ZERO OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V
11	2	TP1,TP3	TEST POINT 43 HOLE 65 PLATED RED	Keystone Electronics	5000
12	1	TP2	TEST POINT 43 HOLE 65 PLATED BLACK	Keystone Electronics	5001
13	1	U1	PSoC Mixed-Signal Array	Cypress Semiconductor	CY8C29000-24AXI
14	2	NA	SMT Spacer/nut	PEM	SMTSO-440-8ET
15	1	PCB	PRINTED CIRCUIT BOARD	Cypress Semiconductor	PDCR-9464 REV*A
16	1	LABEL1	PCA # Label		121R-46400 REV*B
No Lo	ad Co	nponents			
17	3	C12,C13, C14	CAP NO LOAD 0805	NA	NA
18	1	D1	DIODE SCHOTTKY 30V 200mW SOT23	Diodes Inc	BAT54-7-F
19	1	L1	INDUCTOR FIXED SMD 2.2uH 10%	Panasonic-ECG	ELJ-FC2R2KF
20	1	Y1	CRYSTAL 32.768 kHz CYL 12.5PF	Citizen America Corporation	CFS206 32.768KDZF-UB



# A.3.4 CY8C38 Family Processor Module

Item	tem Qty Reference Description M		Manufacturer	Mfr Part Number	
1	1	N/A	Schematic	N/A	REF-14889 REV *D
2	1	N/A	Assembly Drawing	N/A	121R-49400 REV *D
3	1	N/A	Fab Drawing	N/A	N/A
4	1	N/A	Assembly Adhesive Label	N/A	121R-49400 REV *D
5	1	N/A	РСВ	Cypress Semiconductor	PDCR-9494 REV **
6	2	C1,C14	CAP CER 10UF 16V X5R 0805	Murata Electronics North America	GRM21BR61C106KE15L
7	7	C2,C3,C4,C 5,C6,C7,C28	CAP CERAMIC 1.0UF 10V X5R 0603	Kemet	C0603C105K8PACTU
8	2	C8,C11	CAP CERAMIC 1.2UF 10V X5R 0805	Kemet	C0805C125K8PACTU
9	2	C9,C25	CAP .10UF 16V CERAMIC X7R 0603	Kemet	C0603C104J4RACTU
10	2	C10,C12	CAP CER 22UF 10V 10% X5R 1210	Kemet	C1210C226K8PACTU
11	7	C15,C16,C1 7,C21,C22,C 23,C24	CAP .10UF 10V CERAMIC X5R 0402	Kemet	C0402C104K8PACTU
12	1	C18	CAP CER 2200PF 50V 5% C0G 0603	Murata	GRM1885C1H222JA01D
13	2	C19,C20	CAP CERAMIC 22PF 50V 0603 SMD	Panasonic - ECG	ECJ-1VC1H220J
14	1	D1	DIODE SCHOTTKY 40V 1A SOT23	Zetex	ZHCS1000TA
15	4	J1,J2,J3,J4	CONN MALE 32POS DL .050 TH SHRD GOLD	Centronic Precision Elec- tronic Co.	HHLHS32GB1
16	1	J5	CONN HEADER 10 PIN 50MIL KEYED SMD	Samtec	FTSH-105-01-L-DV-K
17	1	L1	INDUCTOR SHIELD PWR 22UH 7032	TDK Corporation	SLF7032T-220MR96-2-PF
18	2	R8,R9	RES 22 OHM 1/16W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF22R0V
19	8	R31,R32,R3 6,R38,R39,R 40,R41,R42	RES ZERO OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V
20	6	TP1,TP3,TP 4,TP5,TP6,T P7	TEST POINT 43 HOLE 65 PLATED RED	Keystone Electronics	5000
21	1	TP2	TEST POINT 43 HOLE 65 PLATED BLACK	Keystone Electronics	5001
22	1	U1	PSoC3 Mixed-Signal Array	Cypress Semiconductor	CY8C3866AXI-040
23	1	Y1	CRYSTAL 32.768 kHz CYL 12.5PF	Citizen America Corporation	CFS206 32.768KDZF-UB
24	2	C26,C27	CAP, CER, 12 pF, 50V, 5%, COG, 0603, SMD	Murata Electronics North America	GRM1885C1H120JA01D
25	1	Y2	CRYSTAL, 24 MHz, 30 ppm, HC49, SMD	ECS Inc.	ECS-240-12-5PX-TR
	t Insta		· · ·		
26	6	R6,R7,R14, R15,R16,R1 7	RES NO LOAD 0805 SMD	NA	NA
27	5	R10,R34,R3 5,R37,R43	RES NO LOAD 0603 SMD	NA	NA



# A.3.5 CY8C58LP Family Processor Module

ltem	Qty	CY Part Number	Reference	Description	Manufacturer	Mfr Part Number
1	1	630-60082-01 REV**	N/A	Schematic	N/A	630-60082-01 REV**
2	1	620-60083-01 REV**	N/A	Assembly Drawing	N/A	620-60083-01 REV**
3	1	610-60081-01 REV**	N/A	Fab Drawing	N/A	610-60081-01 REV**
4	1	620-60083-01 REV**	N/A	Assembly Adhesive Label	N/A	620-60083-01 REV**
5	1	600-60083-01 REV**	N/A	РСВ	N/A	600-60083-01 REV**
6	2	700-00105	C1,C14	CAP, CER, 10 uF, 16 V, 5%, X5R, 0805, SMD	Murata Electronics North America	GRM21BR61C106K E15L
7	7	700-00111	C2,C3,C4,C5,C 6,C7,C28	CAP, CER, 1.0 uF, 10 V, 10%, X5R, 0603, SMD	Kemet	C0603C105K8PACT U
8	2	700-00118	C8,C11	CAP, CER, 1.0 uF, 10 V, 10%, X5R, 0805, SMD	Murata	GRM219R61A105KC 01D
9	2	700-00104	C9,C25	CAP, CER, 0.1 uF, 16 V, 5%, X7R, 0603, SMD	Kemet	C0603C104J4RACT U
10	2	700-00112	C10,C12	CAP, CER, 22 uF, 10 V, 10%, X5R, 1210, SMD	Kemet	C1210C226K8PACT U
11	8	700-00001	C15,C16,C17,C 21,C22,C23,C2 4,C29	CAP, CER, 0.1 uF, 16 V, 10%, X7R, 0402, SMD	Kemet	C0402C104K4RACT U
12	1	700-00094	C18	CAP, CER, 2200 pF, 50V, 5%, COG, 0603, SMD	Murata	GRM1885C1H222JA 01D
13	2	700-00012	C19,C20	CAP, CER, 22 pF, 50V, 5%, COG, 0603, SMD	Panasonic - ECG	ECJ-1VC1H220J
14	1	810-00007	D1	DIODE, SCHOTTKY, 40 V, 1 A, ZHCS1000TA, SOT-23, SMD	Zetex	ZHCS1000TA
15	4	400-00051	J1,J2,J3,J4	CONN, HDR, 2x16, 0.05", GOLD, TH	Centronic Precision Electronic Co.	HHLHS32GB1
16	2	400-00061	J5,J6	CONN, HDR, KEYED, 2x5, 0.050", Gold, SMD	Digilent	161-026



ltem	Qty	CY Part Number	Reference	Description	Manufacturer	Mfr Part Number
17	1	820-00006	L1	IND, FIXED, 22 uH, 20%, .960 A, 7032, SMD	TDK Corporation	SLF7032T- 220MR96-2-PF
18	2	600-00208	R8,R9	RES 22 OHM 1/16W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF22R0V
19	13	620R-10519	R31,R32,R36,R 38,R39,R40,R4 1,R42,R45,R46, R47,R48,R49	RES ZERO OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V
20	1	600-00165	R44	RES, 0.0 Ohms, 1/8 W, 5%, 0805, SMD	Panasonic - ECG	ERJ-6GEY0R00V
21	7	400-00001	TP1,TP3,TP4,T P5,TP6,TP7,TP 8	CONN, TEST POINT, RED, TH	Keystone Electronics	5000
22	2	400-00002	TP2,TP9	CONN, TEST POINT, BLACK, TH	Keystone Electronics	5001
23	1	CY8C5868AXI-LP035	U1	IC, PSoC5LP,CY8C5868A XI-LP035, TQFP-100, SMD	Cypress Semiconductor	CY8C5868AXI- LP035
24	1	850R-13480	Y1	CRYSTAL 32.768 KHZ CYL 12.5PF CFS308	Citizen America Corporation	CFS206 32.768KDZF-UB
25	2	700-00048	C26,C27	CAP, CER, 12 pF, 50V, 5%, COG, 0603, SMD	Murata Electronics North America	GRM1885C1H120JA 01D
26	1	850-00003	Y2	CRYSTAL, 24 MHz, 30 ppm, HC49, SMD	ECS Inc.	ECS-240-12-5PX-TR
DON	IOT	INSTALL				
27	6	NA	R6,R7,R14,R15 ,R16,R17	RES NO LOAD 0805 SMD	NA	NA
28	5	NA	R10,R34,R35,R 37,R43	RES NO LOAD 0603 SMD	NA	NA
29	1	NA	TP10	TEST POINT 43 HOLE 65 PLATED WHITE	NA	NA
Addi	tiona	I assembly instructions:				
		ble primary side (TOP SIDI its last.	E) through hole co	omponents first. Then asser	nble secondary sid	e (BOTTOM SIDE)

2. Ensure J5 and J6 are installed with the keyed side of the header facing the inside of the board.

# Appendix B. MiniProg3



# B.1 MiniProg3 LEDs

MiniProg3 provides five indicator LEDs:

- Upper Left Busy: A red LED that lights when an operation (such as programming or debug) is in progress.
- Lower Left Status: A green LED that lights when the device is enumerated on the USB bus and flashes when the MiniProg3 receives USB traffic.
- Upper Right Target Power: A red LED that lights to indicate that the MiniProg3 is supplying
  power to the target connectors. Note that it does not light when target power is detected but not
  being supplied by MiniProg3.
- Lower Right Aux: A yellow LED reserved for future use.
- Middle No Label: A yellow LED that indicates the configuration state of the device. It flashes briefly during the initial configuration of the device. If this LED lights solid, a configuration error has occurred and MiniProg3 must be disconnected from the USB port and reconnected.

## B.2 Programming in Power Cycle Mode

Do not perform power cycle mode programming with PSoC Programmer on the CY8CKIT-001. This is due to the design of the CY8C38 family module. VTARG of the MiniProg3 is wired exclusively to VDDIO1 of the chip on the module. For power cycle programming to work, VTARG needs to be wired to VDDD.

## B.3 Interface Pin Assignment Table

5-Pin # *	10-Pin # *	JTAG **	SWD	SWV	ISSP	I2C
1	1	Vtarg	Vtarg	Vtarg	Vtarg	Vtarg
2	3,5,7,9	GND	GND	GND	GND	GND
3	10	TRST		SWO	XRES	INT
4	4	тск	SCK		SCLK	SCLK
5	2	TMS	SDIO		SDAT	SDAT
	6	TDO				
	8	TDI				

Notes:

\* The 5- and 10-pin connectors are NOT connected together on the I/O pins

\*\* JTAG is supported only on the 10-pin connector

\*\*\* Future upgrades may be possible to support these modes



# B.4 Protection Circuitry

The Vtarg and I/O pins of the two interface connectors are protected from ESD events and momentary short circuits by a group of TVS (Transient Voltage Suppressor) diodes. These diodes provide a 15 KV ESD event protection for each pin, and will clamp the pin levels to a safe voltage in the event of a short circuit. The Vtarg pins are protected by a shared, 5 V clamp device capable of shunting 350 W of transient power. Each I/O pin is similarly protected by a 5 V, 30 W device.

# B.5 Level Translation

The design provides level translators that interfaces with any I/O voltage in the range of 1.2 V to 5.5 V without damage and function properly. There are two different level translators used in the design.

# Appendix C. MiniProg3 Technical Description



The MiniProg3 is a protocol translation device. It enables PC host software to communicate through high-speed USB to the target device to be programmed or debugged. This is shown in Figure C-1. The device side communication protocol can be one of several standards, and can occur over either of two connectors. Table C-1 lists the protocols that are supported by each connector. MiniProg3 enables communication with target devices using I/O voltage levels from 1.5 V to 5.5 V. In addition, MiniProg3 can provide power to a simple target board, at one of four voltage levels.

Figure C-1. System Block Diagram



Table C-1. Connectors / Communication Protocol Support

Connector	ISSP	JTAG	SWD and SWV <sup>a</sup>	l <sup>2</sup> C
5-pin	Supported	N/A	SWD	Supported
10-pin	N/A	Supported	SWD and SWV	N/A

a. SWV trace is only available with SWD debugging.



# C.1 Interfaces

## C.1.1 ISSP

In-System Serial Programming (ISSP) is a Cypress legacy interface used to program the PSoC 1 family of microcontrollers. MiniProg3 supports programming PSoC 1 devices through the 5-pin connector only.

For more information about the ISSP interface, see the PSoC 1 Technical Reference Manual.

## C.1.2 JTAG

The Joint Test Action Group (JTAG) standard interface is supported by many high end microcontrollers, including the PSoC 3 and PSoC 5LP families. This interface allows a daisy chain bus of multiple JTAG devices. MiniProg3 supports programming and debugging PSoC 3 and PSoC 5 LP devices using JTAG, through the 10-pin connector only.

## C.1.3 SWD/SWV

Recent ARM based devices have introduced a new serial debugging standard called Serial Wire Debug (SWD). The PSoC 3 and PSoC 5LP family implements this standard, which offers the same programming and debug functions as JTAG, except the boundary scan and daisy chain. SWD uses fewer pins of the device than the JTAG standard. MiniProg3 supports programming and debugging PSoC 3 and PSoC 5LP devices, using SWD, through the 5-pin or 10-pin connector.

The Single Wire Viewer (SWV) interface, also introduced by ARM, is used for program and data monitoring, where the firmware may output data in a method similar to 'printf' debugging on PCs, using a single pin. MiniProg3 supports monitoring of PSoC 3 and PSoC 5LP firmware, using SWV, through the 10-pin connector and in conjunction with SWD only.

## C.1.4 I<sup>2</sup>C<sup>™</sup>

A common serial interface standard is the Inter-IC Communication ( $I^2C$ ) standard by Philips. It is mainly used for communication between microcontrollers and other ICs on the same board, but can also be used for intersystem communications. MiniProg3 implements an  $I^2C$  multimaster host controller that allows the tool to exchange data with  $I^2C$  enabled devices on the target board. For example, this feature may be used to tune CapSense designs.

For more information on the PSoC 3 and PSoC 5LP JTAG, SWD, SWV, and I<sup>2</sup>C interfaces, see the PSoC 3 and PSoC 5LP Technical Reference Manual. For more information on PSoC 1 interfaces, see the PSoC 1 Technical Reference Manual.



# C.2 Connectors

**Warning** It is recommended that a keyed 10-pin or 5-pin connector be used on the target board applications as programmer/debugger headers for the MiniProg3. The I/Os of the MiniProg3 have very limited series protection against over current. Therefore, plugging the MiniProg3 into a programming/debugger header backwards can potentially damage the MiniProg3.

## C.2.1 5-Pin Connector

The 5-pin connector is configured as a single row with a 100-mil pitch. It is designed to mate with a Molex model 22-23-2051 (straight) or 22-05-3051 (right angle) male header, with key tab. The signal assignment is shown in this figure.





## C.2.2 10-Pin Connector

The 10-pin connector is configured as a dual row with a 50-mil pitch. It is used with a ribbon cable (provided) to mate to a similar connector on the target board. The recommended mating connectors are the Samtec FTSH-105-01-L-DV-K (surface mount) and the FTSH-105-01-L-D-K (through hole) or similar available from other vendors. The signal assignment is shown in this figure.

Figure C-3. 10-Pin Connector with Pin Assignments





Here is a summary of the protocols and related pin assignments.

Protocol	Signal	5-Pin	10-Pin
	SCLK	4	
ISSP	SDAT	5	
	XRES	3	
	TMS		2
	тск		4
JTAG	TDO		6
	TDI		8
	XRES		10
	SDIO	5	2
	SCK	4	4
SWD / SWV	SWV <sup>a</sup>		6
	XRES	3	10
12C	SCK	4	
	SDA	5	

 Table Appendix C-2.
 Communication Protocol Pin Assignments

a. SWV trace is only available in conjunction with SWD debugging.

## C.3 Power

MiniProg3 requires a connection to the Vddio supply of the target device to set the voltage level used for communication. This is required regardless of the communication protocol and the port selected. One of the connectors' VTARG pins must be connected to the Vddio supply of the target device. For PSoC 3 and PSoC 5LP, this is the Vddio1 supply, because this is the supply used to drive the debug pins. Failing to connect VTARG, or connecting it to the wrong supply results in the MiniProg3 being unable to communicate with the target device.

On boards where there is a single power supply for the entire board, the MiniProg3 can, in some cases, supply power to the board. This supply is limited to approximately 200 mA and is protected against excess current draw. The power supply voltage can be selected from one of 1.8 V, 2.5 V, 3.3 V, or 5 V. The 5-V supply may be as low as 4.25 V or as high as 5.5 V, as it is supplied directly from the USB port.

# Appendix D. PSoC Creator DWR



The PSoC Creator Design-Wide Resources (DWR) system provides a single location to manage all the resources in your design. These resources include pins, clocks, interrupts, DMA, and so on. Each new design project provides a default design-wide resources file (.cydwr) file with the same name as the project.

A brief explanation of each tab is provided here. See **Help > Topics > Using Design Entry Tools > Design-Wide Resources** for more details of each editor in the DWR file.

The Pins tab of the DWR file or the Pin Editor allows you to manually assign the pins used in the schematic to the PSoC.



Figure Appendix D-1. DWR File - Pin Editor

The Analog tab or the Analog Device Editor provides an interconnect view of the PSoC 3 and PSoC 5 devices along with place-and-route results for a particular design. The editor also allows for manual place-and-route with the ability to lock-down all or some of the results.





Figure Appendix D-2. DWR File - Analog Device Editor

The Clocks tab or the Clock Editor is a design-wide resources tool to create and edit clocks. This tool allows you to view all clocks, add and delete design-wide clocks, as well as edit design-wide and system clocks.

Figure Appendix D-3.	<b>DWR File - Clock Editor</b>
i iguie / appendix B er	Difference Clock Ealter

Start G Add D	esign-Wide Clock		LED_wPWM	-					- 4 ▷ :
Туре 🗡	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on . Reset	<ul> <li>Source Clock</li> </ul>
System	USB_CLK	DIGITAL	48.000 MHz	? MHz	±0	-	1		IM0x2
System	Digital Signal	DIGITAL	? MHz	? MHz	±0	-	0		
System	XTAL 32kHz	DIGITAL	32.768 kHz	? MHz	±0	-	0		
System	XTAL	DIGITAL	24.000 MHz	? MHz	±0	-	0		
System	ILO	DIGITAL	? MHz	1.000 kHz	-50, +100	-	0		
System	IMO	DIGITAL	3.000 MHz	3.000 MHz	±5	-	0		
System	BUS_CLK (CPU)	DIGITAL	? MHz	24.000 MHz	±5	-	1		MASTER_CLK
System	MASTER_CLK	DIGITAL	? MHz	24.000 MHz	±5	-	1		PLL_OUT
System	PLL_OUT	DIGITAL	24.000 MHz	24.000 MHz	±5	-	0		IMO
Local	Clock_1	DIGITAL 🐱	? MHz	100.000 Hz	-50, +100	-	10	<b>V</b>	ILO
🐝 P	ins M Analog	Clocks	🍠 Interrupts	So dma	🌶 System 📘	Directives	Fl.	ash Security	4

The Interrupts tab or the Interrupt Editor allows you to change the priority of interrupt service routines (ISRs) in your design.



Start Page Ex	4_USB_HID.c	ydwi	n	- ₫
nstance Name	Priority	1	Vector	
JSBFS_1_sof_int	Default <7>	*	Unknown	
JSBFS_1_arb_int	Default <7>	*	Unknown	
SBFS_1_bus_reset	Default <7>	*	Unknown	
SBFS_1_ep_0	Default <7>	*	Unknown	
SBFS_1_ep_1	Default <7>	*	Unknown	
BFS_1_ord_int	Default <7>	*	Unknown	
SBFS_1_dp_int	Default <7>	~	Unknown	

Figure Appendix D-4. DWR File - Interrupt Editor

**Note** If no interrupts are used in your design, the Interrupt Editor gives the message, as shown in the following figure.

Figure Appendix D-5. Message for No Interrupts



The DMA tab or the DMA Editor displays all the direct memory access (DMA) components that have been directly placed in the design, as well as all the DMA components "inside" placed components.



Figure Appendix D-6. DWR File - DMA Editor

Start Page Ex3_ADC_tDAC.cydwr	<b>→</b> 4 Þ <b>→</b>
Name Priority / Channel Number	
DMA_1 Default <2> V Unknown	
🐗 Pins 🕔 Analog 🕑 Clocks 💉 Interrupts 🔡 DMA 🦻 System 🖺 Directives 🧃 Flash Security	
💣 Pins 🛝 Analog 🕑 Clocks 🕖 Interrupts 🖧 DMA 🍢 System 🖺 Directives 🧃 Flash Security	4 Þ

Similar to the Interrupt editor, if there is no DMA component used in the design DMA editor shows the message that there is no DMA component being used.

Figure Appendix D-7. No DMA Component



The System tab or the System Editor is used to edit various system properties. It contains a table with different categories of properties, such as Configuration, Programming/Debugging, and Operating Conditions. The available categories change based on your design.



Figure Appendix D-8. DWR File - System Editor

Configuration         Configuration Mode       Compressed         Use Dedicated Configuration Data Memory       Image: Configuration Data Memory         Image: Configuration Data Memory       AlowButWarn         Image: Configuration Data Memory       AlowButWarn         Image: Configuration Data Memory       AlowButWarn         Image: Configuration Data Memory       Subol Data Memory         Image: Configuratin Data Memory       Subol Data Memory	Option		Value	
- Use Dedicated Configuration Data Memory	- Configura	ration		
Instruction Cache Enabled         Image: Cache Enabled           Image: Instruction Cache Enabled         Image: Cache Enabled         Image: Cache Enabled           Image:	- Devi	rice Configuration Mode	Compressed	
-         -	- Use	Dedicated Configuration Data Memory		
Heap Size         Dx1000           Stack Size         0x4000           Debug Select         SWD+SWV (serial wire debug and viewer)           Enable Device Protection         0           Detug Select         SUD+SWV (serial wire debug and viewer)           Verting Conditions         0           Vdda         S0.0           Vddio0         S0.0           Vddio1         S0.0           Vddio2         S0.0           Vddio2         S0.0	- Instru	ruction Cache Enabled		
Stack Size         0x4000           Programming\Debugging         SWD+SWV (serial wire debug and viewer)           Enable Device Protection         Image: Conditions           Vddd         S0           Vdda         S0           Vddio0         S0           Vddio0         S0           Vddio0         S0           Vddio1         S0           Vddio2         S0           Vddio2         S0	- Unu:	ised Bonded IO	AllowButWarn	
Programming\Debugging           Debug Select         SWD+SWV (serial wire debug and viewer)           Enable Device Protection	- Heap	ap Size	<mark>0x1000</mark>	
Debug Select         SWD+SWV (serial wire debug and viewer)           Enable Device Protection	- Stac	ck Size	<mark>0x4000</mark>	
nable Device Protection         Image: Conditions           Dereting Conditions         5.0           Vdda         5.0           Vdda         5.0           Vddio0         5.0           Vddio1         5.0           Vddio2         5.0           Vddio2         5.0	Program	ming\Debugging		
Derating Conditions         Derating Conditions           - Vddd         5.0           - Vdda         5.0           - Vddio0         5.0           - Vddio1         5.0           - Vddio2         5.0           - Vddio3         5.0	Debu	pug Select	SWD+SWV (serial wire debug and viewe	)
- Vdd         5.0           - Vdda         5.0           - Vddio0         5.0           - Vddio1         5.0           - Vddio2         5.0           - Vddio2         5.0	- Enat	ble Device Protection		
- Vdda         5.0           - Vddio0         5.0           - Vddio1         5.0           - Vddio2         5.0           - Vddio2         5.0           - Vddio2         5.0	- Operating	ng Conditions		
- Vddio0         5.0           - Vddio1         5.0           - Vddio2         5.0           - Vddio3         5.0	- Vddo	ld	5.0	
- Vddio1         5.0           - Vddio2         5.0           - Vddio3         5.0	- Vdda	la	5.0	
- Vddio2         5.0           - Vddio3         5.0	- Vddi	lioO	5.0	
- Vddio3 5.0	- Vddi	lio1	5.0	
	- Vddi	lio2	5.0	
Temperature Range 40C - 85C	Vddi	lio3	5.0	
	Tem	nperature Range	<mark>-40C - 85C</mark>	

The Directives tab or the Directives Editor is used to add, remove, and edit directives. Directives may be used to influence the implementation of a design. They are used in an iterative fashion to refine, improve, or constrain the results of synthesis. Directives may be applied to components that have been either instantiated in a schematic or inferred by the synthesizer from Verilog HDL code.

See Help > Topics > PSoC Creator > Building a PSoC Creator Project > Directives for more information on the directives available in PSoC Creator.



Figure Appendix D-9. DWR File - Directives Editor



The Flash security tab or the Flash Security Editor allows you to control the read/write access to the device memory. The flash rows are displayed as a table where each editable cell in the table represents a single row of flash (256 bytes). Each flash row can have its protection level independently set.

m row:	0 🛟 to 102 🕻	U	- Unprotec	cted	~	S	et												
	OFFSET:	000	100	200	300	400	500	600	700	800	900	A00	B00	C00	D00	E00	F00	Row	
	BASE ADDR: 0000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	0-15	
	1000	U	U	U	U	U	F • F. 💙	U	U	U	U	U	U	U	U	U	U	16-31	
	2000	U	U	U	U	U	U Unpro F - Factor	ry Upgrade	3		U	U	U	U	U	U	U	32-47	
	3000	U	U	U	U	U	R - Field I W - Full F				U	U	U	U	U	U	U	48-63	
	4000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	64-79	
	5000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	80-95	
	6000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	96-111	
	7000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	112-127	
	8000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	128-143	
	9000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	144-159	
	A000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	160-175	
	B000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	176-191	
	C000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	192-207	
	D000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	208-223	
	E000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	224-239	
	F000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	240-255	
	10000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	256-271	
	11000	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	272-287	
	12000	Ш	ш	Ш	ш	Ш	П	Ш	11	ш	Ш	ш	Ш	Ш	Ш	Ш	Ш	288-303	

Figure Appendix D-10. DWR File - Flash Security Editor