CY8CKIT-001

PSoC® Development Kit Guide

Doc. # 001-48651 Rev. *K
July 4, 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

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. Introduction **7**

1.1 Kit Overview....7
1.2 Kit Contents....7
1.3 Installation....8
1.3.1 Before You Begin....8
1.3.2 Prerequisites....8
1.3.3 Installing PSoC 1 Development Software....8
1.3.4 Installing PSoC 3 Development Software....8
1.3.5 Installing PSoC 5 Development Software....9
1.4 PSoC Development Board....10
1.4.1 Default Switch and Jumper Settings....10
1.5 Kit Revision....12
1.6 Additional Resources....13
1.6.1 Beginner Resources....13
1.6.2 Engineers Looking for More....13
1.6.3 Learning from Peers....13
1.7 Document Conventions....13
1.8 Document Revision History....14

2. Loading My First PSoC Project **15**

2.1 My First PSoC 1 (CY8C28) Project....16
2.1.1 Loading My First PSoC 1 Project....16
2.1.2 Building My First PSoC 1 Project....17
2.1.3 Programming My First PSoC 1 Project....17
2.1.4 Running My First PSoC 1 Project....18
2.2 My First PSoC 1 (CY8C29) Project....20
2.2.1 Loading My First PSoC 1 Project....20
2.2.2 Building My First PSoC 1 Project....21
2.2.3 Programming My First PSoC 1 Project....22
2.2.4 Running My First PSoC 1 Project....23
2.3 My First PSoC 3 (CY8C38) Project....25
2.3.1 Loading My First PSoC 3 Project....25
2.3.2 Building My First PSoC 3 Project....26
2.3.3 Programming My First PSoC 3 Project....27
2.3.4 Running My First PSoC 3 Project....28
2.4 My First PSoC 5 (CY8C55) Project....28
2.4.1 Loading my First PSoC 5 Project....28
2.4.2 Building My First PSoC 5 Project....30
2.4.3 Programming My First PSoC 5 Project....31
2.4.4 Running My First PSoC 5 Project....32

3. Sample Projects 33

3.1 CY8C28 Family Processor Module Code Examples 33
3.1.1 My First PSoC 1 (CY8C28) Project 33
3.1.2 ADC to LCD Project 41
3.1.3 ADC to UART with DAC 47
3.1.4 CapSense 59
3.2 CY8C29 Family Processor Module Code Examples 69
3.2.1 My First PSoC 1 (CY8C29) Project 69
3.2.2 ADC to LCD Project 77
3.2.3 ADC to LCD with DAC and UART 84
3.3 CY8C38 / CY8C55 Family Processor Module Code Examples 100
3.3.1 My First PSoC 3 / PSoC 5 Project 100
3.3.2 ADC to LCD Project 107
3.3.3 ADC to UART with DAC 115
3.3.4 USB HID 135
3.3.5 CapSense 149
3.3.6 SAR ADC (PSoC 5 Only) 158

Appendix A. Board Specifications and Layout 177

A.1 PSoC Development Board 177
A.1.1 Factory Default Configuration 177
A.1.2 Power Supply Configuration Examples 178
A.1.3 Prototyping Components 183
A.1.4 LCD Module 185
A.1.5 CapSense Elements 186
A.1.6 Processor Module 186
A.1.7 Expansion Ports 189
A.2 Schematics 192
A.2.1 CY8CKIT-001 PSoC Development Board 192
A.2.2 CY8C28 Family Processor Module 193
A.2.3 CY8C29 Family Processor Module 194
A.2.4 CY8C38 Family Processor Module 195
A.2.5 CY8C55 Family Processor Module 196
A.2.6 Enabling Boost Component in PSoC 3 and PSoC 5 Processor Modules 197
A.3 Bill of Materials 197
A.3.1 CY8CKIT-001 PSoC Development Board 197
A.3.2 CY8C28 Family Processor Module 199
A.3.3 CY8C29 Family Processor Module 200
A.3.4 CY8C38 Family Processor Module 201
A.3.5 CY8C55 Family Processor Module 202

Appendix B. MiniProg3 203

B.1 MiniProg3 LEDs 203
B.2 Programming in Power Cycle Mode 203
B.3 Interface Pin Assignment Table 203
B.4 Protection Circuitry 204
B.5 Level Translation 204

Appendix C. MiniProg3 Technical Description 205

C.1 Interfaces 206
C.1.1 ISSP 206
C.1.2 JTAG 206

C.1.3 SWD/SWV....206
C.1.4 I2C™....206
C.2 Connectors....207
C.2.1 5-Pin Connector....207
C.2.2 10-Pin Connector....207
C.3 Power....208

Appendix D. PSoC Creator DWR 209

1. Introduction

1.1 Kit Overview

The CY8CKIT-001 PSoC® Development Kit provides a common development platform where you can prototype and evaluate different solutions using either the PSoC 1, PSoC 3, or PSoC 5 architectures. 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 CY8C55 family processor modules.

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 CY8C55 family processor module
- MiniProg3 programmer and debug tool
- USB cable[1]
- 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™ IDE
  - PSoC Programmer software
  - CY8C28 datasheets
  - Kit release notes
  - Software release notes
  - Code example files, firmware, and documentation
- PSoC 3 and PSoC 5 software DVD (contents are installed in the `\CY8CKIT-009A` folder for PSoC 3 module kit and `\PSoC Development Kit CY8C55` folder for PSoC 5 module kit):
  - PSoC Creator™ IDE
  - PSoC Programmer software
  - CY8C38 datasheet
  - CY8C55 datasheet
  - Kit release notes
  - Software release notes
  - Code example files, firmware, and documentation

1. 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 http://www.cypress.com.

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** → **Control Panel** → **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 5 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** → **Register** → **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 → Help → Topics → Getting Started**
- **Programmer → Documentation → 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 → 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 5 Development Software

To use the CY8C55 family processor module (PSoC 5), you need:

- PSoC Creator 2.0 or later
- PSoC Programmer 3.12.3 or later
- PSoC Development Kit example files

Insert the PSoC 3 or PSoC 5 Software DVD; in the menu, select **Install Software for PSoC 5**. 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 5 applications.

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

- **PSoC Creator → Help → Topics → Getting Started**
- **Programmer → Documentation → 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 → 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>:\PSoC Development Kit CY8C55\<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)

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 2×16 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™ 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® applications.

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

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

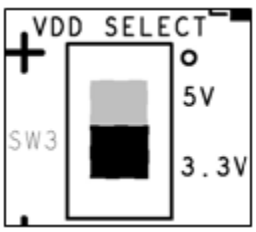
- The PSoC device may get hot or damaged if many I/O pins are configured as strong drive with initial state HIGH and grounded externally using wires.
- The PSoC device may get hot or damaged if many I/O pins are configured as strong drive with initial state LOW and connected to Vcc externally using wires.

1.4.1 Default Switch and Jumper Settings

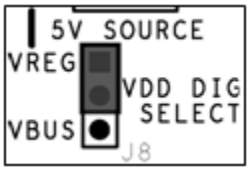
Jumpers on the CY8CKIT-001 PSoC development board have a default setting to operate at 3.3 V. For default configuration, each of the jumpers must be set according to these instructions.

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.

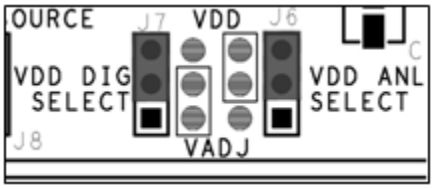
SW3 - VDD Select. Default Position: 3.3 V (down position)



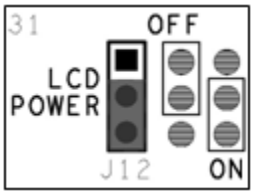
J8 - 5 V Source. Default Position: VREG (upper two pins)



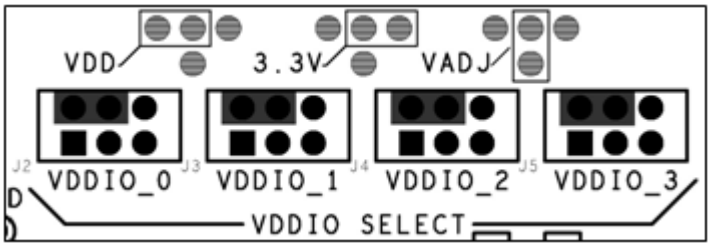
J7, J6 - VDD Digital, VDD Analog. Default Position: VDD (upper two pins, both headers)



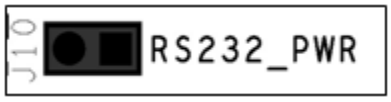
J12 - LCD Power. Default Position: ON (lower two pins)



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-001B Rev **, then congratulations, you own the latest version.

You can also check the silicon marketing part number on the processor module. If the part number is CY8C3866AXI-040, then congratulations, you own the latest version.

To upgrade CY8CKIT-001A to CY8CKIT-001B, the PSoC 3 processor module and kit DVD must be updated. Purchase the latest processor module at http://www.cypress.com/go/CY8CKIT-009 and download the latest DVD ISO image at http://www.cypress.com/go/CY8CKIT-001.

To upgrade CY8CKIT-001 to CY8CKIT-001B, besides the upgrades stated above, you need to purchase the latest PSoC 5 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 Oscillator

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 5.

1.6.3 Learning from Peers

Cypress Developer Community Forums

1.7 Document Conventions

These conventions are used throughout this guide.

Table 1-1. Documentation Conventions

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**]
Bold → With → Arrows	Represents menu paths, user entered text: **File → New Project → Clone**
Bold	Displays commands and selections, and icon names in procedures: Click the **Debugger** icon, and then click **Next**.
Note	Displays functionality unique to PSoC Designer, PSoC Creator, or the PSoC device.
WARNING:	Displays cautions that are important to the subject.

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
*B	11/19/09	AESA	CDT based updates
*C	05/21/10	AESA	Updated with PSoC 5.
*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
*I	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.

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:

1. Install PSoC Designer using the steps listed in Installing PSoC 1 Development Software on page 8.
2. Install PSoC Creator using the steps listed in Installing PSoC 3 Development Software on page 8.
3. Connect the MiniProg3 into your PC using the supplied USB cable. When you connect the MiniProg3, Microsoft Windows® 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 10.
6. 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 16 or My First PSoC 1 (CY8C29) Project on page 20).
7. 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 25).
8. Use the PSoC CY8C55 family processor module for the PSoC 5 version of your first PSoC project (My First PSoC 5 (CY8C55) Project on page 28).
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 5 project, use the JTAG ribbon cable. Connect the ribbon cable to the MiniProg3 and the CY8C38 family processor module or CY8C55 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.

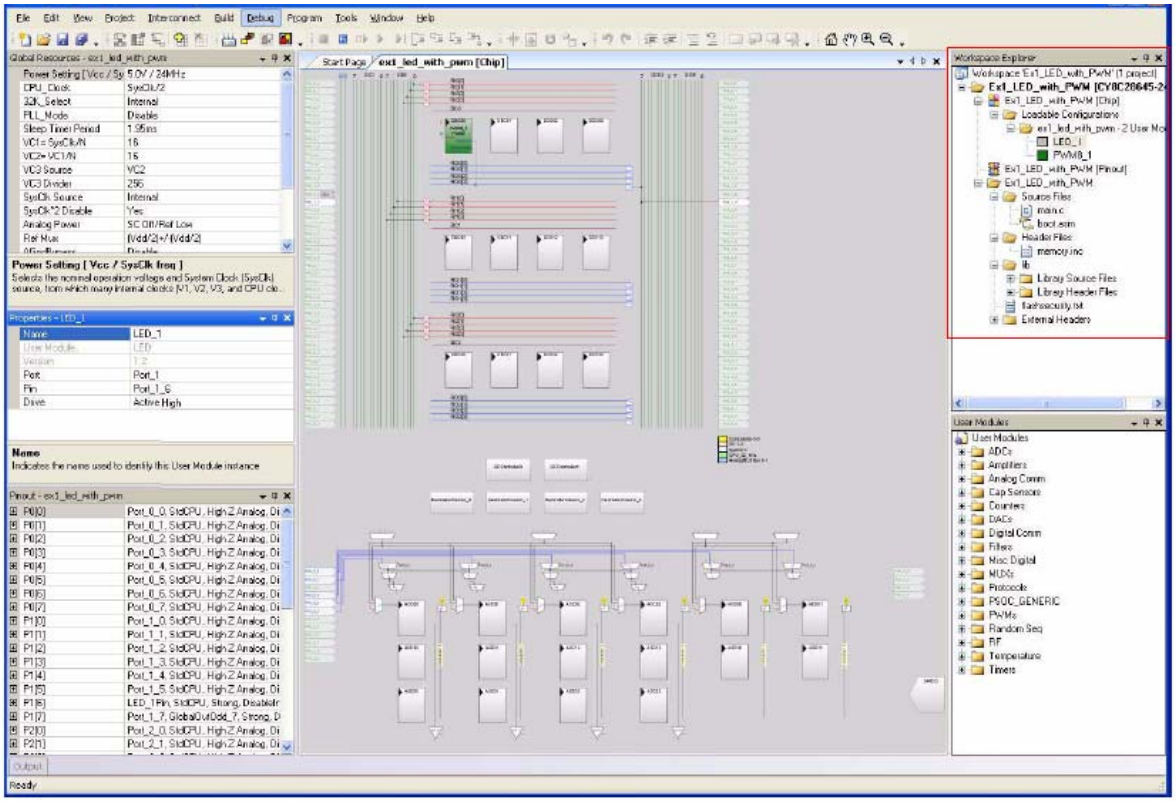
2.1 My First PSoC 1 (CY8C28) Project

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** → **Open Project/Workspace**
3. Navigate to the project directory: `<Install_Directory>:\PSoC Developent 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**.

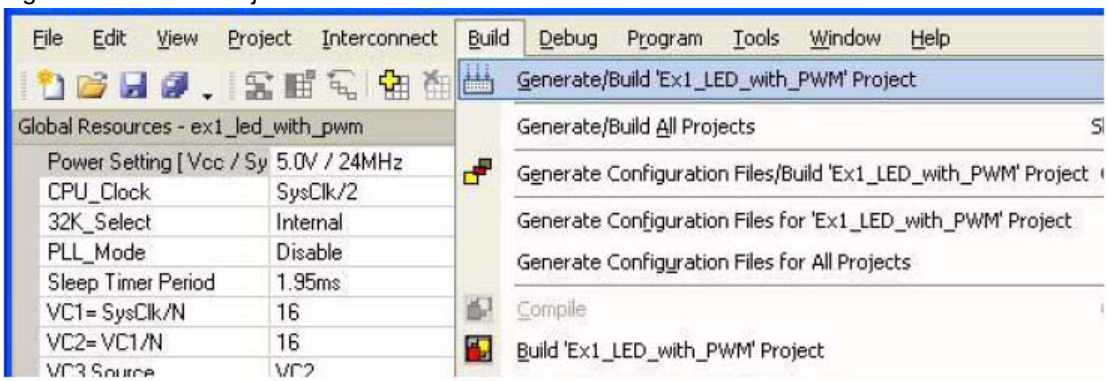
Figure 2-1. Chip Editor View



2.1.2 Building My First PSoC 1 Project

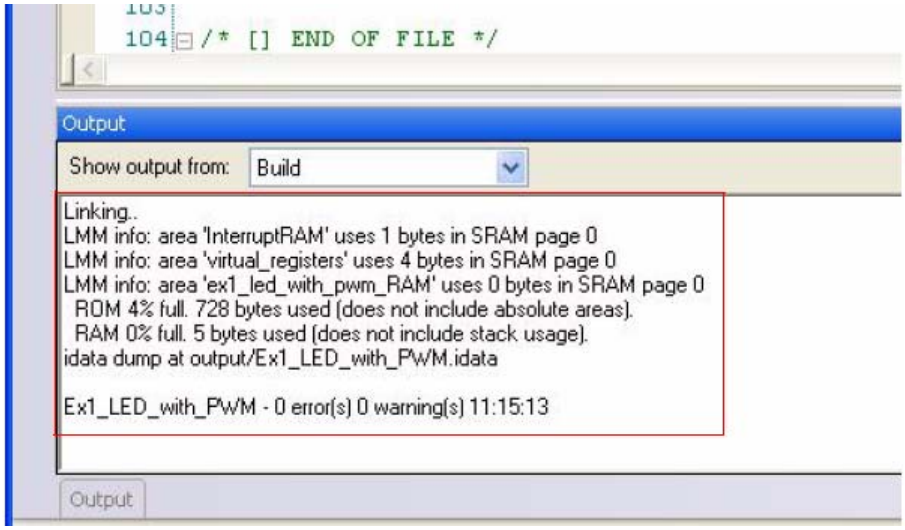
1. Select **Build** → **Generate/Build 'Ex1_LED_with_PWM' Project**.

Figure 2-2. Build Project



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



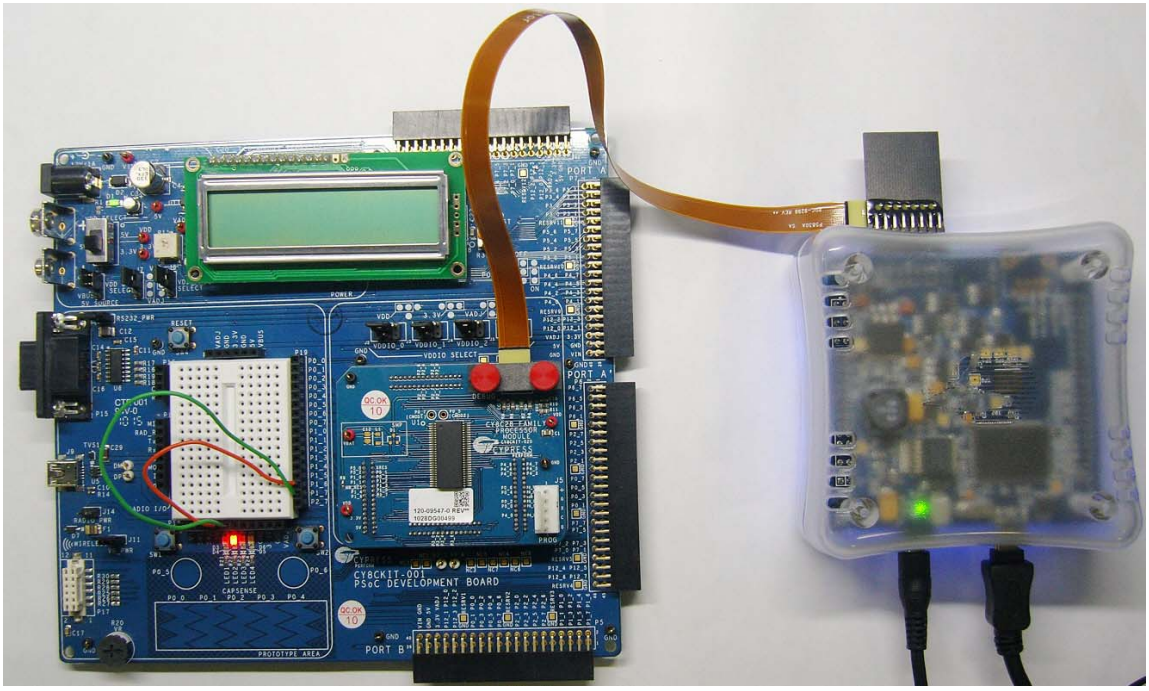
2.1.3 Programming My First PSoC 1 Project

1. Open **Program Part** from within PSoC Designer by selecting **Program** → **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 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.
6. In the **Program Part** window, click the program arrow to program the device.
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** → **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 2-4.

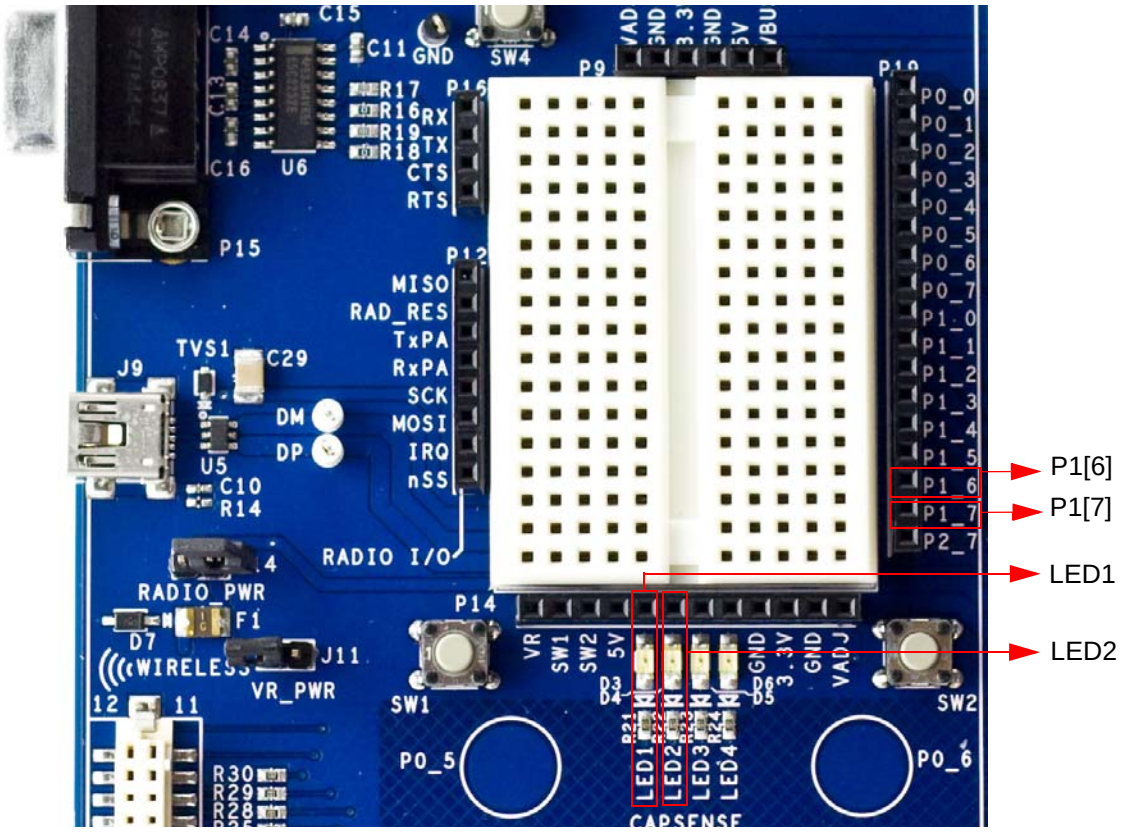
Figure 2-4. ICE-Cube Connected to CY8C28 (PSoC 1) Processor Module - Debugging the Ex1_LED_with_PWM Code Example



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 2-5. 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 33.

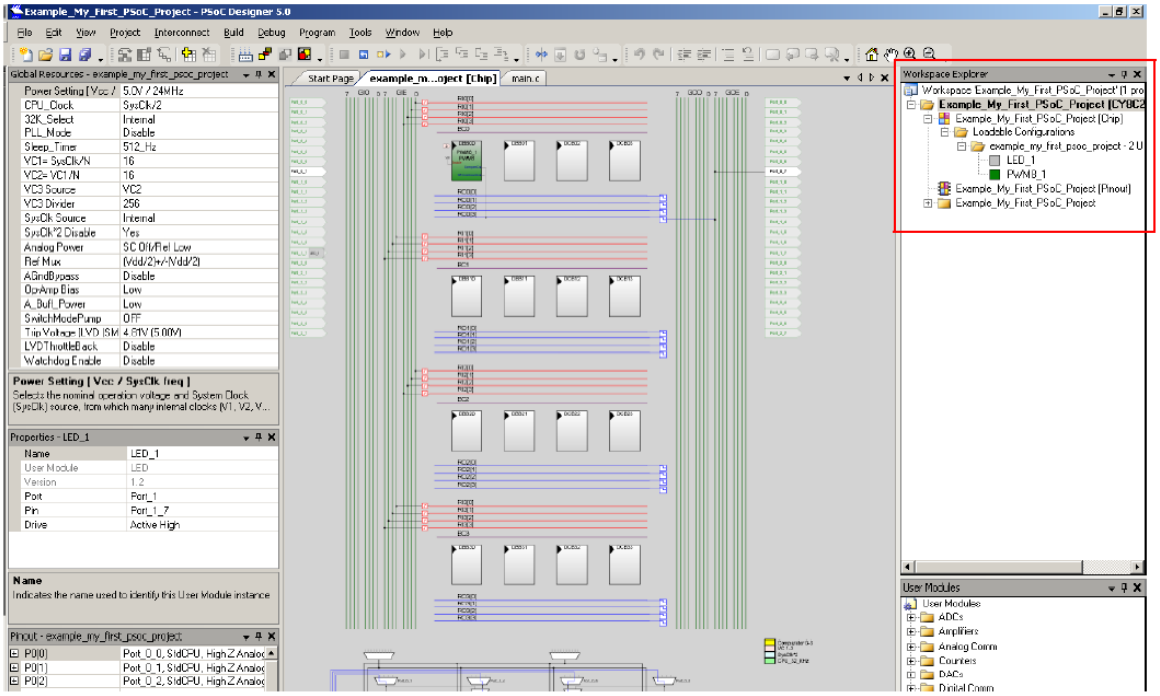
2.2 My First PSoC 1 (CY8C29) Project

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** → **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**.

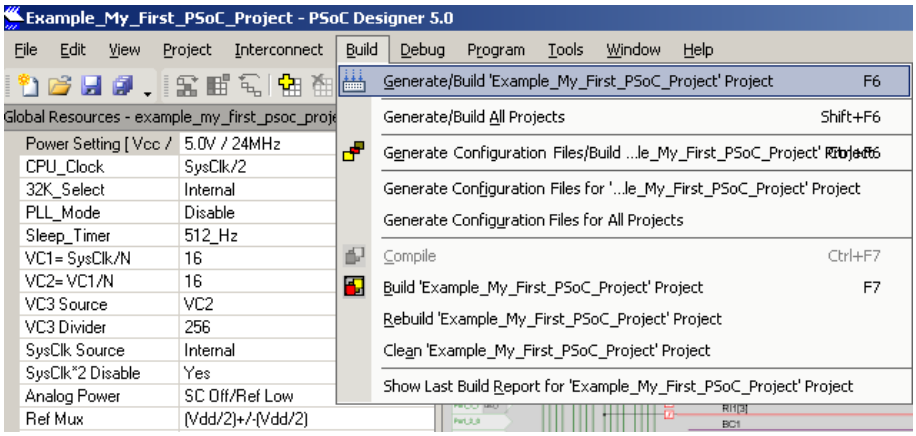
Figure 2-6. Chip Editor View



2.2.2 Building My First PSoC 1 Project

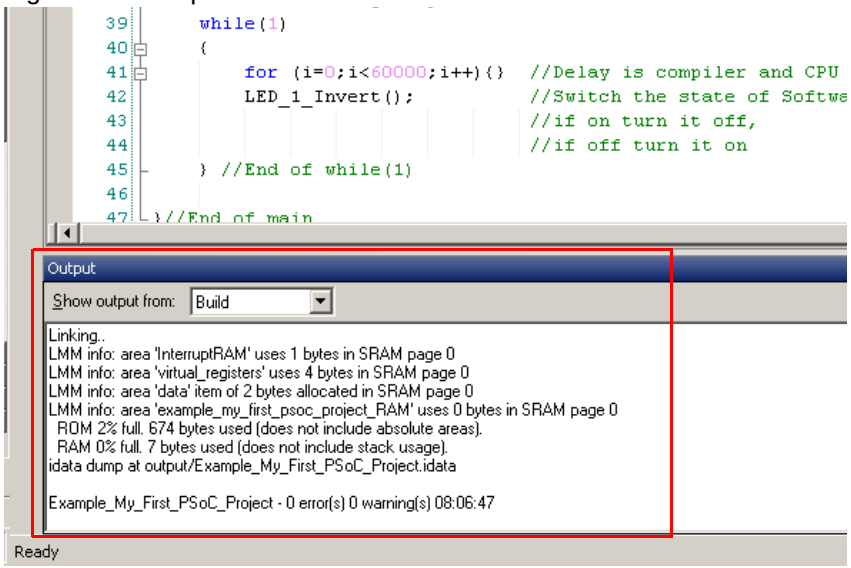
1. Select **Build** → **Generate/Build 'Example_My_First_PSoC_Project' Project**.

Figure 2-7. Build Project



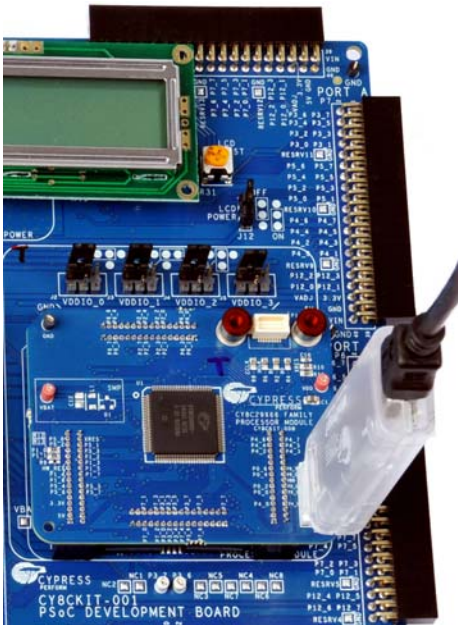
2. PSoC Designer builds the project and displays comments in the **Output** window. When you see the message that the project is built with 0 errors and 0 warnings, you are ready to program the device.

Figure 2-8. Output Window



2.2.3 Programming My First PSoC 1 Project

Figure 2-9. Connect MiniProg3 to J5 on CY8C29 Family Processor Module

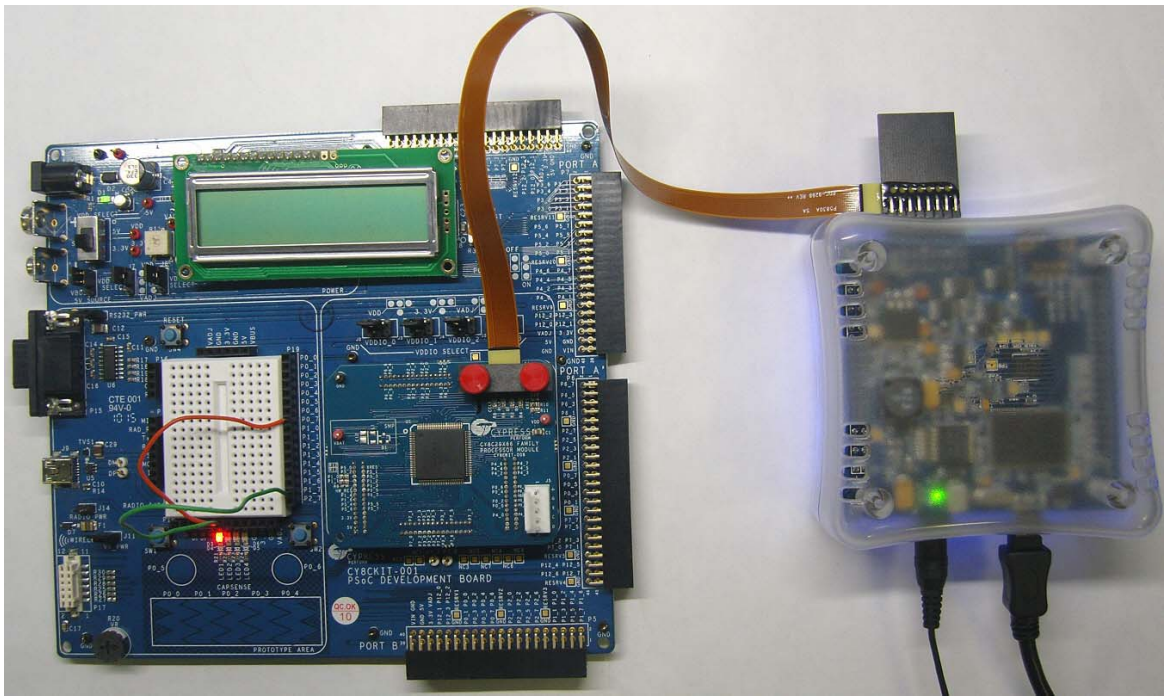


1. Open PSoC Programmer from within PSoC Designer by selecting **Program** → **PSoC Programmer**.
2. In PSoC Programmer, make sure that **MiniProg3** is selected in the **Port Selection** box.
3. In PSoC Programmer, set **Programming Mode** to **Reset**.
4. In PSoC Programmer, set **Verification** to **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** → **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 2-10.

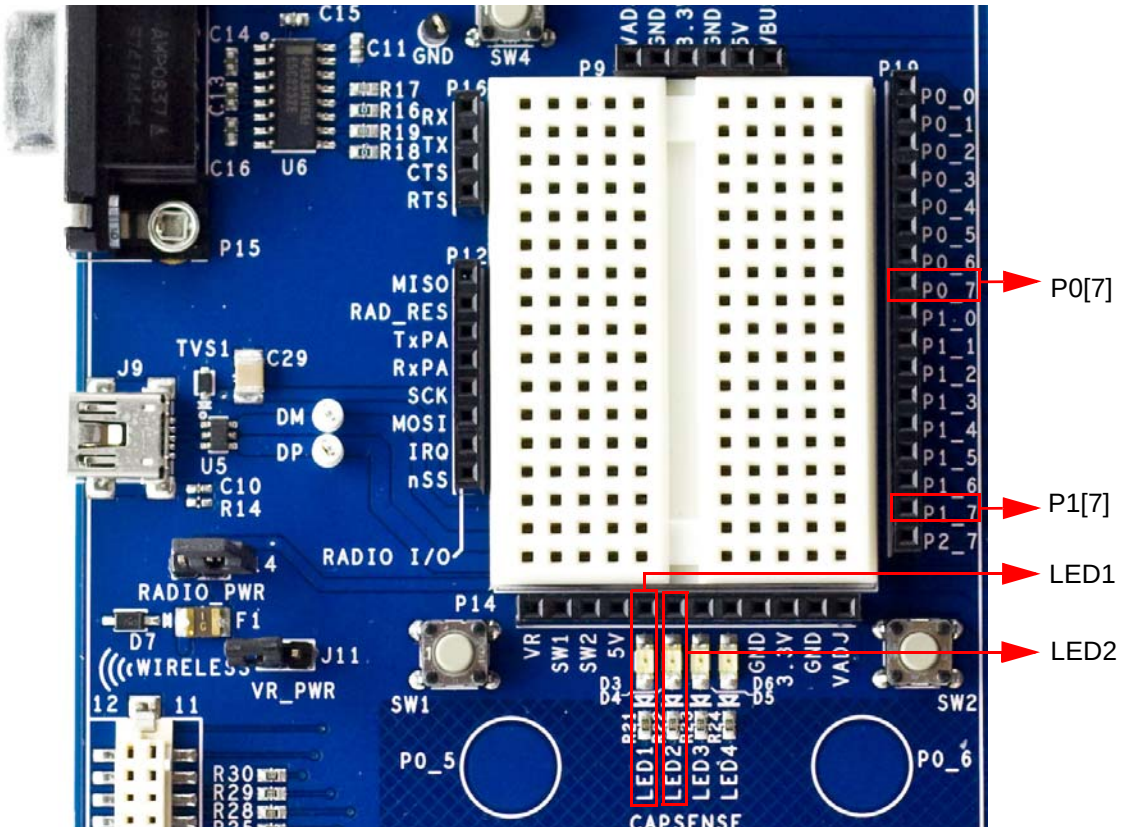
Figure 2-10. ICE-Cube Connected to CY8C29 (PSoC 1) Processor Module - Debugging the Example_My_First_PSoC_Project Code Example



2.2.4 Running My First PSoC 1 Project

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 2-11. 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 69.

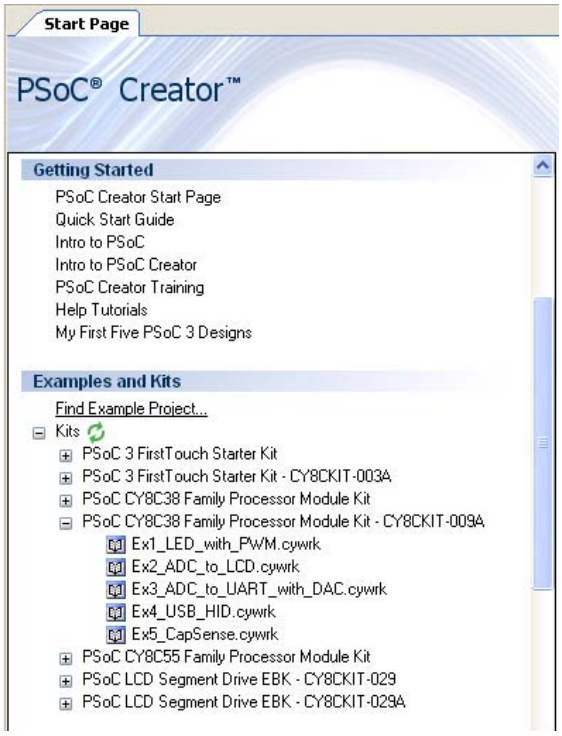
2.3 My First PSoC 3 (CY8C38) Project

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

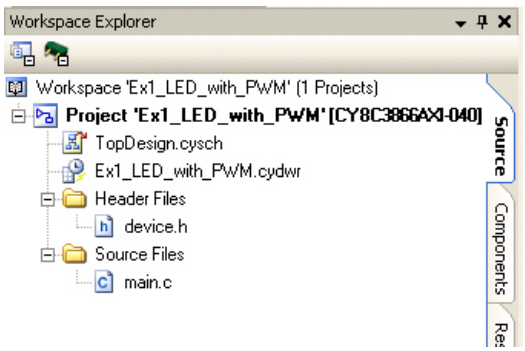
1. Open PSoC Creator.
2. In the **Start Page**, under **Start Page Topics** expand **Kits**.
3. Under **Kits**, expand **PSoC CY8C38 Family Processor Module Kit**.
4. Click **Ex1_LED_with_PWM.cywrk** to open the project.

Figure 2-12. Kits List



5. Select the directory to store the project.
6. After the project opens, you can see the project files in **Workspace Explorer** (see Figure 2-13).

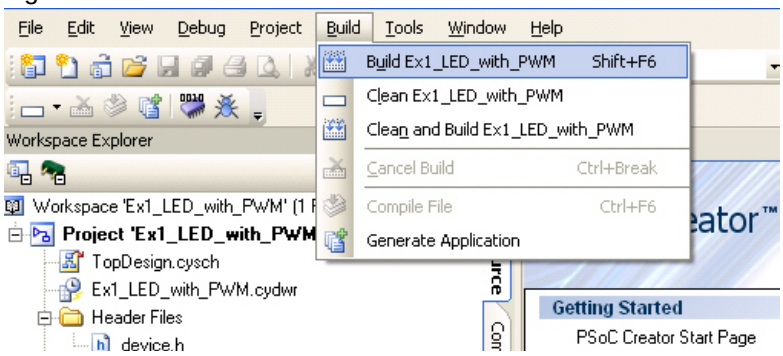
Figure 2-13. Workspace Explorer



2.3.2 Building My First PSoC 3 Project

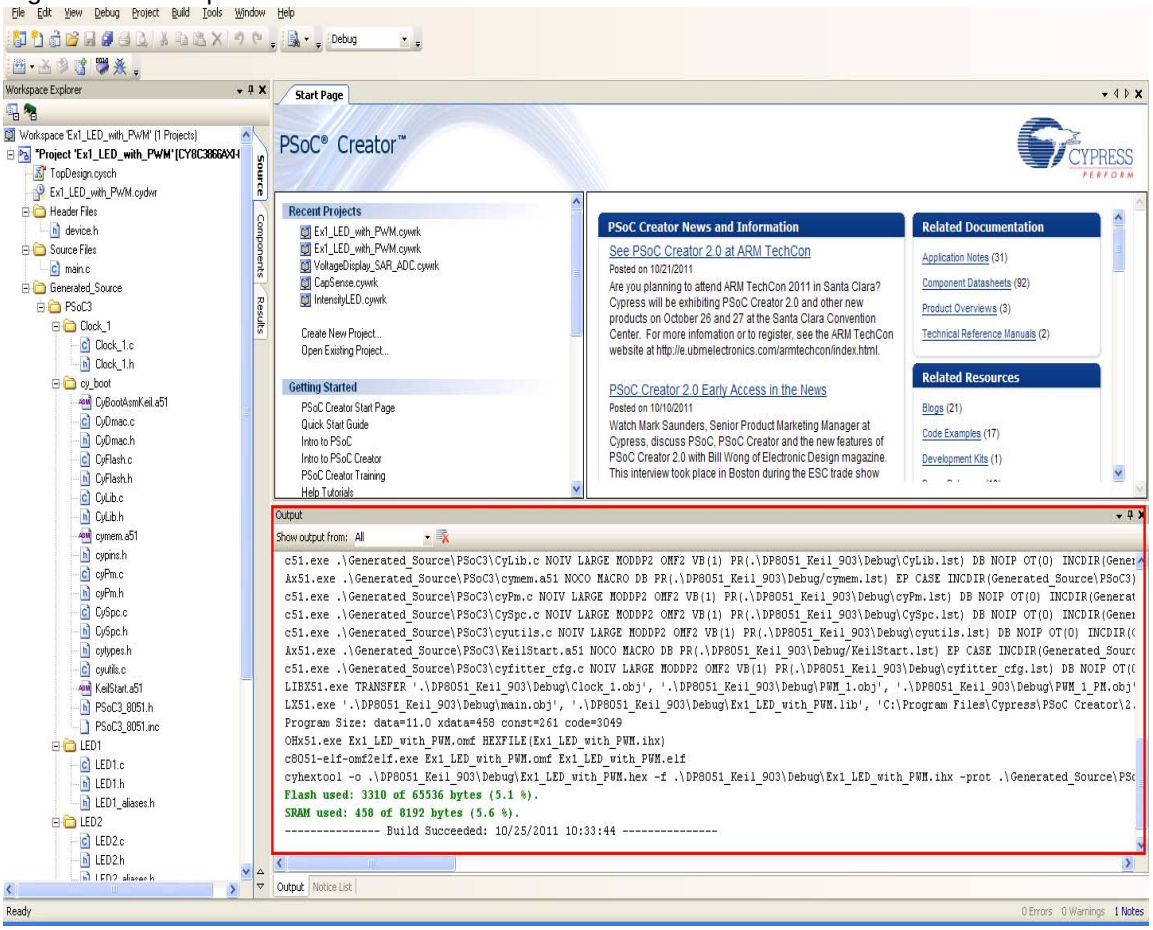
1. Select **Build** → **Build Ex1_LED_with_PWM**.

Figure 2-14. Build Window



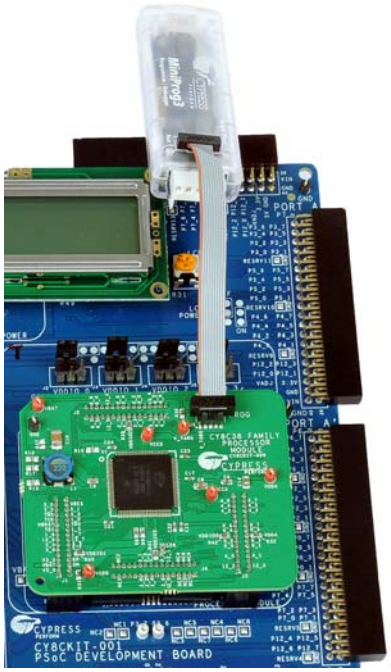
2. PSoC Creator builds the project and displays the comments in the **Output** window. When you see the message "Build Succeeded", you are ready to program the device.

Figure 2-15. Output Window



2.3.3 Programming My First PSoC 3 Project

Figure 2-16. 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.

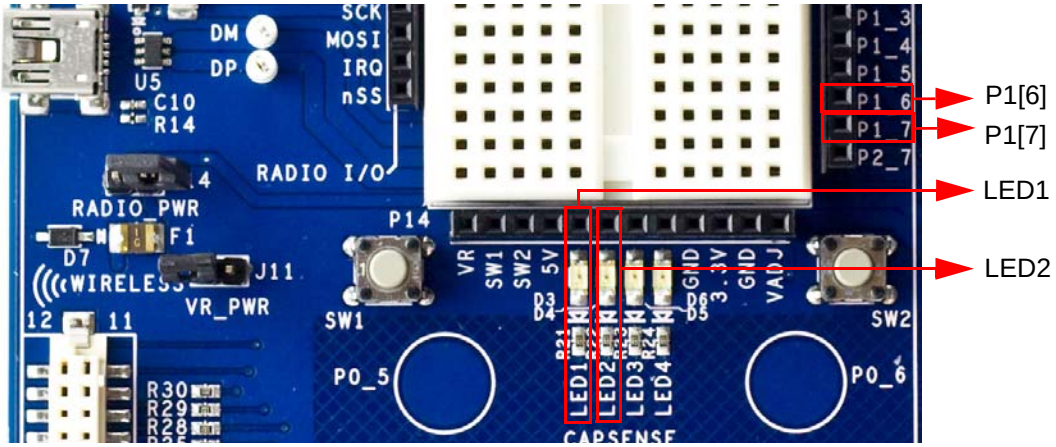
   Note VTARG of the MiniProg3 is wired exclusively to VDDIO1 of the chip on the PSoC CY8C38 family processor module. Because of this, you cannot perform power cycle mode programming.

   - From the **Tools** menu in PSoC Creator, click **Options**. The **Options** window opens.
   - In the **Options** window, select **Program/Debug** → **Port Configuration** → **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.
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 LED2 blinks about three times a second.

Figure 2-17. 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 5 Project on page 100.

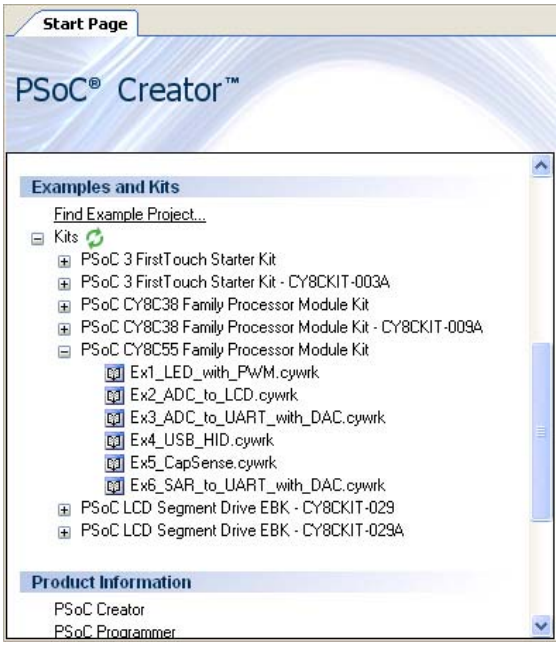
2.4 My First PSoC 5 (CY8C55) Project

This project uses a PWM peripheral programmed from inside PSoC 5 to control the blinking rates of two different LED outputs. For this project, insert the PSoC CY8C55 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 5 Project

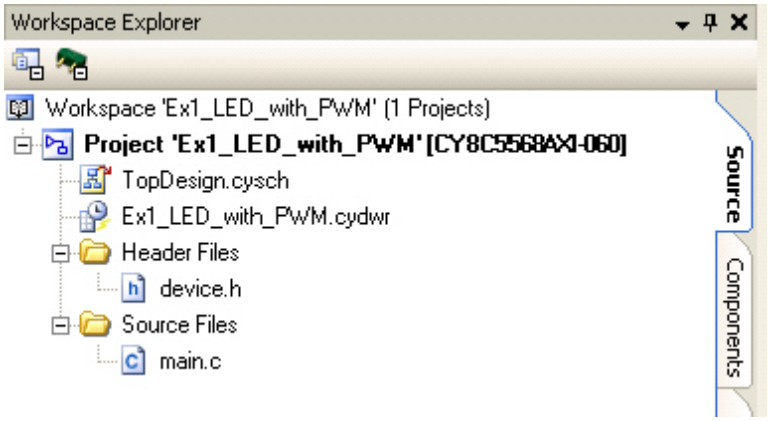
1. Open PSoC Creator.
2. In the **Start Page**, under **Start Page Topics** expand **Kits**.
3. Under **Kits**, expand **PSoC CY8C55 Family Processor Module Kit**.
4. Click **Ex1_LED_with_PWM.cywrk** to open the project.

Figure 2-18. Kits List



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

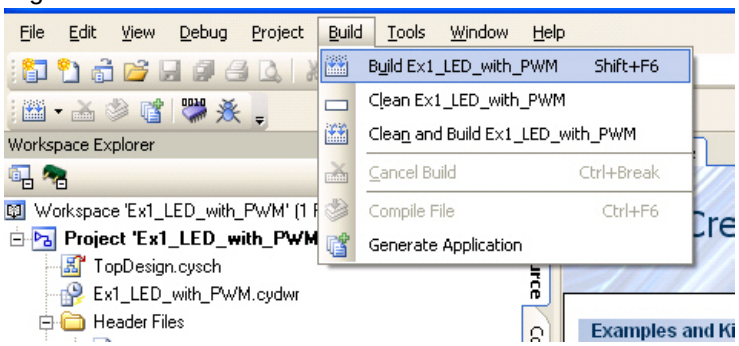
Figure 2-19. Workspace Explorer



2.4.2 Building My First PSoC 5 Project

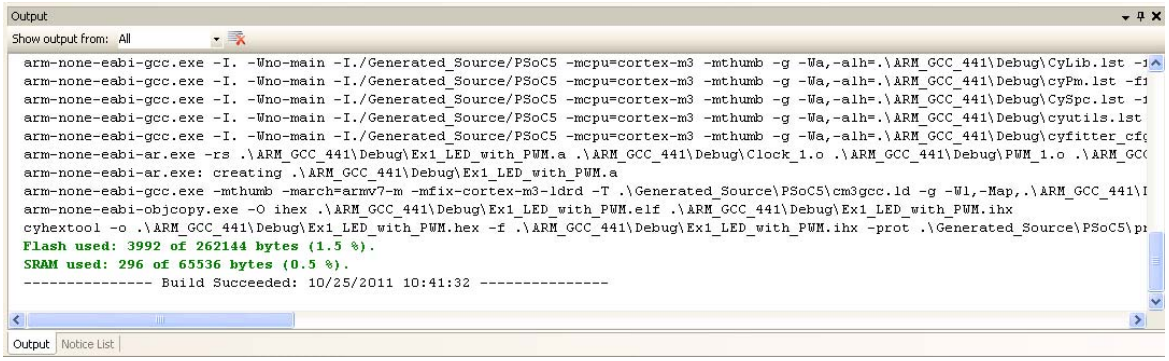
1. Select **Build** → **Build Ex1_LED_with_PWM**.

Figure 2-20. Build Window



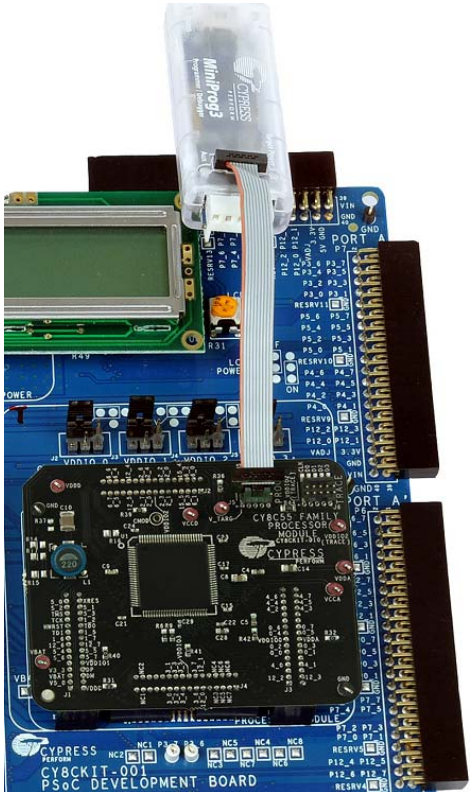
2. PSoC Creator builds the project and displays the comments in the Output window. When you see the message "Build Succeeded", you are ready to program the device.

Figure 2-21. Output Window



2.4.3 Programming My First PSoC 5 Project

Figure 2-22. Connect MiniProg3 to J5 on CY8C55 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.

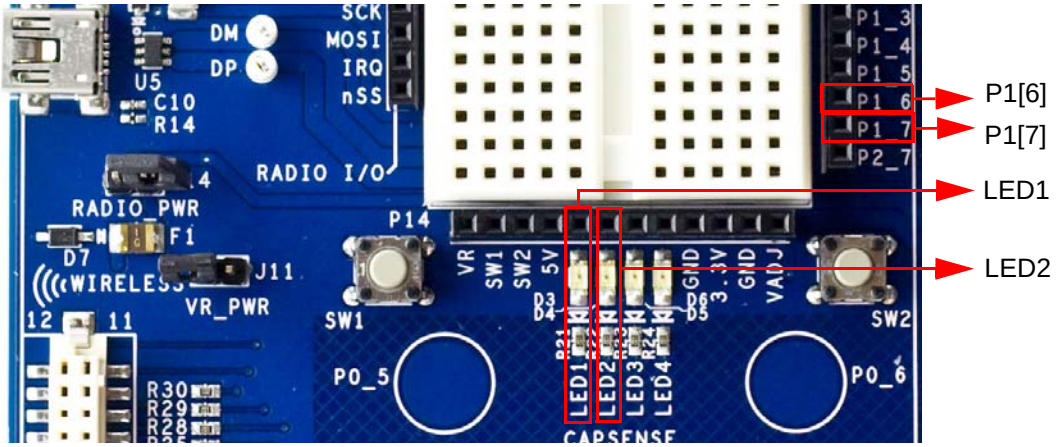
   Note VTARG of the MiniProg3 is wired exclusively to VDDIO1 of the chip on the PSoC CY8C55 family processor module. Because of this, you cannot perform power cycle mode programming.

   - From the **Tools** menu in PSoC Creator, click **Options**.
   - In the Options window, select **Program/Debug** → **Port Configuration** → **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 5 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 2-23. Connect P1[6] to LED1 and P1[7] to LED2



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 CY8C55 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.

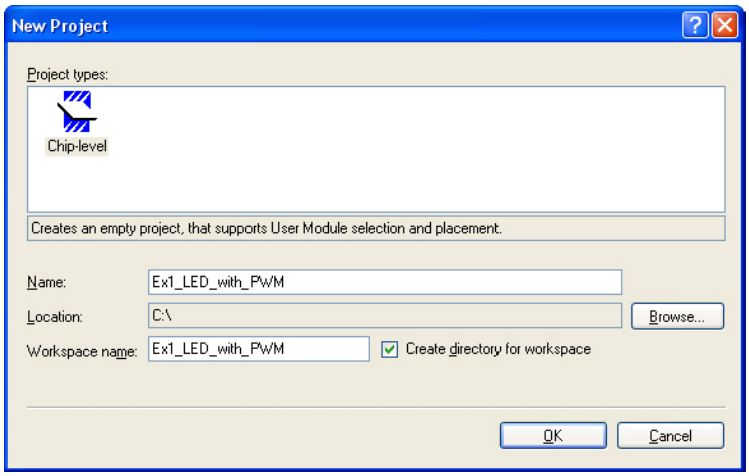
3.1 CY8C28 Family Processor Module Code Examples

3.1.1 My First PSoC 1 (CY8C28) Project

3.1.1.1 Creating My First PSoC 1 (CY8C28) Project

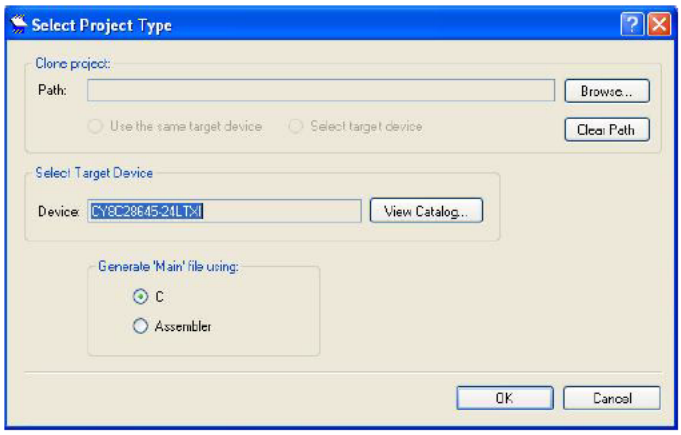
1. Open PSoC Designer.
2. To create a new project, click **File** → **New Project**. The **New Project** window opens.
3. In the **New Project** window, select **Chip-Level Project**. Name the project **Ex1_LED_with_PWM**.
4. In the **Location** field, click **Browse** and navigate to the appropriate directory.

Figure 3-1. New Project Window



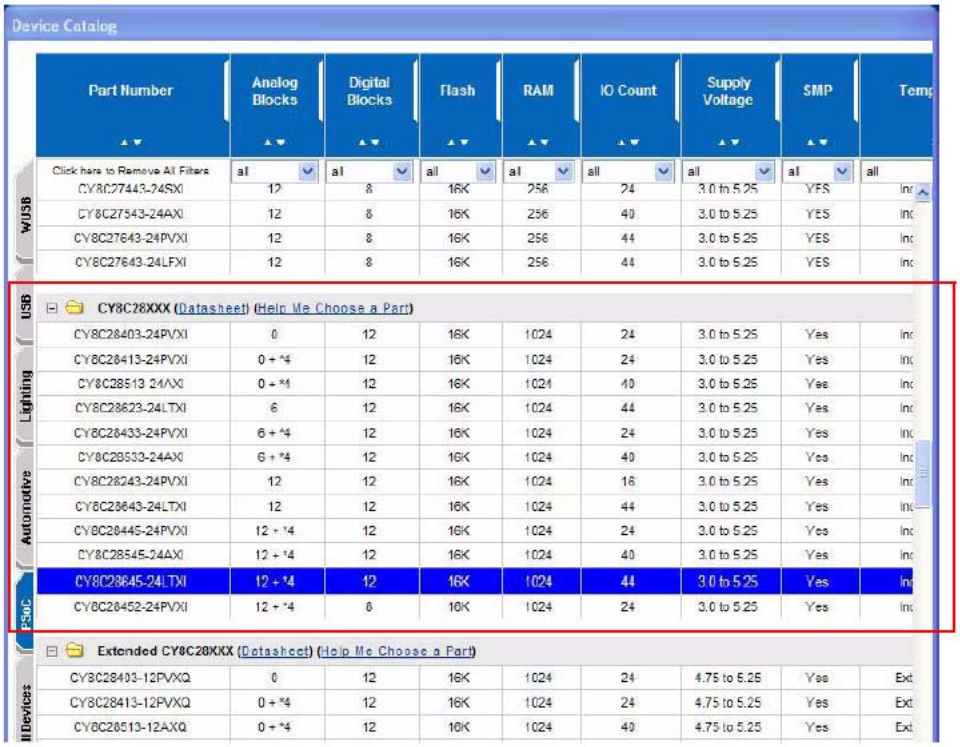
5. Click **OK**. The **Select Project Type** window opens.

Figure 3-2. Select Project Type Window



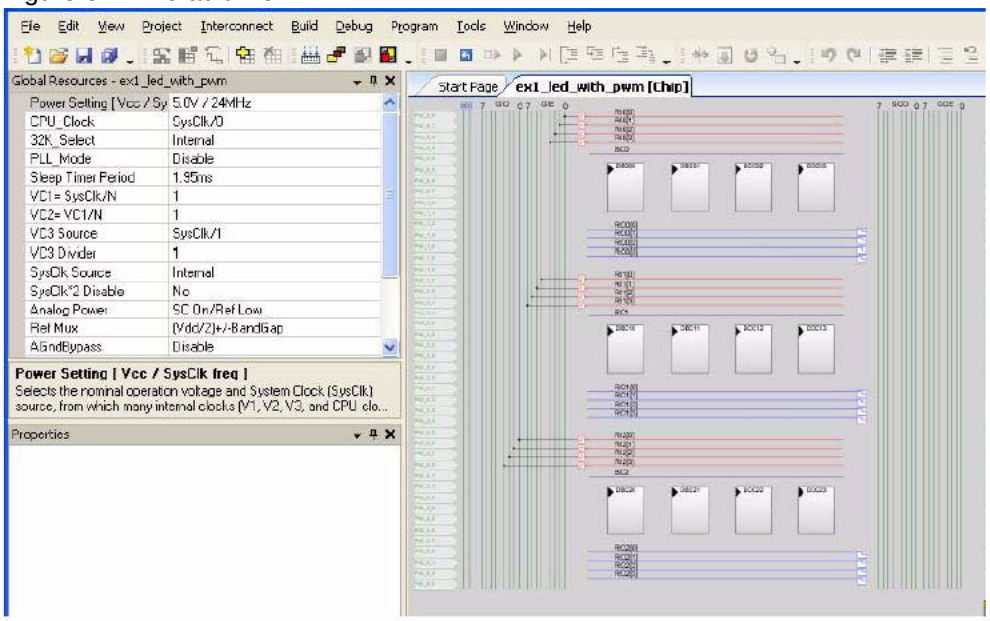
6. Under **Select Target Device**, click **View Catalog**.
7. The **Device Catalog** window opens. Click on the **PSoC** tab and scroll down to the **CY8C28XXX** section.
8. In this section, click the **CY8C28645-24LTXI** device; click **Select**.

Figure 3-3. Device Catalog Window



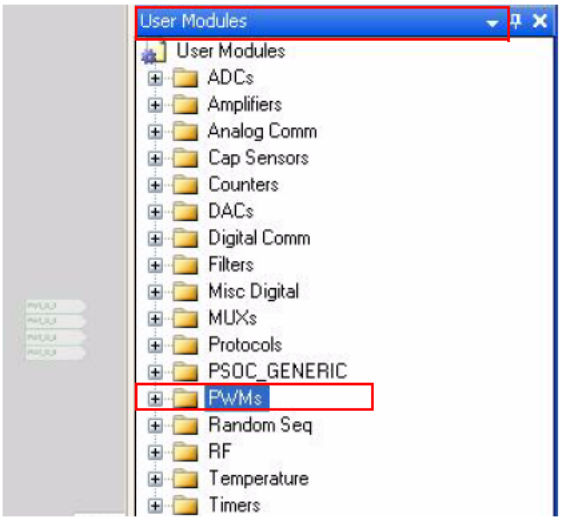
9. Under **Generate 'Main' File Using:**, select **C**; then, click **OK**.
10. By default, the project opens in Chip view.

Figure 3-4. Default View.



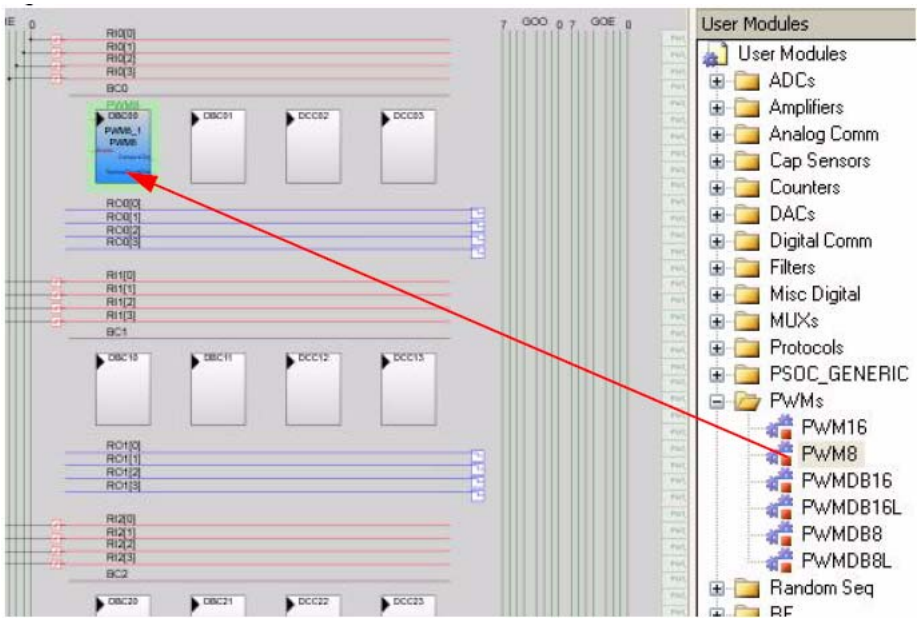
11. In the **User Modules** window, expand the **PWMs** folder.

Figure 3-5. User Modules Window



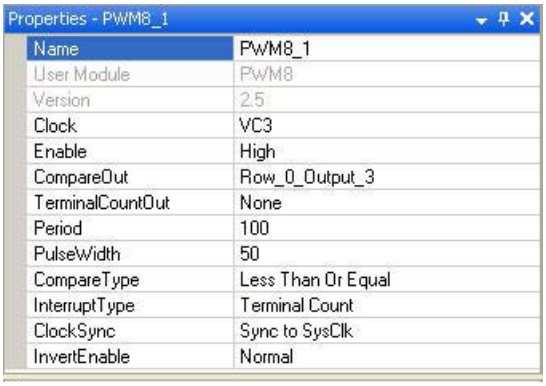
12. In this folder, right-click on **PWM8** and select **Place**. The user module (UM) is placed in the first available digital block.

Figure 3-6. Place User Module PWM8



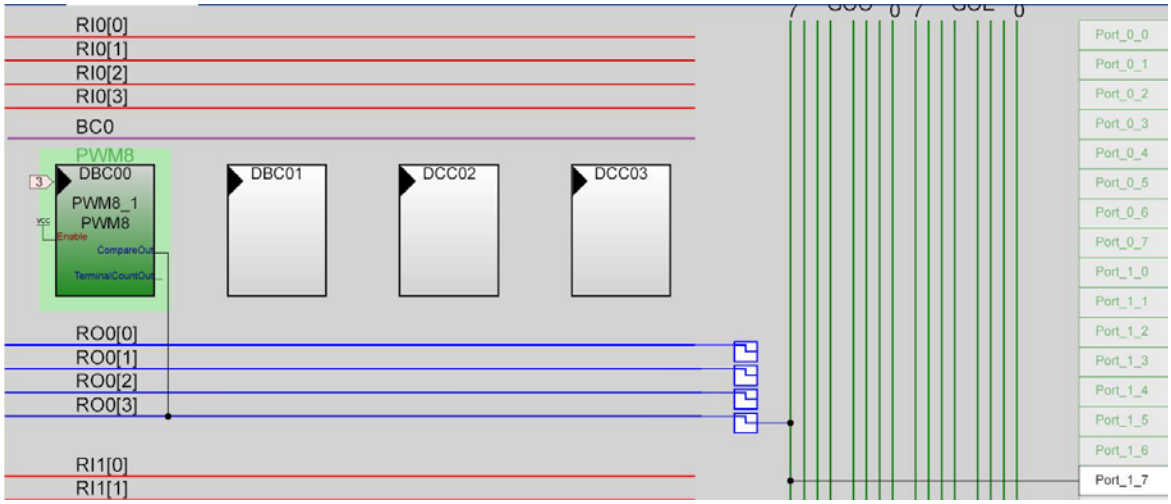
13. Click the placed PWM8_1 UM; the **Properties** window opens on the left side of the screen. Configure the PWM with the settings shown in the following figure. If the **Properties** window does not appear, click **View** → **Properties Window**.

Figure 3-7. Properties Window



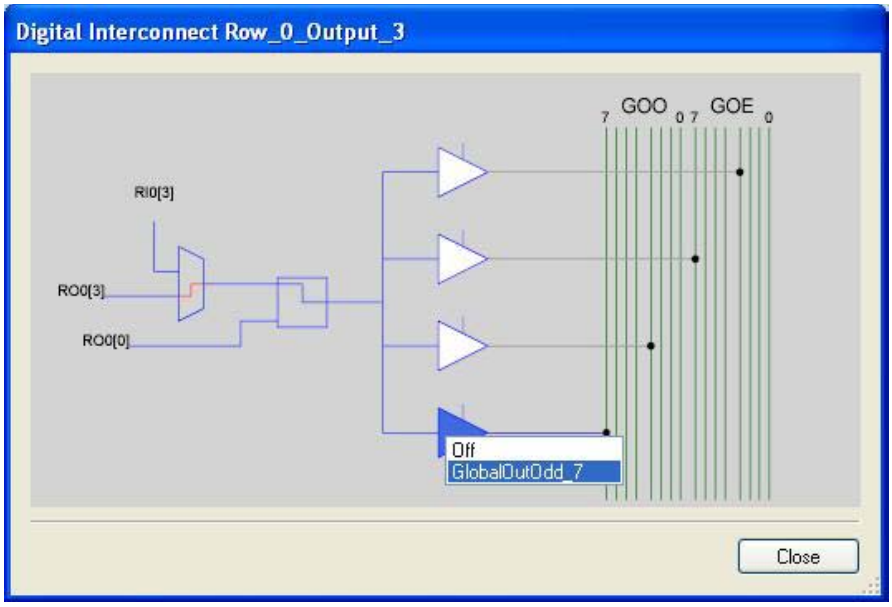
14. Next, route the PWM **CompareOut** signal to P1[7]. The first step is to configure the lookup table (LUT) on **Row_0_Output3**.

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



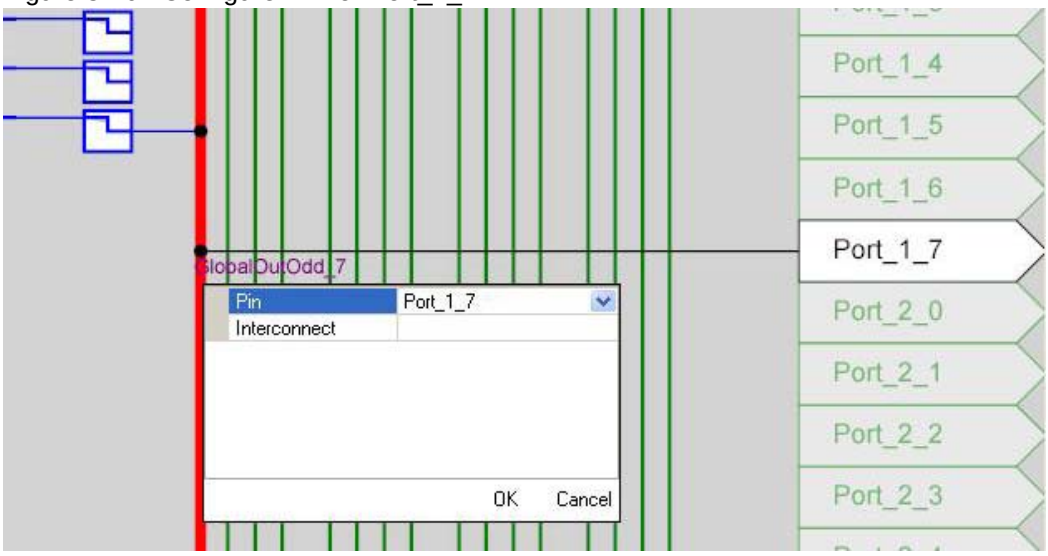
15. Double-click the LUT, the **Digital Interconnect** window opens.
16. In this window, enable **Row_0_Output_3_Drive_3** to connect to **GlobalOutOdd_7**.

Figure 3-9. Digital Interconnect Window



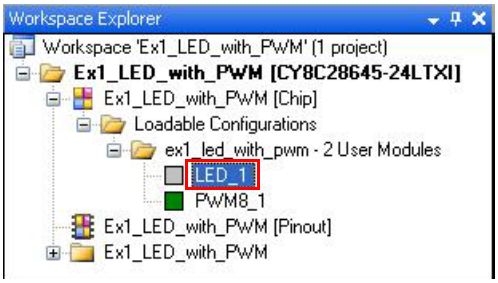
17. Click **Close**.
18. Click on **GlobalOutOdd_7**. In the window that appears, configure **Pin** for **Port_1_7**.

Figure 3-10. Configure Pin for Port_1_7



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

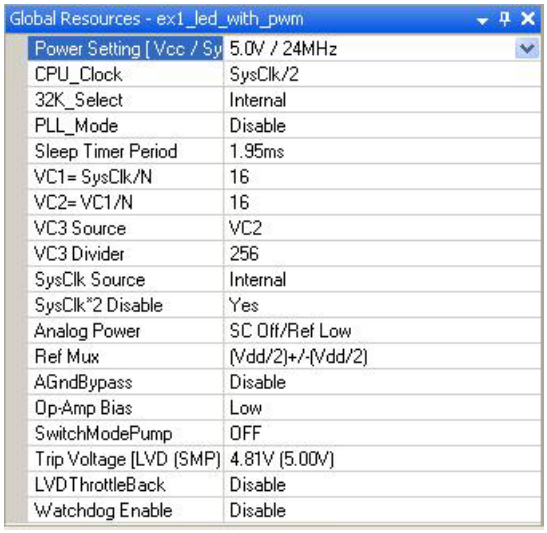


21. Click the **LED_1** UM and navigate to the **Properties** window. Configure the LED for **P1[6].**

Figure 3-12. Properties Window

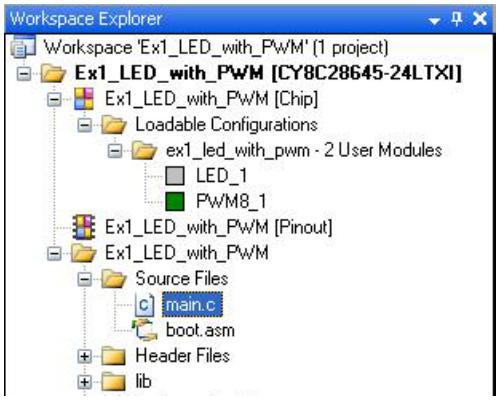
22. Configure the **Global Resources** window to match the following figure.

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** → **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 17 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)
{
   WORD i;            /* Variable used for delay */

   PWM8_1_Start();    /* Turn on the PWM to blink LED on P1.6 */
   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 */
```

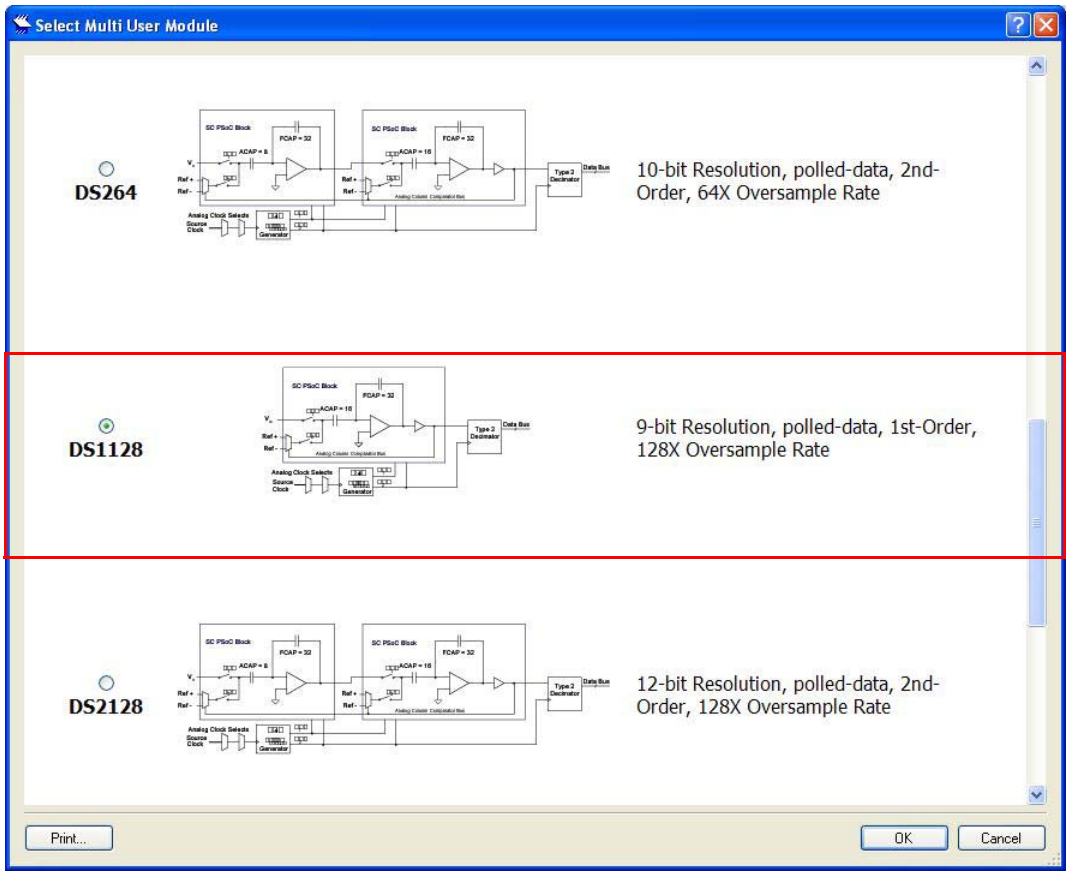
3.1.2 ADC to LCD Project

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

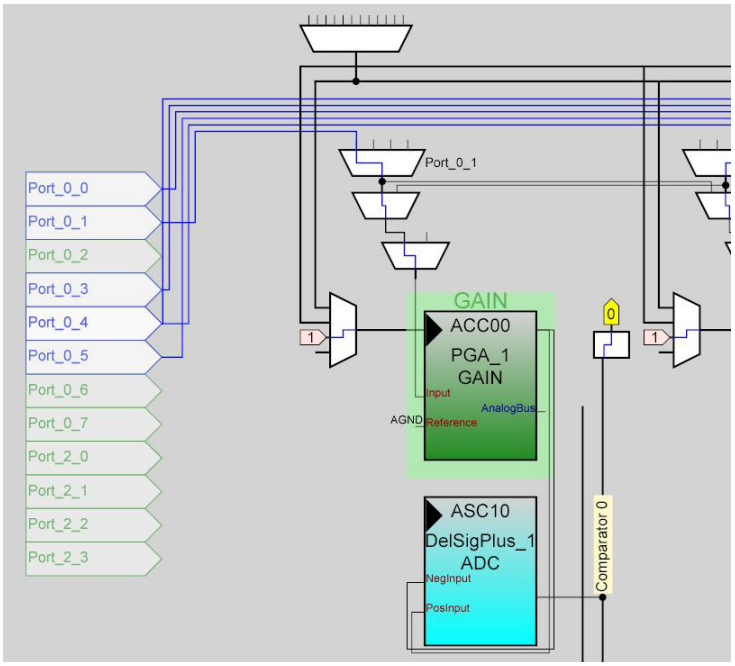
1. Follow steps 1 to 10 in section 3.1.1.1 on page 33; change the project name to **Ex2_ADC_to_LCD**.
2. In the **User Modules** window, expand the **ADCs** folder and right-click **DelSigPlus**; select **Place**. A window opens with multiple options for the DelSigPlus UM. Here, the **DS1128** configuration 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**.

Figure 3-16. Place PGA 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

Properties - PGA_1

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.

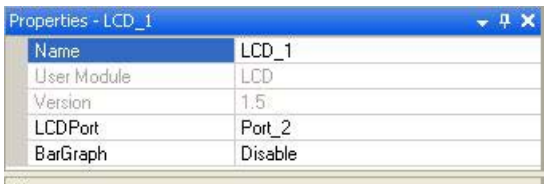
Figure 3-18. DelSigPlus_1 Properties

Properties - DelSigPlus_1

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

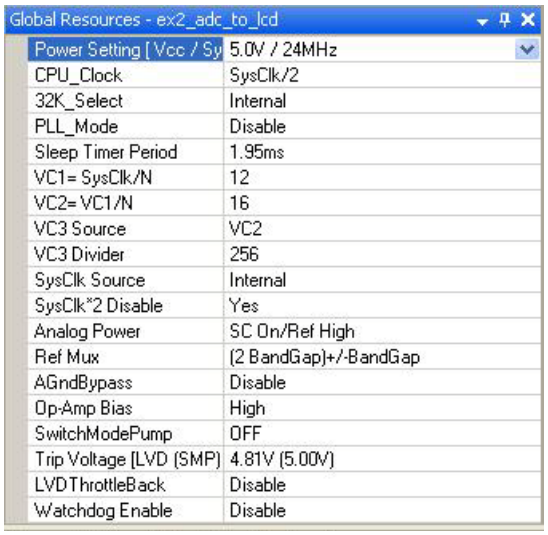
9. Click **LCD_1** and configure the properties to match this figure.

Figure 3-19. LCD_1 Properties



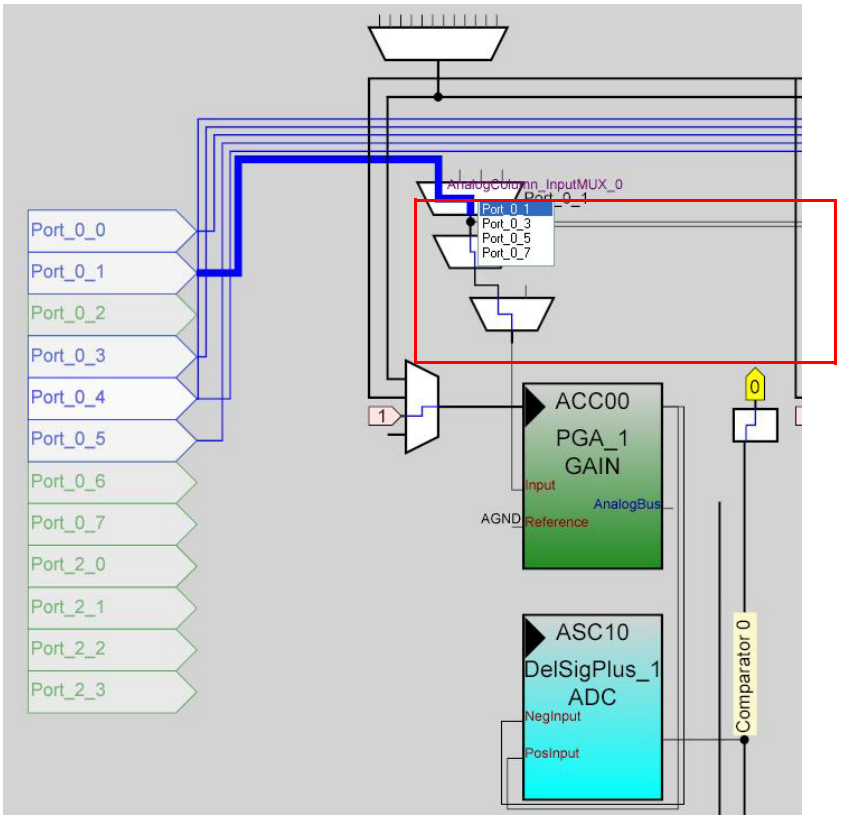
10. Configure the **Global Resources** to match the following figure.

Figure 3-20. Global Resources



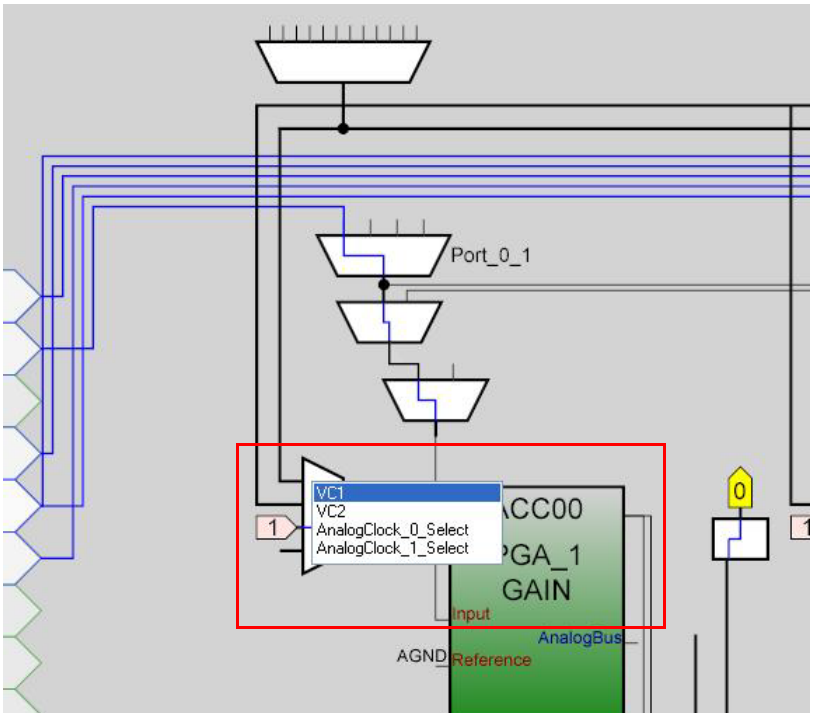
11. Ensure that **AnalogColumn_InputMUX_0** is connected to **Port_0_1**. If it is not configured for this port, double-click the mux and choose **Port_0_1**.

Figure 3-21. AnalogColumn_InputMUX_0 Connected to Port_0_1



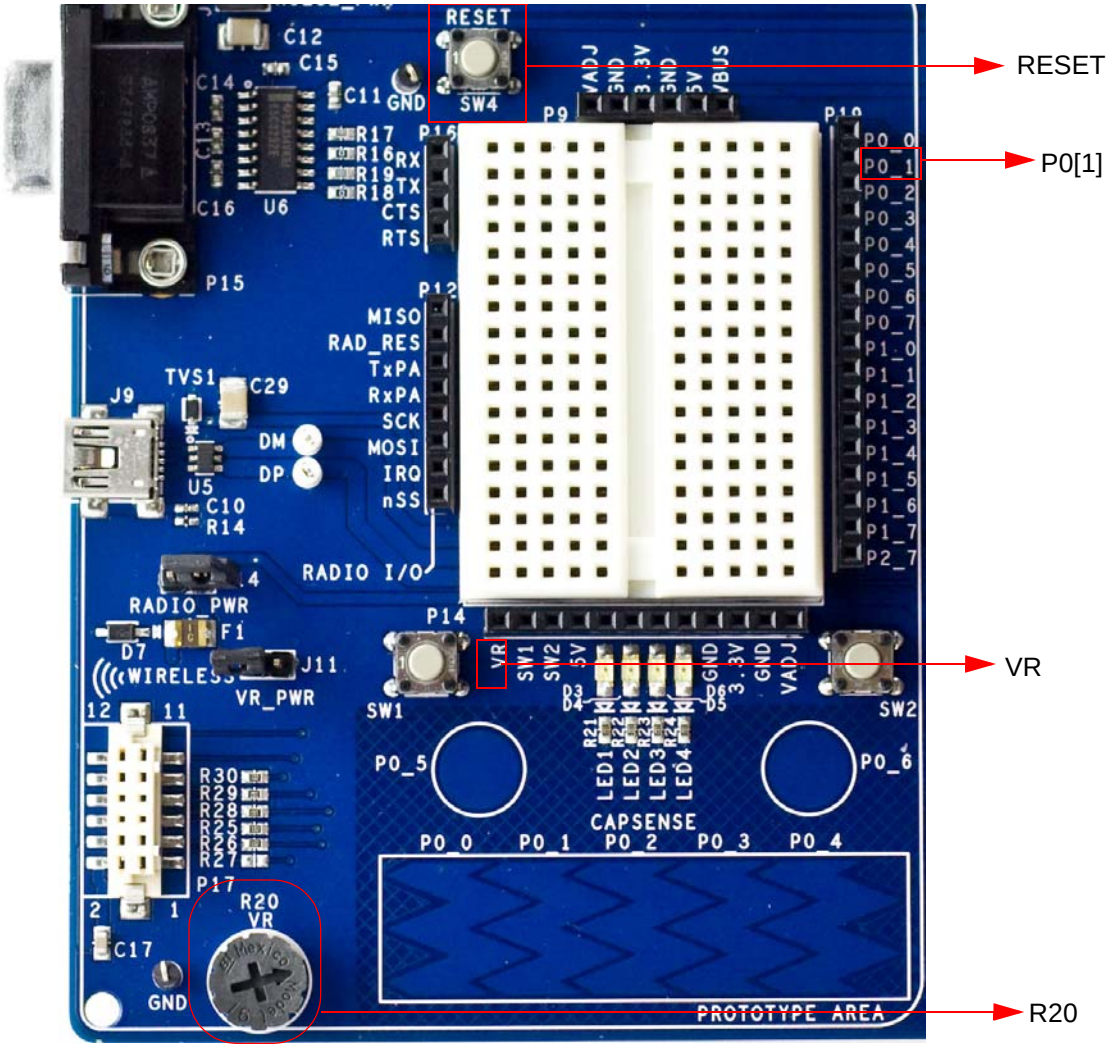
12. Ensure that **AnalogColumn_Clock_0** is connected to **VC1**. If it is not, double-click the mux and choose **VC1**.

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** → **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 17 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.

```
#include <m8c.h>         /* part specific constants and macros */
#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 */
```

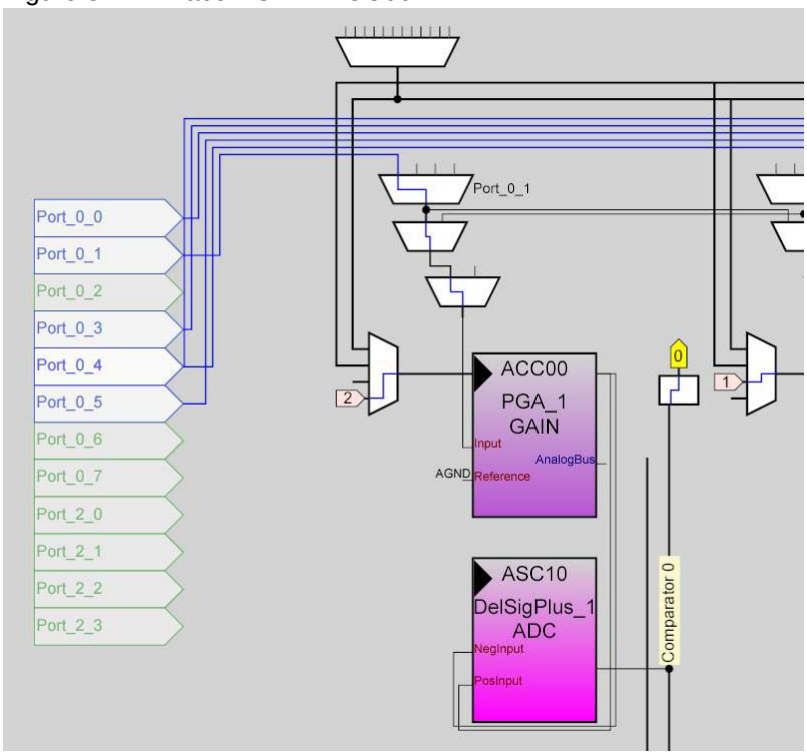
3.1.3 ADC to UART with DAC

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

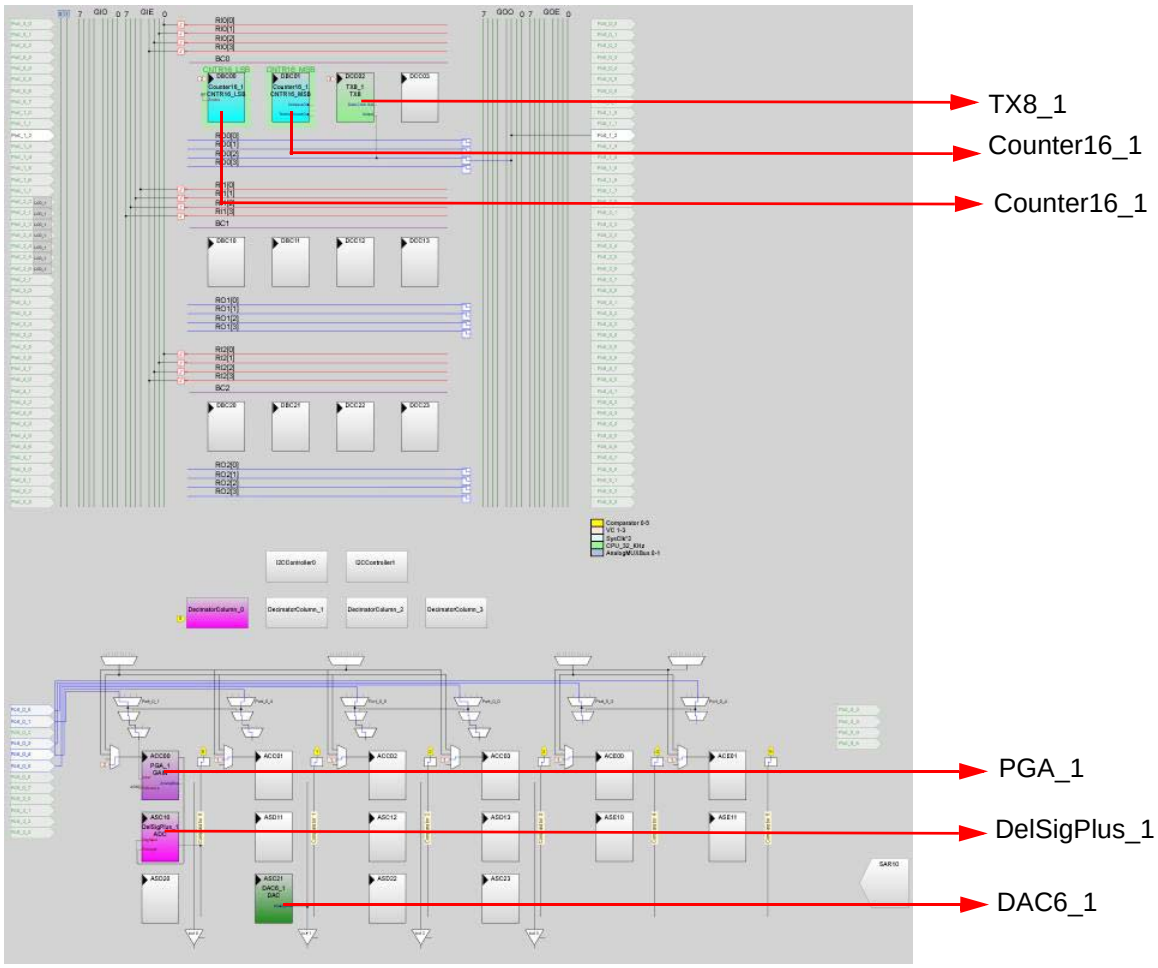
1. Follow steps 1 to 10 in section 3.1.1.1 on page 33; change the project name to **Ex3_ADC_to_UART_with_DAC**.
2. In the **User Modules** window expand the **ADCs** folder and right-click **DelSigPlus**; select **Place**. A window opens with multiple options for the DelSigPlus UM. Here, the **DS1128** 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**.

Figure 3-24. Place PGA in ACC00



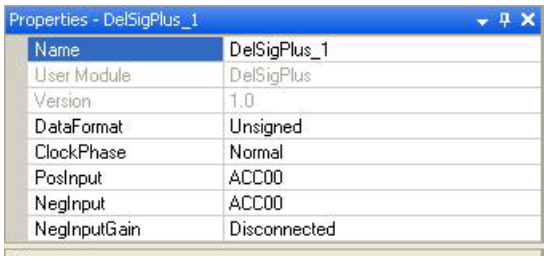
6. In the **User Modules** window, expand **Misc Digital**, right-click **LCD**, and select **Place**.
7. In the **User Modules** window, expand **Counters**, right-click **Counter16**, and select **Place**.
8. In the **User Modules** window, expand **Digital Comm**, right-click **TX8**, and select **Place**.
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



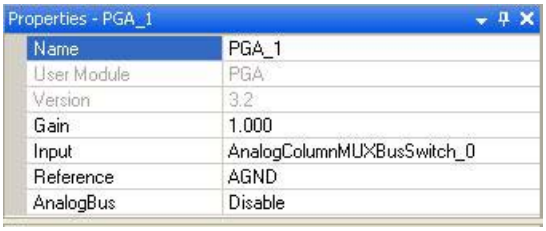
11. Click on **DelSigPlus_1** and configure it to match this figure.

Figure 3-26. DelSigPlus_1 Properties



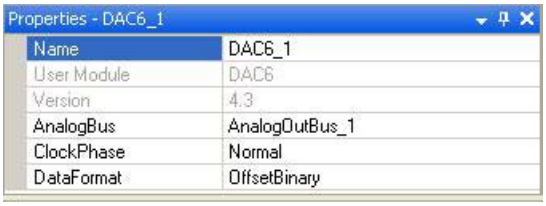
12. Click **PGA_1** and configure it to match this figure.

Figure 3-27. PGA_1 Properties



13. Click **DAC6_1** and configure it to match this figure.

Figure 3-28. DAC6_1 Properties

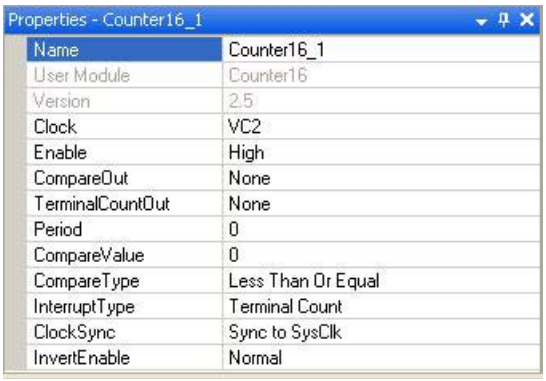


14. Click **LCD_1** and configure it to match this figure.

Figure 3-29. LCD_1 Properties

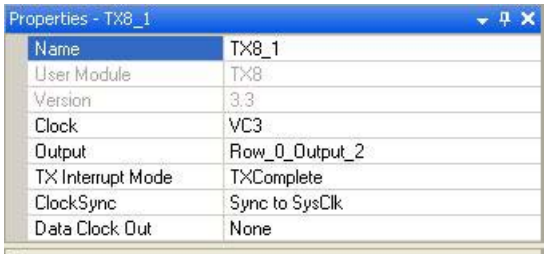
15. Click on **Counter16_1** and configure it to match this figure.

Figure 3-30. Counter16_1 Properties



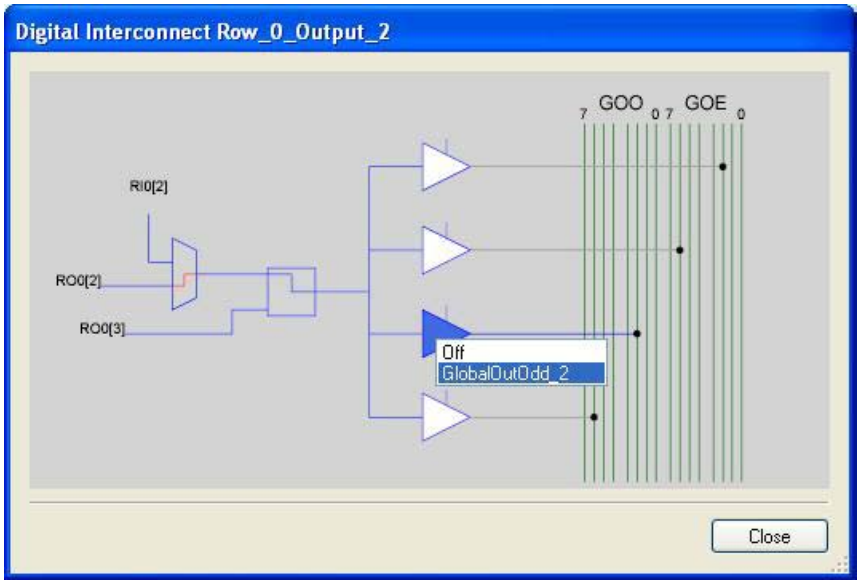
16. Click **TX8_1** and configure it to match this figure.

Figure 3-31. TX8_1 Properties



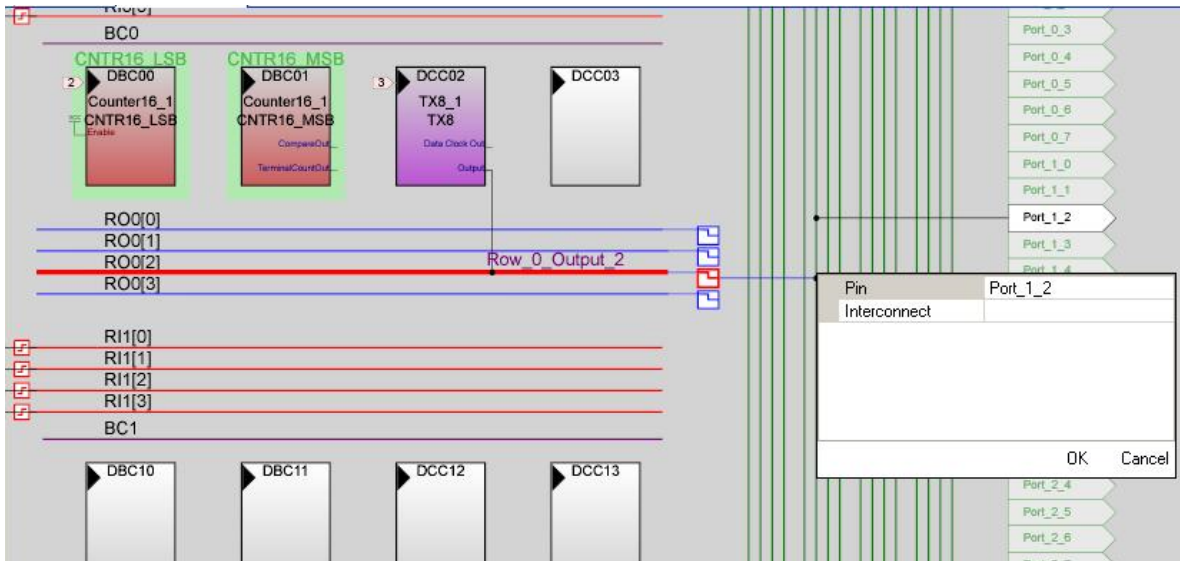
17. Click **RO0[2] LUT**, enable **Row_0_Output_2_Drive_2** to connect **GlobalOutOdd_2**.

Figure 3-32. Digital Interconnect Window



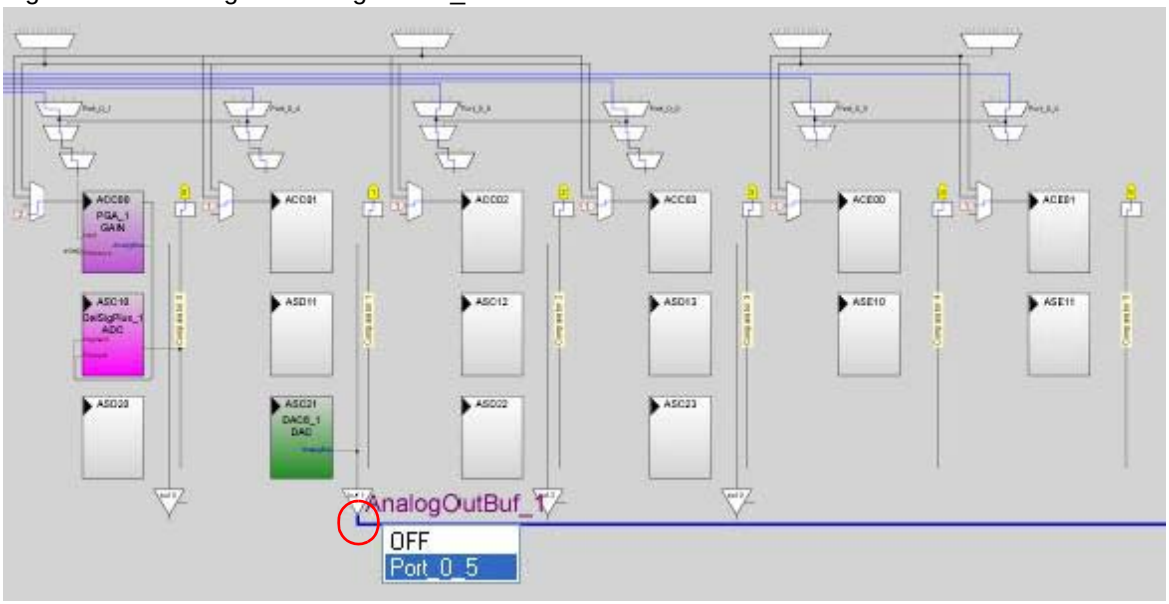
18. Click **GlobalOutOdd_2**. In the window that appears, configure **Pin** for **Port_1_2**.

Figure 3-33. Configure Pin for Port_1_2



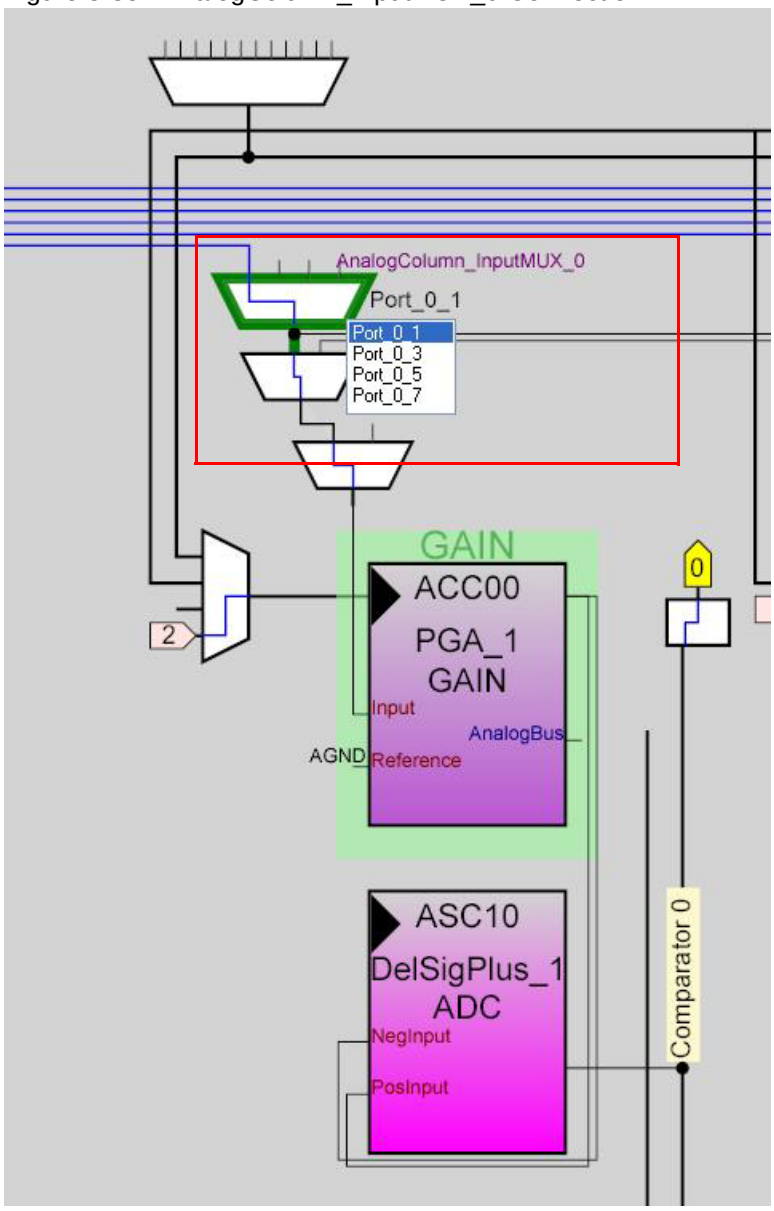
19. Click **OK** to continue.
20. Click **AnalogOutBuf_1** and configure it for **Port_0_5**.

Figure 3-34. Configure AnalogOutBuf_1



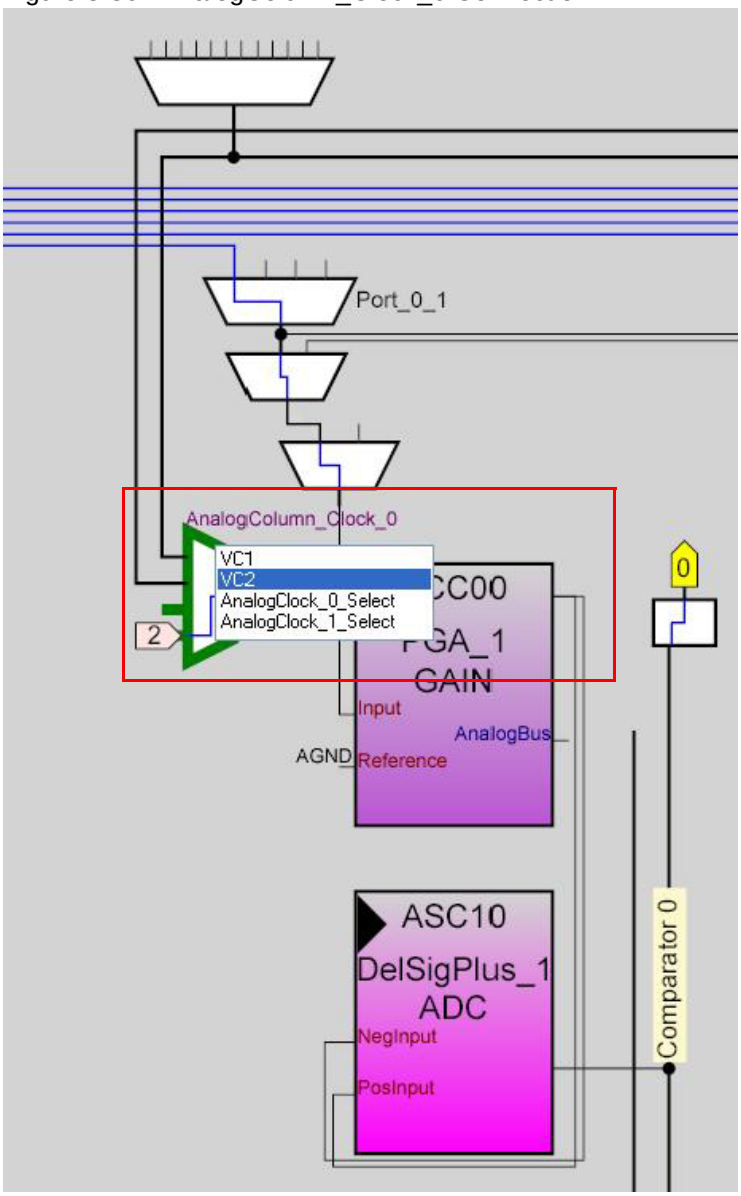
21. Verify that **AnalogColumn_InputMUX_0** is connected to **Port_0_1**. If it is not configured for this port, double-click the mux and choose **Port_0_1**.

Figure 3-35. AnalogColumn_InputMUX_0 Connection



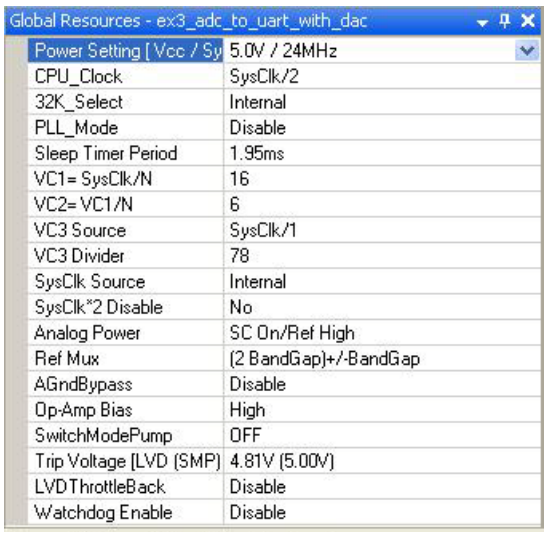
22. Verify that **AnalogColumn_Clock_0** and **AnalogColumn_Clock_1** are connected to **VC2**. If it is not, double-click the mux and chose **VC2**.

Figure 3-36. AnalogColumn_Clock_0 Connection



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

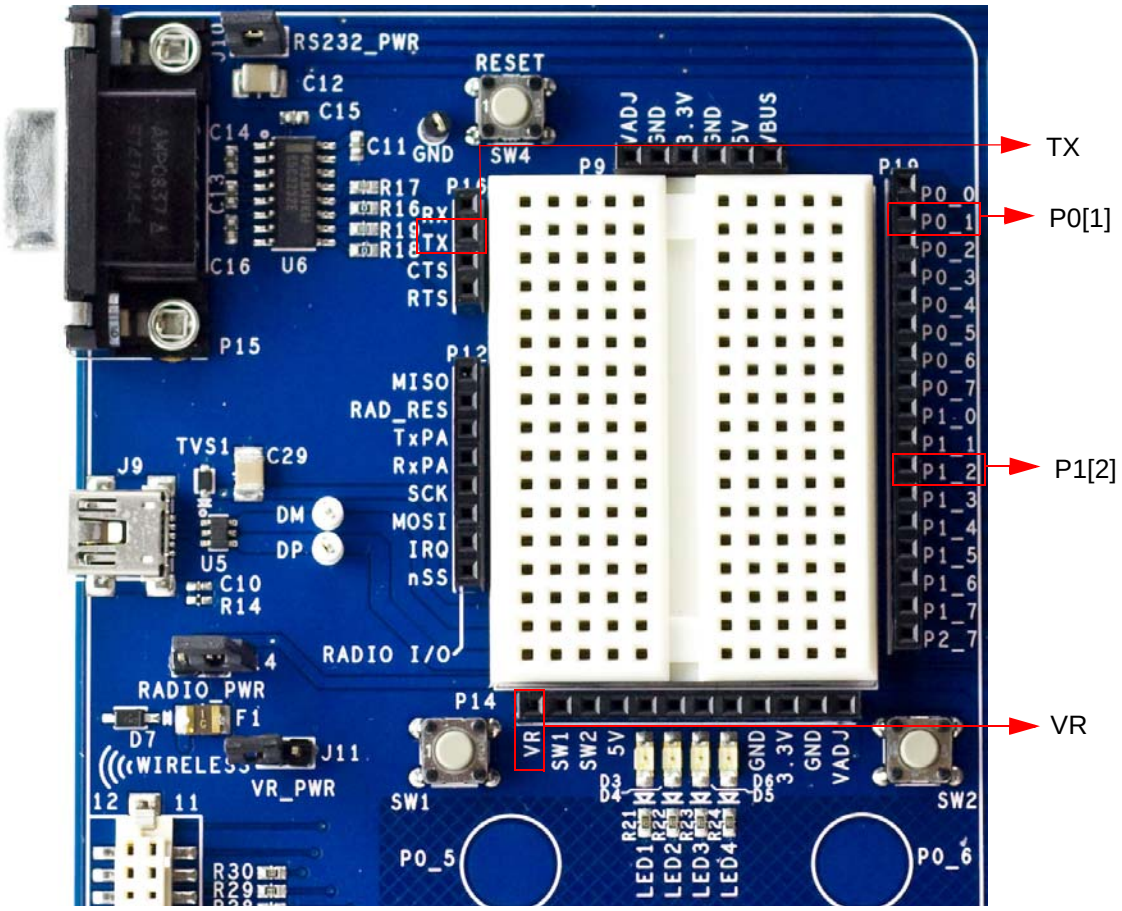
Figure 3-37. Configure Global Resources



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** → **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** → **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 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 3-38. 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 17 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.

```
#include <m8c.h>        /* part specific constants and macros */
#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,  0,  1,  2,  3,  4,  6,  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 */
    DelSigPlus_1_StartAD();             /* Start reading values on the ADC */
    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);

            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 */
            TX8_1_PutCRLF();                 /* Write return character to TX8 */
        }
    } /* 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
```

```
*
*******************************************************************************/
#ifdef HI_TECH_C
void Counter16_C_ISR(void) @ 0x24
#else
void Counter16_C_ISR(void)
#endif
{
    // Check to see if we have reached the //
    if (tablePos >= sizeof(sinTable))
    {
        tablePos = 0;
    }
    DAC6_1_WriteBlind(sinTable[tablePos++]);
}


/* [] END OF FILE */
```

3.1.4 CapSense

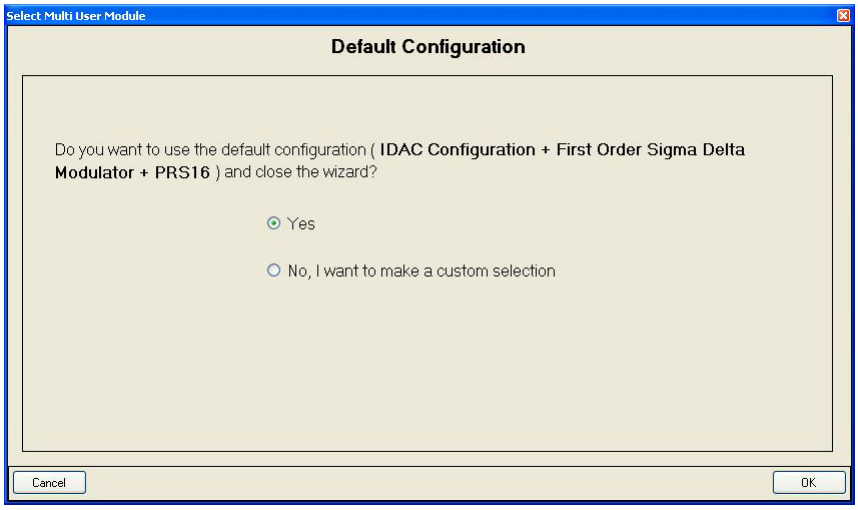
This project demonstrates CapSense. The firmware displays the CapSense button presses on the LCD (row 1) and associated LEDs. It also displays the CapSense slider position on the LCD (row 2).

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

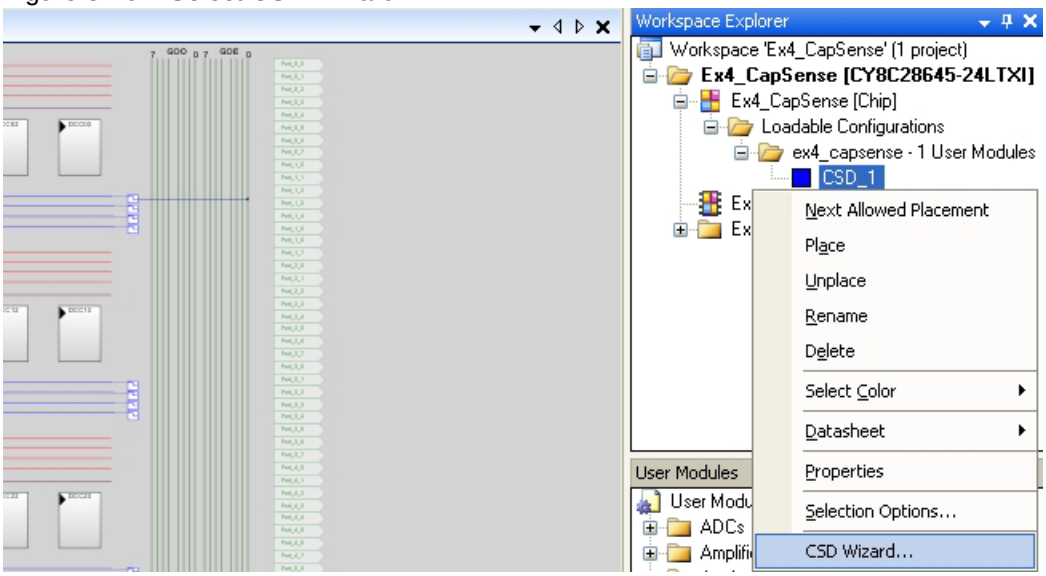
1. Follow steps 1 to 10 in section 3.1.1.1 on page 33; change the project name to **Ex4_CapSense**.
2. In the User Modules window, expand the **Cap Sensors** folder. Right-click **CSD** and select **Place**. A window appears with the option to use the default configuration.

Figure 3-39. Select Multi User Module Window



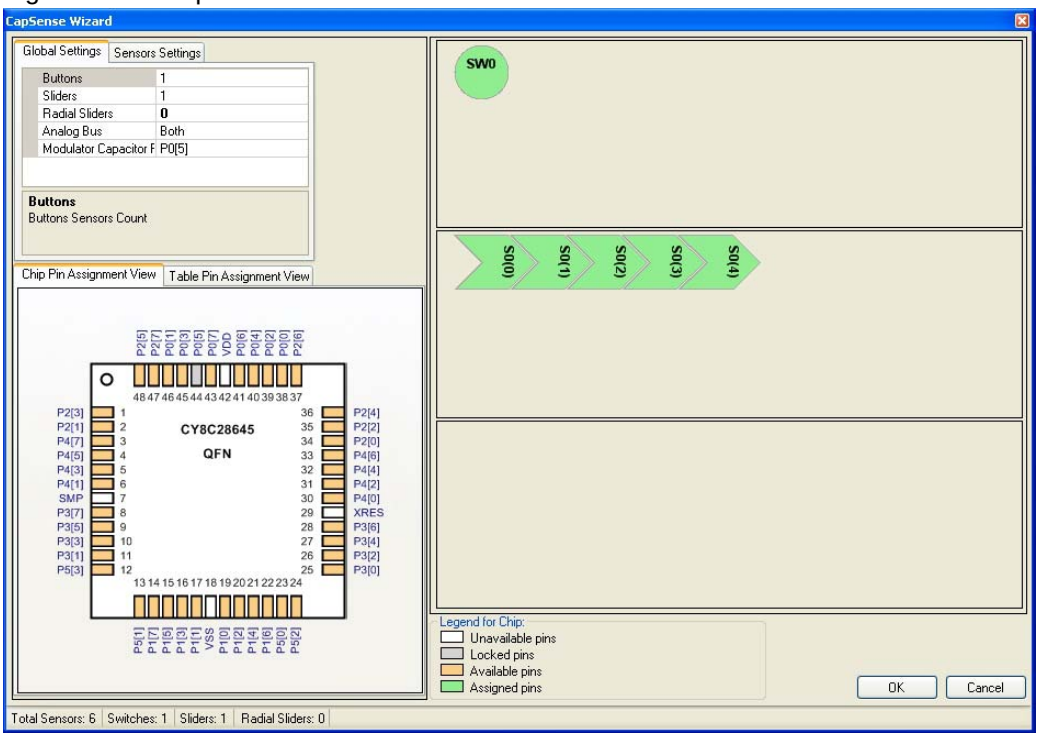
3. Select **Yes** and click **OK**.
4. Right-click the **CSD** user module in the workspace explorer and select **CSD Wizard**.

Figure 3-40. Select CSD Wizard



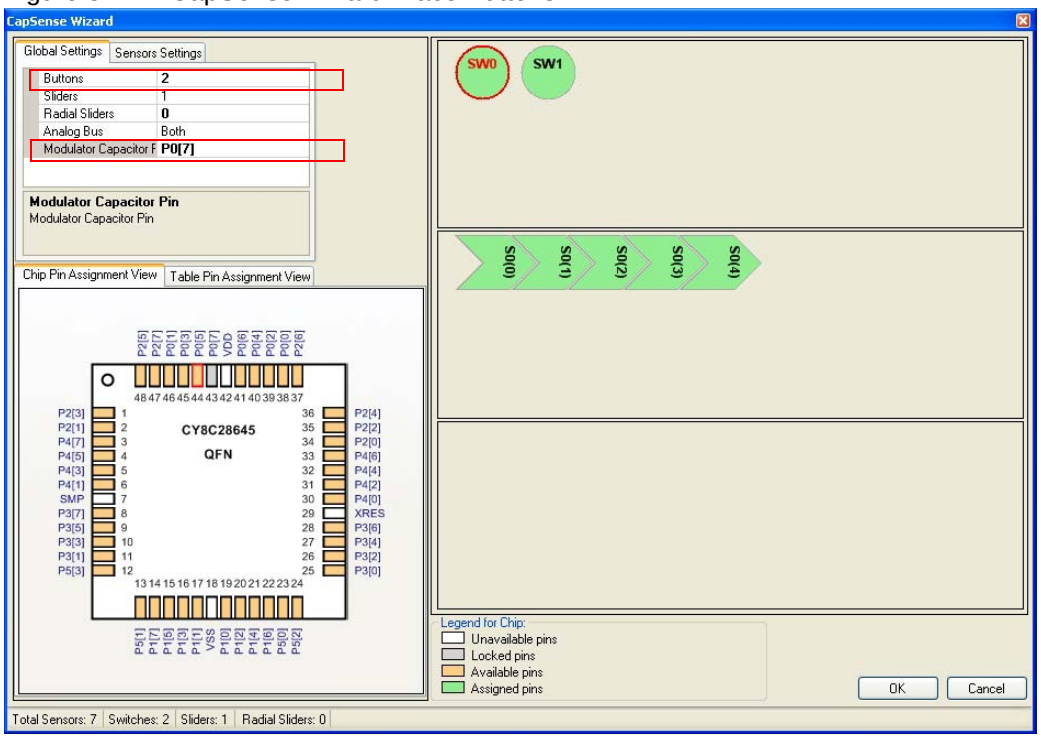
5. The CapSense Wizard window opens.

Figure 3-41. CapSense Wizard



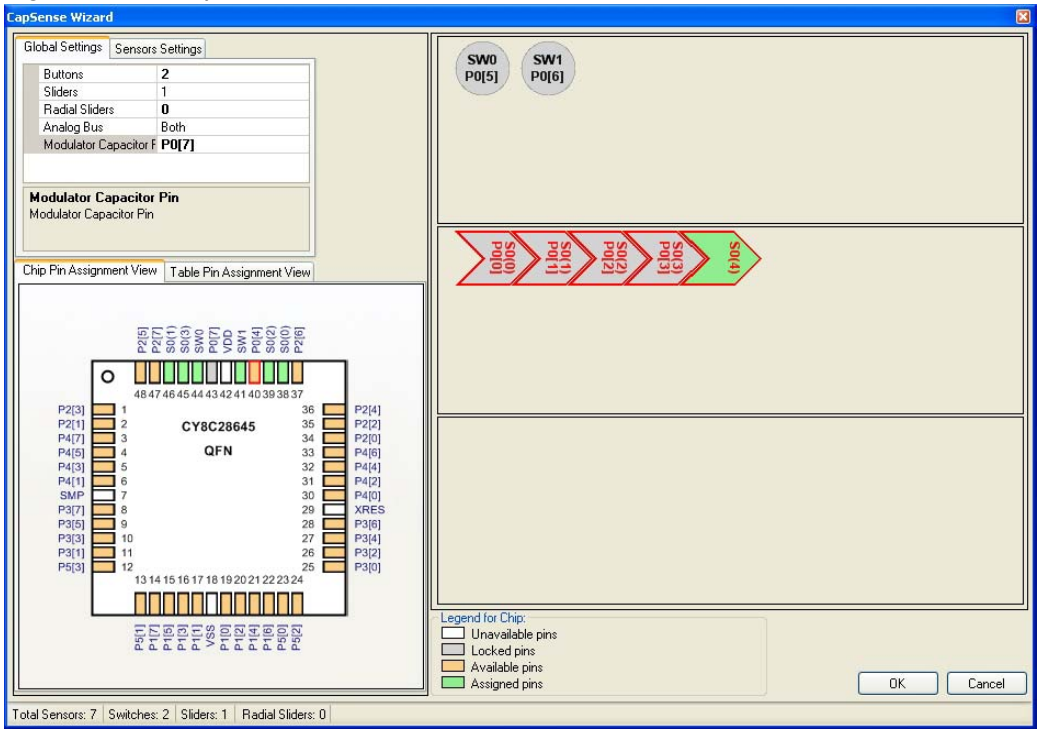
6. In the CapSense Wizard window, under the **Global Settings** tab, set the # of buttons to ‘2’.
7. Select P0[7] as the Modulator Capacitor Pin.

Figure 3-42. CapSense Wizard Place Buttons



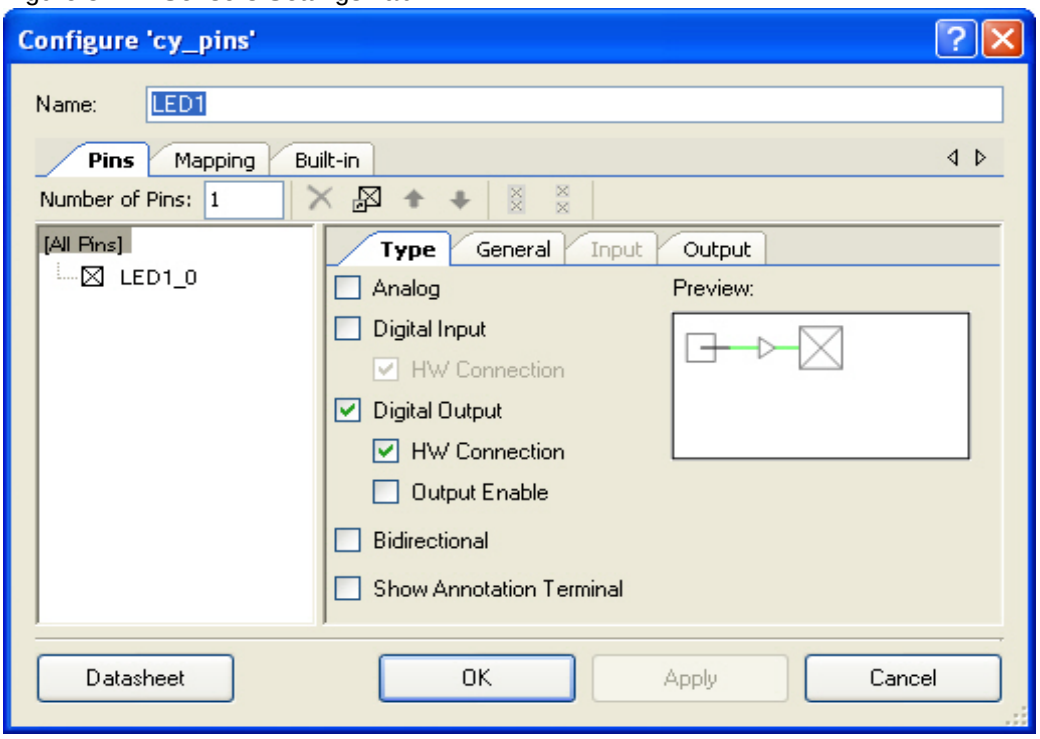
8. Click and hold **SW0** and drag it to P0[5].
9. Click and hold **SW1** and drag it to P0[6].

Figure 3-43. CapSense Wizard Slider Sensors



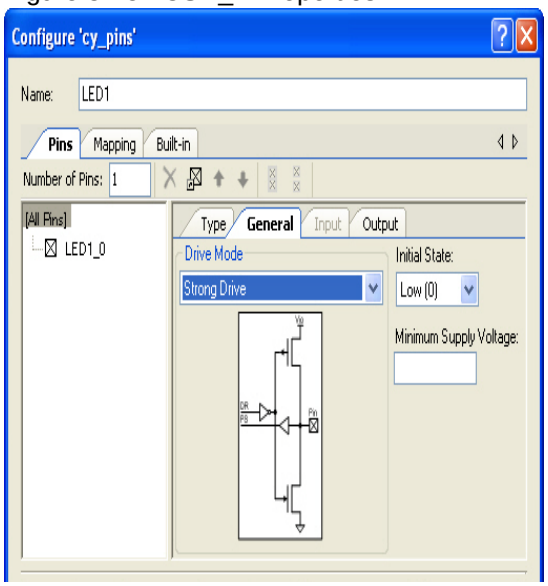
10. Repeat for each slider sensor and corresponding pin.
    - S0[0] to P0[0]
    - S0[1] to P0[1]
    - S0[2] to P0[2]
    - S0[3] to P0[3]
    - S0[4] to P0[4]
11. Select the **Sensors Settings** tab.
12. Set the **Resolution** to 80.

Figure 3-44. Sensors Settings Tab



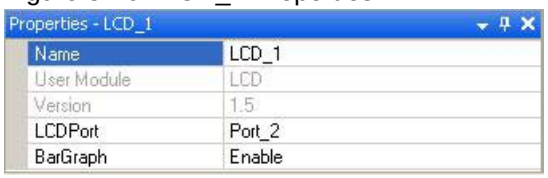
13. Click **OK**.
14. In the **User Modules** window, expand **Misc Digital**, right-click **LCD**, and select **Place**.
15. In the **User Modules** window, expand **Misc Digital**, right-click **LED**, and select **Place**.
16. In the **User Modules** window, expand **Misc Digital**, right-click **LED**, and select **Place**.
17. Click **CSD_1** and configure it to match this figure.

Figure 3-45. CSD_1 Properties



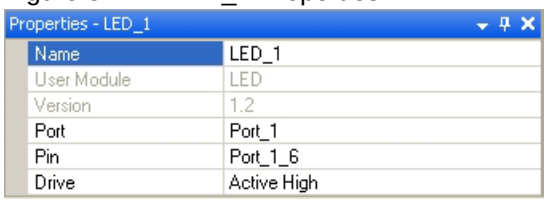
18. Click **LCD_1** and configure it to match this figure.

Figure 3-46. LCD_1 Properties



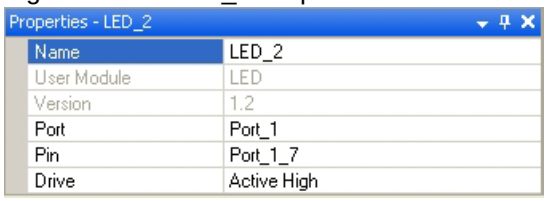
19. Click **LED_1** and configure it to match this figure.

Figure 3-47. LED_1 Properties



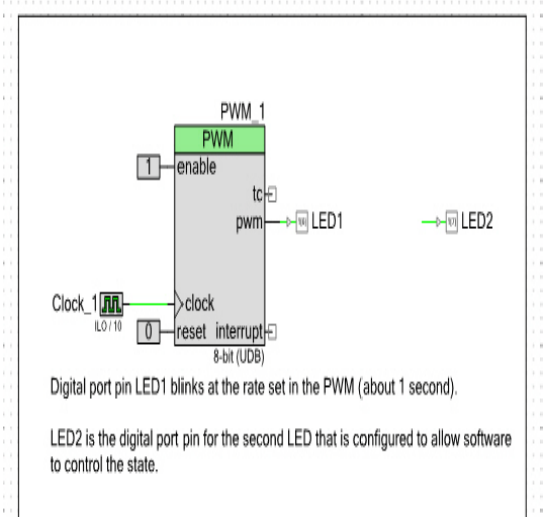
20. Click **LED_2** and configure it to match this figure.

Figure 3-48. LED_2 Properties



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 **Build** → **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 17 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 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 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 */
```

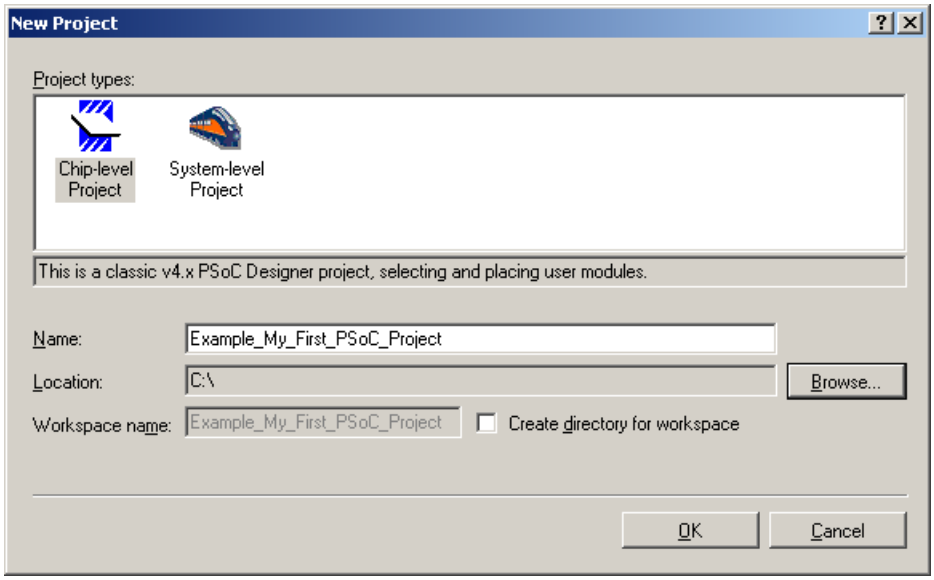
3.2 CY8C29 Family Processor Module Code Examples

3.2.1 My First PSoC 1 (CY8C29) Project

3.2.1.1 Creating My First PSoC 1 (CY8C29) Project

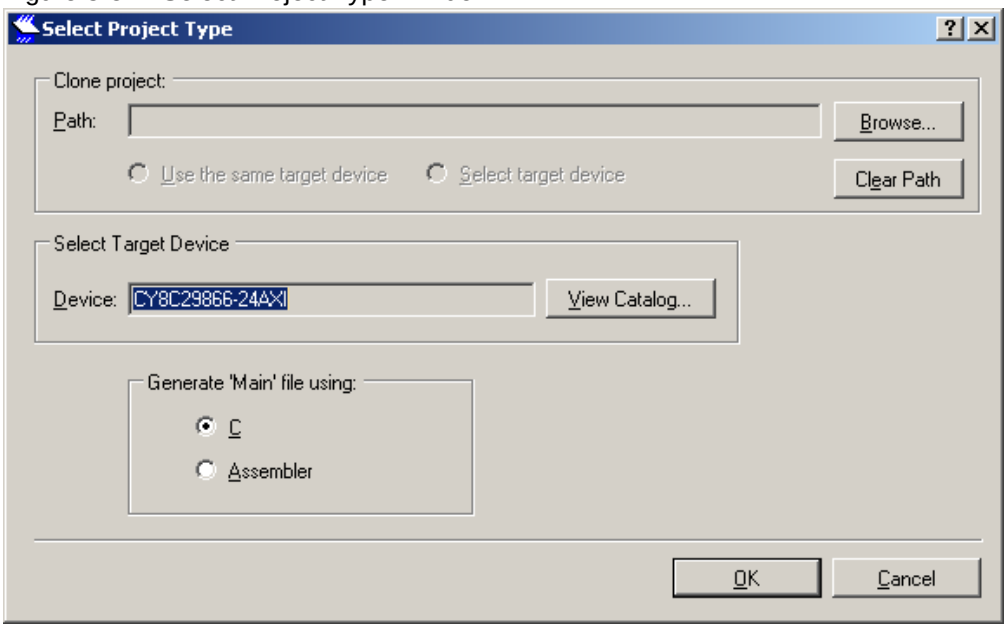
1. Open PSoC Designer
2. To create a new project, click **File** → **New Project**. The **New Project** window opens.
3. In the **New Project window**, select the **Chip-Level Project**. Name the project **Example_My_First_PSoC_Project**.
4. In the **Location** field, click **Browse** and navigate to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

Figure 3-50. New Project Window



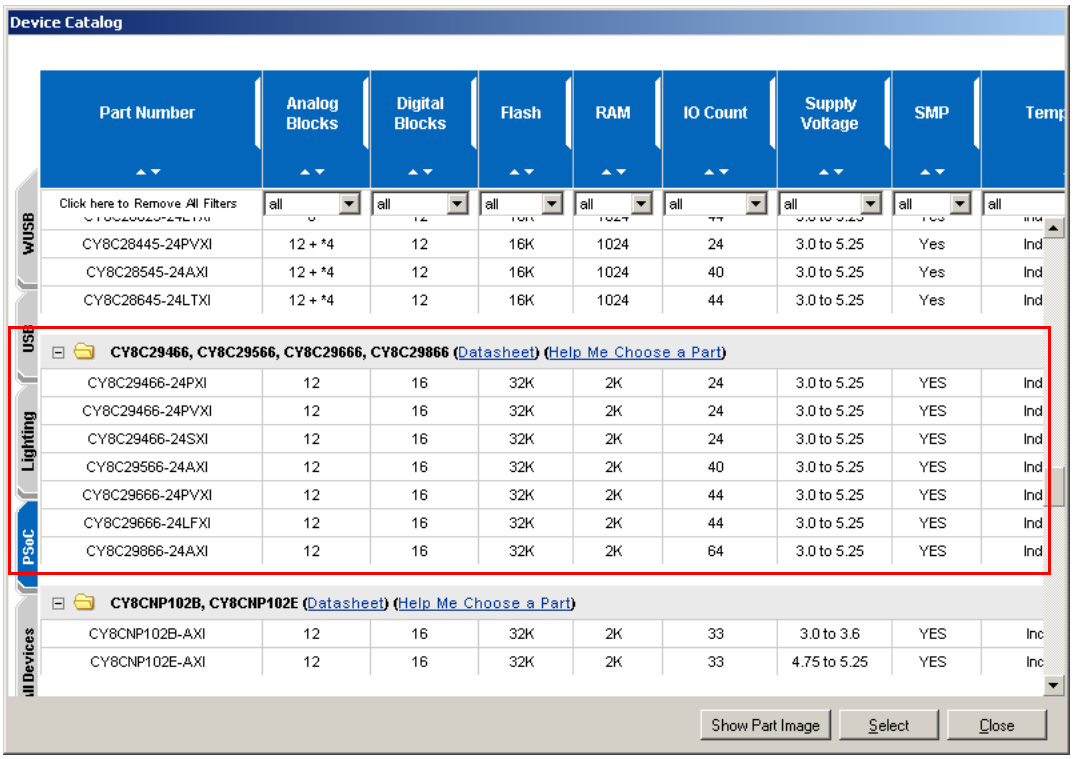
5. Click **OK**. The **Select Project Type** window opens.

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**.

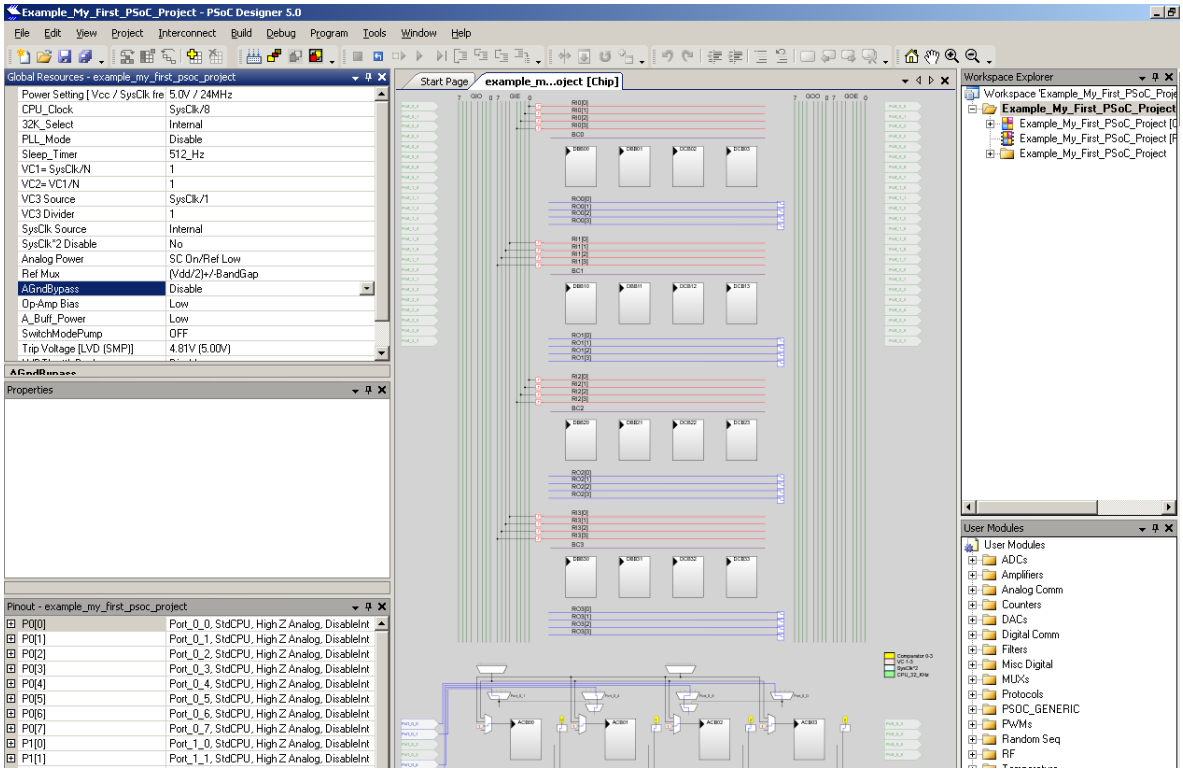
Figure 3-52. Device Catalog Window



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

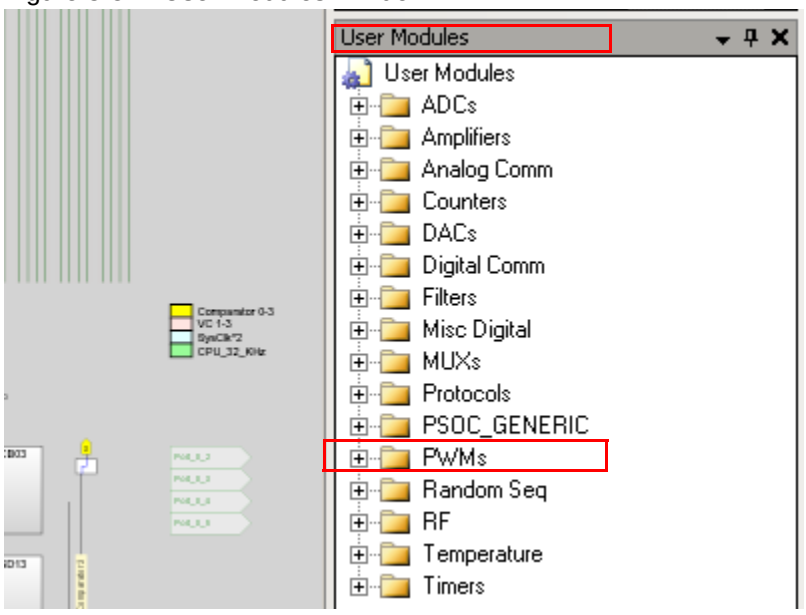
10. By default, the project opens in Chip view.

Figure 3-53. Default View.



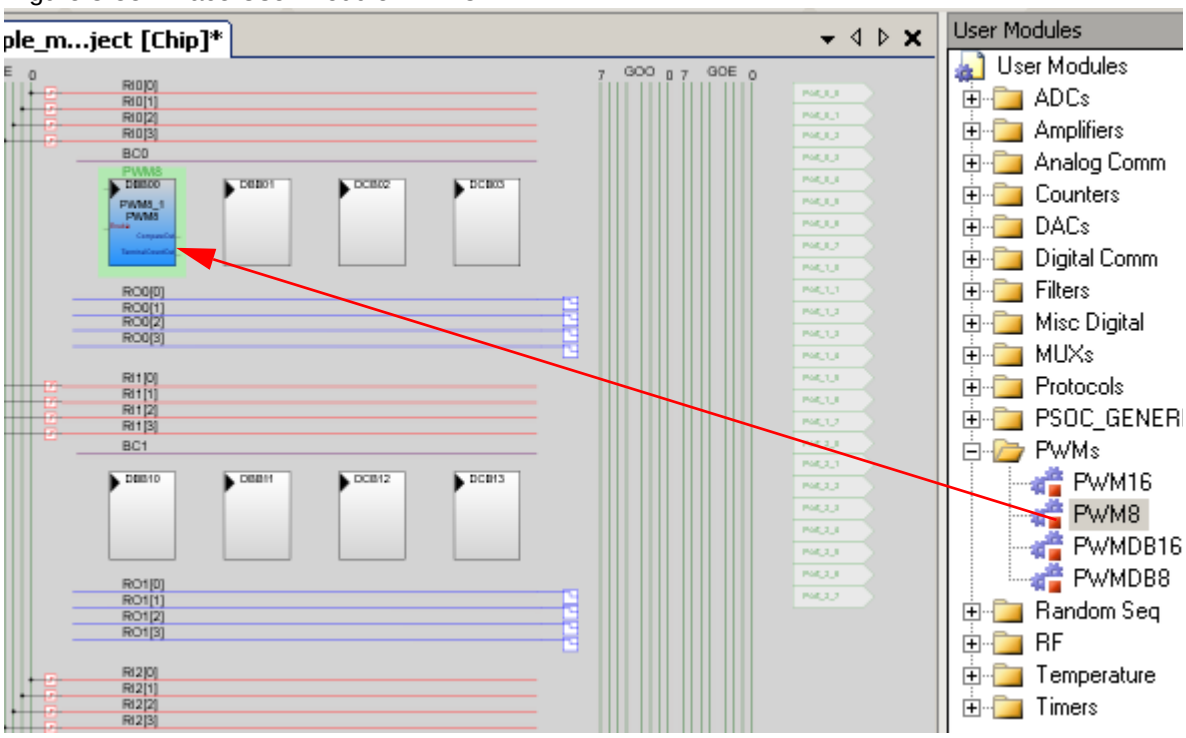
11. In the **User Modules** window, expand the **PWMs** folder.

Figure 3-54. User Modules Window



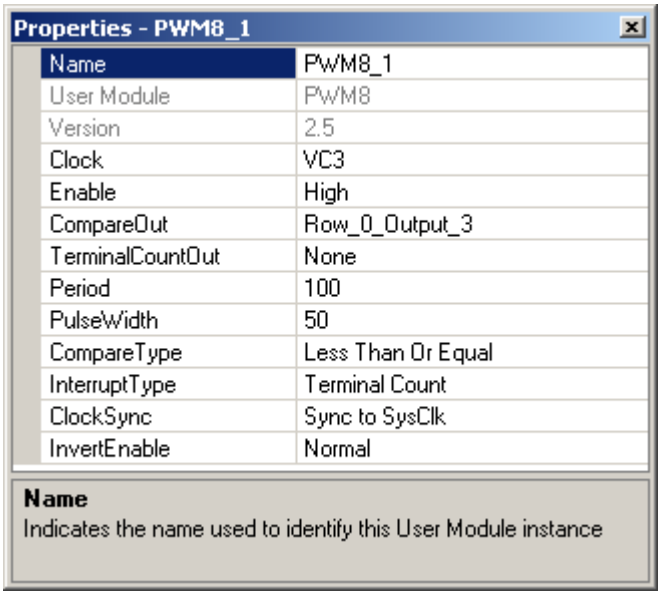
12. In this folder, right-click on **PWM8** and select **Place**. The User Module (UM) is placed in the first available digital block.

Figure 3-55. Place User Module PWM8



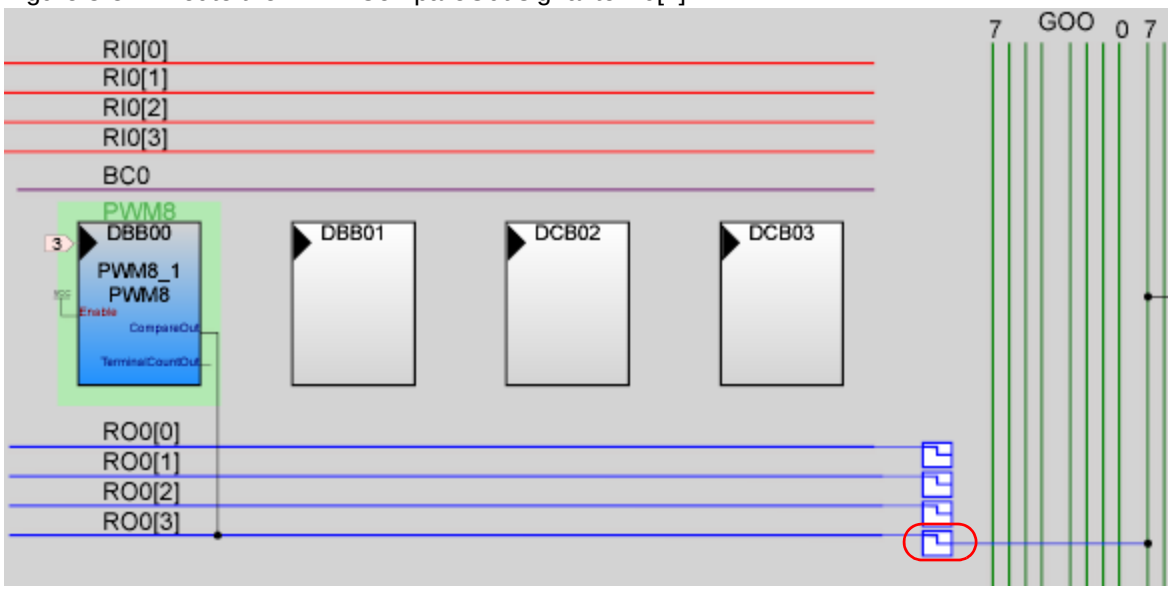
13. Double-click the placed PWM8_1 UM; the **Properties** window opens on the left side of the screen. Configure the PWM with the settings as in the following figure. If the **Properties** window does not appear, click **View** → **Properties Window**.

Figure 3-56. Properties Window



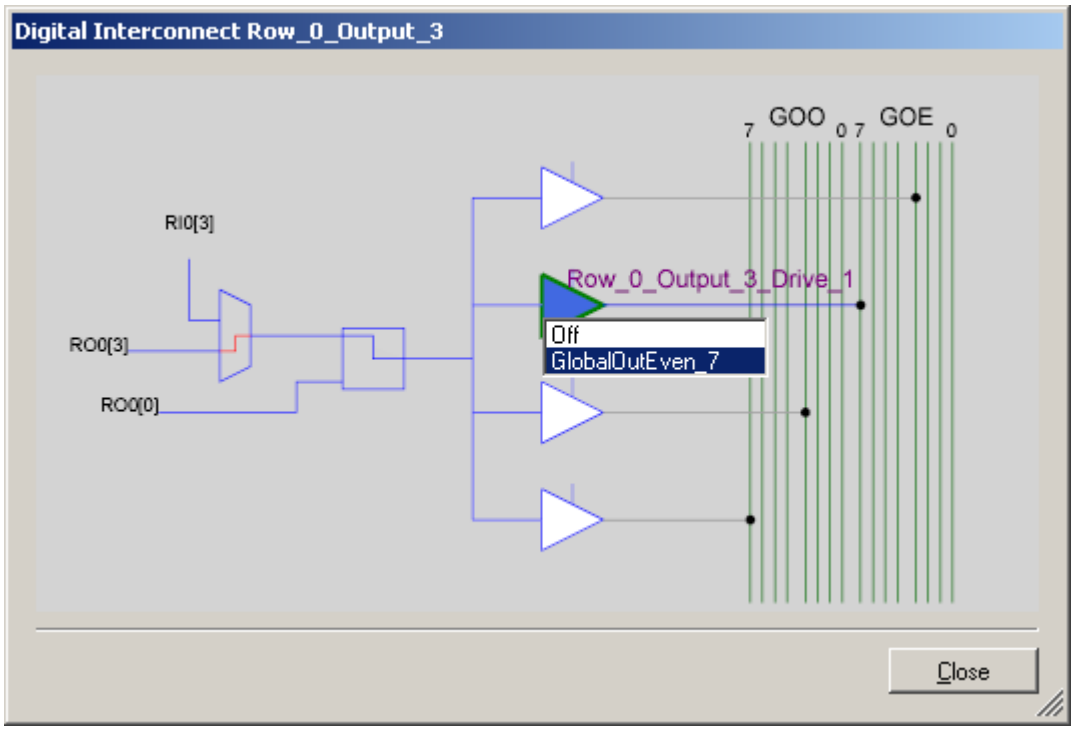
14. Next, route the PWM **CompareOut** signal to P0[7]. The first step is to configure the lookup table (LUT) on **Row_0_Output3**.

Figure 3-57. Route the PWM CompareOut signal to P0[7]



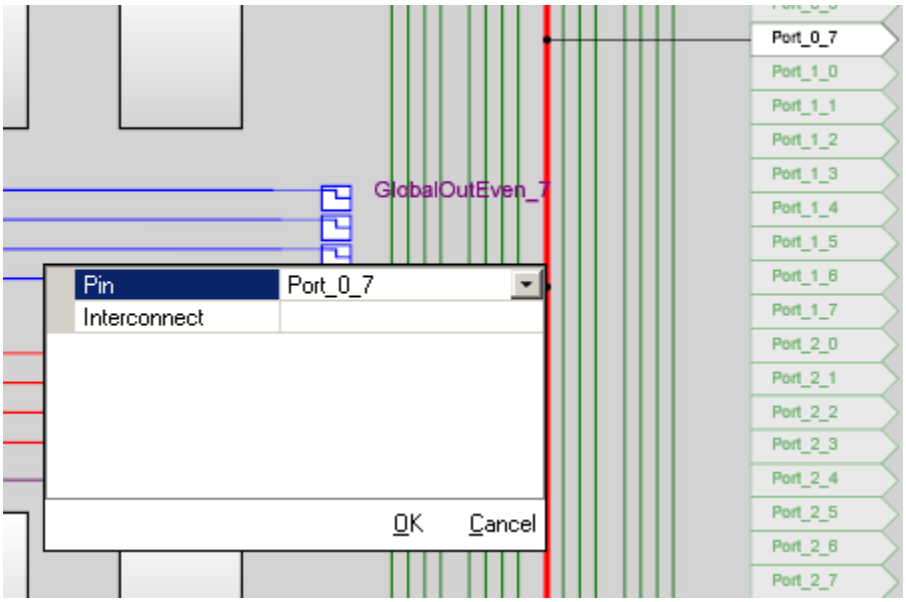
15. Double-click the LUT, the **Digital Interconnect** window opens.
16. In this window, enable **Row_0_Output_3_Drive_1** to connect to **GlobalOutEven_7**.

Figure 3-58. Digital Interconnect Window



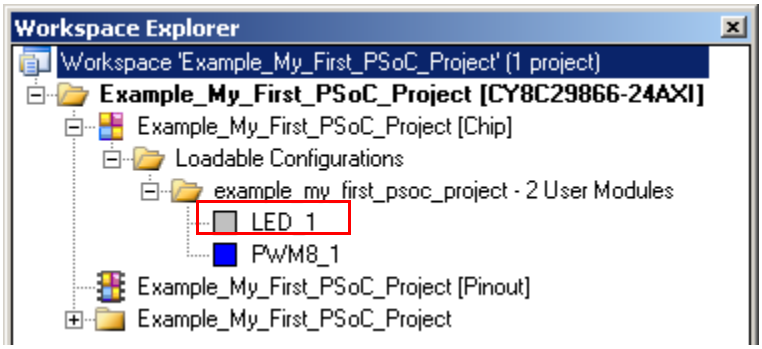
17. Click **Close**.
18. Click on **GlobalOutEven_7**. In the window that appears, configure **Pin** for **Port_0_7**.

Figure 3-59. Configure Pin for Port_0_7



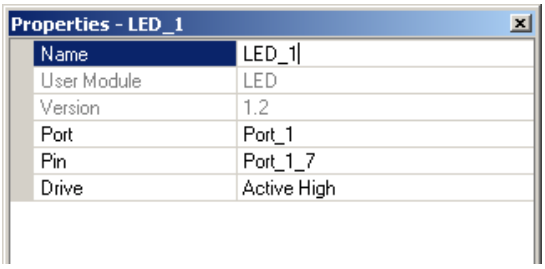
19. Click **OK** to continue.
20. In the **User Modules** window expand the **Misc Digital** folder. In this folder, right-click the **LED** and select **Place**; this adds the UM to the project. This UM does not use digital or analog blocks. It appears in **Workspace Explorer** → **Example_My_First_PSoC_Project[CY8C29]** → **Example_My_First_PSoC_Project[Chip]** → **Loadable Configurations** → **example_my_first_psoc_project - 2 User Modules**.

Figure 3-60. Workspace Explorer



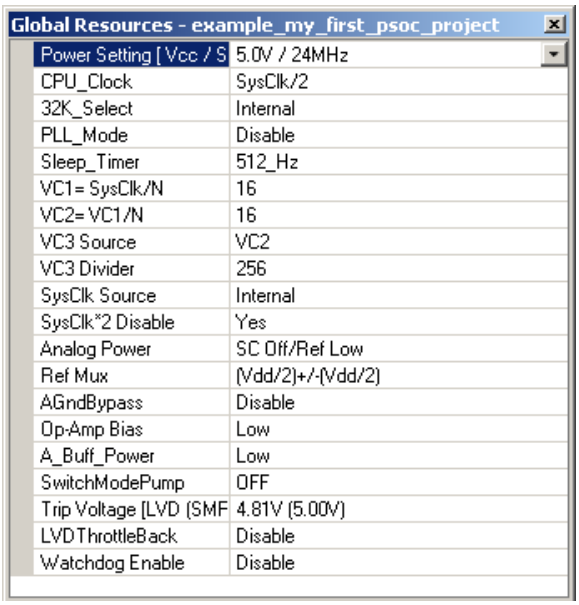
21. Double-click the **LED_1** UM and navigate to the **Properties** window. Configure the LED for **Port_1_7.**

Figure 3-61. Properties Window



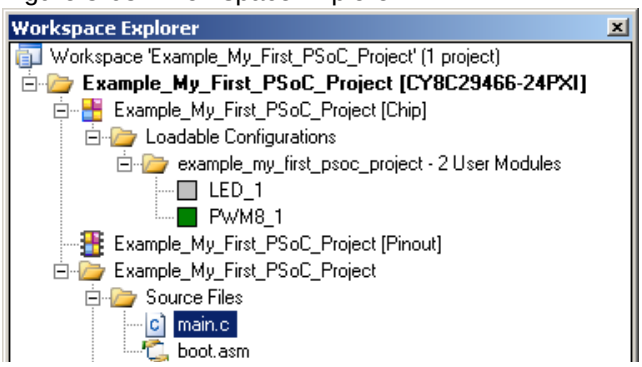
22. Configure the **Global Resources** window to match the following figure.

Figure 3-62. Global Resources Window



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** → **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 22 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           = High
*     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
*     ClockSync        = Sync to SysClk
*     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
```

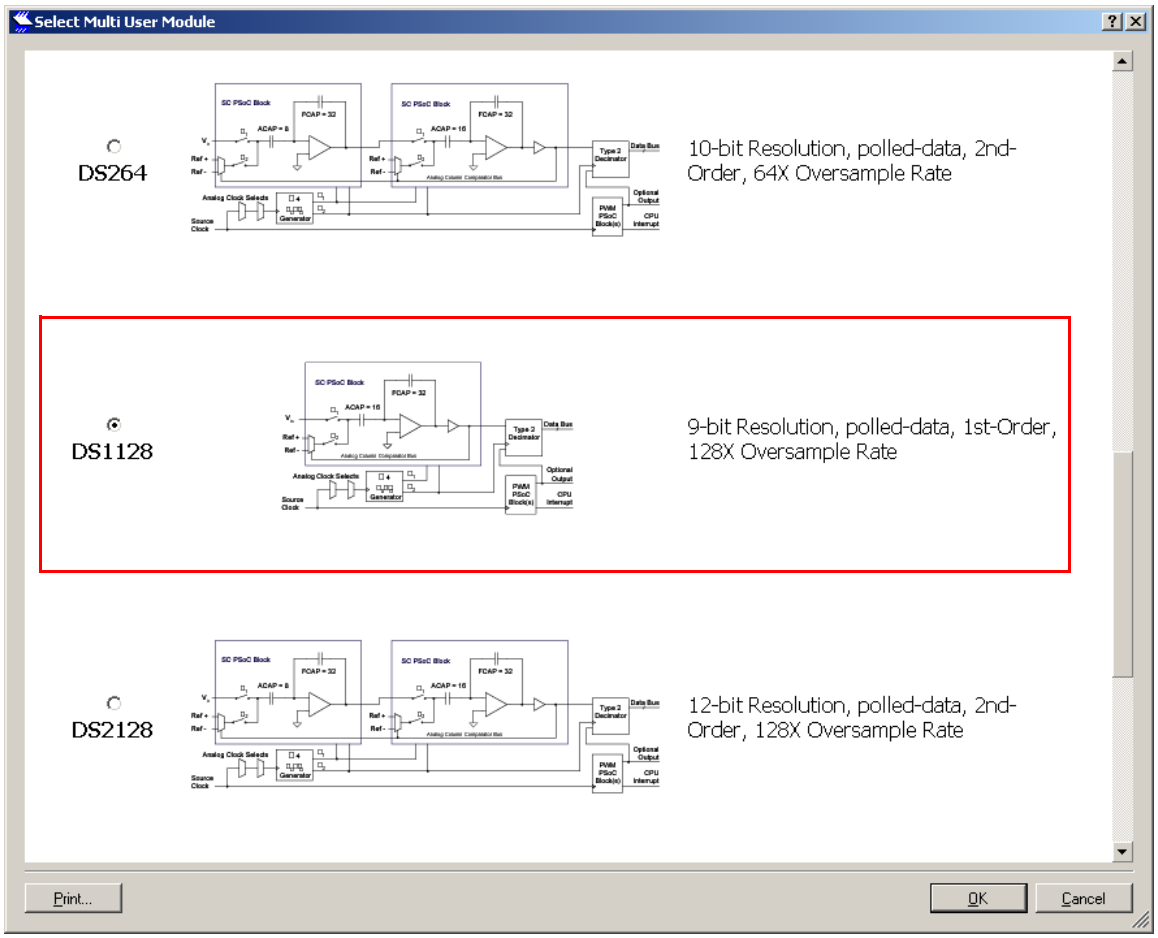
3.2.2 ADC to LCD Project

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

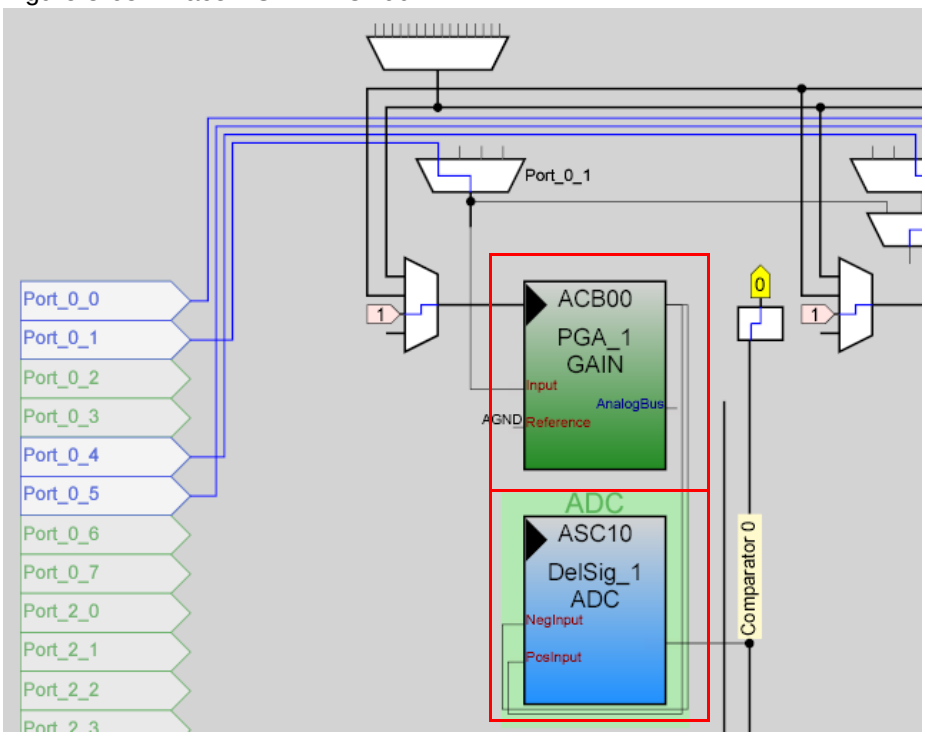
1. Follow steps 1 to 10 in section 3.2.1.1 on page 69; change the project name to **Example_ADC_to_LCD**.
2. In the **User Modules** window, expand the **ADCs** folder; right-click **DelSig** and select **Place**. A window opens with multiple options for the DelSig UM. Scroll down, if necessary, and select the **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**.

Figure 3-65. Place PGA 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

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.

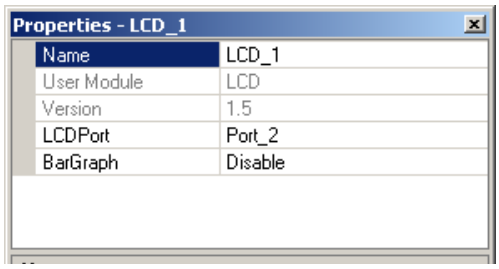
Figure 3-67. DelSig_1 Properties

Properties - DelSig_1

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

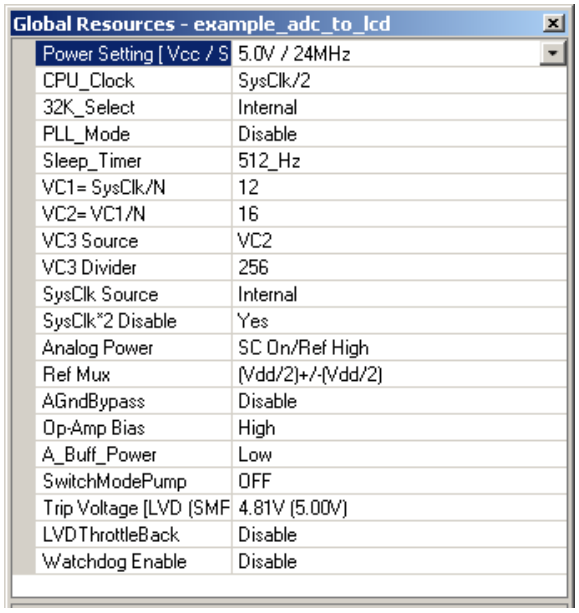
9. Double-click **LCD_1** and configure the properties to match this figure.

Figure 3-68. LCD_1 Properties



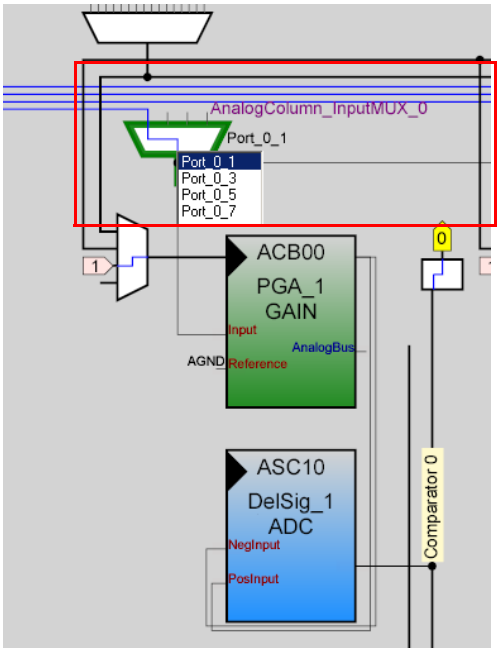
10. Configure the **Global Resources** to match the following figure.

Figure 3-69. Global Resources Properties



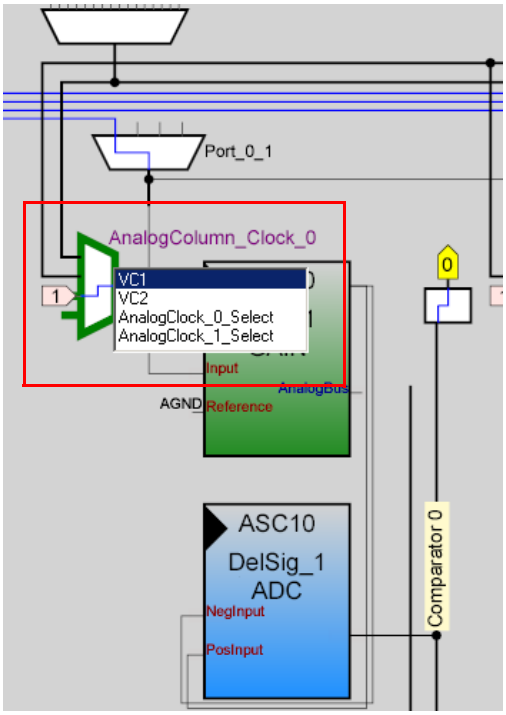
11. Ensure that **AnalogColumn_InputMUX_0** is connected to **Port_0_1**. If it is not configured for this port, double-click the mux and choose **Port_0_1**.

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**.

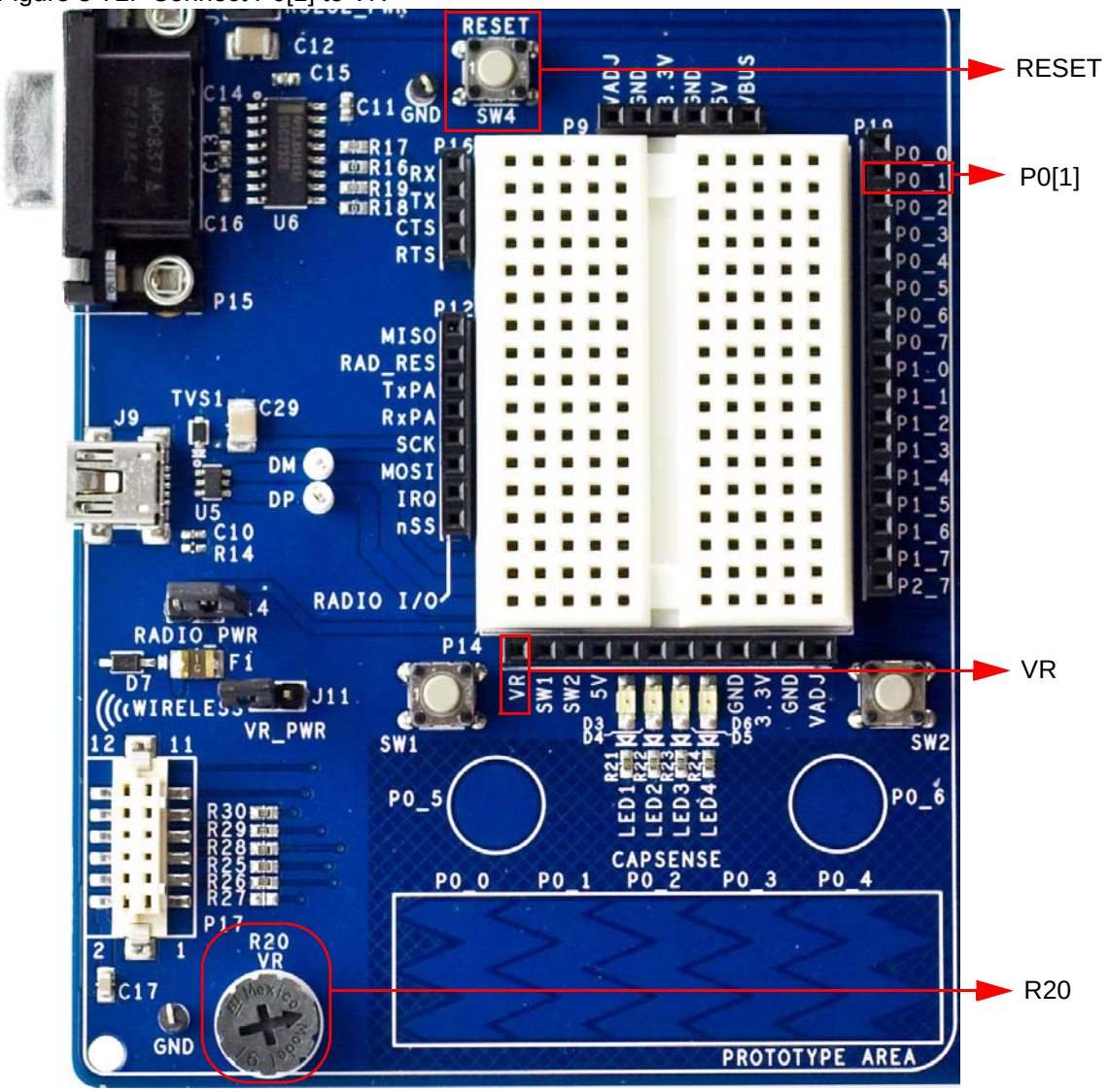
Figure 3-71. AnalogColumn_Clock_0 is connected to VC1



13. Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *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** → **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

Figure 3-72. 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.

3.2.2.2 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
*     DataClock    = VC1       // VC1 = 24MHz/12 = 2MHz
*     ClockPhase   = Normal
*     PosInput     = ACB00 (PGA_1)
*     NegInput     = ACB00 *Note, this parameter is unused
*     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
   }


}
```

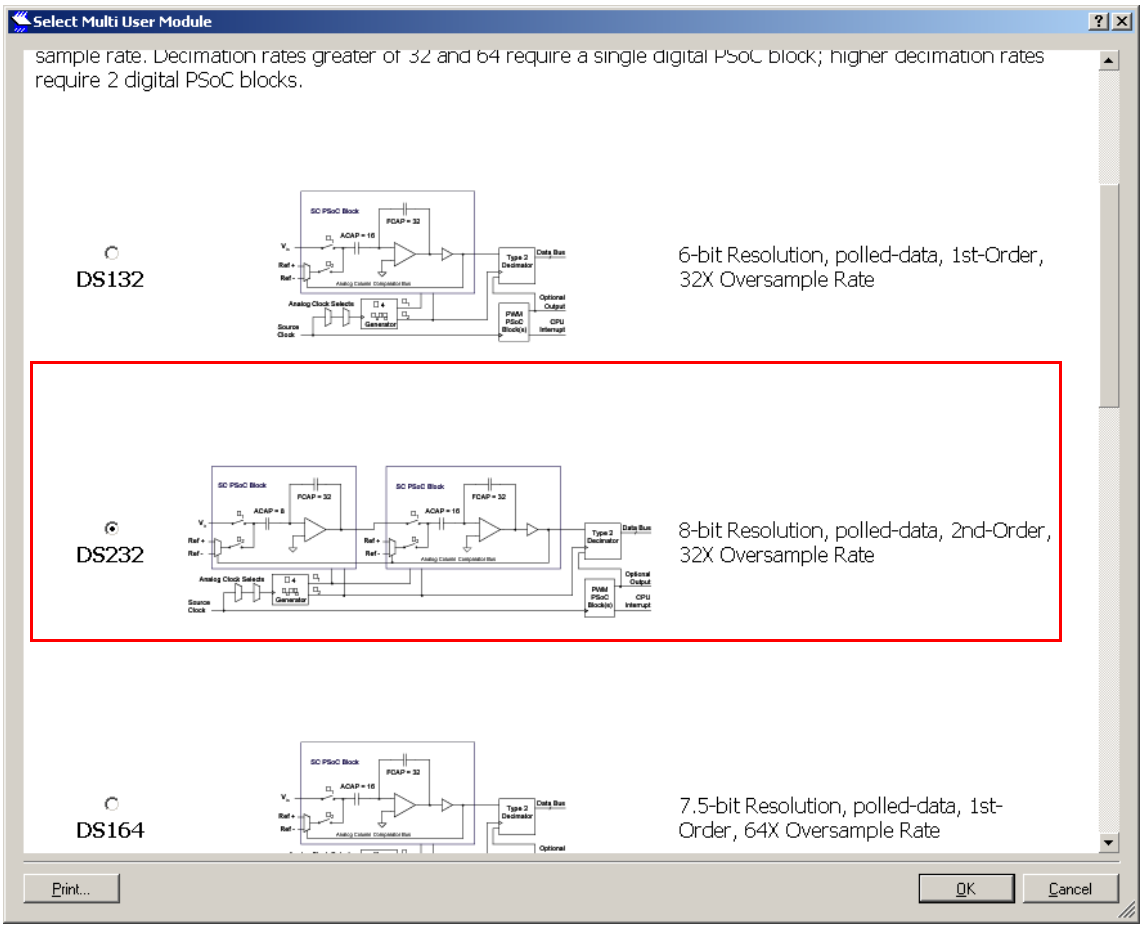
3.2.3 ADC to LCD with DAC and UART

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 section 3.2.1.1 on page 69; 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**.

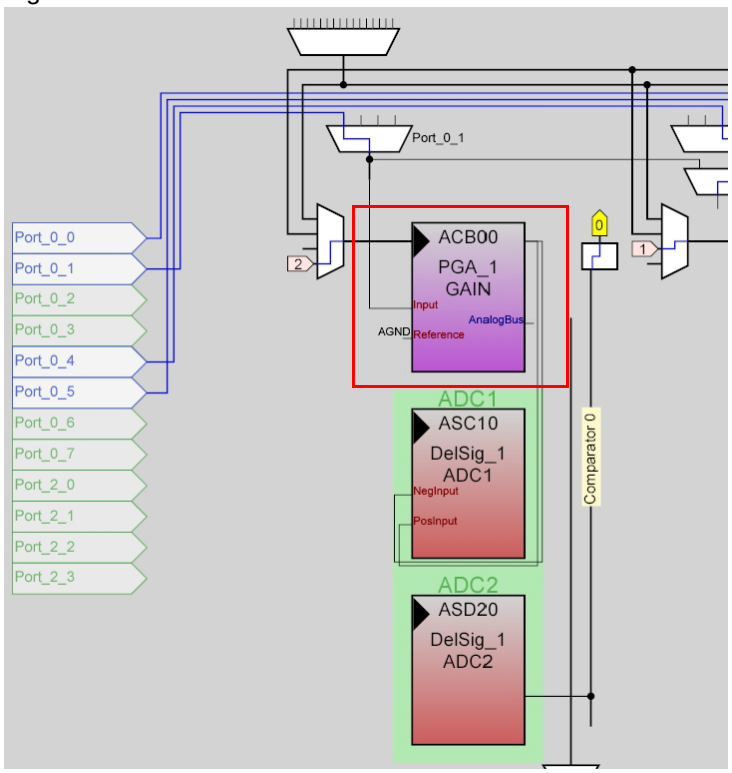
Figure 3-73. Select Multi User Module Window



3. Click **OK**.

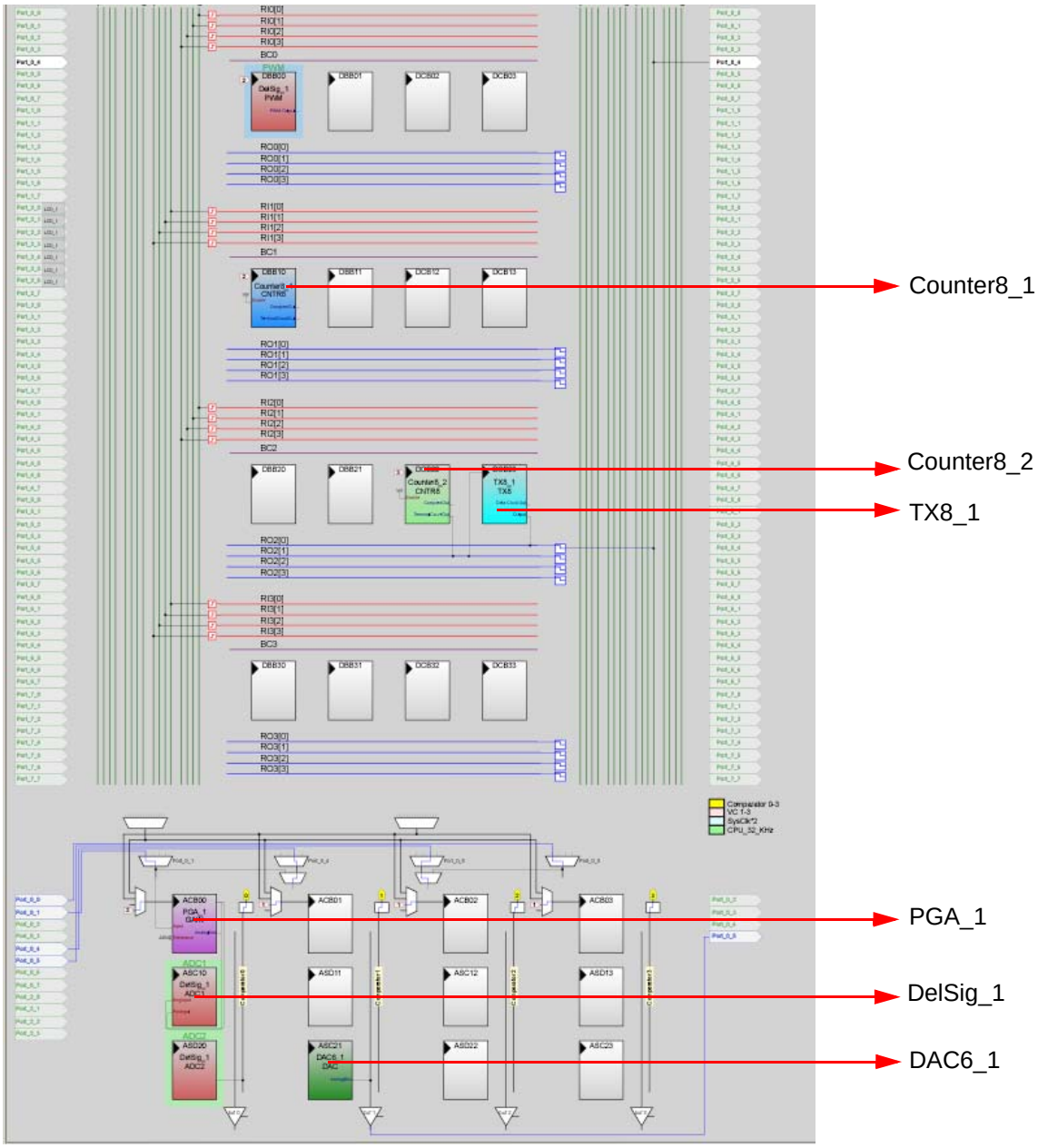
4. Verify that the UM is placed in **ASC10**.
5. In the **User Modules** window, expand the **Amplifiers** window. Right-click **PGA** and select **Place**. Ensure that the **PGA** is placed in **ACB00**.

Figure 3-74. Place PGA in ACB00



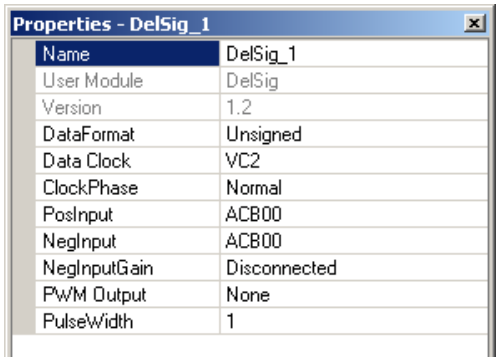
6. In the **User Modules** window, expand **Misc Digital**, right-click **LCD**, and select **Place**.
7. In the **User Modules** window, expand **Counters**, right-click **Counter8**, and select **Place**. Complete this step twice to place two **Counter8**s.
8. In the **User Modules** window, expand **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 86.

Figure 3-75. Configure User Modules



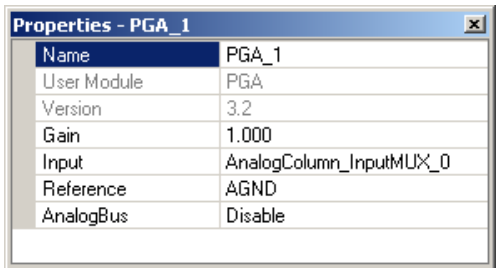
11. Double-click **DelSig_1** and configure it to match this figure.

Figure 3-76. DelSig_1 Properties



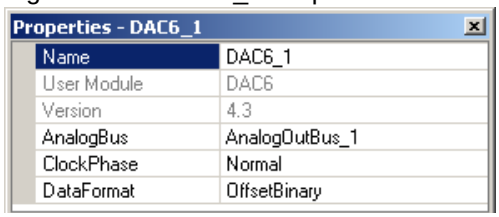
12. Double-click **PGA_1** and configure it to match this figure.

Figure 3-77. PGA_1 Properties



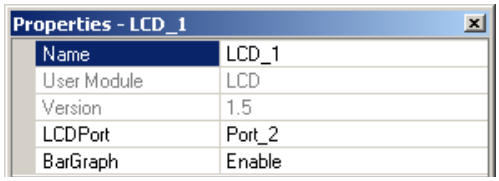
13. Double-click **DAC6_1** and configure it to match this figure.

Figure 3-78. DAC6_1 Properties



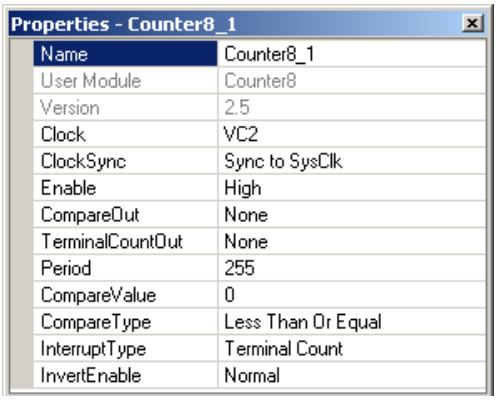
14. Double-click **LCD_1** and configure it to match this figure.

Figure 3-79. LCD_1 Properties



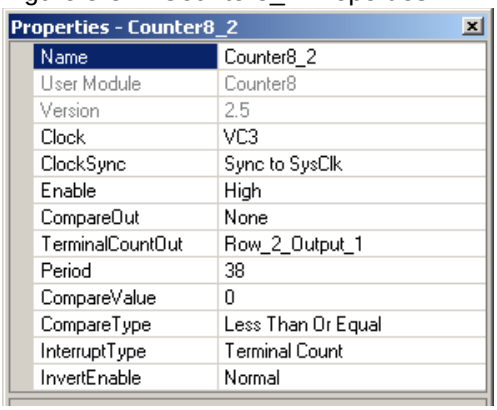
15. Double-click **Counter8_1** and configure it to match this figure.

Figure 3-80. Counter8_1 Properties



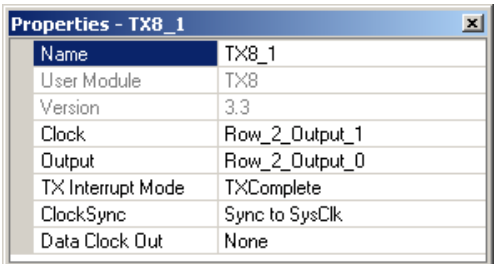
16. Double-click **Counter8_2** and configure it to match this figure.

Figure 3-81. Counter8_2 Properties



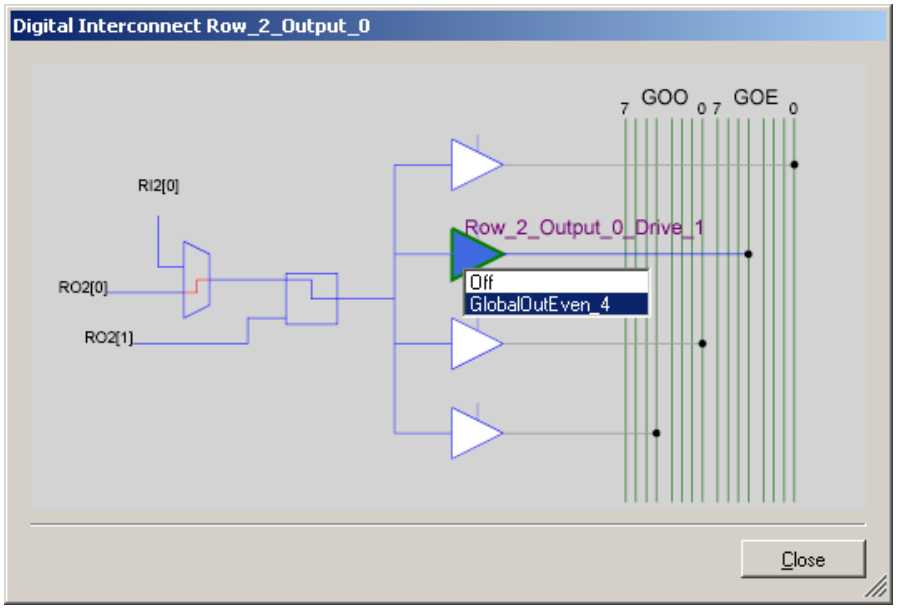
17. Double-click **TX8_1** and configure it to match this figure.

Figure 3-82. TX8_1 Properties



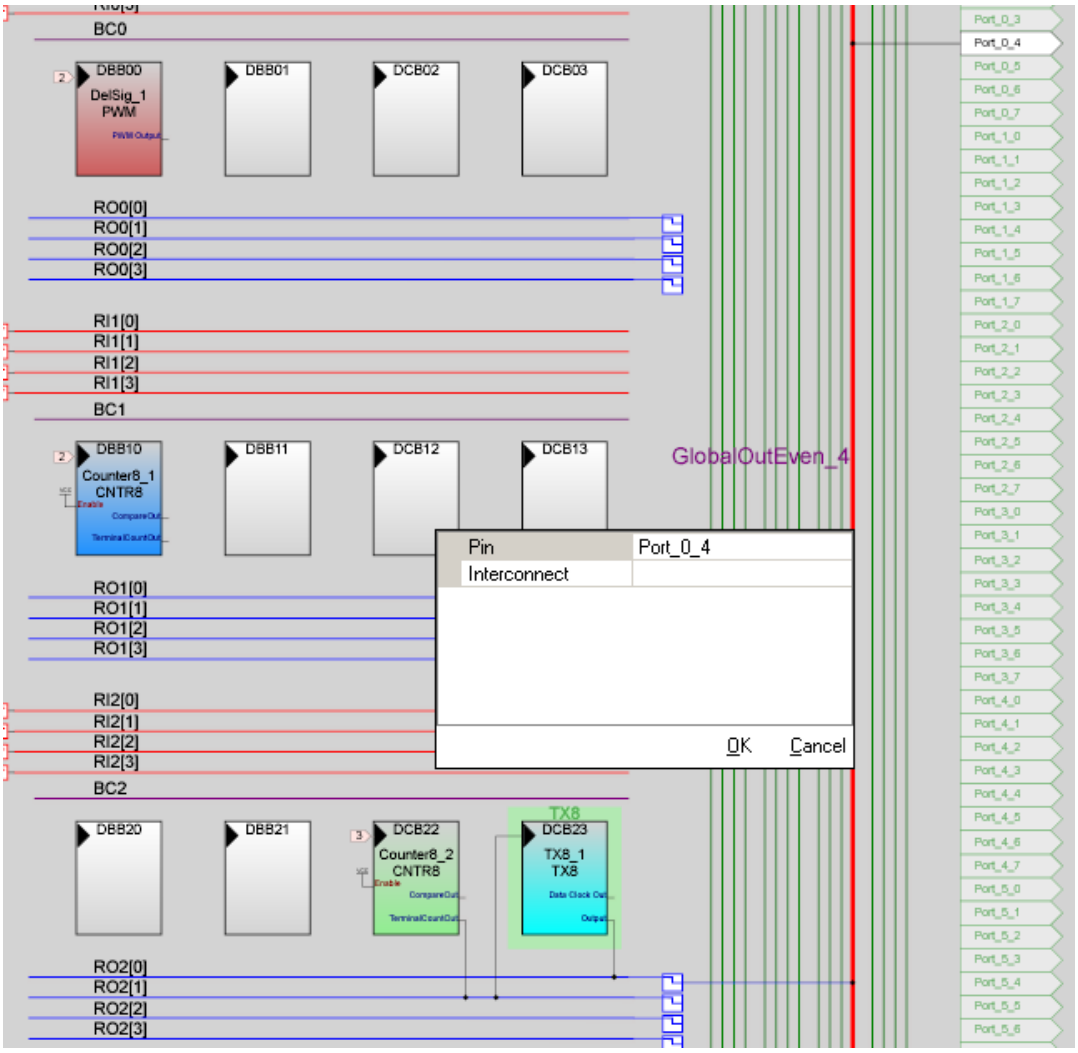
18. Double-click **RO2[0] LUT**, enable **Row_2_Output_0_Drive_1** to connect **GlobalOutEven_4**.

Figure 3-83. Digital Interconnect Window



19. Double-click **GlobalOutEven_4**. In the window that appears, configure **Pin** for **Port_0_4**.

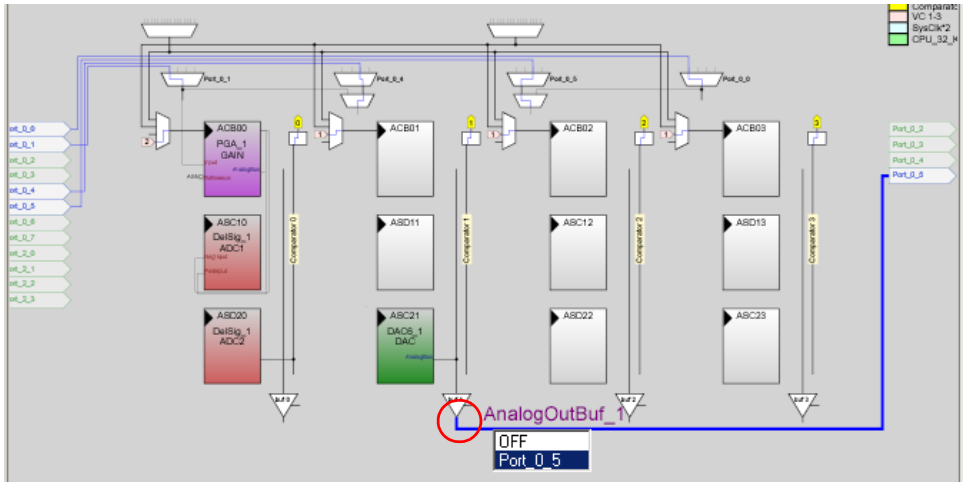
Figure 3-84. Configure Pin for Port_0_4



20. Click **OK** to continue.

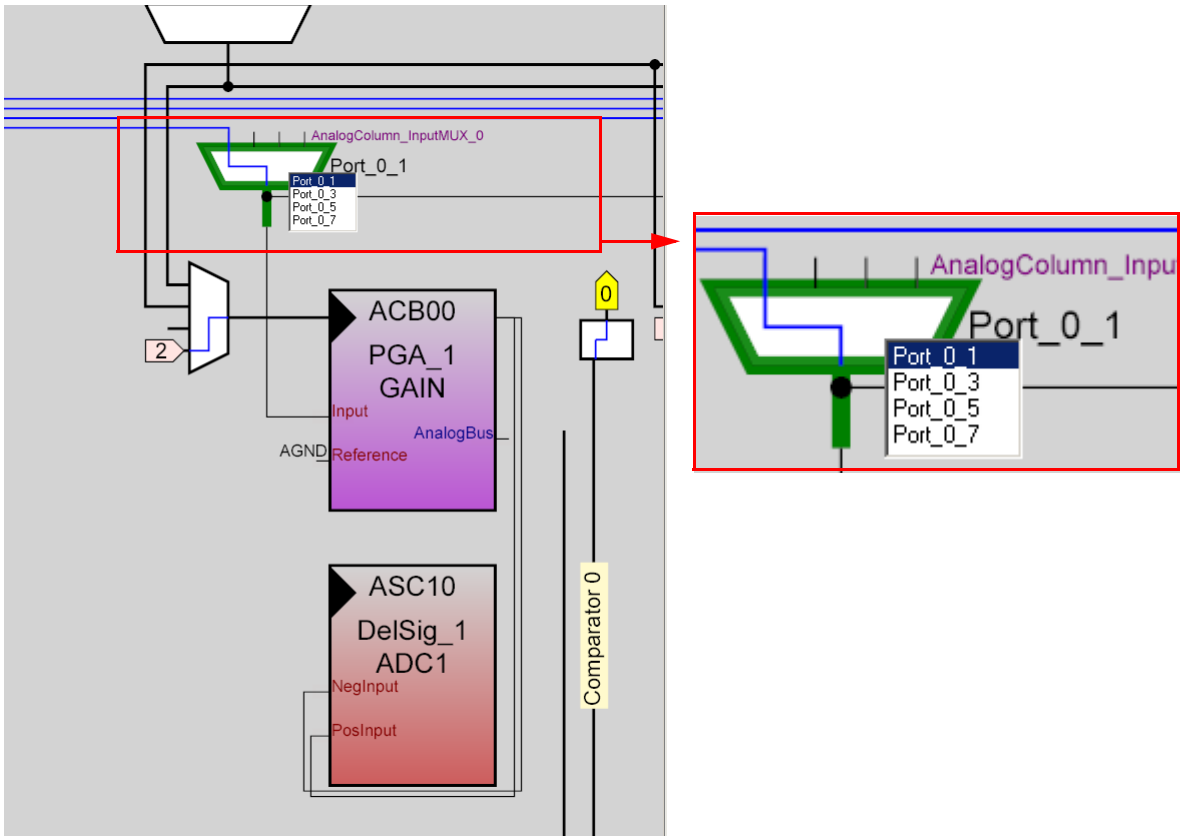
21. Click **AnalogOutBuf_1** and configure it for **Port_0_5**.

Figure 3-85. Configure AnalogOutBuf_1



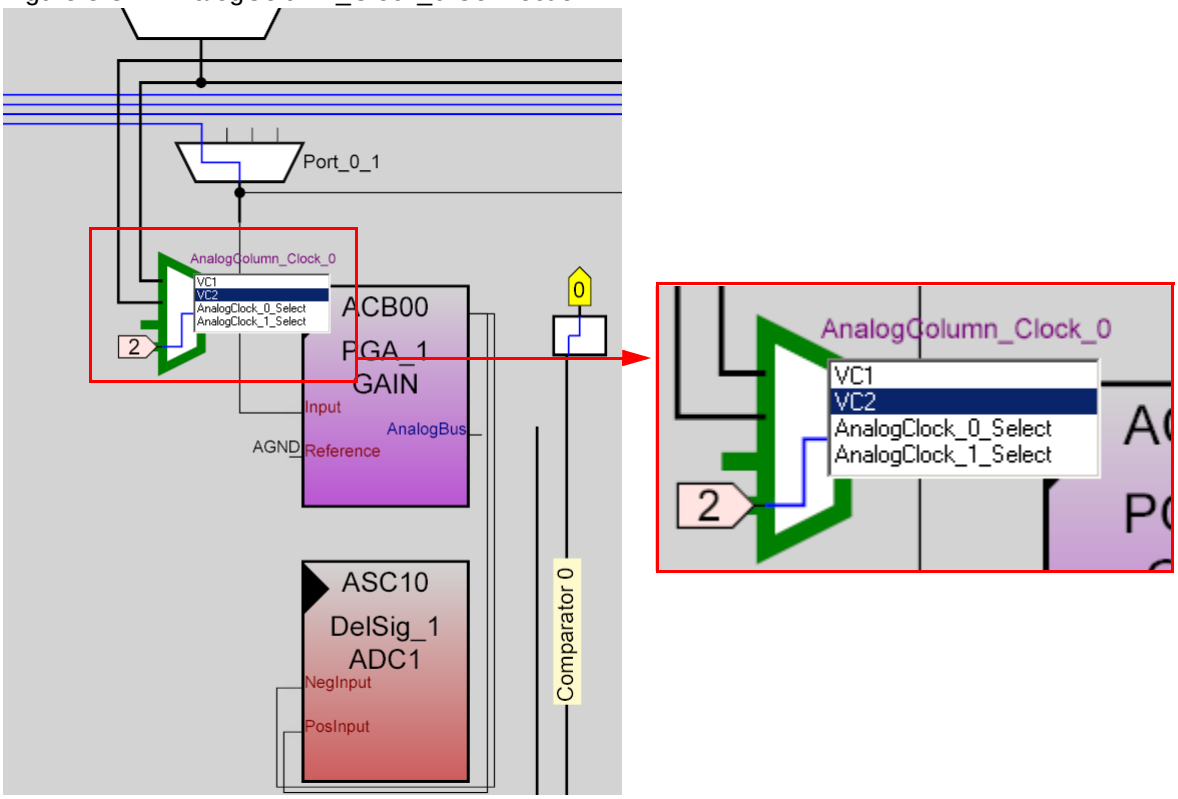
22. Verify that **AnalogColumn_InputMUX_0** is connected to **Port_0_1**. If it is not configured for this port, double-click the mux and choose **Port_0_1**.

Figure 3-86. AnalogColumn_InputMUX_0 Connection



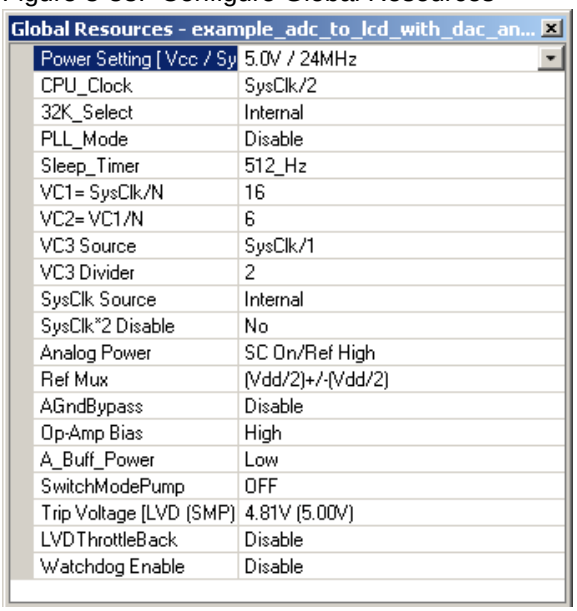
23. Verify that **AnalogColumn_Clock_0** is connected to **VC2**. If it is not, double-click the mux and chose **VC2**.

Figure 3-87. AnalogColumn_Clock_0 Connection



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

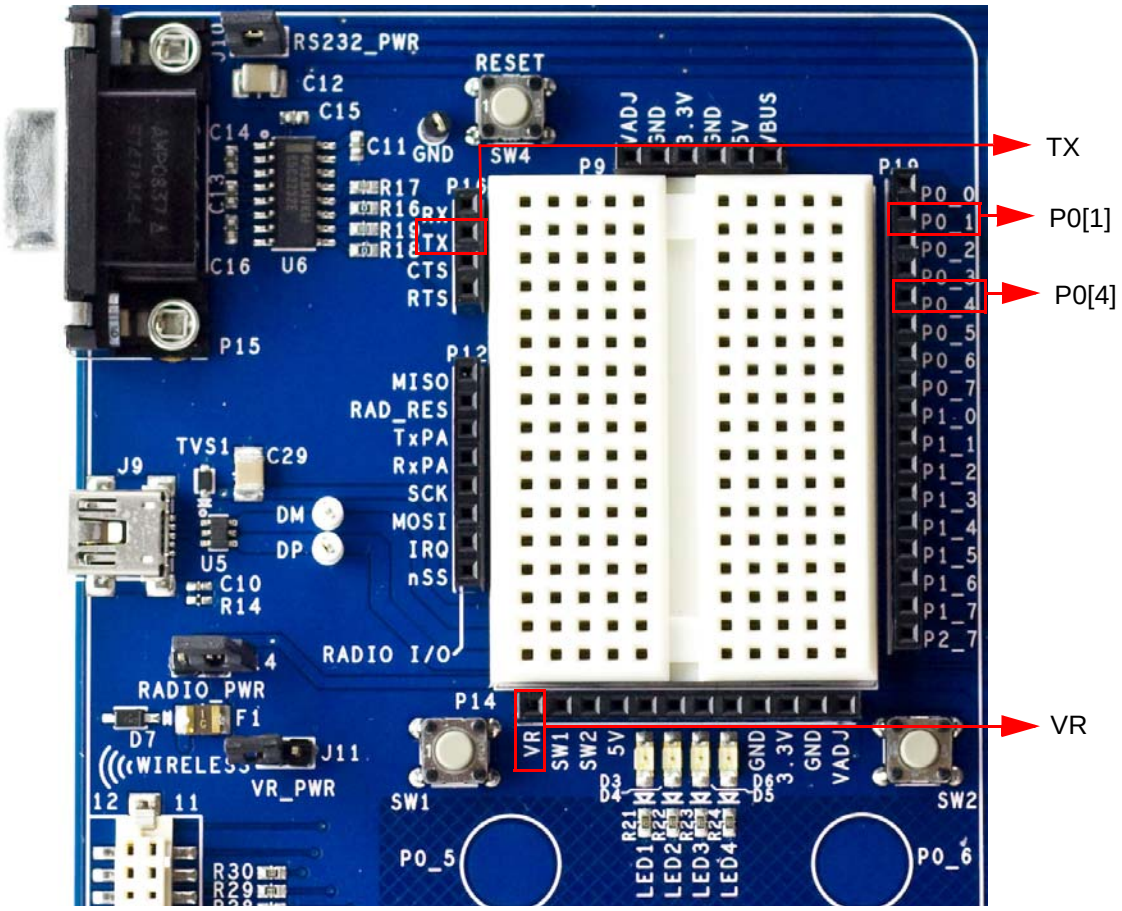
Figure 3-88. Configure Global Resources



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** → **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 22 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)
*
*     Clock            = VC2 // VC2 = 24MHz/16/16 = 250kHz
*     ClockSync        = Sync to SysClk
*     Enable           = High
*     CompareOut       = None
*     TerminalCountOut = None
*     Period           = 255   *Note this parameter is updated in the main loop
*     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
*     ClockSync        = Sync to SysClk
*     Enable           = High
*     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.
*
*     Clock             = Row_2_Output_1 (From Counter8_1)
*     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** → **lib** → **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:
   index   _SINtable;Get the value in the SINtable pointed to by [bTablePos]
   lcall   DAC6_1_WriteBlind;Write value from SINtable (stored in A) to the DAC

   pop X
   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 / CY8C55 Family Processor Module Code Examples

3.3.1 My First PSoC 3 / PSoC 5 Project

This project demonstrates basic hardware and software functionality with the PSoC 3 or PSoC 5 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.

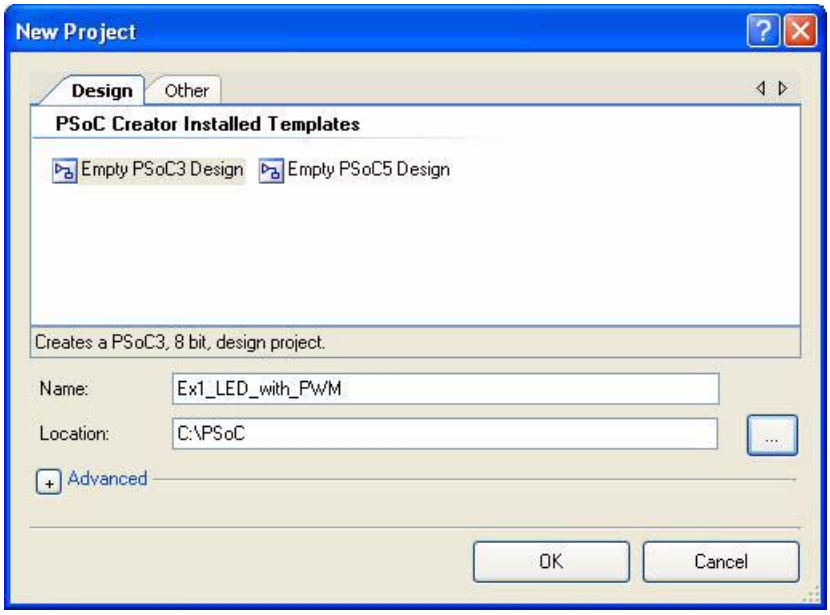
This code example uses these components:

- DigitaL Output Pin (**Component Catalog → Ports and Pins → Digital Output Pin**)
- PWM (**Component Catalog → Digital Functions → PWM**)
- Clock (**Component Catalog → System → Clock**)
- Logic Low (**Component Catalog → Digital → Logic → Logic Low**)
- Logic High (**Component Catalog → Digital → Logic → Logic High**)

3.3.1.1 Creating My First PSoC 3 / PSoC 5 Project

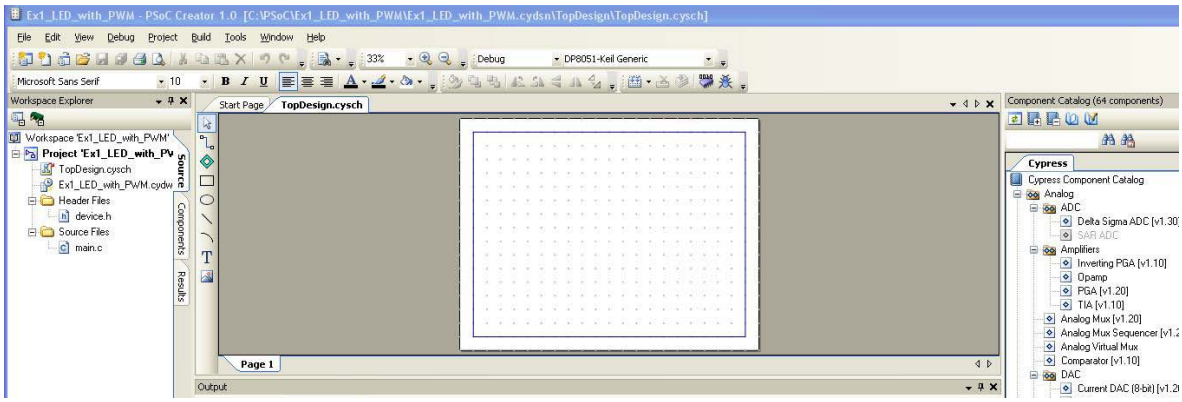
1. Open PSoC Creator.
2. Create a new project by clicking **Create New Project…** in the **Start Page** of PSoC Creator.
3. In the **New Project** window, select the **Empty PSoC3 Design** template for a PSoC 3 design, or **Empty PSoC5 Design** template for a PSoC 5 design and name the project **Ex1_LED_with_PWM**.
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



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



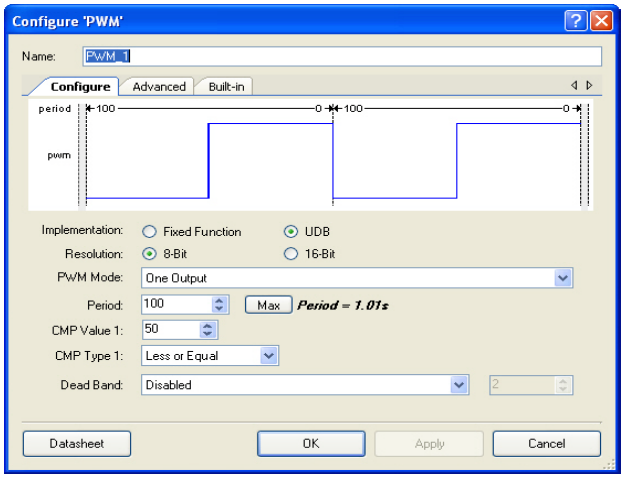
3.3.1.2 *Placing and Configuring PWM*

1. Drag and drop the **PWM** component (**Component Catalog** → **Digital** → **Functions** → **PWM**) to-workspace.
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

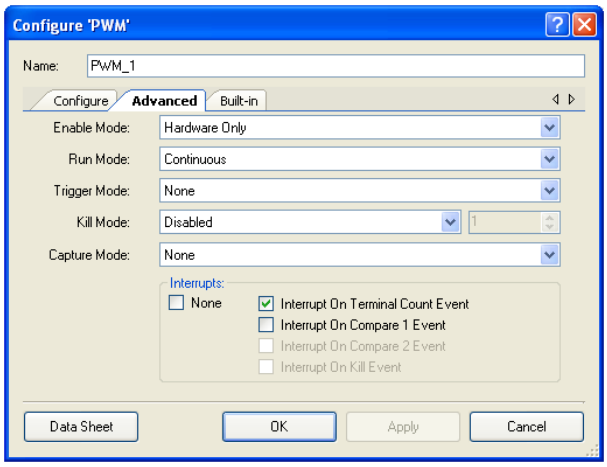
Figure 3-92. PWM Component Configuration



   Advanced Tab

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

Figure 3-93. PWM Component Advanced Tab Configuration



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

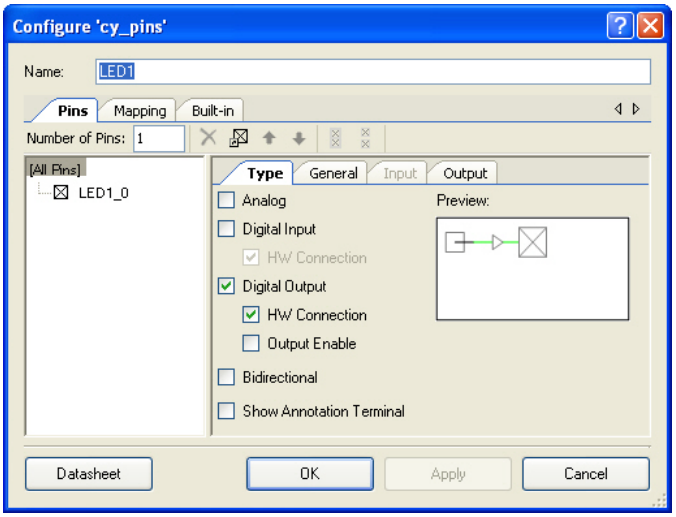
3.3.1.3 *Placing and Configuring Digital Output Pin Hardware*

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

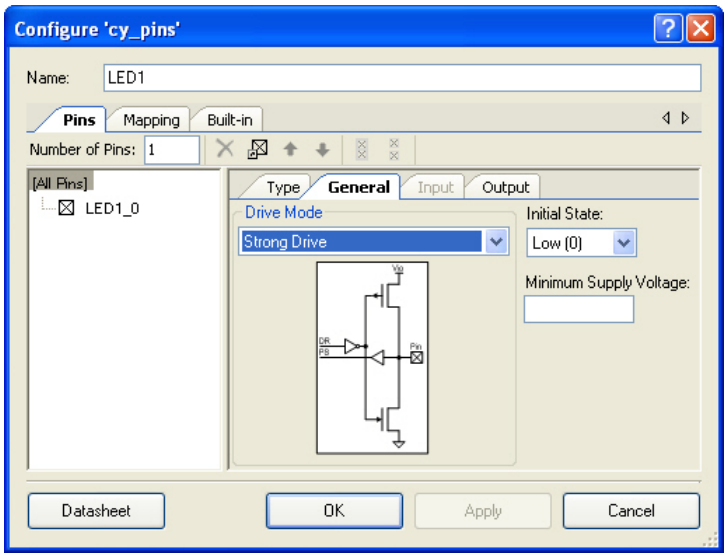
Figure 3-94. LED1 Component Configuration



General Tab

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

Figure 3-95. Pins - LED1 Component Configuration



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

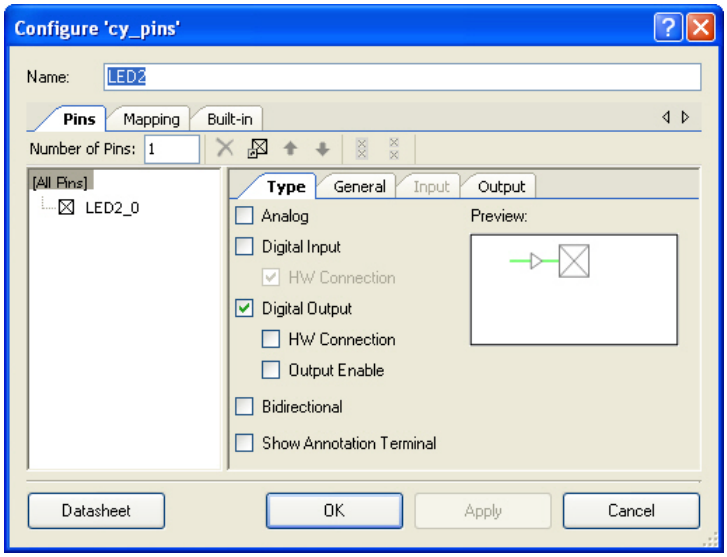
3.3.1.4 *Placing and Configuring Software Digital Output Pin*

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

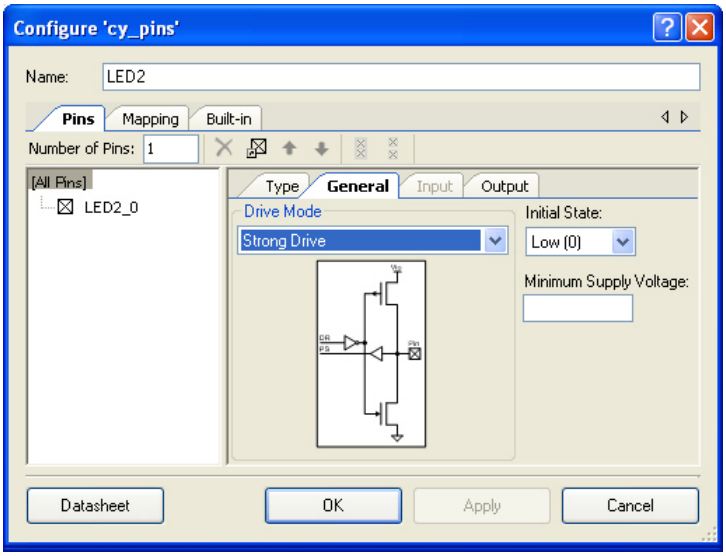
Figure 3-96. LED2 Component Configuration



General Tab

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

Figure 3-97. Pins - LED2 Component Configuration



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

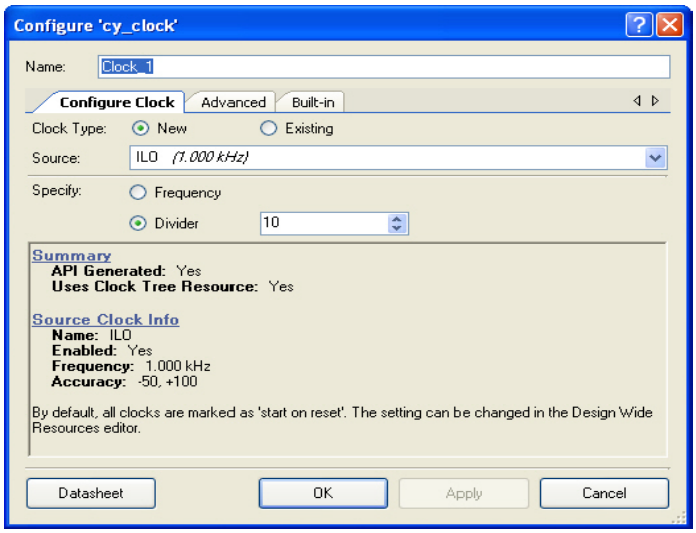
3.3.1.5 *Connecting the Components Together*

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** → **Digital** → **Logic** → **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** → **System** → **Clock**) to the **clock** on the PWM.
5. Double-click the **Clock_1** component to configure.
6. Configure the clock:

   Configure Clock Tab

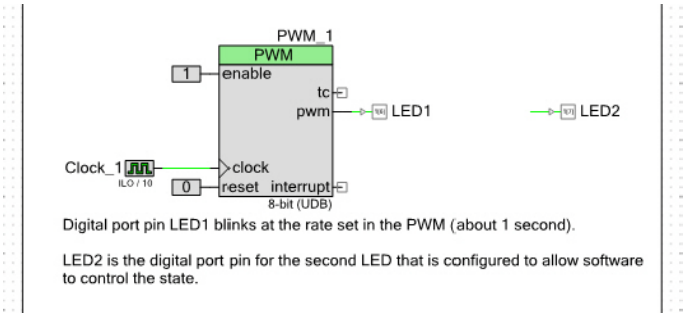
   - **Name**: Clock_1
   - **Source**: ILO (1.000 kHz)
   - Select **Divider** and set the value as 10
   - Leave the remaining parameters as default

Figure 3-98. Clock Component Configuration



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

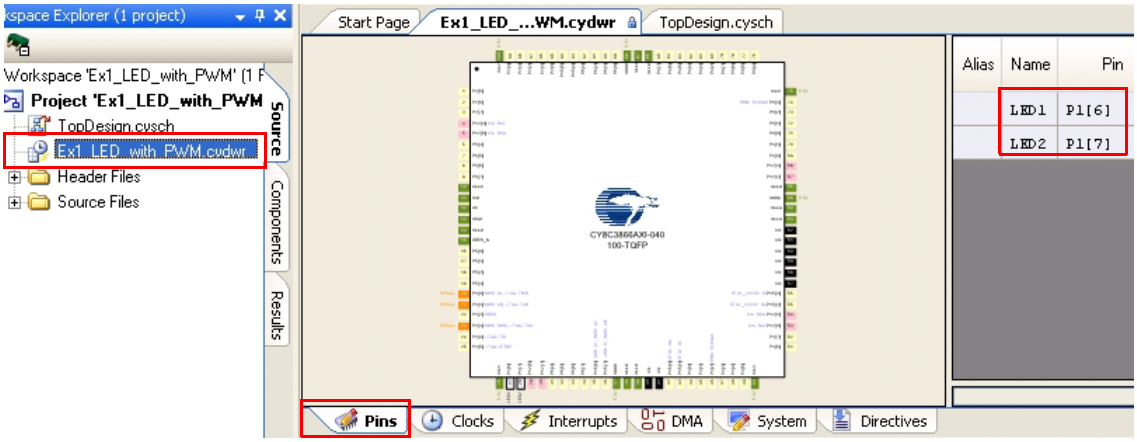
Figure 3-99. Connected Components



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**.
2. Replace the existing *main.c* content with the content of the embedded *CY8C38_main_Ex1.c* file, which is available within the attachments feature of this PDF document.

Notes

- 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.
- Use the PSoC 3 family processor module file *CY8C38_main_Ex1.c* to replace the *main.c* content for PSoC 5 CY8C55 family processor module.

```
#include <device.h>

#define MS_DELAY 167u /* For delay, about 167ms */


/*******************************************************************************
* Function Name: main
********************************************************************************
*
* Summary:
*  The main function initializes the PWM and starts the PWM clock which will
*  blink LED1 at about once a second. 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)
{
    uint8 ledState = 0x00; /* Initially set LED2 to off */

    Clock_1_Enable(); /* Start the clock */
    PWM_1_Start();    /* Enable PWM      */
```

```
    /* Following loop does software blinking of LED2 connected to P1.7 */
    while (1)
    {

        CyDelay(MS_DELAY);   /* Have software loop blink control   */
        ledState ^= 0x01u;   /* Toggle LED2 setting between low and high */
        LED2_Write(ledState); /* Set LED2 */
    }
}


/* [] END OF FILE */
```

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 27 or Programming My First PSoC 5 Project on page 31 to program the device.
6. 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 5 Project and therefore have a basic understanding of the PSoC Creator software environment.

This code example uses these components:

- Delta Sigma ADC (**Component Catalog** → **Analog** → **ADC** → **Delta Sigma ADC**)
- Character LCD (**Component Catalog** → **Display** → **Character LCD**)
- Analog Pin (**Component Catalog** → **Ports and Pins** → **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.

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 5 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

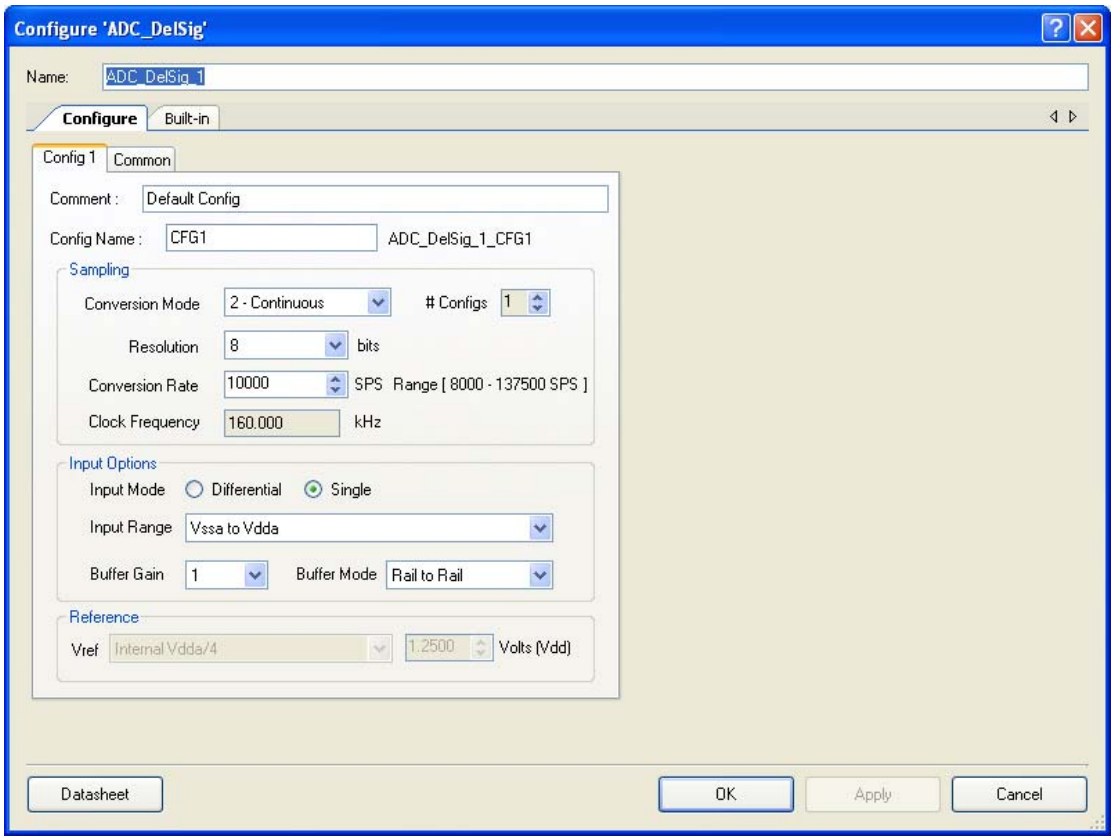
1. Drag and drop the Delta Sigma ADC component (**Component Catalog** → **Analog** → **ADC** → **Delta Sigma ADC**).
2. Double-click the **ADC_DelSig_1** component in the schematic to open the configuration window.
3. Configure the Delta Sigma ADC as follows:

   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

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

Figure 3-101. ADC Component Configuration



3.3.2.3 *Placing and Configuring an Analog Pin*

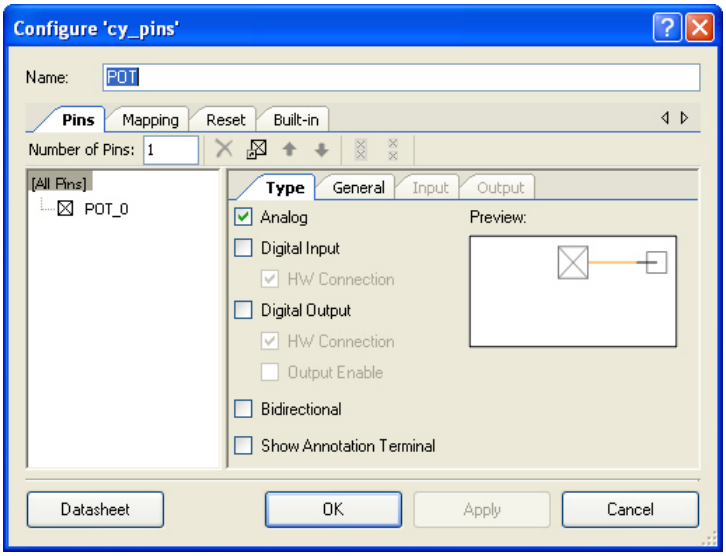
1. Drag and drop the analog pin component (**Component Catalog → Ports and Pins → Analog Pin**).
2. Double-click on the **Pin_1** component in the schematic to open the configuration window.
3. Configure the analog pin as follows:

   Type Tab

   - **Name**: POT
   - Select **Analog** check box only

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

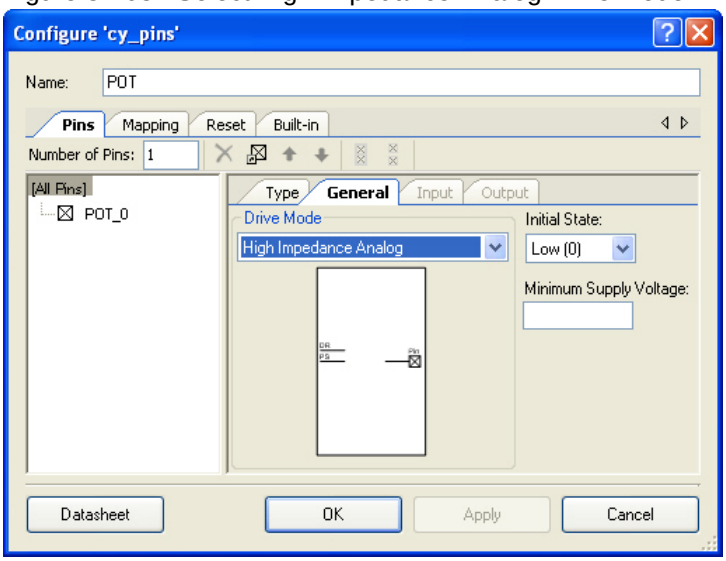
Figure 3-102. Analog Pin Component Configuration



General Tab

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

Figure 3-103. Select High Impedance Analog Drive Mode

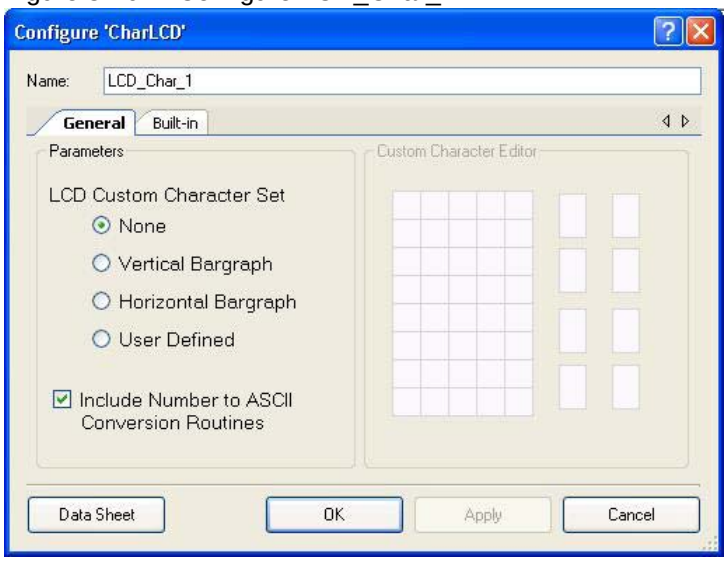


3.3.2.4 *Placing and Configuring Character LCD*

1. Drag and drop the character LCD component (**Component Catalog** → **Display** → **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.

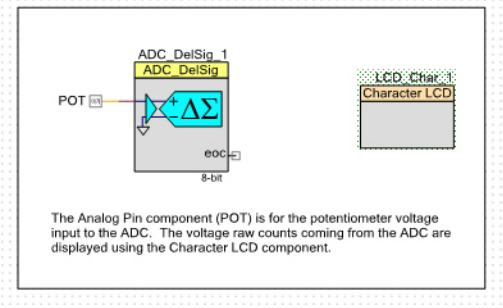
Figure 3-104. Configure LCD_Char_1



3.3.2.5 *Connecting the Components Together*

1. Using the Wire Tool, connect **POT** to **ADC_DelSig (ADC_DelSig_1)**.
2. When complete, the schematic looks similar to Figure 3-105.

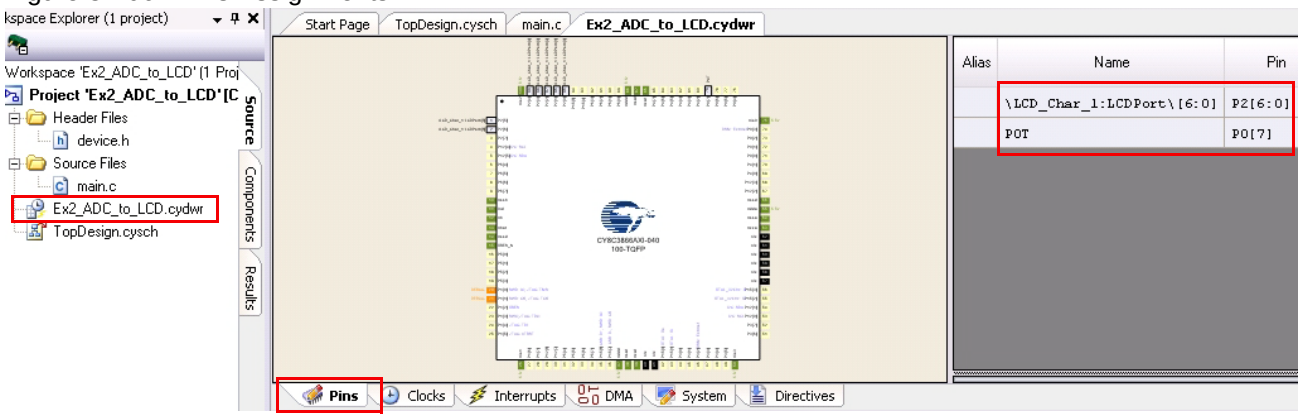
Figure 3-105. Connected Components



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



3.3.2.7 *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_Ex2.c* file, which is available within the attachments feature of this PDF document.

Notes

- 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.
- Use the PSoC 3 family processor module file *CY8C38_main_Ex2.c* to replace the *main.c* content for PSoC 5 CY8C55 family processor module.

```
#include <device.h>

/* LCD specific */
#define ROW_0       0  /* LCD row 0     */
#define COLUMN_0    0  /* LCD column 0  */
#define COLUMN_9    9  /* LCD column 9  */
#define COLUMN_10  10  /* LCD column 10 */
#define COLUMN_11  11  /* LCD column 11 */
/* For clearing Tens and Hundreds place */
#define CLEAR_TENS_HUNDREDS "    "
/* For clearing Hundreds place */
#define CLEAR_HUNDREDS      "   "

void UpdateDisplay(uint16 voltageRawCount);


/*******************************************************************************
* 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:
```

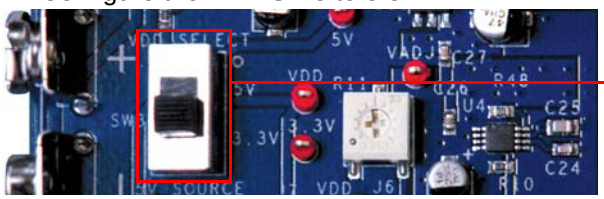
```
*  void
*
*******************************************************************************/
void main()
{
    uint16 voltageRawCount;

    ADC_DelSig_1_Start();  /* Configure and power up ADC */
    LCD_Char_1_Start(); /* Initialize and clear the LCD */

    LCD_Char_1_Position(ROW_0,COLUMN_0); /* Move the cursor to Row 0 Column 0 */

    /* Print Label for the pot voltage raw count */
    LCD_Char_1_PrintString("V Count: ");

    ADC_DelSig_1_StartConvert(); /* Force ADC to initiate a conversion */

    while(1)
    {
        /* Wait for end of conversion */
        ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT);
        voltageRawCount = ADC_DelSig_1_GetResult16(); /* Get converted result */

        /* Set range limit */
        if (voltageRawCount > 0x7FFF)
        {
            voltageRawCount = 0;
        }
        else
        {
            /* Continue on */
        }

        UpdateDisplay(voltageRawCount); /* Print result on LCD */
    }
}

/*******************************************************************************
* Function Name: UpdateDisplay
********************************************************************************
*
* Summary:
*   Print voltage raw count result to the LCD. Clears some characters if
*   necessary.
*
* Parameters:
*   voltageRawCount: The voltage raw counts being received from the ADC
*
* Return:
*   void
*
*******************************************************************************/
void UpdateDisplay (uint16 voltageRawCount)
{
    /* Move the cursor to Row 0, Column 9 */
    LCD_Char_1_Position(ROW_0,COLUMN_9);
    LCD_Char_1_PrintNumber(voltageRawCount); /* Print the result */
```

```
    if (voltageRawCount < 10)
    {
        /* Move the cursor to Row 0, Column 10 */
        LCD_Char_1_Position(ROW_0,COLUMN_10);
        LCD_Char_1_PrintString(CLEAR_TENS_HUNDREDS); /* Clear last characters */
    }
    else if (voltageRawCount < 100)
    {
        /* Move the cursor to Row 0, Column 11 */
        LCD_Char_1_Position(ROW_0,COLUMN_11);
        LCD_Char_1_PrintString(CLEAR_HUNDREDS); /* Clear last characters */
    }
    else
    {
        /* Continue on */
    }
}


/* [] END OF FILE */
```

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.



Move VDD SELECT
Switch to 3.3 V

3. 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 27 or Programming My First PSoC 5 Project on page 31 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** → **Analog** → **ADC** → **Delta Sigma ADC**)
- Voltage DAC (**Component Catalog** → **Analog** → **DAC** → **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**)

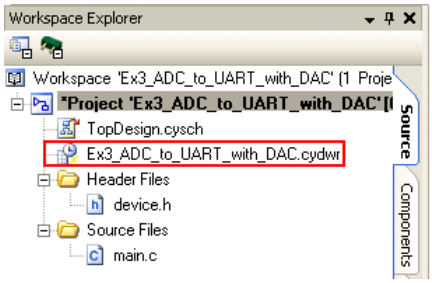
3.3.3.1 Creating ADC to UART with DAC Project

1. Open PSoC Creator.
2. Create a new project by clicking **Create New Project…** in the **Start Page** of PSoC Creator.
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 5 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

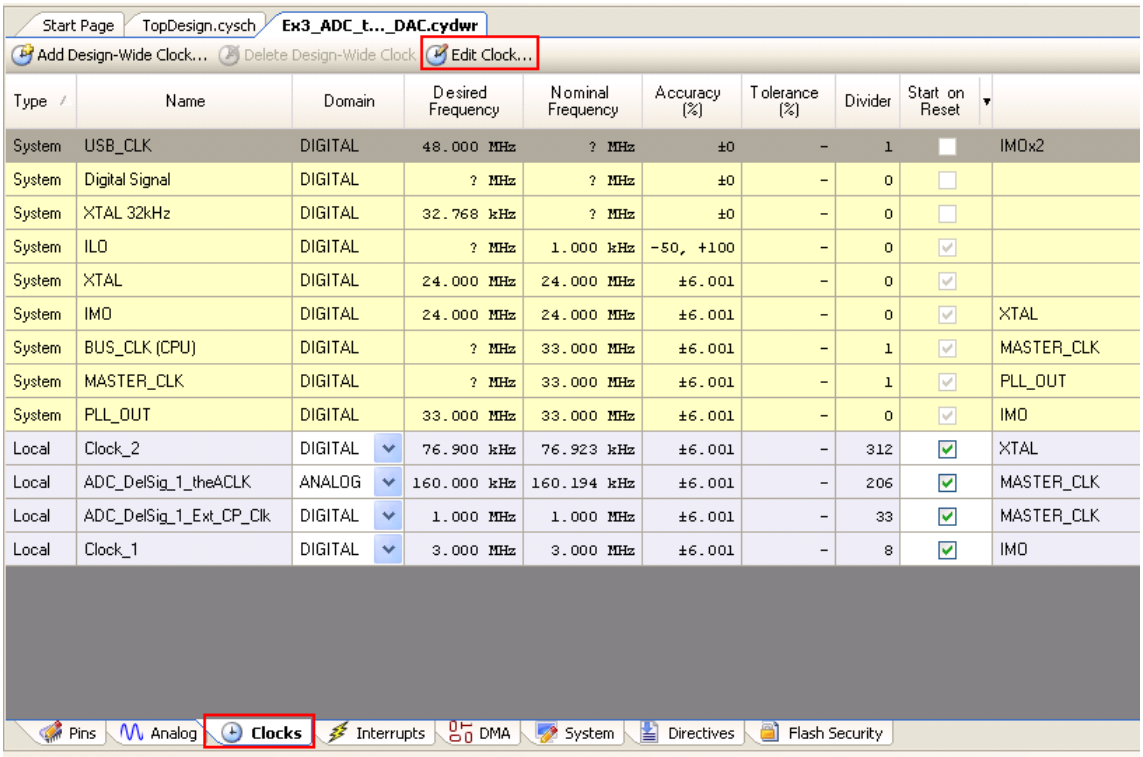
1. Open the Ex3_ADC_to_UART_with_DAC.cydwr file from Workspace Explorer. See figure below

Figure 3-107. Workspace Explorer



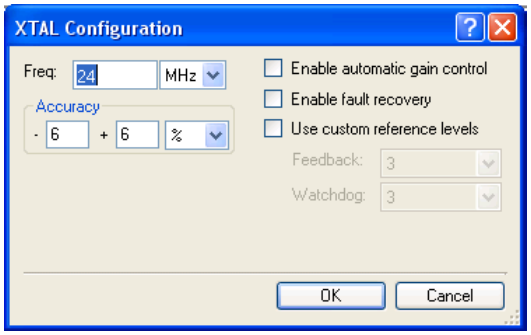
2. Select the **Clocks** tab and click on **Edit Clock…**.

Figure 3-108. Edit Clock



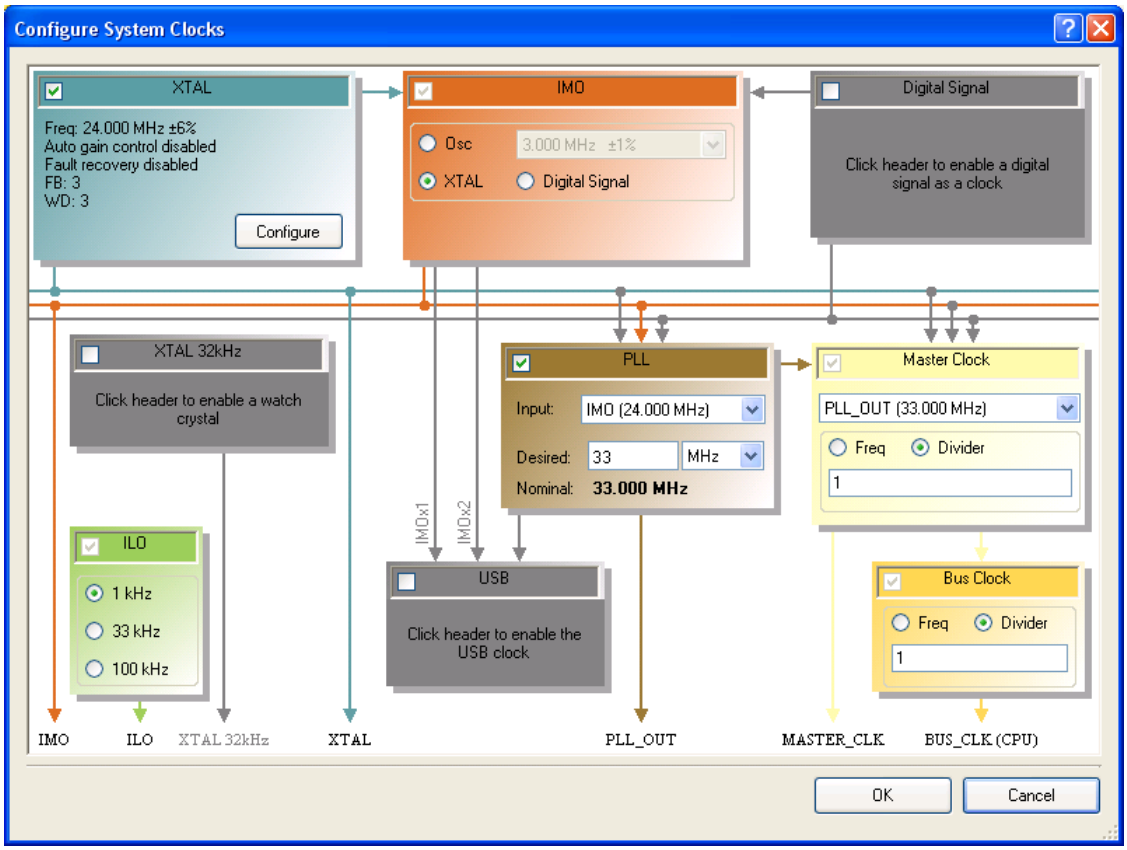
3. In the 'Configure System Clocks' window, enable and configure the XTAL to **24 MHz** with accuracy as **± 6%**.

Figure 3-109. Configure XTAL



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*

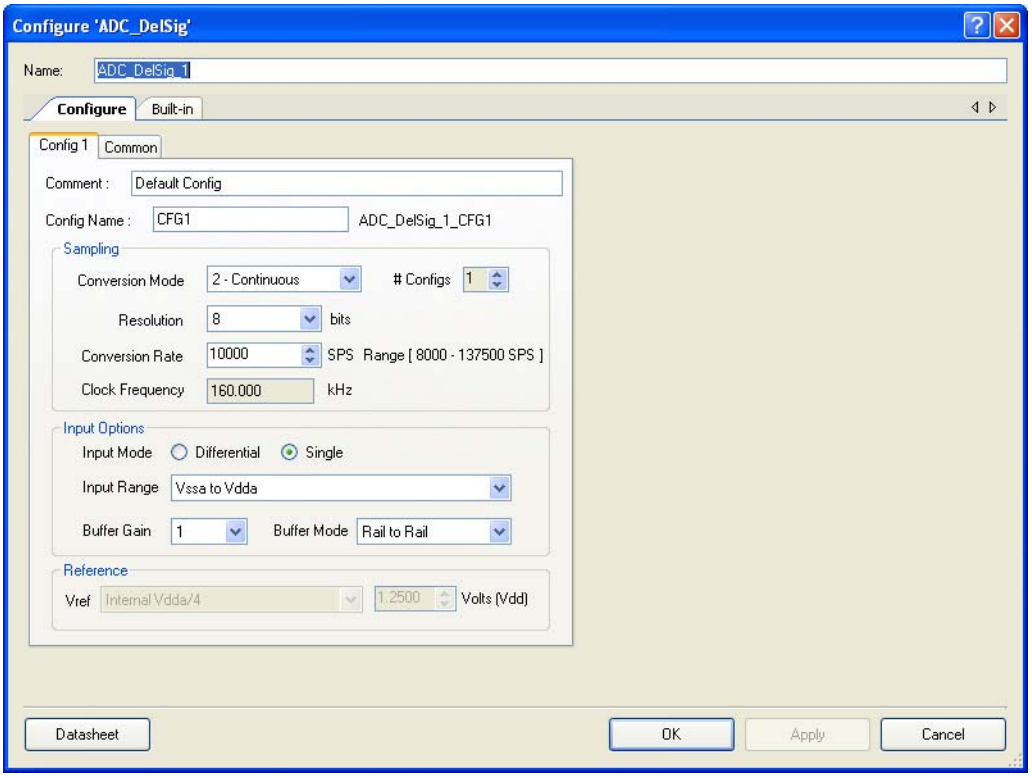
1. Drag and drop the Delta Sigma ADC component (**Component Catalog** → **Analog** → **ADC** → **Delta Sigma ADC**)
2. Double-click the **ADC_DelSig_1** component in the schematic to open the configuration window.
3. Configure the Delta Sigma ADC as follows:

   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

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

Figure 3-111. Delta Sigma ADC Component Configuration



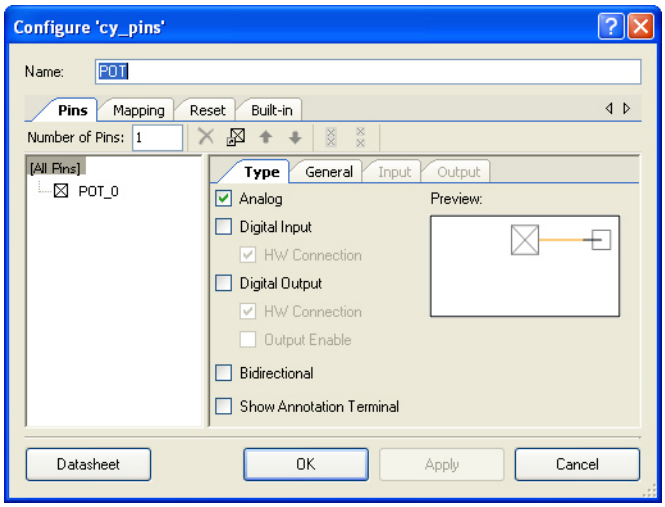
3.3.3.4 *Placing and Configuring an Analog Pin*

1. Drag and drop the Analog Pin component (**Component Catalog** → **Ports and Pins** → **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 3-112. POT Component Configuration



General Tab

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

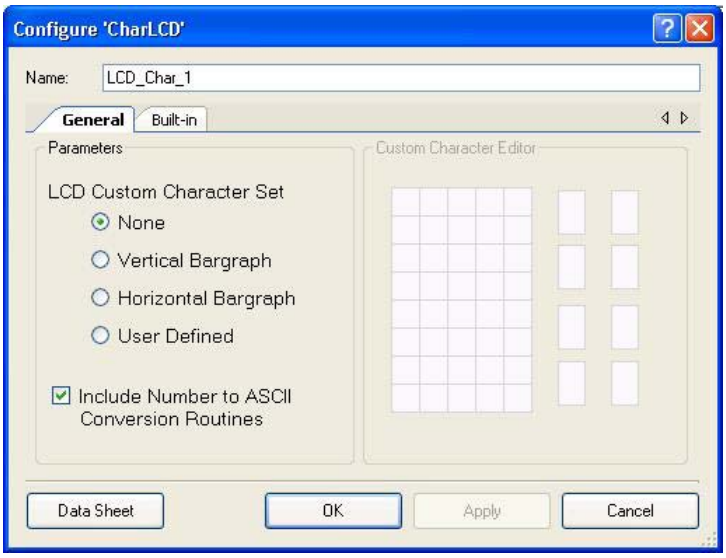
For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

3.3.3.5 *Placing and Configuring Character LCD*

1. Drag and drop the character LCD component (**Component Catalog** → **Display** → **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 3-113. Character LCD Component Configuration



3.3.3.6 *Placing and Configuring Voltage DAC*

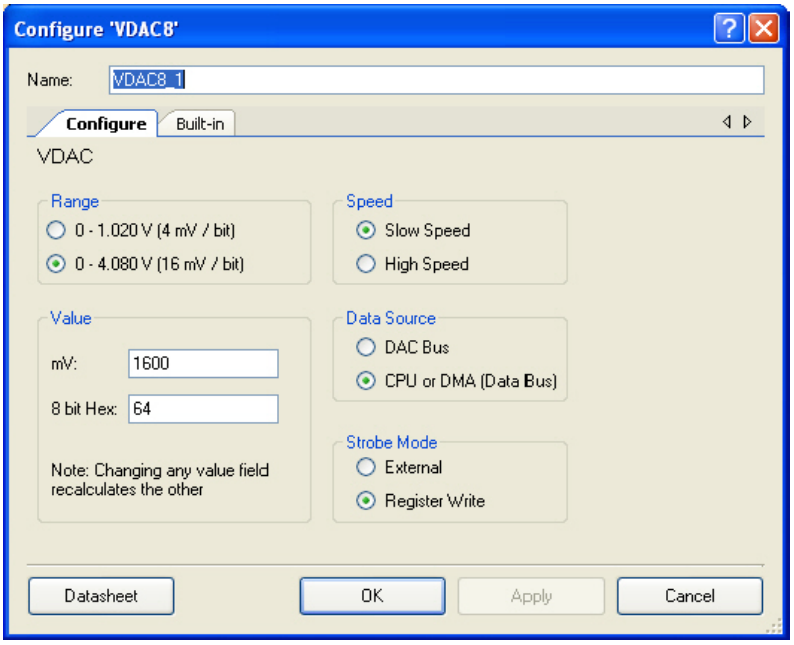
1. Drag and drop the Voltage DAC component (**Component Catalog** → **Analog** → **DAC** → **Voltage DAC**)
2. Double-click the **VDAC8_1** component in the schematic to open the configuration window.
3. Configure the VDAC:

   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

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

Figure 3-114. Voltage DAC Component Configuration



3.3.3.7 *Placing and Configuring Opamp*

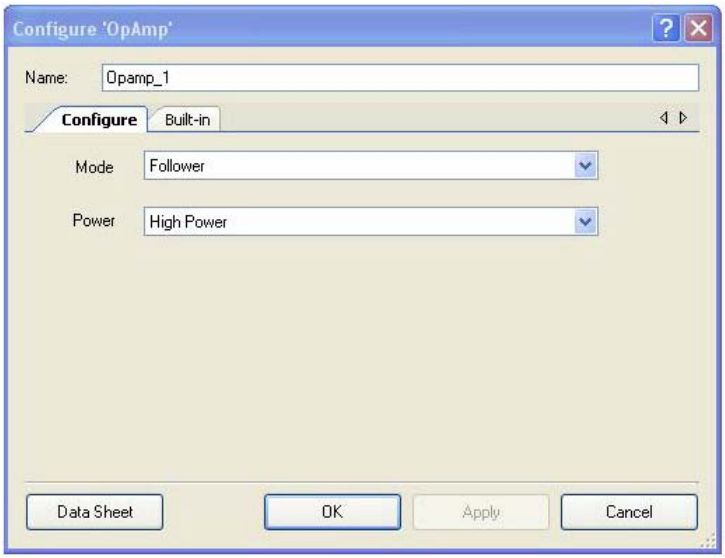
1. Drag and drop the Opamp component (**Component Catalog** → **Analog** → **Amplifiers** → **Opamp**)
2. Double-click the **Opamp_1** component in the schematic to open the configuration window.
3. Configure the Opamp:

   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 3-115. Opamp Component Configuration



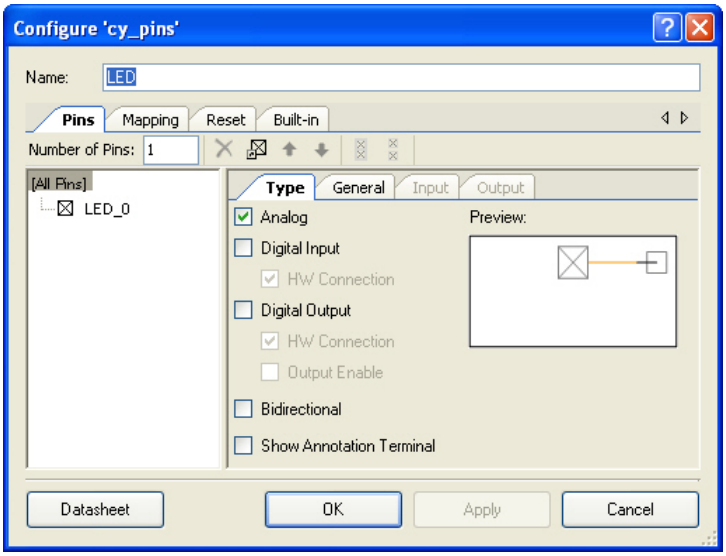
3.3.3.8 *Placing and Configuring Analog Pin*

1. Drag and drop the analog pin component (**Component Catalog → Ports and Pins → 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

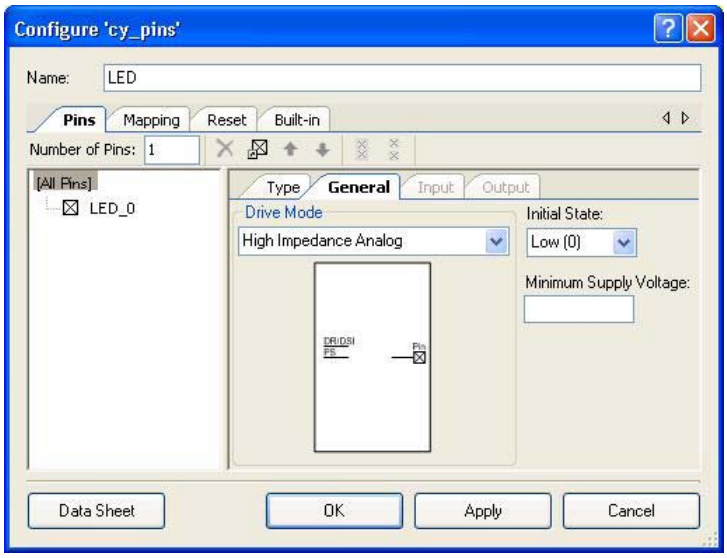


General Tab

- **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



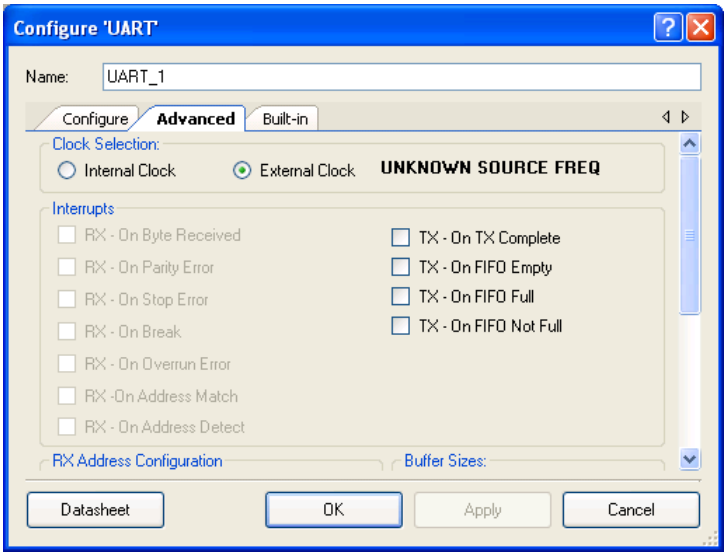
3.3.3.9 *Placing and Configuring UART*

1. Drag and drop the UART component (**Component Catalog** → **Communications** → **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



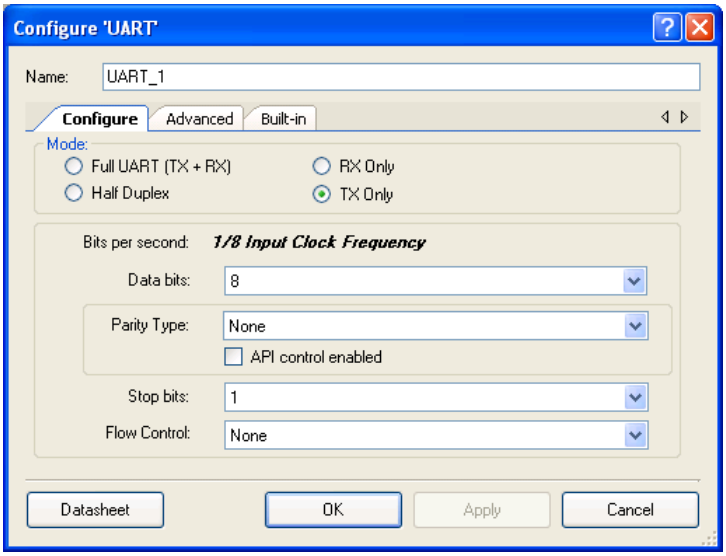
Configure Tab

- **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 3-119. UART Component Configuration



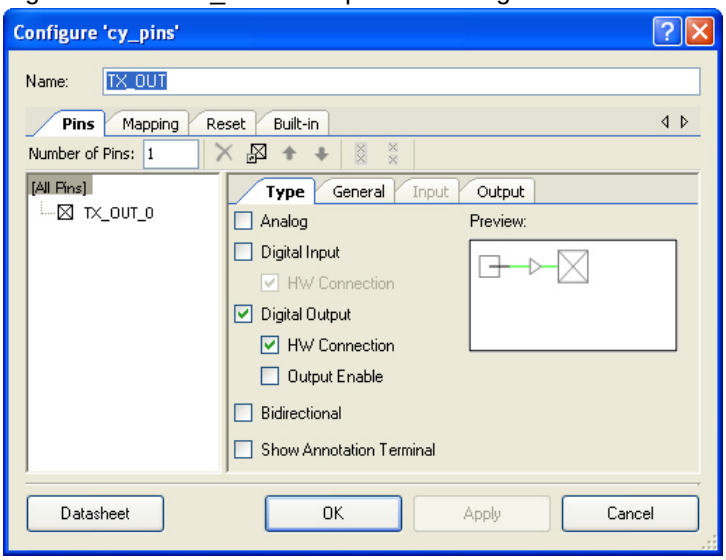
3.3.3.10 *Placing and Configuring Digital Output Pin*

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

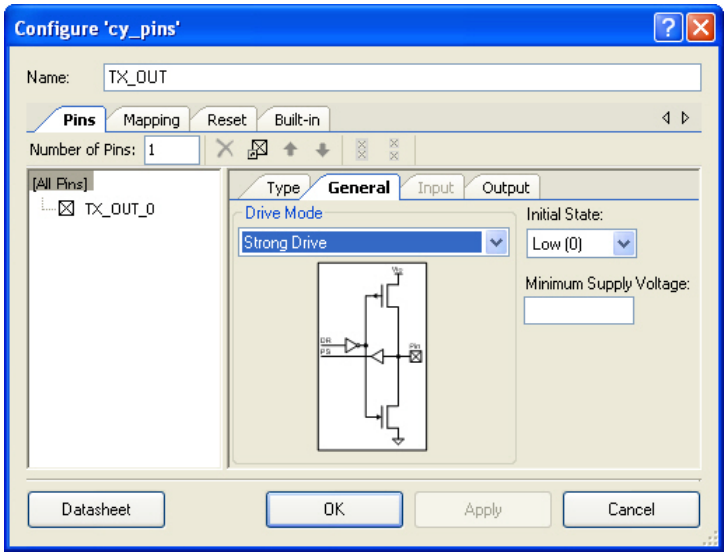
Figure 3-120. TX_OUT Component Configuration



General Tab

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

Figure 3-121. Pins - TX_OUT Component Configuration



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** → **System** → **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

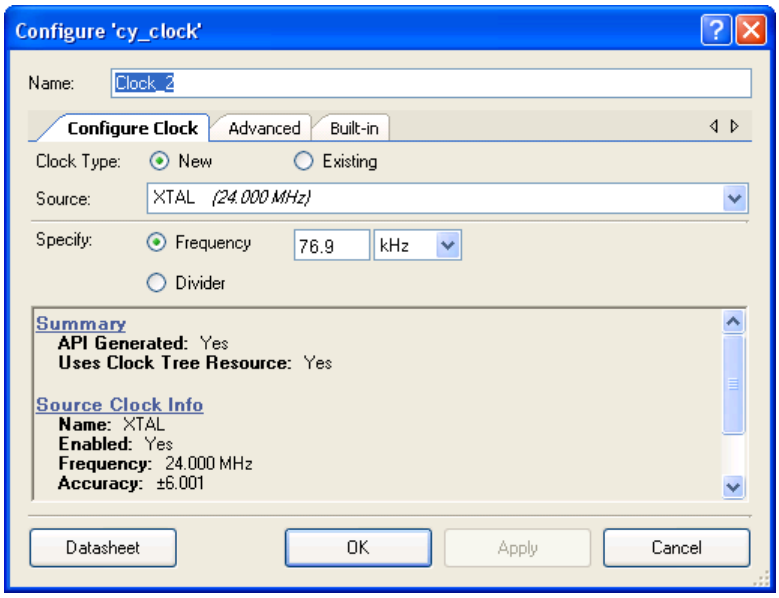


Figure 3-123. UART Configure Window After Assigning Clock Source - Configure Tab

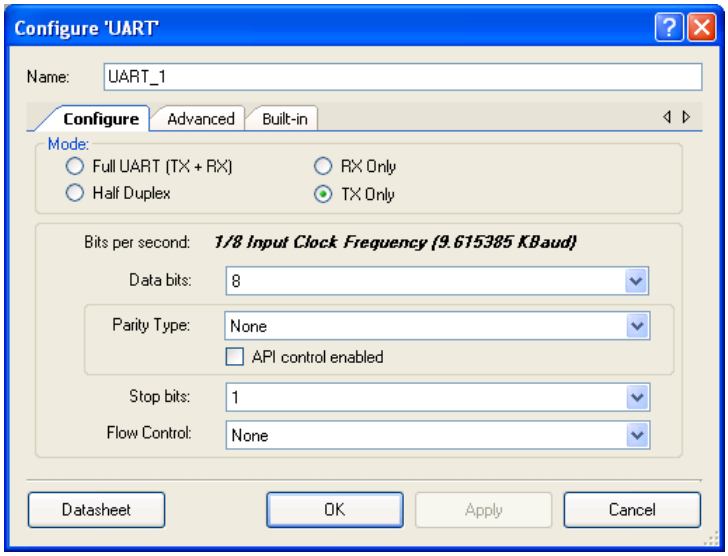
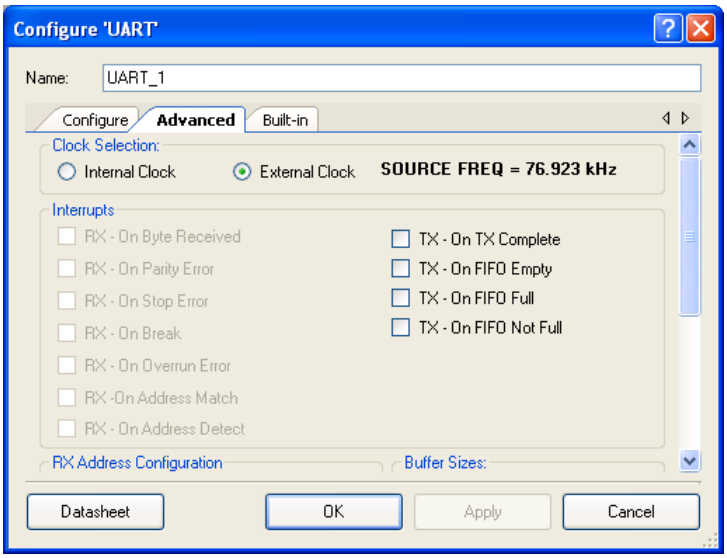


Figure 3-124. UART Configure Window After Assigning Clock Source - Advanced Tab



3.3.3.12 *Placing and Configuring DMA*

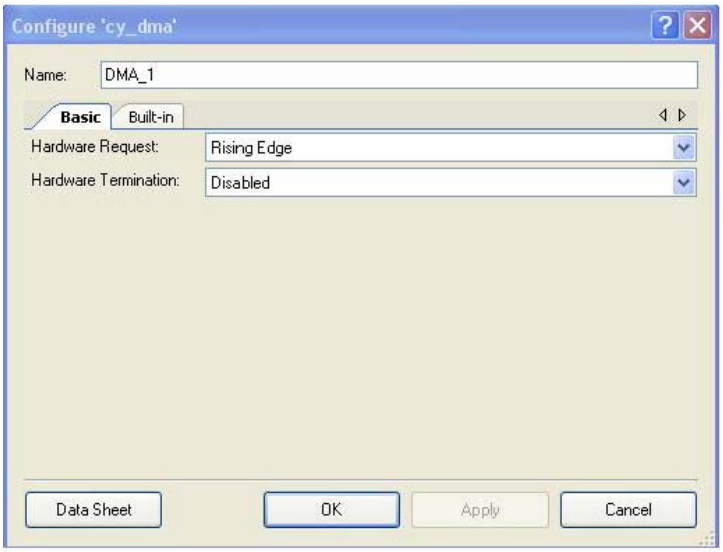
1. Drag and drop the DMA component (**Component Catalog** → **System** → **DMA**)
2. Double-click the **DMA_1** component in the schematic to open the configuration window.
3. Configure the DMA:

   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

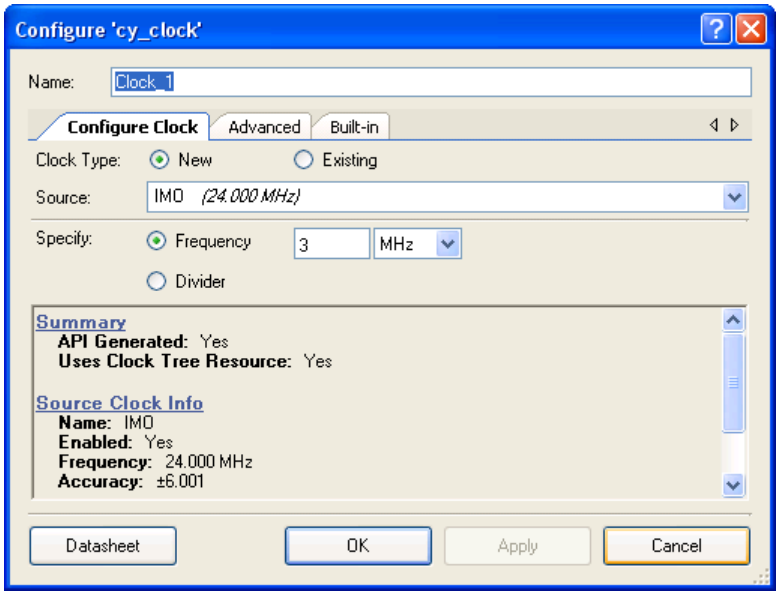


3.3.3.13 *Connecting the Components Together*

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

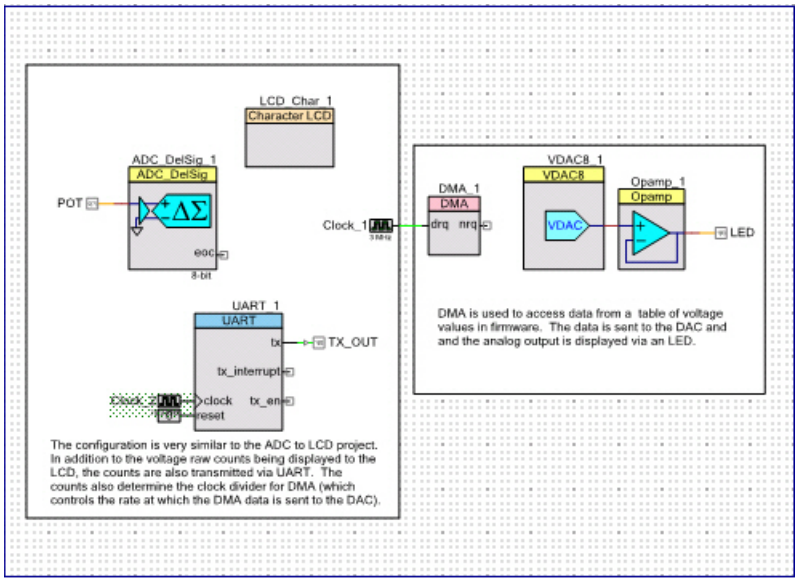
   Configure Clock Tab
   - **Name**: Clock_1
   - **Source**: IMO (24.000 MHz)
   - **Desired Frequency**: 3 MHz
   - Leave the remaining parameters set to their default values

Figure 3-126. Clock Component Configuration



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, connect **VDAC8 (VDAC8_1)** to **Opamp (Opamp_1)**.
7. Using the Wire Tool, 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.

Figure 3-127. Connected Components



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

Alias	Name	Pin
	LED	P1[6]
	TX_OUT	P1[2]
	\LCD_Char_1:LCDPort\[6:0]	P2[6:0]
	POT	P0[7]

3.3.3.15 *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 *CY8C38_main_Ex3.c* file, which is available within the attachments feature of this PDF document.

Notes

- 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.
- Use the PSoC 3 family processor module file *CY8C38_main_Ex3.c* to replace the *main.c* content for PSoC 5 CY8C55 family processor module.

```
#include <device.h>

/* LCD specific */
```

```
#define ROW_0       0  /* LCD row 0     */
#define COLUMN_0    0  /* LCD column 0  */
#define COLUMN_9    9  /* LCD column 9  */
#define COLUMN_10  10  /* LCD column 10 */
#define COLUMN_11  11  /* LCD column 11 */
/* For clearing Tens and Hundreds place */
#define CLEAR_TENS_HUNDREDS "    "
/* For clearing Hundreds place */
#define CLEAR_HUNDREDS      "   "

/* DMA specific */
#define REQUEST_PER_BURST   1 /* One request per burst */
#define BURST_BYTE_COUNT    1 /* Bursts are one byte each */
/* Upper 16-bits of the source address are zero */
#define SOURCE_ADDRESS      0
/* Upper 16-bits of the destination address are zero */
#define DESTINATION_ADDRESS 0

void UpdateDisplay(uint16 * voltageRawCount);
void TxHex (uint16 voltageRawCount);

/* Table of voltage values for DMA to send to the DAC. These values range
 * between 0x3D and 0x9F because these are the two points where the LED
 * is not visible and where the LED is saturated */
const uint8 voltageWave[] =
{
0x6D, 0x6F, 0x71, 0x73, 0x75, 0x77, 0x79, 0x7B, 0x7D, 0x7F, 0x81, 0x83, 0x85,
0x87, 0x89, 0x8B, 0x8D, 0x8F, 0x91, 0x93, 0x95, 0x97, 0x99, 0x9B, 0x9C, 0x9D, 0x9D,
0x9E,
0x9E, 0x9F, 0x9F, 0x9F, 0x9E, 0x9E, 0x9E, 0x9C, 0x9C, 0x9B, 0x99, 0x97, 0x95, 0x93,
0x91,
0x8F, 0x8D, 0x8B, 0x89, 0x87, 0x85, 0x83, 0x81, 0x7F, 0x7D, 0x7B, 0x79, 0x77,
0x75, 0x73, 0x71, 0x6F, 0x6D, 0x6B, 0x69, 0x67, 0x65, 0x63, 0x61, 0x5F, 0x5D,
0x5B, 0x59, 0x57, 0x55, 0x53, 0x51, 0x4F, 0x4D, 0x4B, 0x49, 0x47, 0x45, 0x43,
0x41, 0x40, 0x40, 0x3F, 0x3F, 0x3D, 0x3D, 0x3D, 0x3D, 0x3D, 0x3D, 0x3F, 0x41,
0x43, 0x45, 0x47, 0x49, 0x4B, 0x4D, 0x4F, 0x51,
0x53, 0x55, 0x57, 0x59, 0x5B, 0x5D, 0x5F, 0x61, 0x63, 0x65, 0x67, 0x69, 0x6B
};

/*******************************************************************************
* Function Name: main
********************************************************************************
*
* Summary:
*   The main function initializes the ADC, LCD, VDAC, Analog Buffer, and UART.
*   It also initializes DMA by allocating/configuring a DMA channel and
*   Transaction Descriptor and also copies the voltage table address to the DAC
*   address. In the main loop, it starts and waits for an ADC conversion, then
*   it displays the ADC raw count to the LCD, transmits the raw count serially,
*   and sets the DMA clock divider proportional to the raw count.
*
* Parameters:
*  void
*
* Return:
*  void
*
*******************************************************************************/
```

```
void main()
{
    uint16 voltageRawCount;
    uint8  myChannel;
    uint8  myTd;

    ADC_DelSig_1_Start(); /* Configure and power up ADC */
    LCD_Char_1_Start();   /* Initialize and clear the LCD */
    VDAC8_1_Start();      /* Initializes VDAC8 with default values */
    Opamp_1_Start();      /* Enables Opamp and sets power level */
    UART_1_Start();       /* Enable UART */
    CyDmacConfigure();    /* Set DMA configuration register */

    /* Allocate and initialize a DMA channel to be used by the caller
     * PSoC 3 */
    #if(CYDEV_CHIP_DIE_EXPECT == CYDEV_CHIP_DIE_LEOPARD)
        myChannel = DMA_1_DmaInitialize(BURST_BYTE_COUNT,
                                        REQUEST_PER_BURST,
                                        SOURCE_ADDRESS,
                                        DESTINATION_ADDRESS);
    #endif

    /* PSoC 5 */
    #if(CYDEV_CHIP_DIE_EXPECT == CYDEV_CHIP_DIE_PANTHER)
        myChannel = DMA_1_DmaInitialize(BURST_BYTE_COUNT,
                                        REQUEST_PER_BURST,
                                        (uint16)((uint32)voltageWave >> 16),
                                        (uint16)(VDAC8_1_viDAC8__D >> 16));
    #endif

    /* Allocate a Transaction Descriptor (TD) from the free list */
    myTd = CyDmaTdAllocate();

    /* Move the LCD cursor to Row 0, Column 0 */
    LCD_Char_1_Position(ROW_0, COLUMN_0);

    /* Print Label for the pot voltage raw count */
    LCD_Char_1_PrintString("V Count: ");

    CyDmaTdSetConfiguration(myTd, sizeof(voltageWave),
                            myTd, TD_INC_SRC_ADR ); /* Configure the TD */

    /* Copy address of voltageWave to address of DAC. Set the lower 16-bits of
     *  the source and destination addresses for this TD
     * PSoC 3 */
    #if(CYDEV_CHIP_DIE_EXPECT == CYDEV_CHIP_DIE_LEOPARD)
        CyDmaTdSetAddress(myTd,
                          (uint16)(voltageWave),
                          (uint16)VDAC8_1_viDAC8__D);
    #endif

    /* PSoC 5 */
    #if(CYDEV_CHIP_DIE_EXPECT == CYDEV_CHIP_DIE_PANTHER)
        CyDmaTdSetAddress(myTd,
                          (uint16)((uint32)voltageWave),
                          (uint16)VDAC8_1_viDAC8__D);
    #endif
```

```
    /* Associate TD with channel */
    CyDmaChSetInitialTd(myChannel, myTd);

    /* Enable DMA channel */
    CyDmaChEnable(myChannel, 1);

    /* Clock will make burst requests to the DMAC */
    Clock_1_Start();

    ADC_DelSig_1_StartConvert(); /* Force ADC to initiate a conversion */

    while(1)
    {
        /* Wait for end of conversion */
        ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT);
        voltageRawCount = ADC_DelSig_1_GetResult16(); /* Get converted result */

        /* Set range limit */
        if (voltageRawCount > 0x7FFF)
        {
            voltageRawCount = 0;
        }
        else
        {
            /* Continue on */
        }

        UpdateDisplay(&voltageRawCount); /* Print the result to LCD */

        TxHex(voltageRawCount); /* Transmit result to UART */

        /*
         * The LED blinking frequency is dependent on the Voltage raw count.
         *  With a 3 MHz clock, the lowest divider (for raw count of 0) should be
         *  1000 to blink at a significantly fast pace. The highest value is
         *  about 52,200 (for a raw count of 256) to blink at a significantly
         *  slow pace. The following equation is necessary to make the adjusted
         *  clock frequency (with the updated divider) linear with the ADC
         *  output.
         */
        Clock_1_Stop();

        Clock_1_SetDivider((((uint32)voltageRawCount * 1000) /                \
                (261 - (uint32)voltageRawCount)) + 1000);
        Clock_1_Start();
    }
}

/*******************************************************************************
* Function Name: UpdateDisplay
********************************************************************************
*
* Summary:
*   Print voltage raw count result to the LCD. Clears some characters if
*   necessary. The voltageRawCount parameter is also updated for use in other
*   functions.
*
* Parameters:
```

```
*   voltageRawCount: Voltage raw count from ADC
*
* Return:
*   void
*
*******************************************************************************/
void UpdateDisplay (uint16 * voltageRawCount)
{
    /* Move the cursor to Row 0, Column 9 */
    LCD_Char_1_Position(ROW_0, COLUMN_9);
    LCD_Char_1_PrintNumber(voltageRawCount[0]); /* Print the result */

    if (voltageRawCount[0] < 10)
    {
        /* Move the cursor to Row 0, Column 10 */
        LCD_Char_1_Position(ROW_0,COLUMN_10);
        LCD_Char_1_PrintString(CLEAR_TENS_HUNDREDS); /* Clear last characters */
    }
    else if (voltageRawCount[0] < 100)
    {
        /* Move the cursor to Row 0, Column 11 */
        LCD_Char_1_Position(ROW_0,COLUMN_11);
        LCD_Char_1_PrintString(CLEAR_HUNDREDS); /* Clear last characters */
    }
    else
    {
        /* Continue on */
    }
}

/*******************************************************************************
* Function Name: TxHex
********************************************************************************
*
* Summary:
*   Convert voltage raw count to hex value and TX via UART.
*
* Parameters:
*   voltageRawCount: The voltage raw counts being received from the ADC
*
* Return:
*   void
*
*******************************************************************************/
void TxHex (uint16 voltageRawCount)
{
    static char8 const hex[16] = "0123456789ABCDEF";

    /* TX converted MSnibble */
    UART_1_PutChar(hex[(voltageRawCount>>12)&0xF]);
    /* TX converted second nibble */
    UART_1_PutChar(hex[(voltageRawCount>>8)&0xF]);
    /* TX converted third nibble */
    UART_1_PutChar(hex[(voltageRawCount>>4)&0xF]);
    /* TX converted LSnibble */
    UART_1_PutChar(hex[voltageRawCount&0xF]);
    UART_1_PutString("h\r");  /* h for hexadecimal and carriage return */
}
```

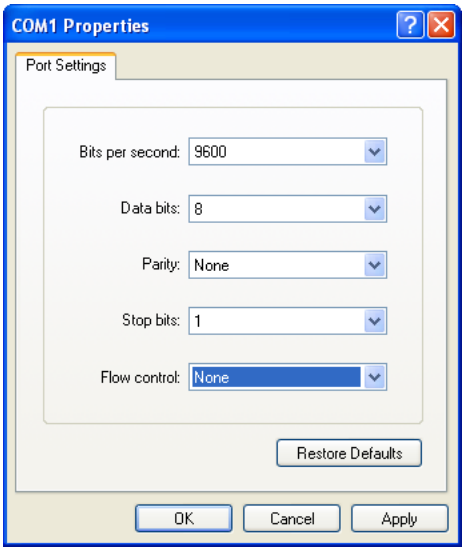
```
/* [] END OF FILE */
```

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.
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 3-129. HyperTerminal Settings



9. Use PSoC Creator as described in Programming My First PSoC 3 Project on page 27 or Programming My First PSoC 5 Project on page 31 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

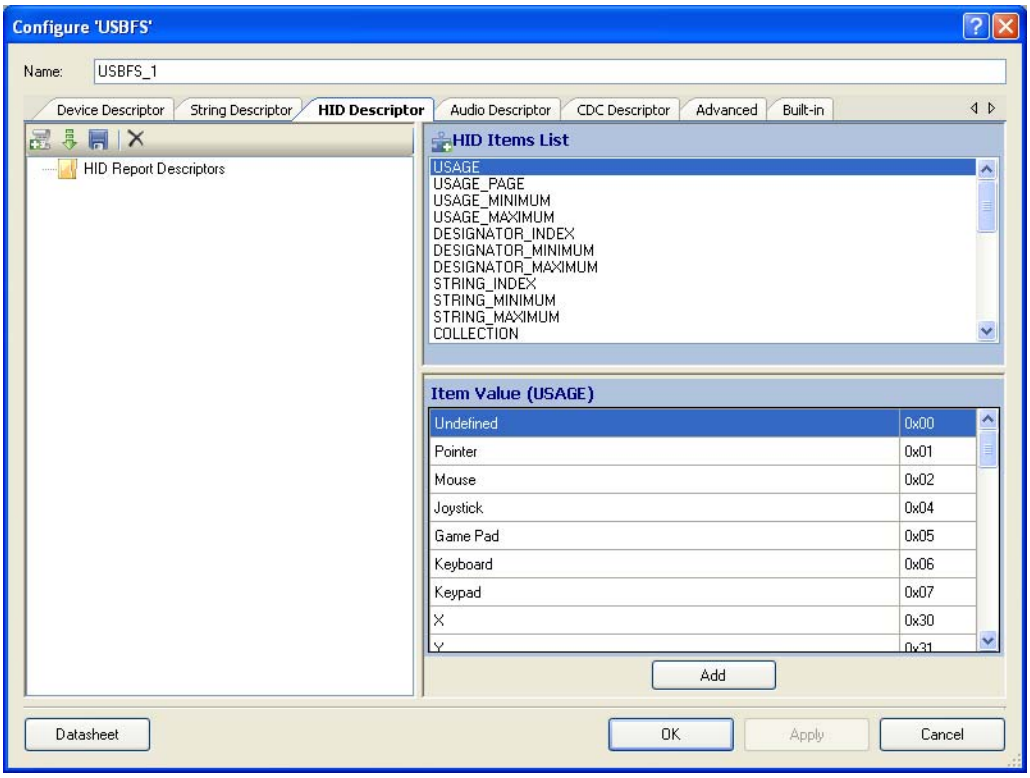
1. Open PSoC Creator.
2. Create a new project by clicking on **Create New Project…** in the **Start Page** of PSoC Creator.
3. In the **New Project** window, select the **Empty PSoC3 Design** template for a PSoC 3 design, or **Empty PSoC5 Design** template for a PSoC 5 design and name the project **Ex4_USB_HID**.
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** → **Communication** → **USBFS** to the workspace.
2. Double-click the **USBFS_1** component.
3. Select the **HID Descriptor** tab.

Figure 3-130. USBFS Component Configuration



4. Click  to import a report.

Figure 3-131. USB Template

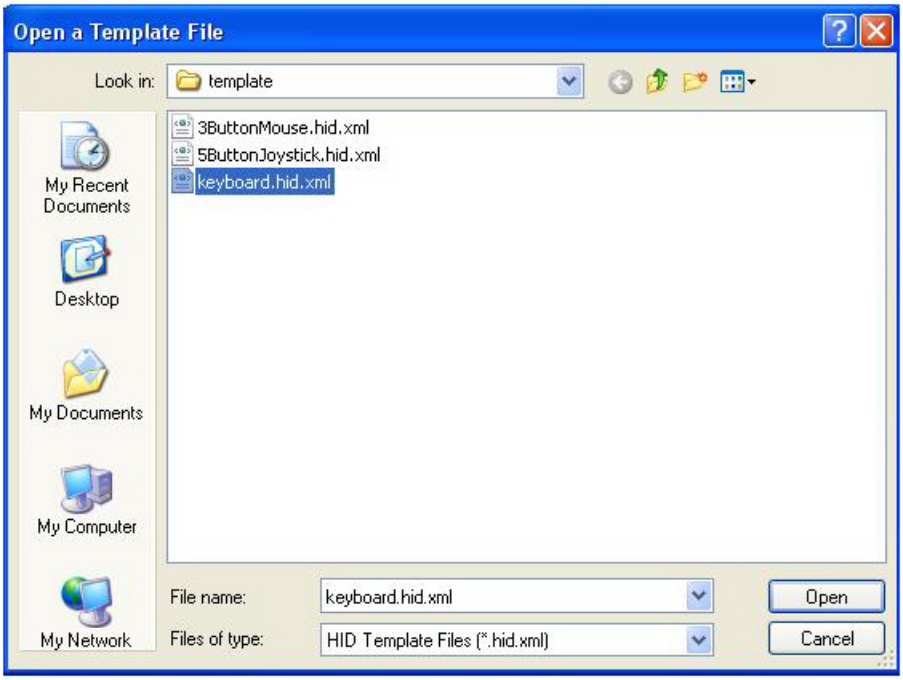
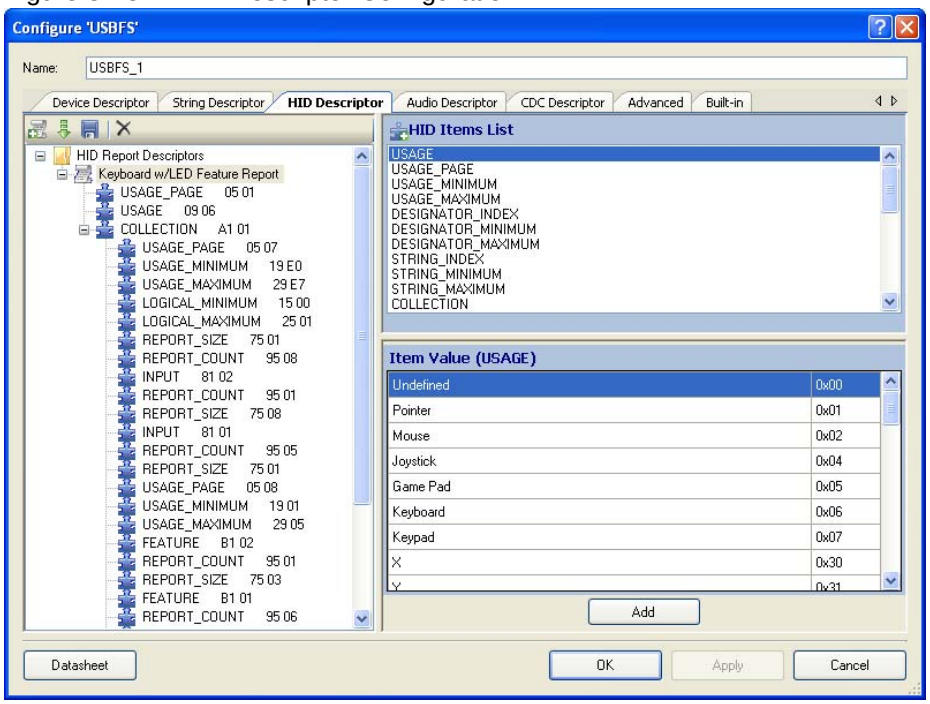
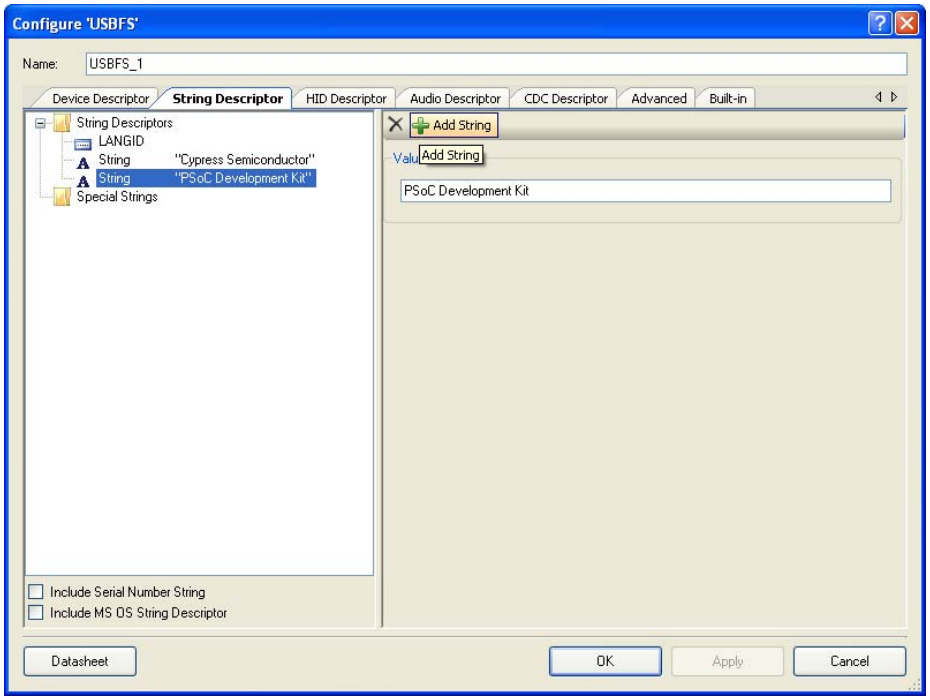


Figure 3-132. HID Descriptor Configuration



5. Select the **String Descriptor** tab.

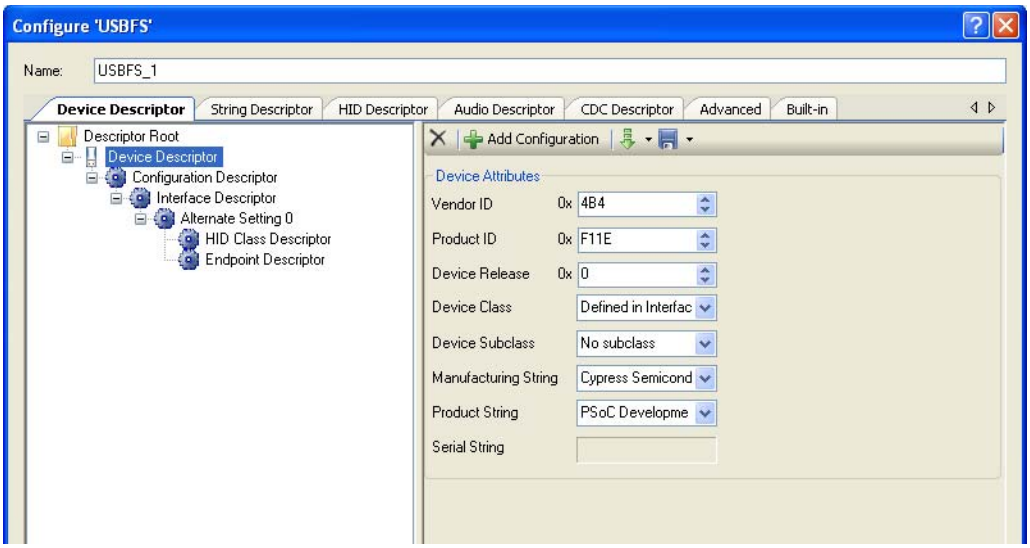
Figure 3-133. String Descriptor Configuration



6. Select **String Descriptors** in the left window.
7. Click **Add String**.

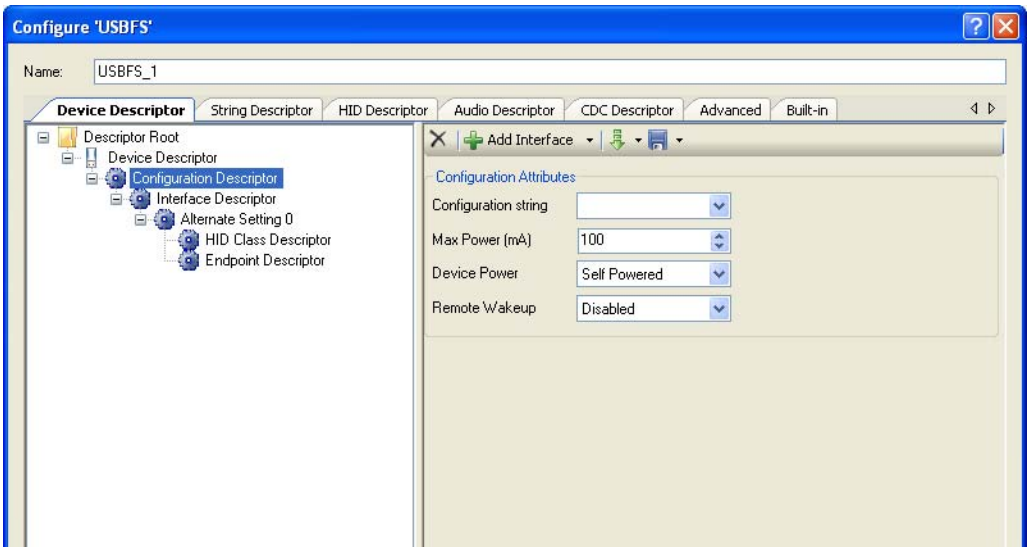
8. Click **Add String** a second time to add a total of two strings.
9. Click the **String** that shows up at the top in the left window.
10. Type **Cypress Semiconductor** in the **Value** field.
11. Click the **String** that shows up at the bottom in the left window.
12. Type **PSoC Development Kit** 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**.

Figure 3-134. Device Descriptor Configuration



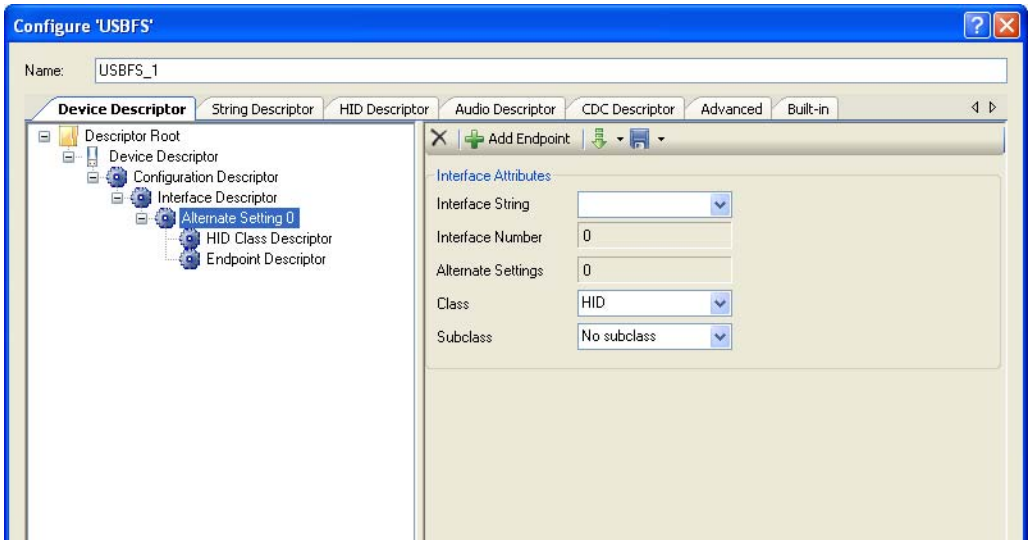
18. Select **Configuration Descriptor**.
19. Set **Device Power** to **Self Powered**.
20. Set **Max Power** to **100 mA**.

Figure 3-135. Configuration Descriptor Configuration



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

Figure 3-136. Interface Descriptor Configuration



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**.

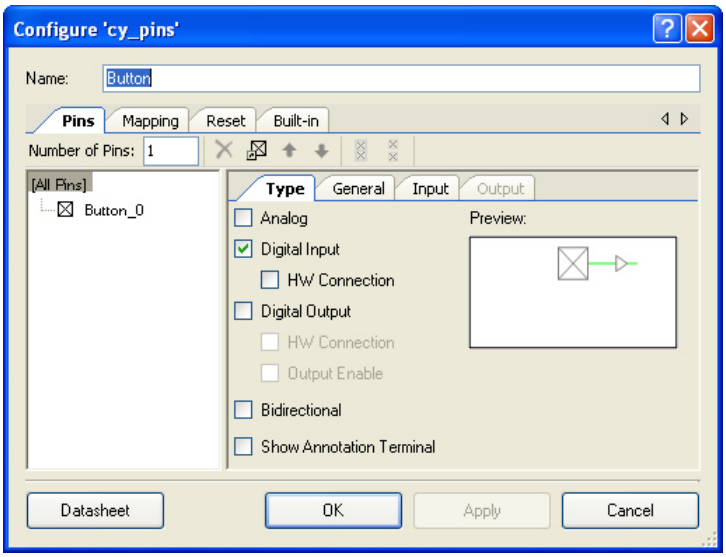
3.3.4.3 Placing and Configuring Software Digital Input Pin

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

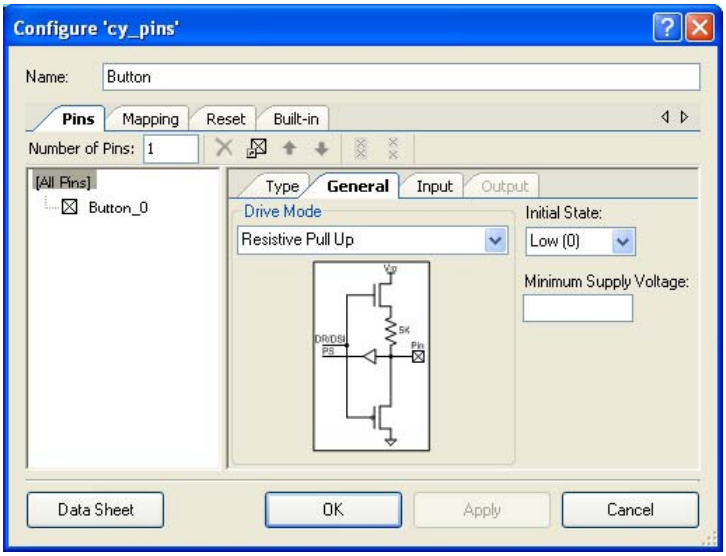
Figure 3-137. SW Digital Input Pin Configuration



General Tab

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

Figure 3-138. Pins - SW Digital Input Pin Configuration



3. Click **OK**.

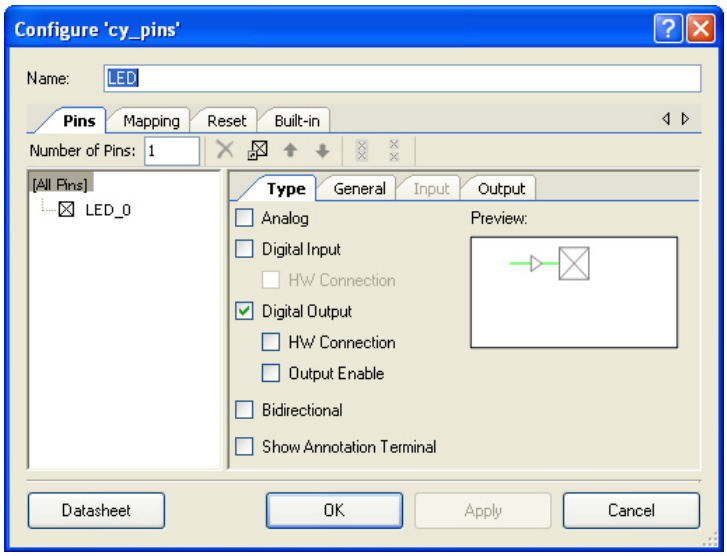
3.3.4.4 *Placing and Configuring LED*

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

   Type Tab

- **Name**: LED

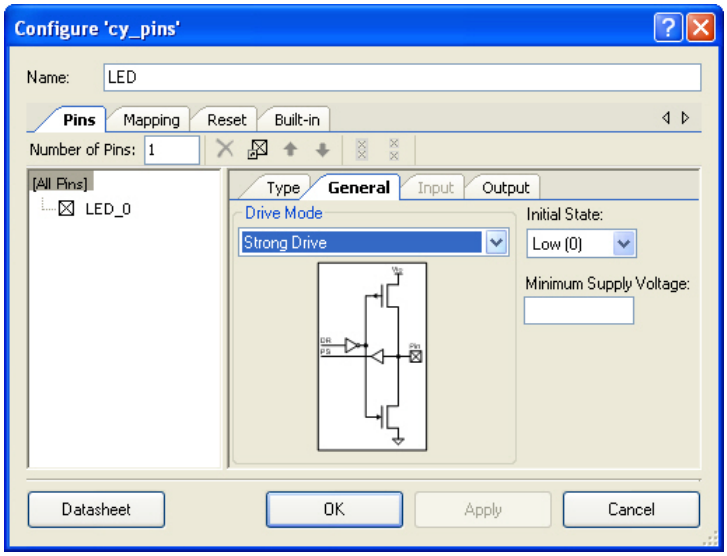
Figure 3-139. LED Component Configuration



General Tab

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

Figure 3-140. Pins - LED Component Configuration

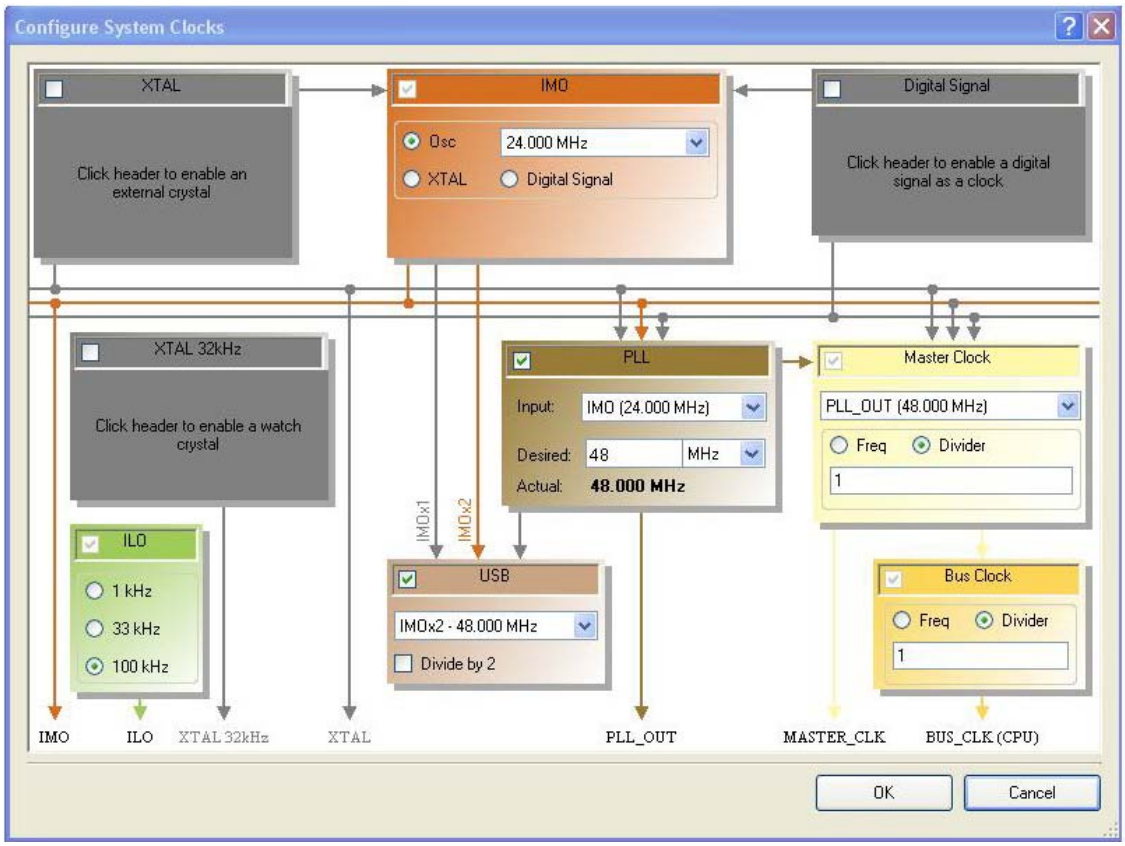


3.3.4.5 *Configuring Clocks for CY8C38 Family Processor Module*

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 CY8C55 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. 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

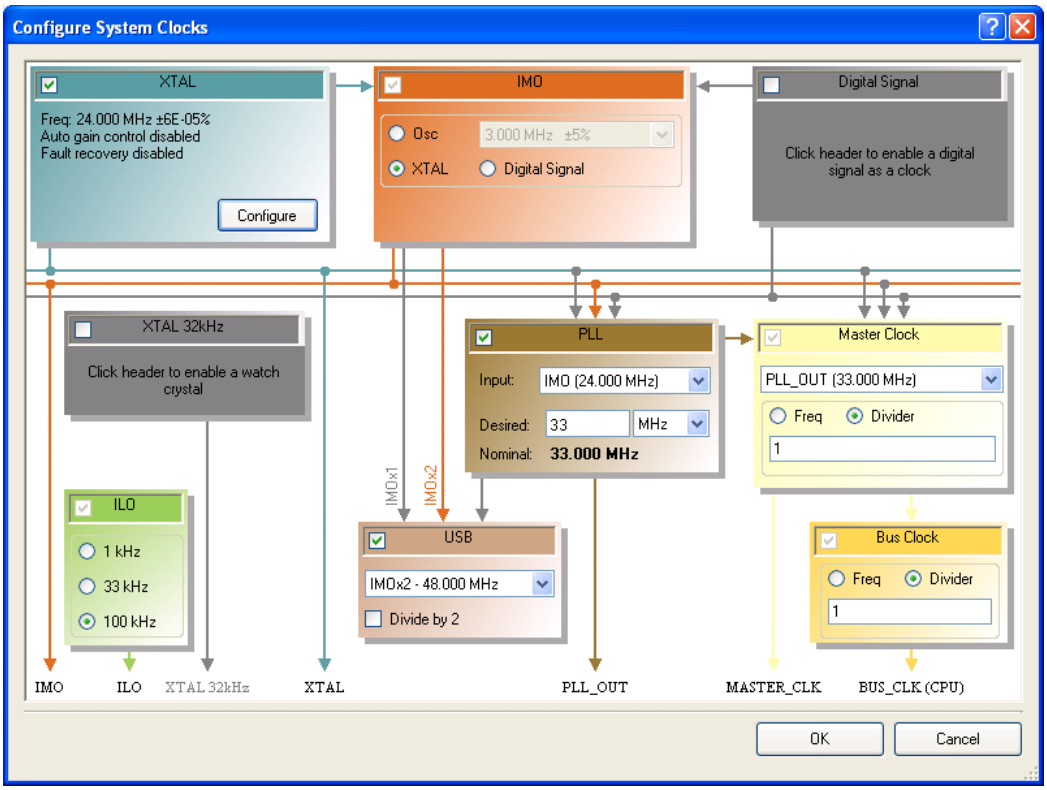
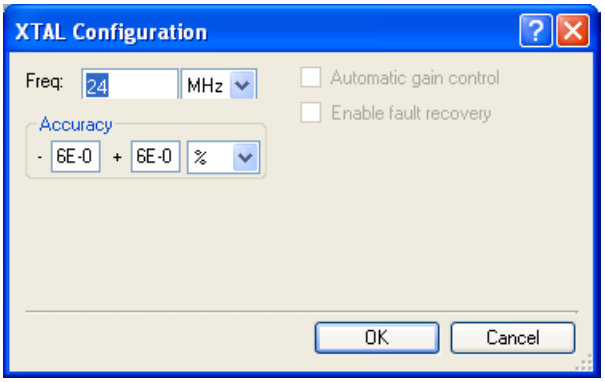


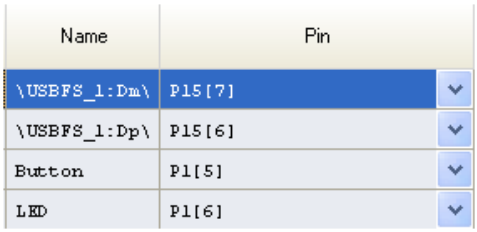
Figure 3-143. XTAL Configuration



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



3.3.4.8 *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_Ex4.c* file, which is available within the attachments feature of this PDF document.

Notes

- 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.
- Use the PSoC 3 family processor module file *CY8C38_main_Ex4.c* to replace the *main.c* content for PSoC 5 CY8C55 family processor module.

```
#include "device.h"

/* Keyboard scan codes */
#define WINDOWS_LEFT_MODIFIER 0x80u /* This is the left windows key */
#define LETTER_R              0x15u
#define CARRIAGE_RETURN       0x28u
#define KEY_RELEASE           0x00u

/* For button setup */
#define SET_BUTTON      0x01u

/* USB related */
#define KEYBOARD_ENDPOINT       0x01u
#define KEYBOARD_DEVICE         0x00u
#define KEYBOARD_DATA_SIZE      0x08u
#define KEY_DATA_INDEX          0x02u
#define MODIFIER_KEY_DATA_INDEX 0x00u

/* MACRO for button detection */
#define IS_BUTTON_PRESSED (!Button_Read())

#define BUTTON_PRESSED        1
#define BUTTON_RELEASED       0

static uint8 ButtonEventDetected(void);
static void  SendKey(uint8 key);
static void  StartWindowsRun(void); /* Open Windows Run App */
static void  GetAckLoadEp(uint8 * keyboardData);

/*******************************************************************************
* Function Name: main
********************************************************************************
*
* Summary:
```

```
*  The main function starts out enabling global interrupts, setting up the
*  button (SW1), then initializing USB for 5 V operation. Then allows the HID
*  device to enumerate, and loads the keyboard endpoint to allow an ack to be
*  sent before sending the first keyboard data. Then it continuously checks for
*  USB plug-and-play and a button press to see if the keyboard data needs sent.
*
* Parameters:
*  void
*
* Return:
*  void
*
*******************************************************************************/
void main()
{
    uint8 ledState = 0; /* Set initial LED state to off */
    uint8 i;

    /* Data array for the keyboard device endpoint */
    uint8 keyboardData[] = { 0, 0, 0, 0, 0, 0, 0, 0 };

    /* Keyboard scan codes for the cypress website ("www.cypress.com<cr>") */
    uint8 cypressWebsiteCharSequence[] = { 0x1A, 0x1A, 0x1A, 0x37, 0x06, 0x1C,
                                0x13, 0x15, 0x08, 0x16, 0x16, 0x37,
                                0x06, 0x12, 0x10, 0x28 };

    CYGlobalIntEnable; /* Enable global interrupts */

    Button_Write(SET_BUTTON);  /* Set port pin for button (SW1) */

    /* Start USBFS operation using keyboard device (0) and with 5 V operation */
    USBFS_1_Start(KEYBOARD_DEVICE, USBFS_1_5V_OPERATION);

    while(!USBFS_1_bGetConfiguration()); /* Wait for Device to enumerate */

    /* Enumeration is completed load keyboard endpoint to set up ACK for first
    key */
    USBFS_1_LoadInEP(KEYBOARD_ENDPOINT, keyboardData, KEYBOARD_DATA_SIZE);

    while (1)
    {
        if (ButtonEventDetected())
        {
            ledState ^= 0x01u;    /* Set/clear (toggle) the LED port pin */
            LED_Write(ledState);  /* Toggle LED */

            StartWindowsRun(); /* Open Windows Run App */

            /* Send the cypress website key sequence */
            for (i = 0; i < sizeof(cypressWebsiteCharSequence); i++)
            {
                SendKey(cypressWebsiteCharSequence[i]);
            }
        }
        else
        {
            /* Continue on */
        }
```

```
    }
}

/*******************************************************************************
* Function Name: ButtonEventDetected
********************************************************************************
*
* Summary:
*  Check to see if the button is pressed.
*
* Parameters:
*  void
*
* Return:
*  TRUE: button pressed
*  FALSE: button not pressed
*
*******************************************************************************/
static uint8 ButtonEventDetected(void)
{
    if(!IS_BUTTON_PRESSED)
    {
        return BUTTON_RELEASED;
    }
    else
    {
        /* Continue on */
    }

    CyDelay(50); /* Debounce mechanical switch for ~50 msec */

    if(IS_BUTTON_PRESSED)
    {
        /* Wait for the button to be released */
        while(IS_BUTTON_PRESSED);
        return BUTTON_PRESSED;
    }
    else
    {
     /* Continue on */
    }

    return BUTTON_RELEASED;
}

/*******************************************************************************
* Function Name: SendKey
********************************************************************************
*
* Summary:
*  Sends keyboard key.
*
* Parameters:
*  key: Keyboard scan code.
*
* Return:
*  void
*
```

```
*******************************************************************************/
static void SendKey(uint8 key)
{
    /* Data array for the keyboard device endpoint */
    uint8 keyboardData[] = { 0, 0, 0, 0, 0, 0, 0, 0 };

    keyboardData[KEY_DATA_INDEX] = key; /* Send key-down data, make */
    GetAckLoadEp(keyboardData); /* Send USB keyboard data */

    keyboardData[KEY_DATA_INDEX] = KEY_RELEASE; /* Send key-up data, break */
    GetAckLoadEp(keyboardData); /* Send USB keyboard data */
}

/*******************************************************************************
* Function Name: StartWindowsRun
********************************************************************************
*
* Summary:
*  Sends the Windows Run command by sending Windows Modifier and 'r' (while
*  the Windows modifier key is in a down state) and then releasing both keys.
*
* Parameters:
*  void
*
* Return:
*  void
*
*******************************************************************************/
static void StartWindowsRun(void)
{
    /* Data array for the keyboard device endpoint */
    uint8 keyboardData[] = { 0, 0, 0, 0, 0, 0, 0, 0 };

    /* Send Windows modifier key-down data, modifier make */
    keyboardData[MODIFIER_KEY_DATA_INDEX] = WINDOWS_LEFT_MODIFIER;
    GetAckLoadEp(keyboardData); /* Send USB keyboard data */

    /* While Windows modifier key is down send r key, r make */
    keyboardData[KEY_DATA_INDEX] = LETTER_R;
    GetAckLoadEp(keyboardData); /* Send USB keyboard data */

    /* Send up keys for both Windows modifier key and r key */
    keyboardData[KEY_DATA_INDEX] = KEY_RELEASE; /* r break */
   keyboardData[MODIFIER_KEY_DATA_INDEX] = KEY_RELEASE; /* Windows modifier break
*/
    GetAckLoadEp(keyboardData); /* Send USB keyboard data */
    CyDelay(500); /* Delay about 0.5 seconds to allow Run window to pop-up */
}

/*******************************************************************************
* Function Name: GetAckLoadEp
********************************************************************************
*
* Summary:
*  It first confirms that an Acknowledge transaction occurred on the keyboard
*  endpoint. When the ACK is confirmed, the endpoint is enabled and loaded.
*
* Parameters:
```

```
*  keyboardData: Data array for the keyboard device endpoint
*
* Return:
*  void
*
*******************************************************************************/
static void GetAckLoadEp(uint8 * keyboardData)
{
    /* Wait for ACK before loading data */
    while(!USBFS_1_bGetEPAckState(KEYBOARD_ENDPOINT));
    /* ACK has occurred, load the endpoint */
    USBFS_1_LoadInEP(KEYBOARD_ENDPOINT, keyboardData, KEYBOARD_DATA_SIZE);
}


/* [] END OF FILE */
```

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 180.

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 27 or Programming My First PSoC 5 Project on page 31 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 5 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.

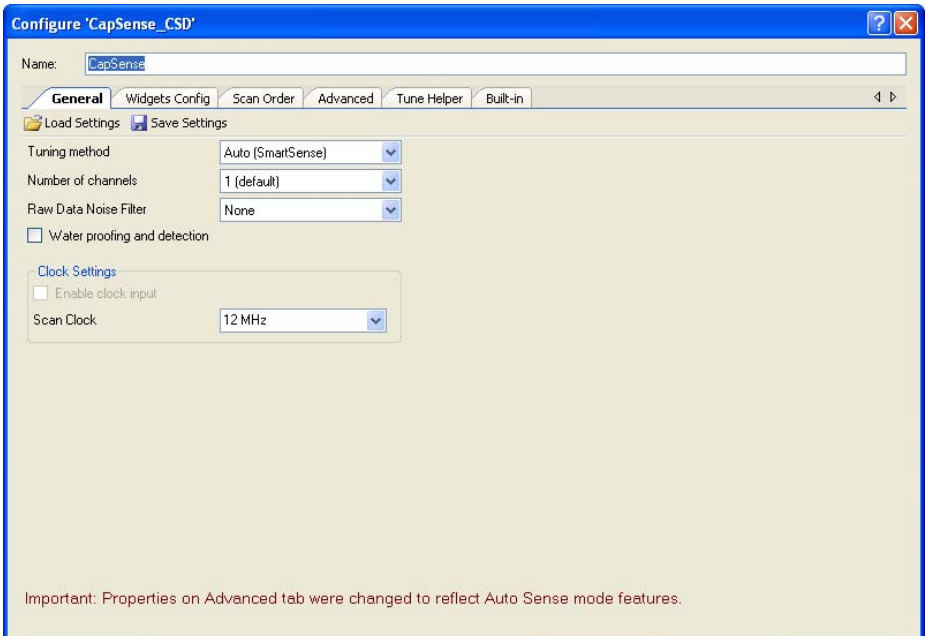
3.3.5.2 Placing and Configuring CapSense

1. Drag and drop a CapSense component from the **Component Catalog** → **CapSense** → **CapSense_CSD** to the workspace.
2. Double-click the **CapSense_1** component
3. Configure CapSense as follows:

   General Tab

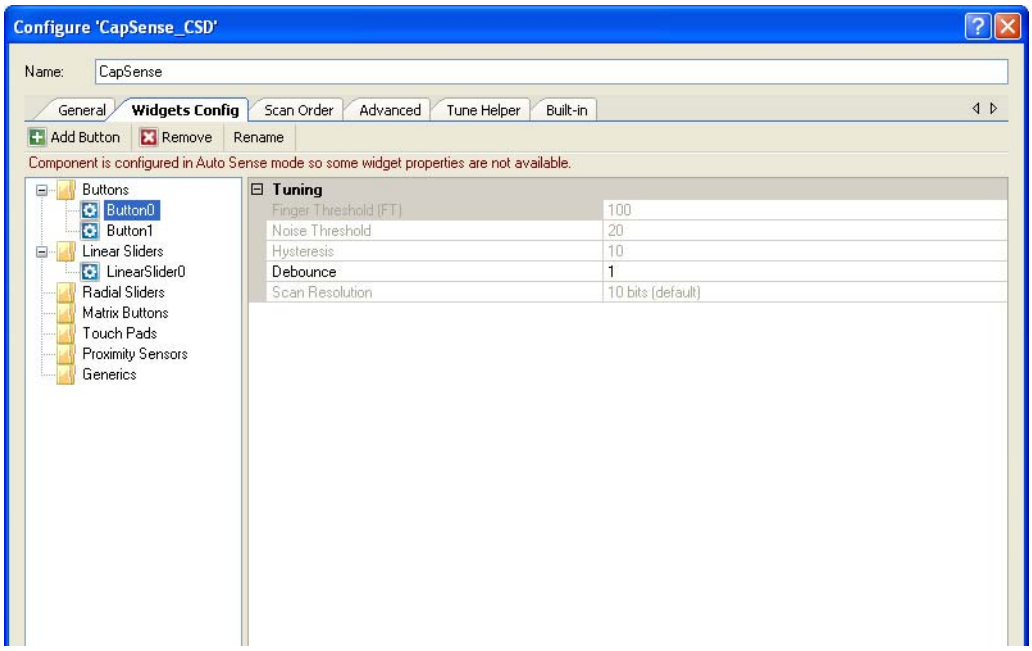
   - Name: **CapSense**
   - Set parameters as shown in the following figure

Figure 3-145. CapSense Component Configuration



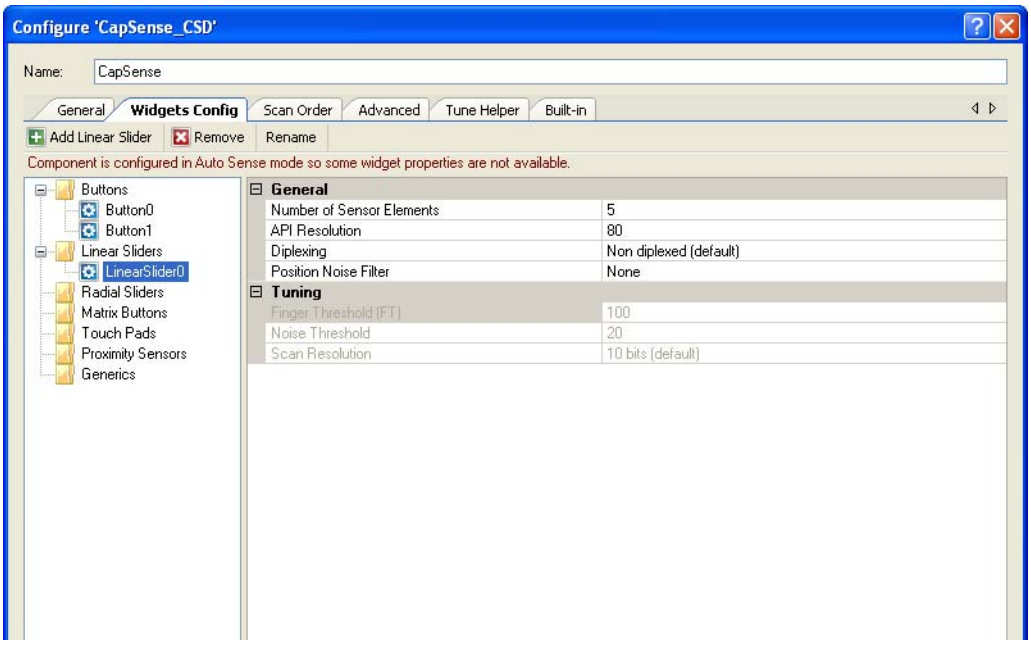
4. Select the **Widget Config** Tab.
5. Add two buttons by clicking on **Add Button**. Leave the button parameters as default.

Figure 3-146. Buttons - CapSense Component Configuration



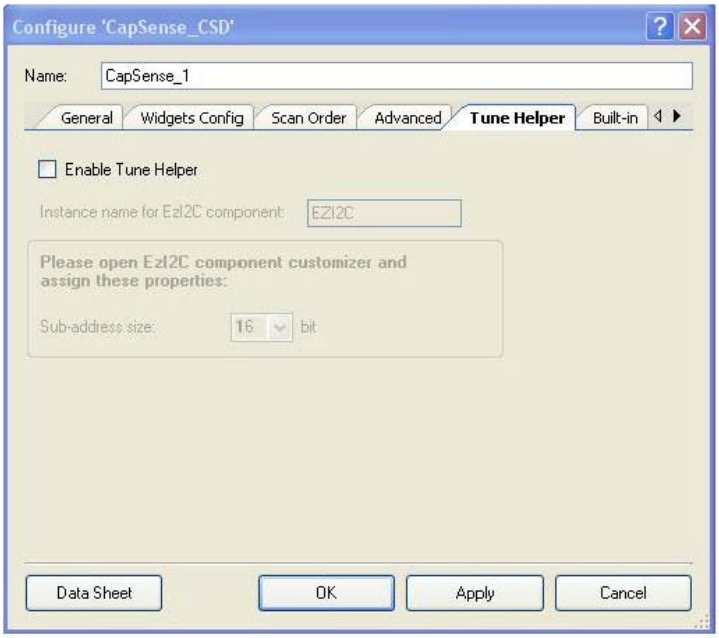
6. Select Linear Slider and click on **Add Linear Slider**.
7. Change API resolution parameter to **80**.

Figure 3-147. Slider Configuration



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

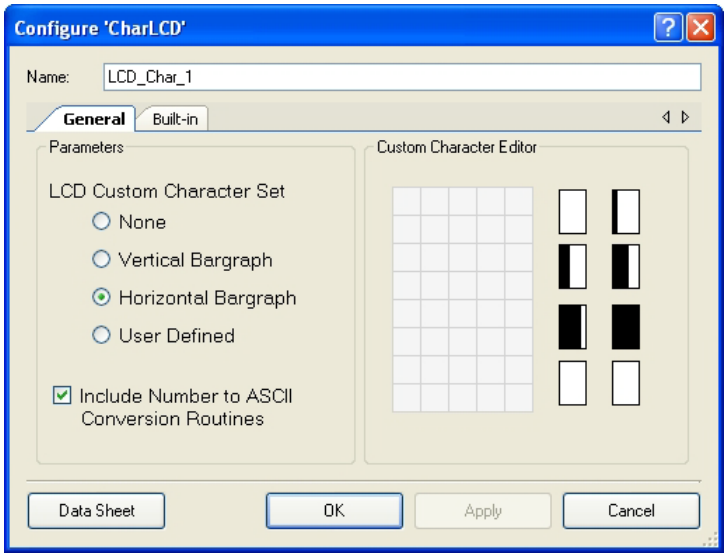
Figure 3-148. Scan Slots Slider Terminals Configuration



3.3.5.3 *Placing and Configuring Character LCD*

1. Drag and drop a character LCD component from the **Component Catalog** → **Display** → **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



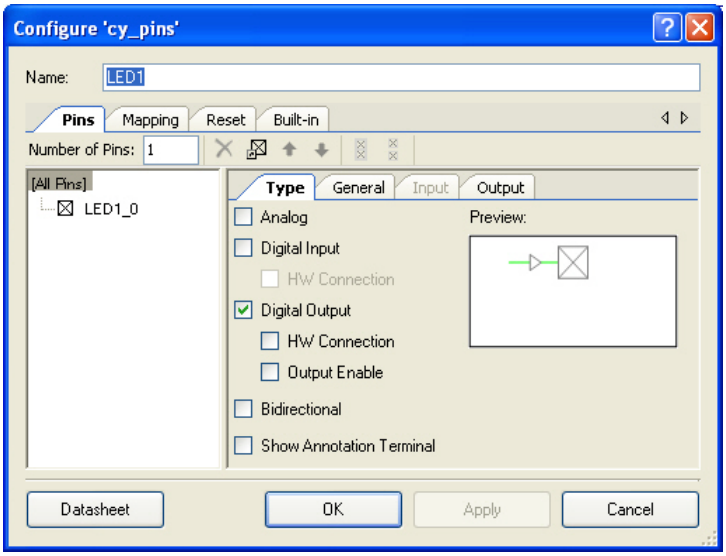
3.3.5.4 *Placing and Configuring Digital Output Pin*

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**

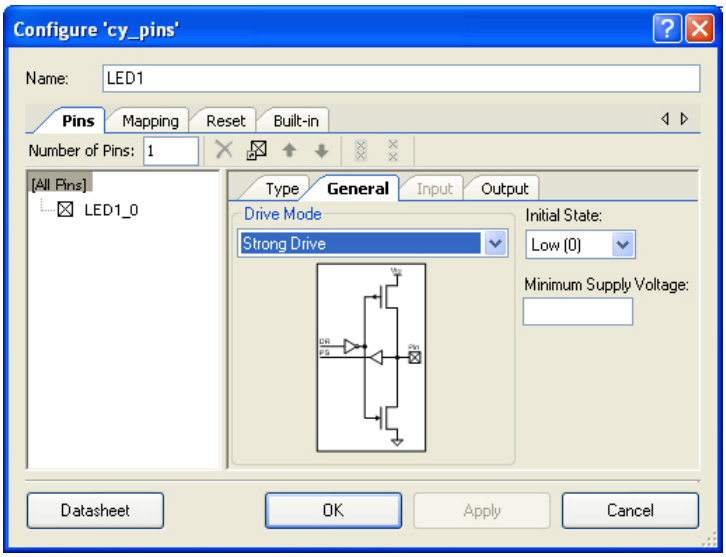
Figure 3-150. LED Configuration



General Tab

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

Figure 3-151. Pins - LED Configuration



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 CY8C55 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

Alias	Name	Pin
sCmod	\CapSense_1:sbCSD:cCmod\	P2[7]
LS_Position_e4	\CapSense_1:sbCSD:cPort\[6]	P0[4]
LS_Position_e3	\CapSense_1:sbCSD:cPort\[5]	P0[3]
LS_Position_e2	\CapSense_1:sbCSD:cPort\[4]	P0[2]
LS_Position_e1	\CapSense_1:sbCSD:cPort\[3]	P0[1]
LS_Position_e0	\CapSense_1:sbCSD:cPort\[2]	P0[0]
BTN_B2	\CapSense_1:sbCSD:cPort\[1]	P0[6]
BTN_B1	\CapSense_1:sbCSD:cPort\[0]	P0[5]
	\LCD_Char_1:LCDPort\[6:0]	P2[6:0]
	LED1	P1[6]
	LED2	P1[7]

Figure 3-153. Pin Assignment for CY8C55 Family Processor Module

Alias	Name	Pin
LS_Position_e4	\CapSense_1:sbCSD:cPort\[6]	P0[4]
LS_Position_e3	\CapSense_1:sbCSD:cPort\[5]	P0[3]
LS_Position_e2	\CapSense_1:sbCSD:cPort\[4]	P0[2]
LS_Position_e1	\CapSense_1:sbCSD:cPort\[3]	P0[1]
LS_Position_e0	\CapSense_1:sbCSD:cPort\[2]	P0[0]
BTN_B2	\CapSense_1:sbCSD:cPort\[1]	P0[6]
BTN_B1	\CapSense_1:sbCSD:cPort\[0]	P0[5]
sCmod	\CapSense_1:sbCSD:cCmod\	P15[5]
	\LCD_Char_1:LCDPort\[6:0]	P2[6:0]
	LED1	P1[6]
	LED2	P1[7]

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* file, which is available within the attachments feature of this PDF document.

Notes

- 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.
- Use the PSoC 3 family processor module file *CY8C38_main_Ex5.c* to replace the *main.c* content for PSoC 5 CY8C55 family processor module.

```
#include <device.h>

/* LCD specific */
#define ROW_0     0  /* LCD row 0    */
#define ROW_1     1  /* LCD 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"

/* LED specific */
#define LED_ON  1 /* For setting LED pin high */
#define LED_OFF 0 /* For setting LED pin low */

/* CapSense specific */
#define SLIDER_RESOLUTION 80

//extern const uint8 LCD_Char_1_customFonts[];

void UpdateButtonState(uint8 slot_1, uint8 slot_2);
void UpdateSliderPosition(uint8 value);

/*******************************************************************************
* Function Name: main
********************************************************************************
*
* Summary:
*  The main function initializes CapSense and the LCD. Then it continuously
*  scans all CapSense slots (slider slots and buttons), gets the state of the
*  buttons and slider and updates the LCD with the current state.
*
* Parameters:
*  void
*
```

```
* Return:
*  void
*
*******************************************************************************/
void main()
{
    uint8 pos;      /* Slider Position */
    uint8 stateB_1; /* Button1 State */
    uint8 stateB_2; /* Button2 State */

    CYGlobalIntEnable; /* Enable global interrupts */

    /* LCD Initialization */
    LCD_Char_1_Start();
    /* For Bargraph display on LCD */
    //    LCD_Char_1_LoadCustomFonts(LCD_Char_1_customFonts);

    /* Start capsense and initialize baselines and enable scan */
    CapSense_Start();
    CapSense_InitializeAllBaselines();
    CapSense_ScanEnabledWidgets();

    while(1)
    {
        /* If scanning is completed update the baseline count and check if sensor
is active */
        while(CapSense_IsBusy());

        /* Update baseline for all the sensors */
        CapSense_UpdateEnabledBaselines();

        CapSense_ScanEnabledWidgets();

        /* Test if button widget is active */
        stateB_1 = CapSense_CheckIsWidgetActive(CapSense_BUTTON0__BTN);
        stateB_2 = CapSense_CheckIsWidgetActive(CapSense_BUTTON1__BTN);
        pos =(uint8)CapSense_GetCentroidPos(CapSense_LINEARSLIDER0__LS);

        /* 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:
*  slot_1: Button state for B1
*  slot_2: Button state for B2
*
* Return:
```

```
*  void
*
*******************************************************************************/
void UpdateButtonState(uint8 slot_1, uint8 slot_2)
{
    LCD_Char_1_Position(ROW_0,COLUMN_0);

    /* Check the state of the buttons and update the LCD and LEDs */
    if (slot_1 && slot_2)
    {
        /* Display both Button strings on LCD if both button slots are active */
        LCD_Char_1_PrintString(BUTTON_1_2_STR);
        /* Both LED's are on in this state */
        LED1_Write(LED_ON);
        LED2_Write(LED_ON);
    }
    else if (slot_1 || slot_2)
    {
        if (slot_1)
        {
            /* Display Button 1 state on LCD and LED1 */
            LCD_Char_1_PrintString(BUTTON_1_STR);
            LED1_Write(LED_ON);
            /* Button 2 is not active */
            LED2_Write(LED_OFF);
        }
        if (slot_2)
        {
            /* Display Button 2 state on LCD and LED2 */
            LCD_Char_1_PrintString(BUTTON_2_STR);
            LED2_Write(LED_ON);

            /* Button 1 is not active */
            LED1_Write(LED_OFF);
        }
    }
    else
    {
        /* Display default string on LCD and set LED's to off */
        LCD_Char_1_PrintString(DEFAULT_ROW_0_STR);

        /* Set both LED's off in this state */
        LED1_Write(LED_OFF);
        LED2_Write(LED_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(uint8 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_Char_1_Position(ROW_1, COLUMN_0);
        LCD_Char_1_PrintString(DEFAULT_ROW_1_STR);
    }
    else
    {
        /* Update the bargraph with the current finger position */
        LCD_Char_1_DrawHorizontalBG(ROW_1, COLUMN_0, NUM_CHARACTERS, value +1);
    }
}


/* [] END OF FILE */
```

3. 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 27 or Programming My First PSoC 5 Project on page 31 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 5 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** → **Analog** → **ADC** → **SAR ADC**)
- Voltage DAC (**Component Catalog** → **Analog** → **DAC** → **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** → **Digital** → **Logic** → **Logic High**)

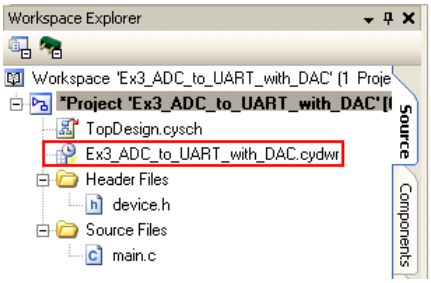
3.3.6.1 Creating ADC to UART with DAC Project

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

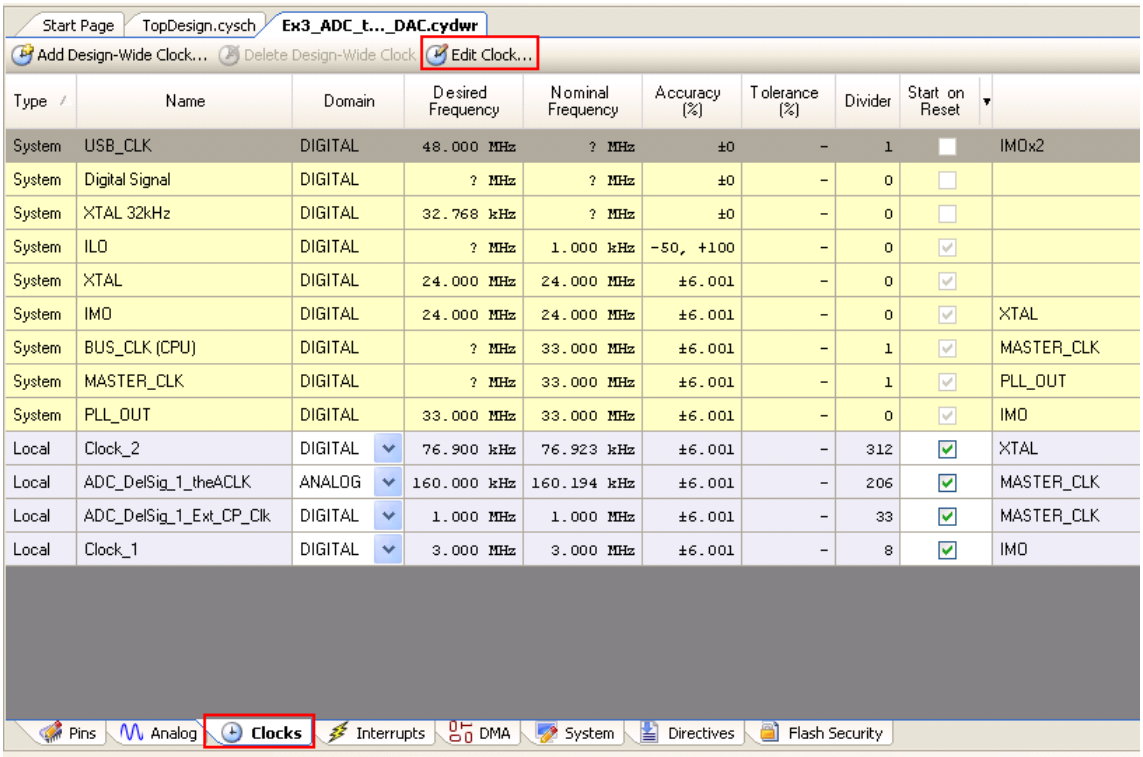
1. Open the Ex3_ADC_to_UART_with_DAC.cydwr file from Workspace Explorer. See figure below

Figure 3-154. Workspace Explorer



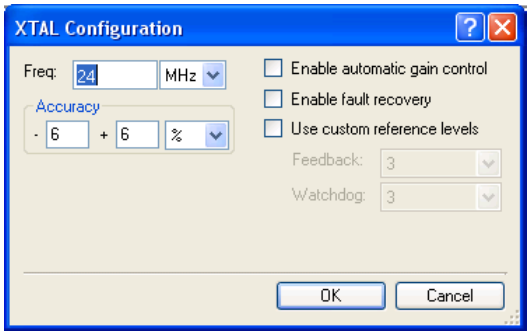
2. Select the **Clocks** tab and click on **Edit Clock…**.

Figure 3-155. Edit Clock



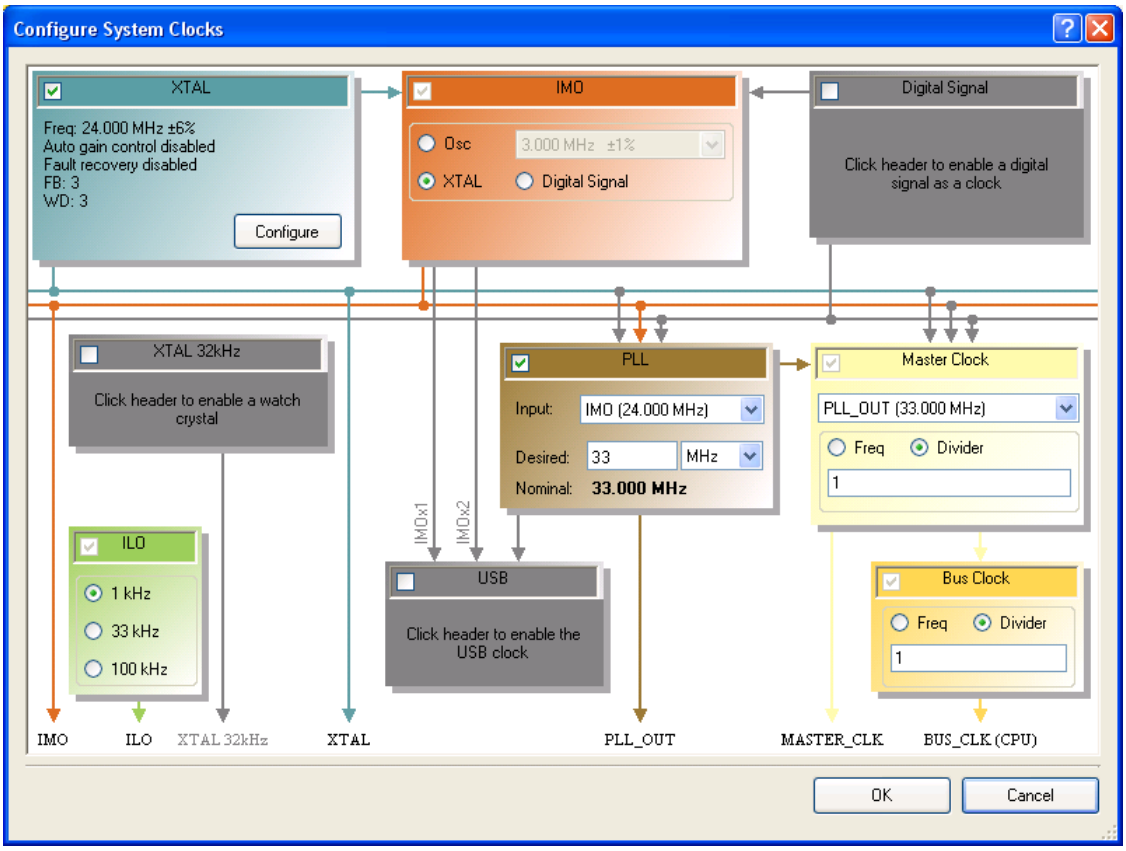
3. In the 'Configure System Clocks' window, enable and configure the XTAL to **24 MHz** with accuracy as **± 6%**.

Figure 3-156. Configure XTAL



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*

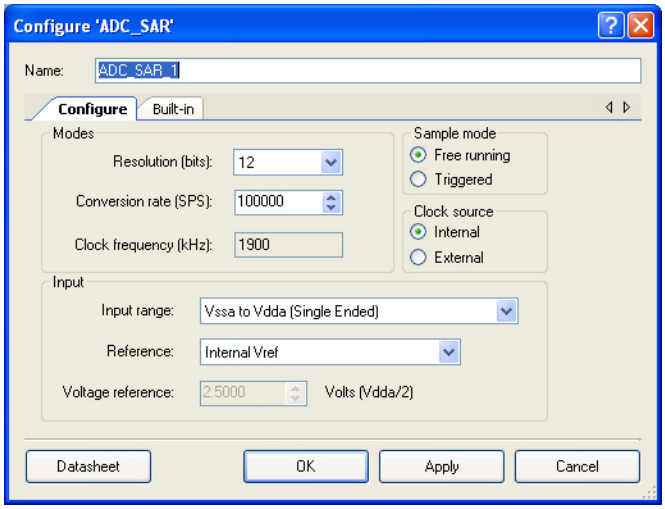
1. Drag and drop the SAR ADC component (**Component Catalog** → **Analog** → **ADC** → **SAR ADC**)
2. Double-click the **ADC_SAR_1** component in the schematic to open the configuration window.
3. Configure the SAR ADC as follows:

   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

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

Figure 3-158. SAR ADC Component Configuration



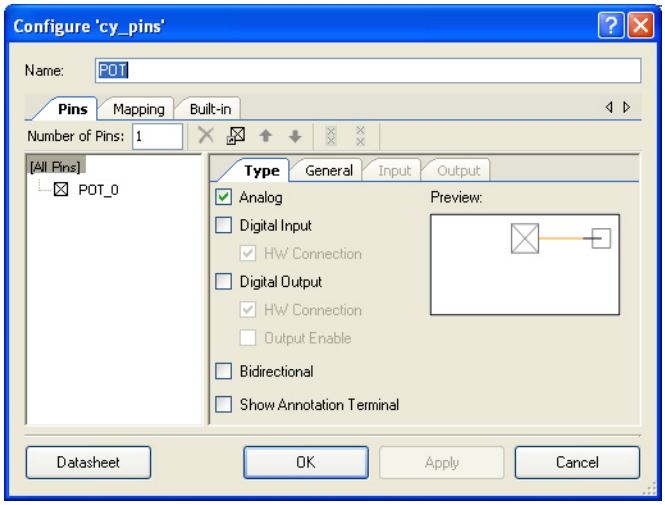
3.3.6.4 *Placing and Configuring an Analog Pin*

1. Drag and drop the Analog Pin component (**Component Catalog** → **Ports and Pins** → **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



General Tab

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

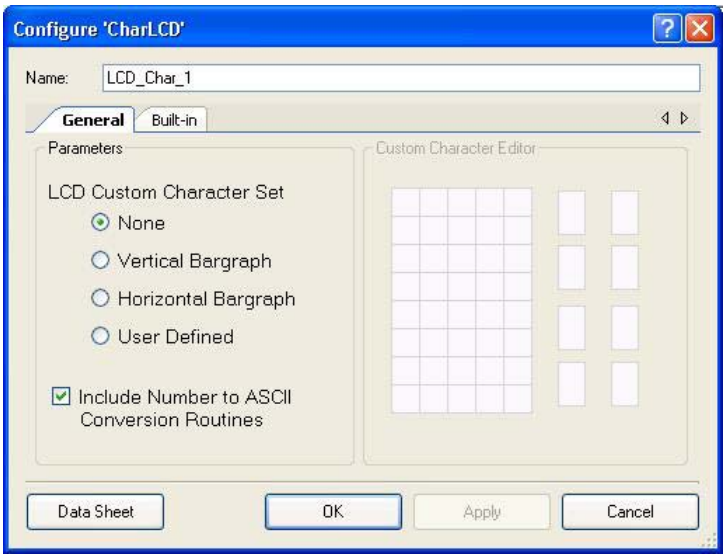
For more information about what the parameters mean, click the **Datasheet** button in the configuration window.

3.3.6.5 *Placing and Configuring Character LCD*

1. Drag and drop the Character LCD component (**Component Catalog → Display → 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 3-160. Character LCD Component Configuration



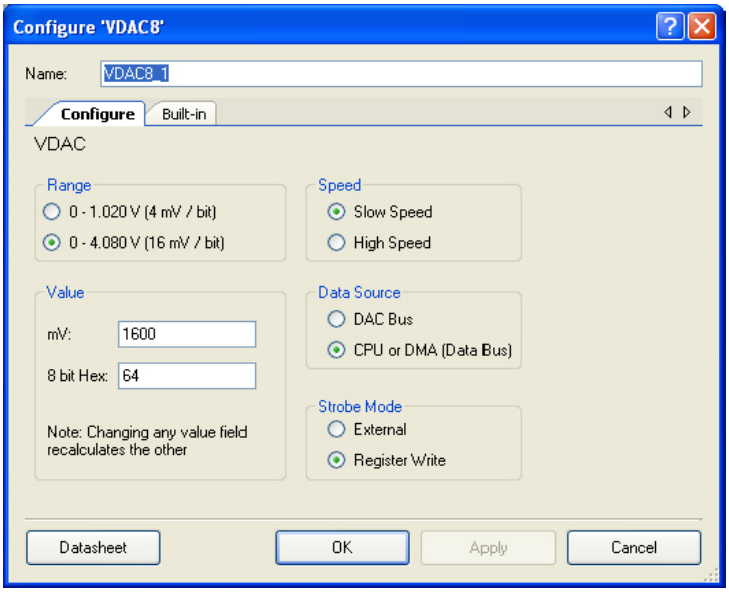
3.3.6.6 *Placing and Configuring Voltage DAC*

1. 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)
   - **Initial_Value**: 1600
   - **Value_mv**: Register Write
   - **Value_8 bit hex**: 64
   - **VDAC_Range**: 0 - 4.080 V (16 mV/bit)
   - **VDAC_Speed**: Slow Speed

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

Figure 3-161. Voltage DAC Component Configuration



3.3.6.7 *Placing and Configuring Opamp*

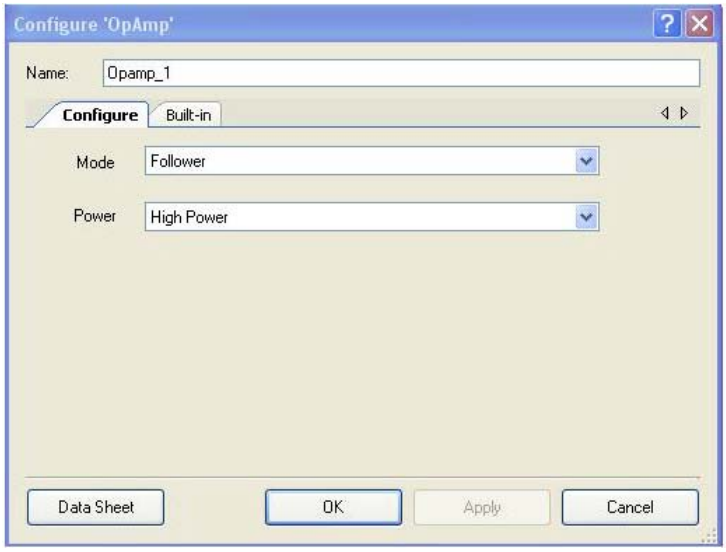
1. Drag and drop the Opamp component (**Component Catalog** → **Analog** → **Amplifiers** → **Opamp**)
2. Double-click the **Opamp_1** component in the schematic to open the configuration window.
3. Configure the Opamp:

   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



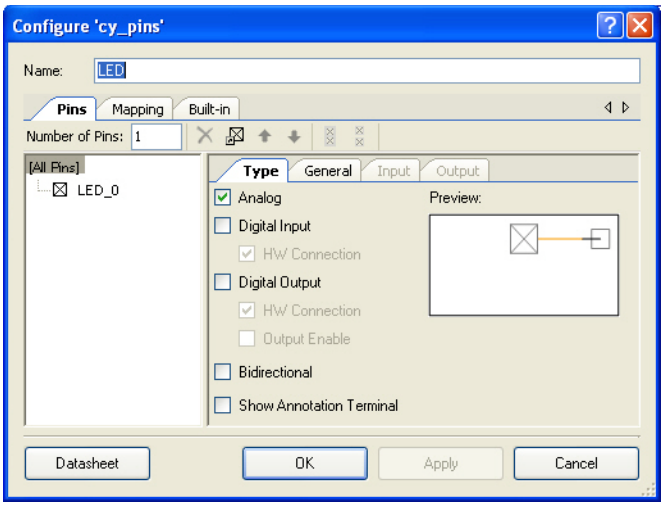
3.3.6.8 *Placing and Configuring Analog Pin*

1. Drag and drop the analog pin component (**Component Catalog** → **Ports and Pins** → **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

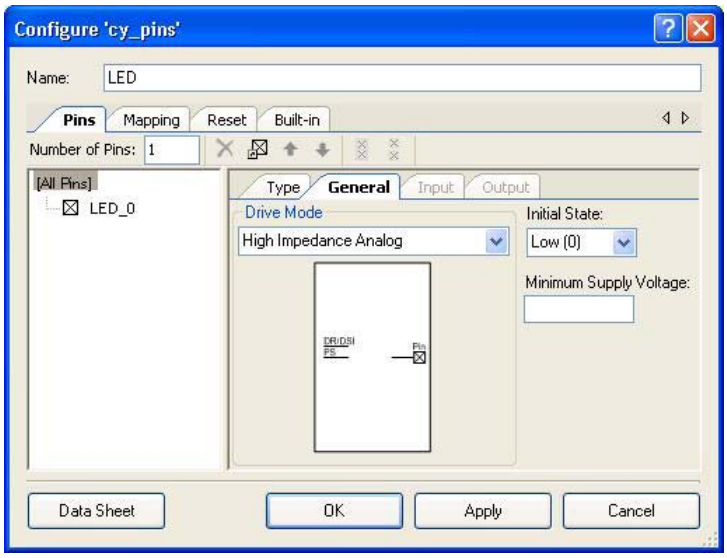


General Tab

- **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 3-164. LED Component Configuration



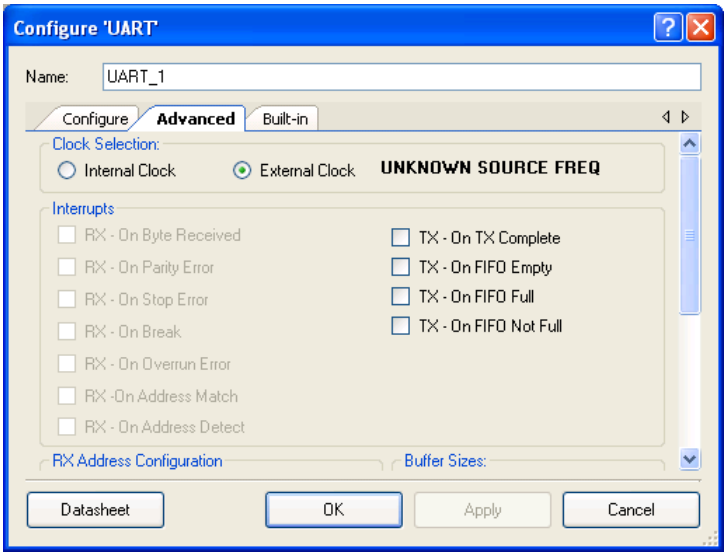
3.3.6.9 *Placing and Configuring UART*

1. Drag and drop the UART component (**Component Catalog** → **Communications** → **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

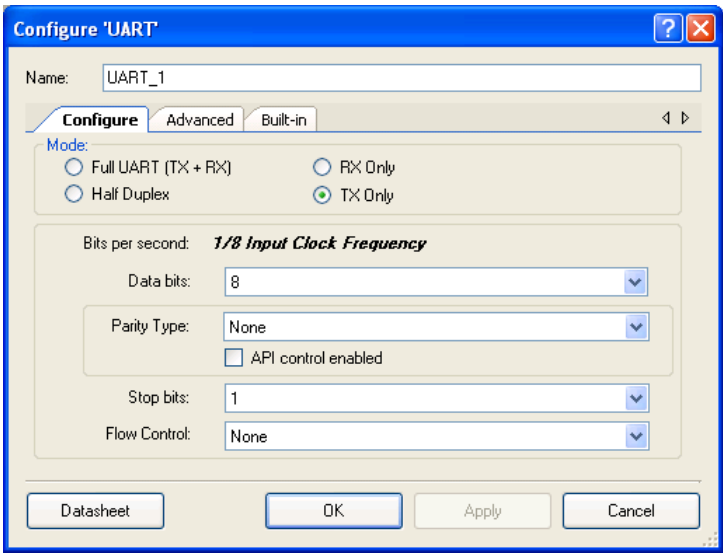


   Configure Tab

   - **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



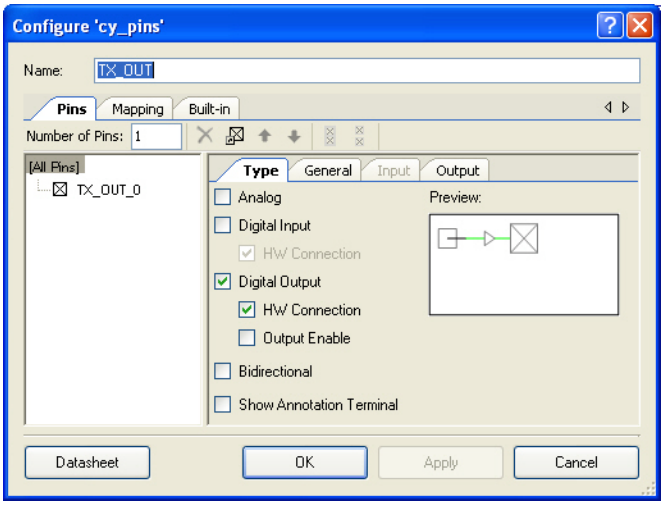
3.3.6.10 *Placing and Configuring Digital Output Pin*

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 3-167. TX_OUT Component Configuration

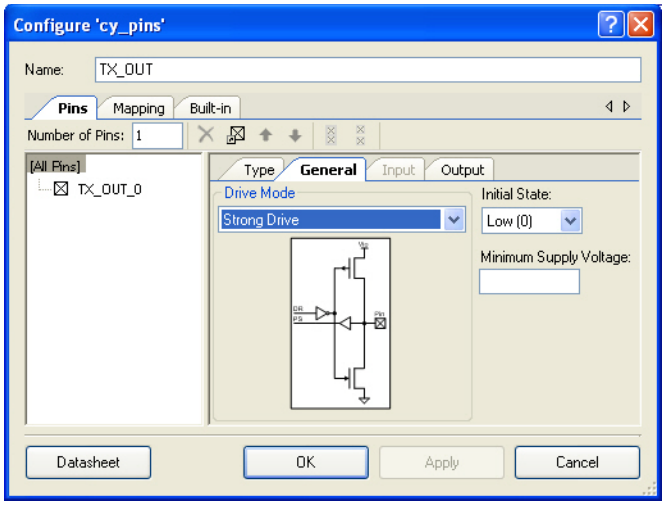


General Tab

- Under **Drive Mode**: Strong Drive

- Leave the remaining parameters as default

Figure 3-168. Pins - TX_OUT Component Configuration



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** → **System** → **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

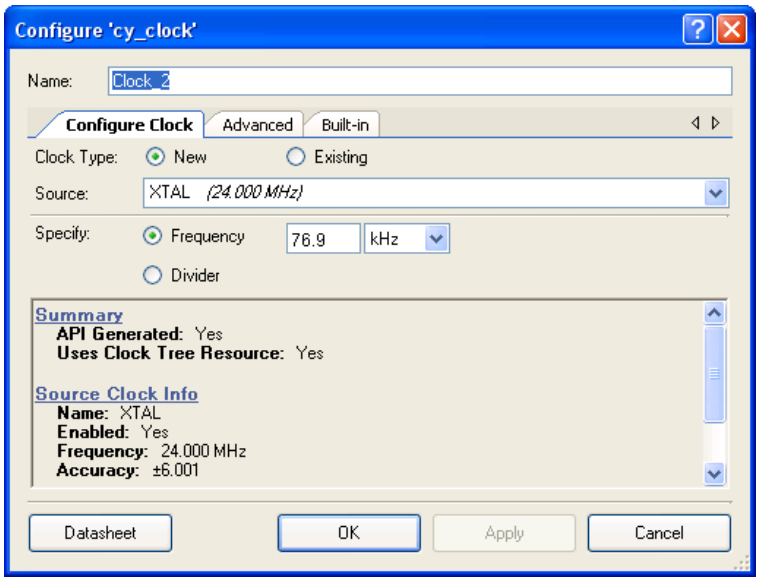


Figure 3-170. UART Configure Window After Assigning Clock Source - Configure Tab

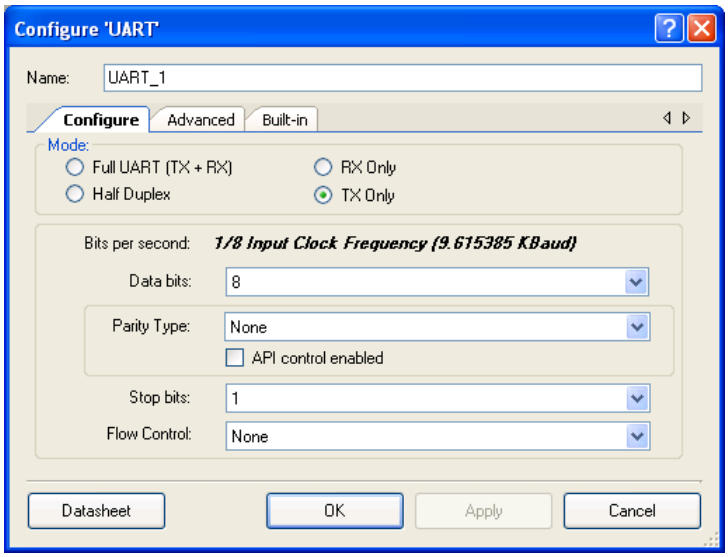
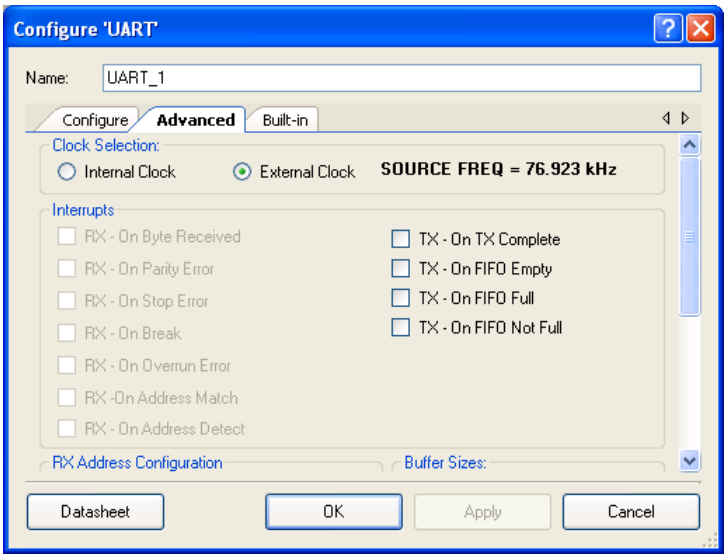


Figure 3-171. UART Configure Window After Assigning Clock Source - Advanced Tab



3.3.6.12 *Placing and Configuring DMA*

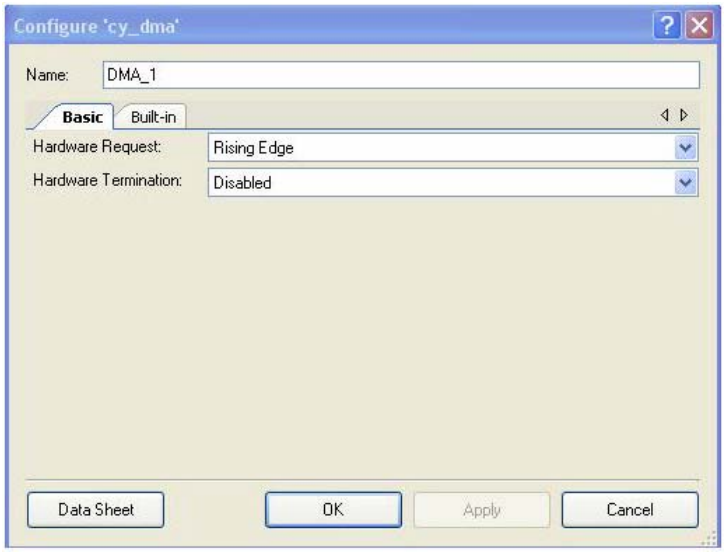
1. Drag and drop the DMA component (**Component Catalog** → **System** → **DMA**)
2. Double-click the **DMA_1** component in the schematic to open the configuration window.
3. Configure the DMA:

   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

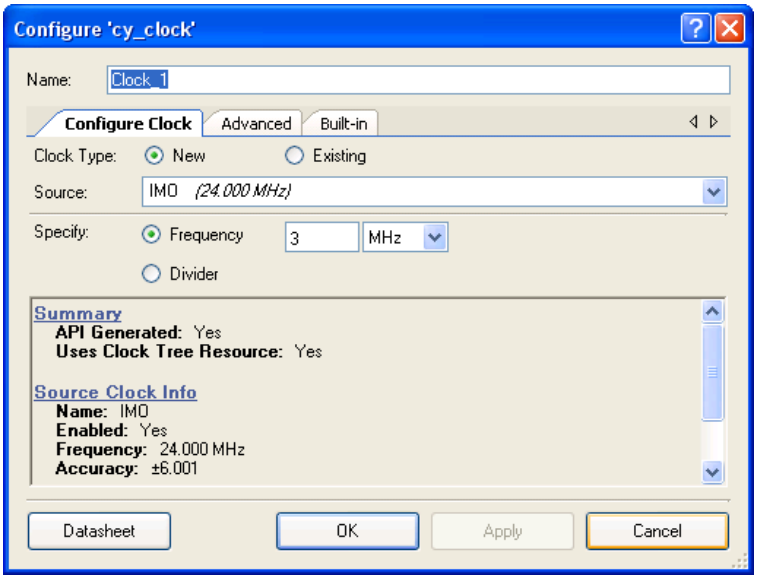


3.3.6.13 *Connecting the Components Together*

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

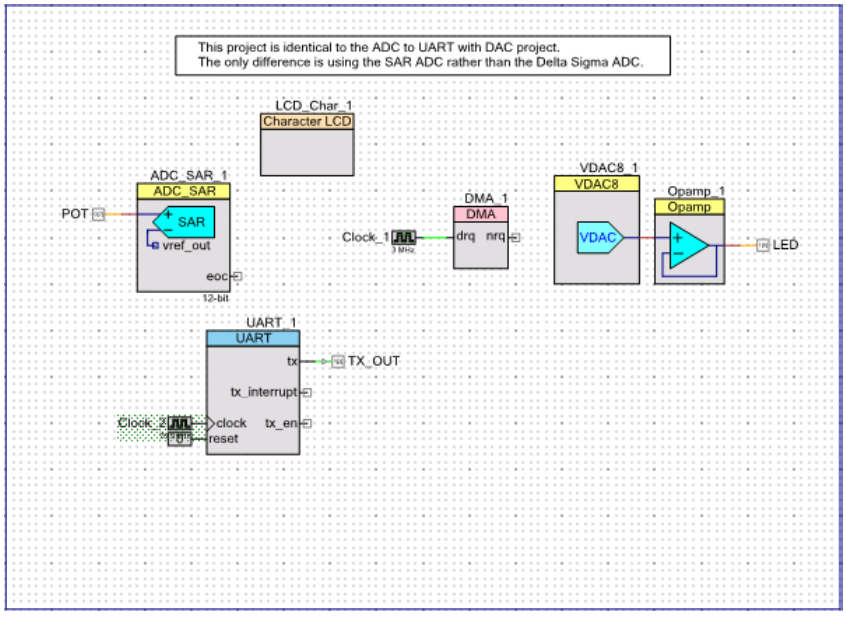
   Configure Clock Tab
   - **Name**: Clock_1
   - **Source**: IMO (24.000 MHz)
   - **Desired Frequency**: 3 MHz
   - Leave the remaining parameters set to their default values

Figure 3-173. Clock Component Configuration



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, connect **VDAC8 (VDAC8_1)** to **Opamp (Opamp_1)**.
7. Using the Wire Tool, 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.

Figure 3-174. Connected Components



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

Alias	Name	Pin
	LED	P1[6]
	TX_OUT	P1[2]
	\LCD_Char_1:LCDPort\[6:0]	P2[6:0]
	POT	P0[7]

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 *CY8C55_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.

```
#include <device.h>

/* LCD specific */
#define ROW_0       0  /* LCD row 0     */
#define COLUMN_0    0  /* LCD column 0  */
#define COLUMN_9    9  /* LCD column 9  */
```

```
#define COLUMN_10  10  /* LCD column 10 */
#define COLUMN_11  11  /* LCD column 11 */
/* For clearing Tens and Hundreds place */
#define CLEAR_TENS_HUNDREDS "    "
/* For clearing Hundreds place */
#define CLEAR_HUNDREDS      "   "

/* DMA specific */
#define REQUEST_PER_BURST   1 /* One request per burst */
#define BURST_BYTE_COUNT    1 /* Bursts are one byte each */
/* Upper 16-bits of the source address are zero */
#define SOURCE_ADDRESS      0
/* Upper 16-bits of the destination address are zero */
#define DESTINATION_ADDRESS 0

void UpdateDisplay(uint16 * voltageRawCount);
void TxHex (uint16 voltageRawCount);

/* Table of voltage values for DMA to send to the DAC. These values range
 * between 0x3D and 0x9F because these are the two points where the LED
 * is not visible and where the LED is saturated */
const uint8 voltageWave[] =
{
 0x6D, 0x6F, 0x71, 0x73, 0x75, 0x77, 0x79, 0x7B, 0x7D, 0x7F, 0x81, 0x83, 0x85,
 0x87, 0x89, 0x8B, 0x8D, 0x8F, 0x91, 0x93, 0x95, 0x97, 0x99, 0x9B, 0x9C, 0x9D, 0x9D, 0x9E,
 0x9E, 0x9F, 0x9F, 0x9F, 0x9E, 0x9E, 0x9E, 0x9C, 0x9C, 0x9B, 0x99, 0x97, 0x95, 0x93, 0x91,
 0x8F, 0x8D, 0x8B, 0x89, 0x87, 0x85, 0x83, 0x81, 0x7F, 0x7D, 0x7B, 0x79, 0x77,
 0x75, 0x73, 0x71, 0x6F, 0x6D, 0x6B, 0x69, 0x67, 0x65, 0x63, 0x61, 0x5F, 0x5D,
 0x5B, 0x59, 0x57, 0x55, 0x53, 0x51, 0x4F, 0x4D, 0x4B, 0x49, 0x47, 0x45, 0x43,
 0x41, 0x40, 0x40, 0x3F, 0x3F, 0x3D, 0x3D, 0x3D, 0x3D, 0x3D, 0x3D, 0x3F, 0x41,
 0x43, 0x45, 0x47, 0x49, 0x4B, 0x4D, 0x4F, 0x51,
 0x53, 0x55, 0x57, 0x59, 0x5B, 0x5D, 0x5F, 0x61, 0x63, 0x65, 0x67, 0x69, 0x6B
};

/*******************************************************************************
* Function Name: main
********************************************************************************
*
* Summary:
*   The main function initializes the ADC, LCD, VDAC, Analog Buffer, and UART.
*   It also initializes DMA by allocating/configuring a DMA channel and
*   Transaction Descriptor and also copies the voltage table address to the DAC
*   address. In the main loop, it starts and waits for an ADC conversion, then
*   it displays the ADC raw count to the LCD, transmits the raw count serially,
*   and sets the DMA clock divider proportional to the raw count.
*
* Parameters:
*  void
*
* Return:
*  void
*
*******************************************************************************/
void main()
{
    uint16 voltageRawCount;
    uint8  myChannel;
    uint8  myTd;
```

```
ADC_SAR_1_Start();    /* Configure and power up ADC */
LCD_Char_1_Start();   /* Initialize and clear the LCD */
VDAC8_1_Start();      /* Initializes VDAC8 with default values */
Opamp_1_Start();      /* Enables Opamp and sets power level */
UART_1_Start();       /* Enable UART */
CyDmacConfigure();    /* Set DMA configuration register */

/* Allocate and initialize a DMA channel to be used by the caller */
myChannel = DMA_1_DmaInitialize(BURST_BYTE_COUNT,
                                REQUEST_PER_BURST,
                                (uint16)((uint32)voltageWave >> 16),
                                (uint16)(VDAC8_1_viDAC8__D >> 16));

/* Allocate a Transaction Descriptor (TD) from the free list */
myTd = CyDmaTdAllocate();

/* Move the LCD cursor to Row 0, Column 0 */
LCD_Char_1_Position(ROW_0, COLUMN_0);

/* Print Label for the pot voltage raw count */
LCD_Char_1_PrintString("V Count: ");

CyDmaTdSetConfiguration(myTd, sizeof(voltageWave),
                        myTd, TD_INC_SRC_ADR ); /* Configure the TD */

/* Copy address of voltageWave to address of DAC. Set the lower 16-bits of
 *  the source and destination addresses for this TD */
CyDmaTdSetAddress(myTd,
                  (uint16)((uint32)voltageWave),
                  (uint16)VDAC8_1_viDAC8__D);

/* Associate TD with channel */
CyDmaChSetInitialTd(myChannel, myTd);

/* Enable DMA channel */
CyDmaChEnable(myChannel, 1);

/* Clock will make burst requests to the DMAC */
Clock_1_Start();

ADC_SAR_1_StartConvert(); /* Force ADC to initiate a conversion */

while(1)
{
    /* Wait for end of conversion */
    ADC_SAR_1_IsEndConversion(ADC_SAR_1_WAIT_FOR_RESULT);

    /* Get converted result */
    /* Shifting for silicon errata workaround to get 8-bit value */
    voltageRawCount = (ADC_SAR_1_GetResult16()>>4);

    UpdateDisplay(&voltageRawCount); /* Print the result to LCD */

    TxHex(voltageRawCount); /* Transmit result to UART */

    /*
     * The LED blinking frequency is dependent on the Voltage raw count.
     *  With a 3 MHz clock, the lowest divider (for raw count of 0) should be
```

```
            *  1000 to blink at a significantly fast pace. The highest value is
            *  about 52,200 (for a raw count of 256) to blink at a significantly
            *  slow pace. The following equation is necessary to make the adjusted
            *  clock frequency (with the updated divider) linear with the ADC
            *  output.
            */
            Clock_1_Stop();
            Clock_1_SetDivider((((uint32)voltageRawCount * 1000) /
                   (261 - (uint32)voltageRawCount)) + 1000);
            Clock_1_Start();
        }
    }

    /*******************************************************************************
    * Function Name: UpdateDisplay
    ********************************************************************************
    *
    * Summary:
    *   Print voltage raw count result to the LCD. Clears some characters if
    *   necessary. The voltageRawCount parameter is also updated for use in other
    *   functions.
    *
    * Parameters:
    *   voltageRawCount: Voltage raw count from ADC
    *
    * Return:
    *   void
    *
    *******************************************************************************/
    void UpdateDisplay (uint16 * voltageRawCount)
    {
        /* Move the cursor to Row 0, Column 9 */
        LCD_Char_1_Position(ROW_0, COLUMN_9);
        LCD_Char_1_PrintNumber(voltageRawCount[0]); /* Print the result */

        if (voltageRawCount[0] < 10)
        {
            /* Move the cursor to Row 0, Column 10 */
            LCD_Char_1_Position(ROW_0,COLUMN_10);
            LCD_Char_1_PrintString(CLEAR_TENS_HUNDREDS); /* Clear last characters */
        }
        else if (voltageRawCount[0] < 100)
        {
            /* Move the cursor to Row 0, Column 11 */
            LCD_Char_1_Position(ROW_0,COLUMN_11);
            LCD_Char_1_PrintString(CLEAR_HUNDREDS); /* Clear last characters */
        }
        else
        {
            /* Continue on */
        }
    }

    /*******************************************************************************
    * Function Name: TxHex
    ********************************************************************************
    *
    * Summary:
```

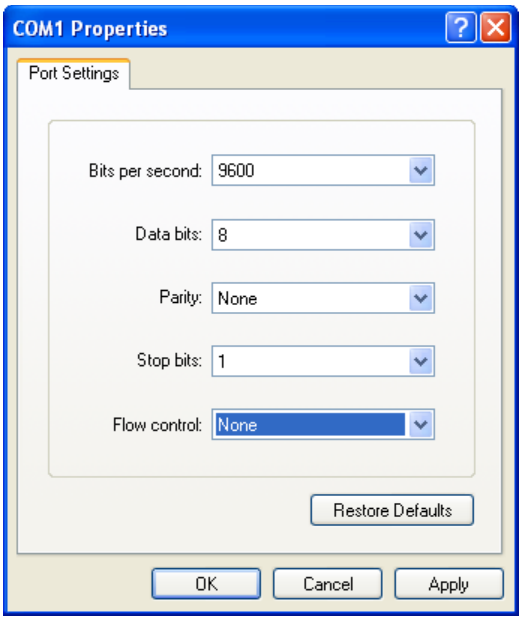
```
*   Convert voltage raw count to hex value and TX via UART.
*
* Parameters:
*   voltageRawCount: The voltage raw counts being received from the ADC
*
* Return:
*   void
*
********************************************************************************/
void TxHex (uint16 voltageRawCount)
{
    static char8 const hex[16] = "0123456789ABCDEF";

    /* TX converted MSnibble */
    UART_1_PutChar(hex[(voltageRawCount>>12)&0xF]);
    /* TX converted second nibble */
    UART_1_PutChar(hex[(voltageRawCount>>8)&0xF]);
    /* TX converted third nibble */
    UART_1_PutChar(hex[(voltageRawCount>>4)&0xF]);
    /* TX converted LSnibble */
    UART_1_PutChar(hex[voltageRawCount&0xF]);
    UART_1_PutString("h\r");  /* h for hexadecimal and carriage return */
}


/* [] END OF FILE */
```

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



9. Use PSoC Creator as described in Programming My First PSoC 5 Project on page 31 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.

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.

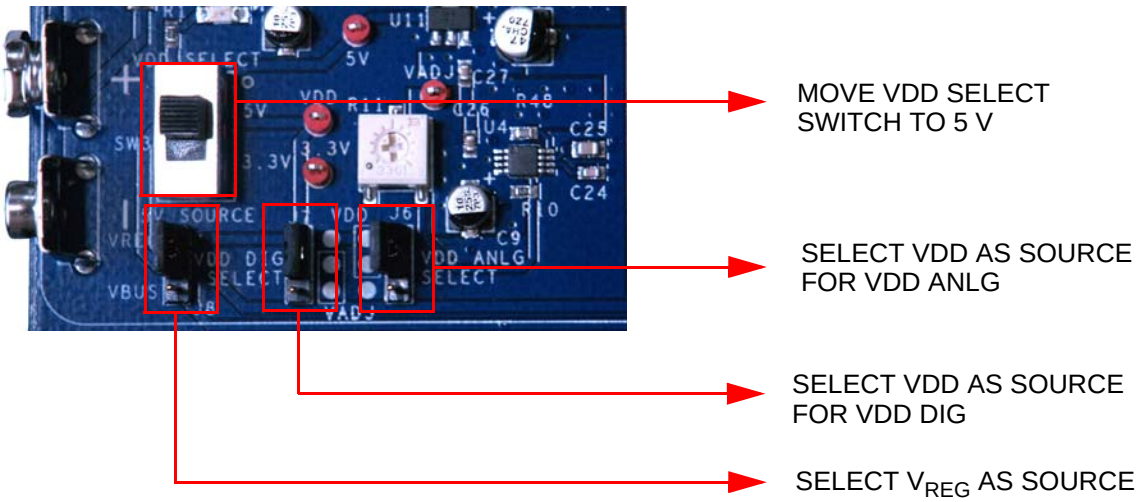
VDDIO - This is power derived from either VDD or VADJ. It is used to power digital I/O on the PSoC device. There are four sections of GPIO, which can be powered to 5 V, 3.3 V, or VADJ using four headers. It enables you to power the PSoC GPIOs at different voltages.

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.

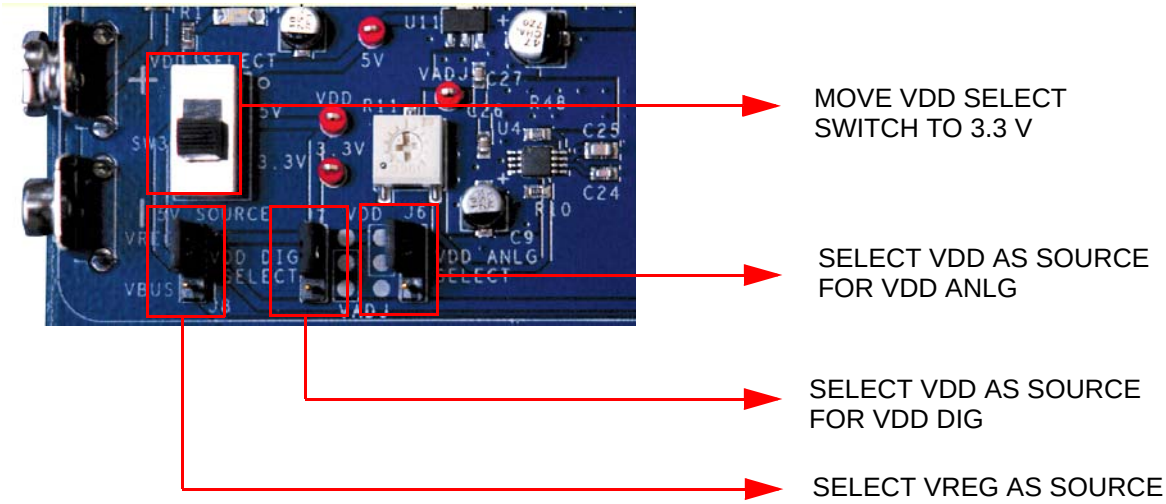
Figure A-1. Setting a 5-V Supply from VREG



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.

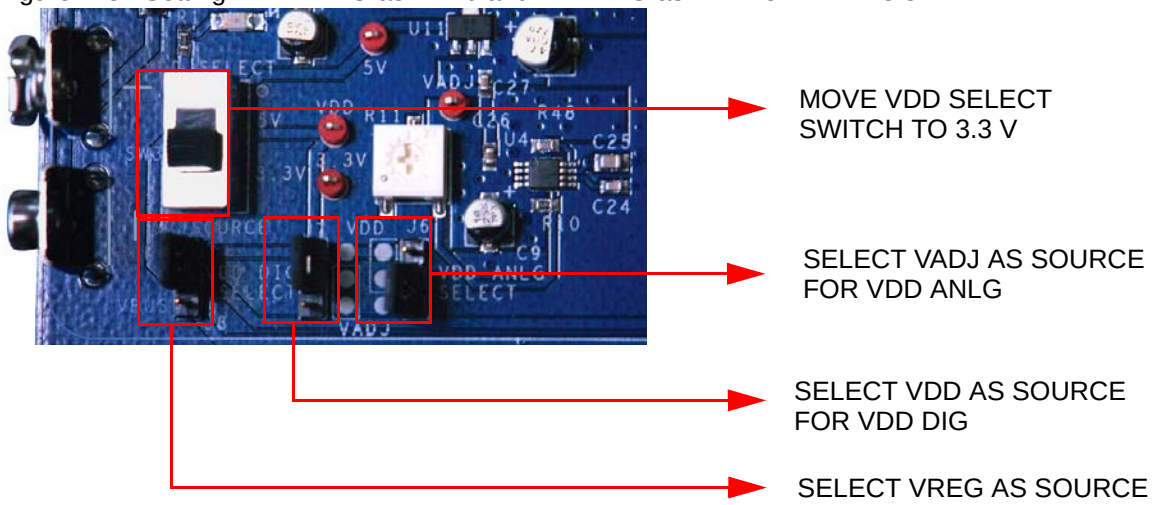
Figure A-2. Setting a 3.3-V Supply from VREG



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

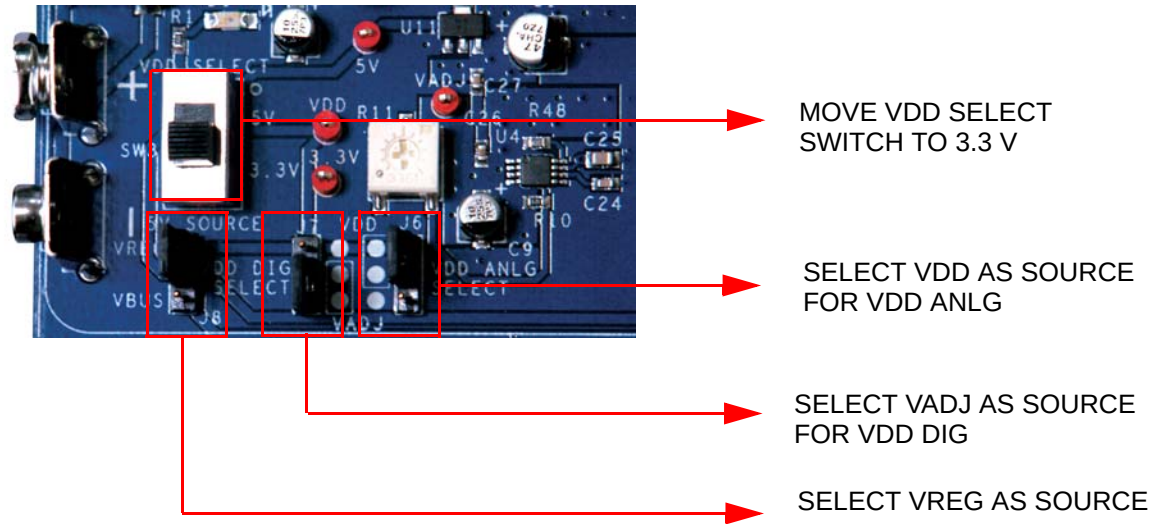


This helps to separate the analog supply from the digital supply and VDD.

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

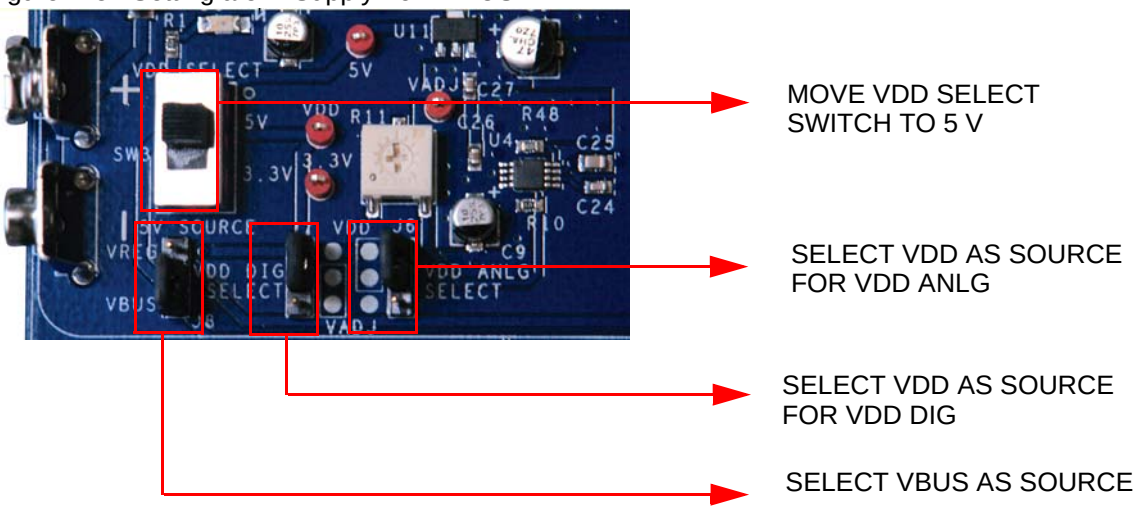


This helps to separate the digital supply from the analog supply and VDD.

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



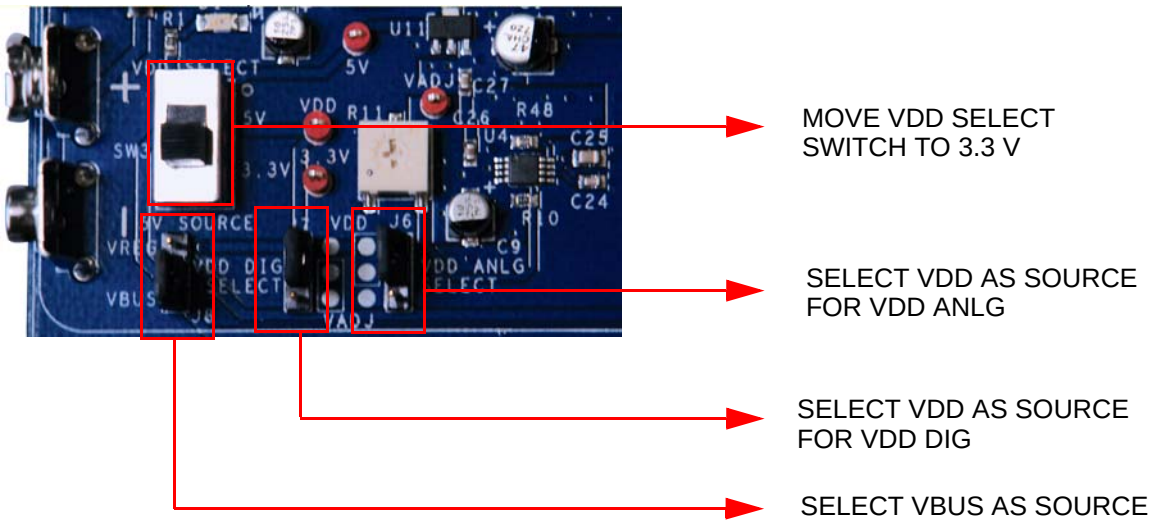
A.1.2.6 *Setting a 3.3-V Supply from VBUS*

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.

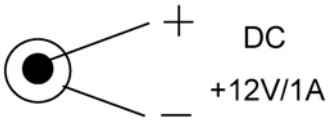
Figure A-6. Setting a 3.3-V supply from VBUS



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*

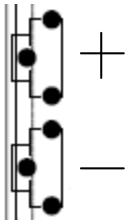
Figure A-7. DC Power Jack



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).

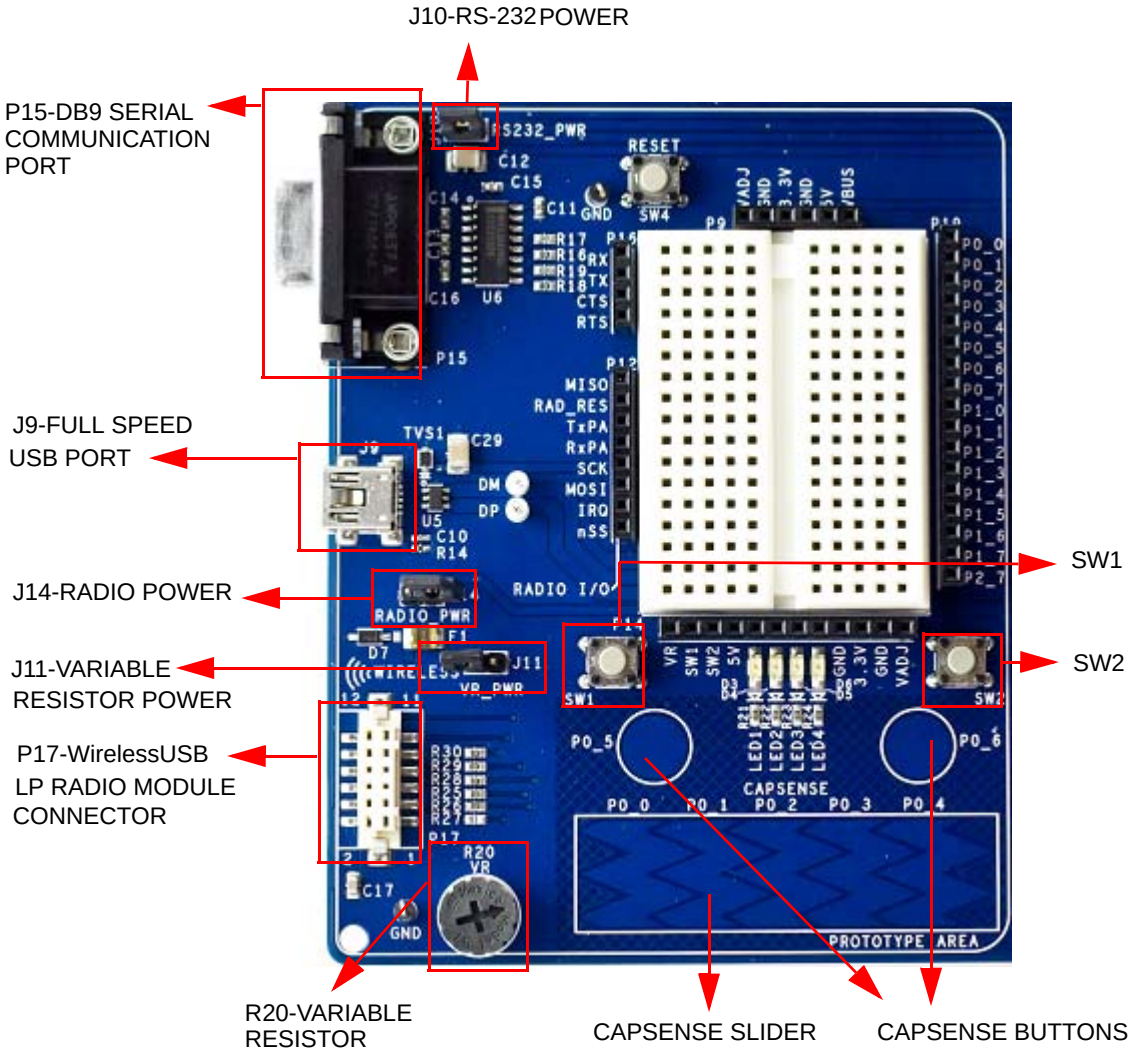
A.1.2.13 R11 - Adjustable Regulator Variable Resistor

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. CY8C55 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 CY8C55 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	TX
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.

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

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 *(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 kΩ 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.

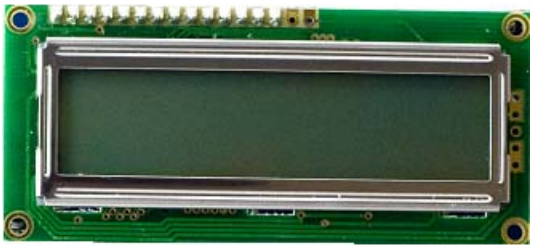
A.1.3.8 J11 - Variable Resistor Power

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 and SW2 - Multipurpose Push Button Switches

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

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).

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

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

A.1.4.1 R31 - LCD Contrast Adjustment

The board is equipped with an LCD contrast adjustment resistor R31. Turning the wiper counter-clockwise 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 0Ω is used for general-purpose CapSense applications, but a value of 560 Ω should be used to achieve best performance. The board is loaded with 0Ω series resistors by default. The presence of CapSense elements does not affect the general purpose use of port zero pins.

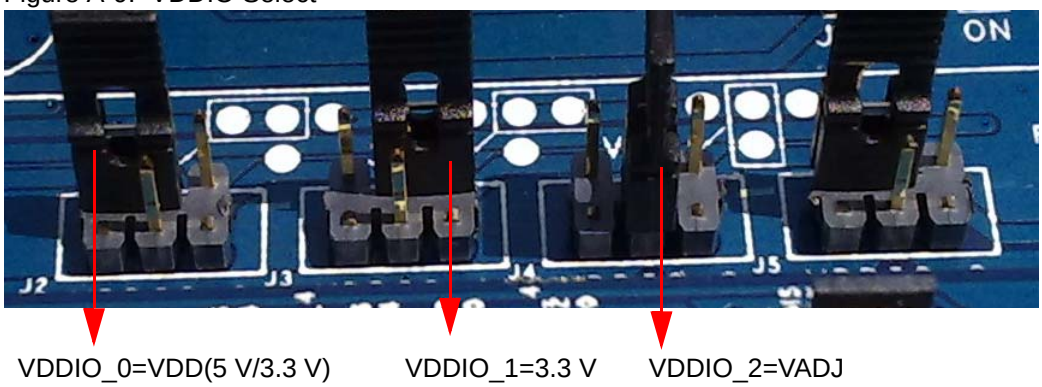
A.1.6 Processor Module

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.

Figure A-9. VDDIO Select



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 5 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.

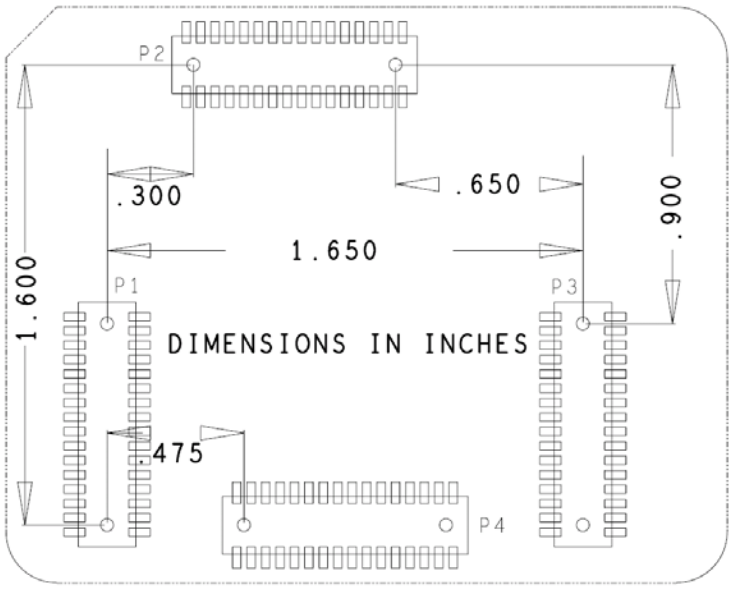
Table A-4. Connector Pin Assignments - Processor Module Sockets

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

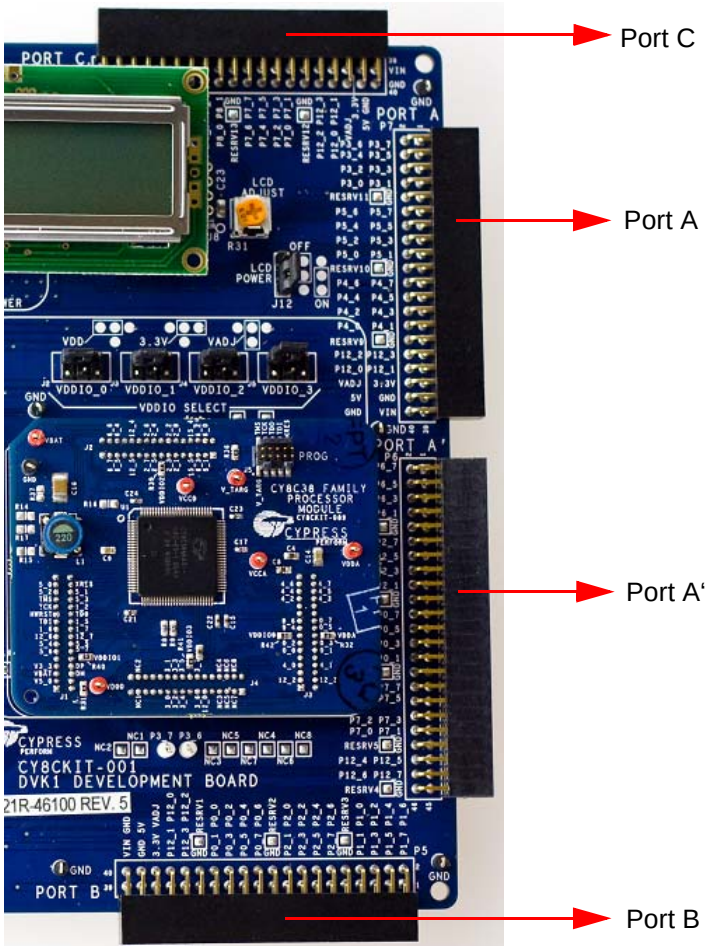
Table A-4. Connector Pin Assignments - Processor Module Sockets *(continued)*

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

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.

Table A-5. Connector Pin Assignments - Expansion Port Sockets

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 *(continued)*

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

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

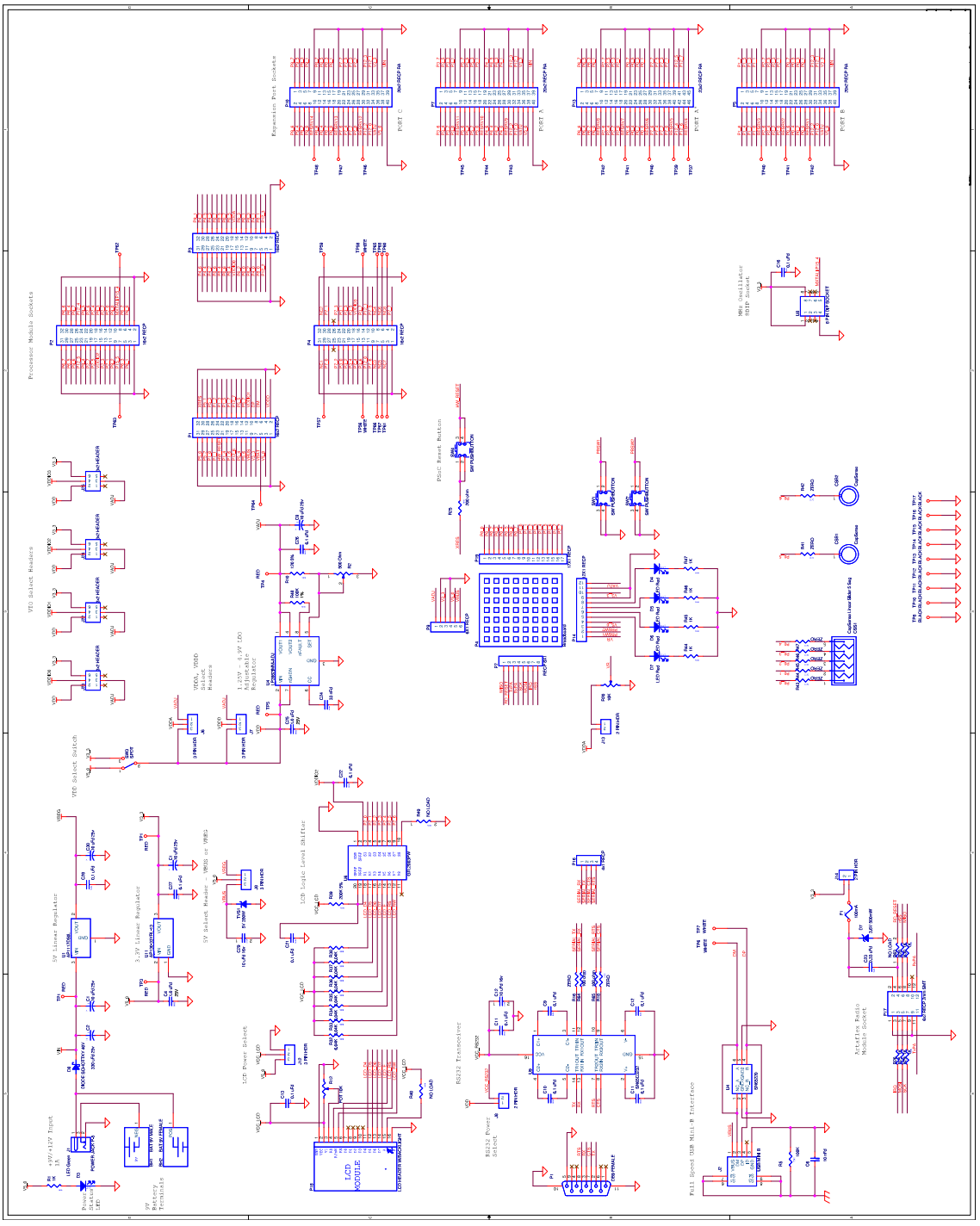
Expansion port B can be used as I/O ports with three full 8-bit ports: port0, port1, and port2. It has four special I/Os as well as ground and voltage pins. It can be used to join processor module I/Os port0, port1, and port2 with external I/Os through the use of daughter boards. It is mainly used in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

A.1.7.3 Expansion Port C

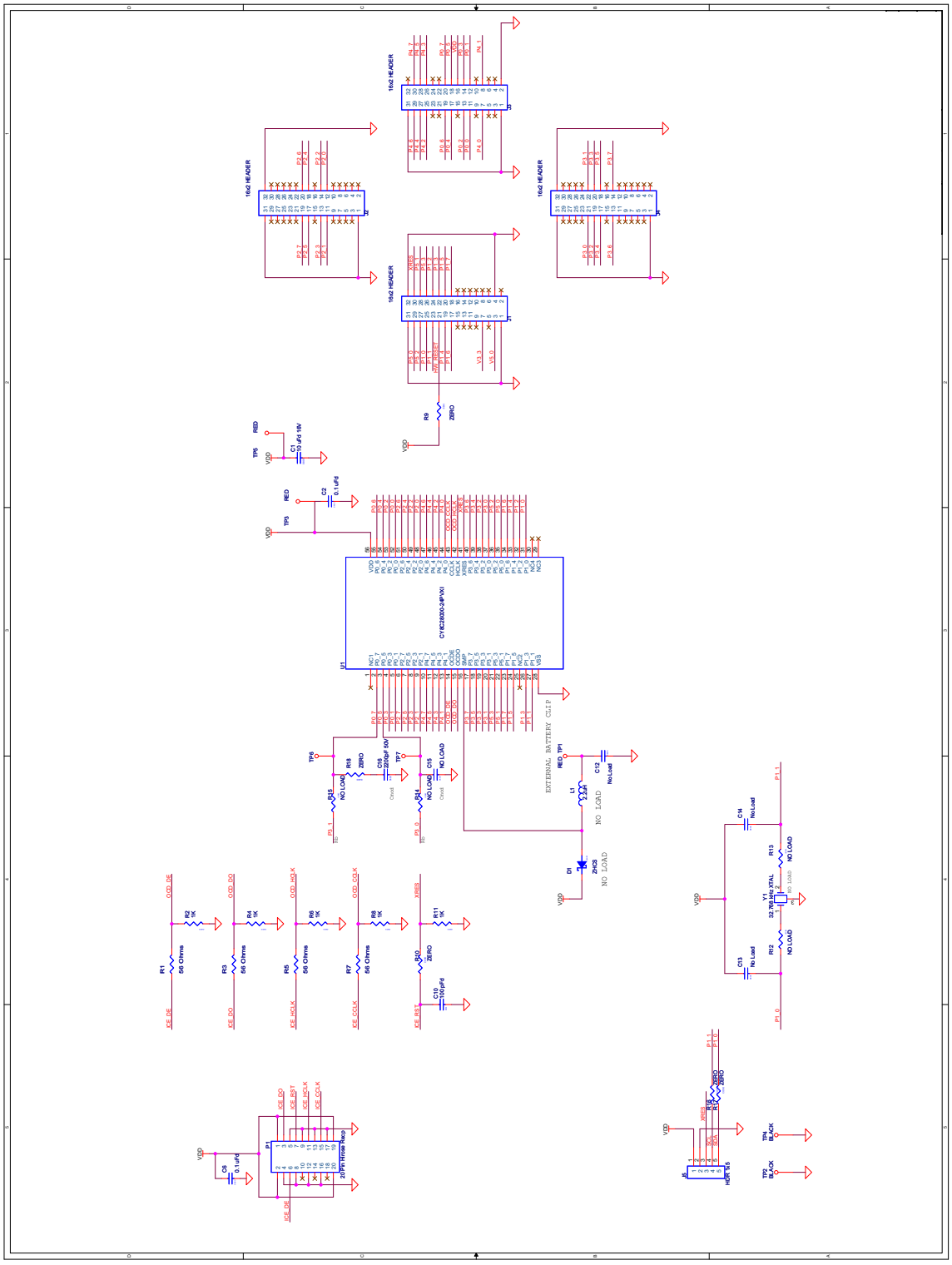
Expansion port C can be used as I/O ports with three full 8-bit ports: port7, port8, and port9. It has four special I/Os as well as ground and voltage pins. It can be used to join processor module I/Os port7, port8, and port9 with external I/Os through the use of daughter boards. It is used for devices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

A.2 Schematics

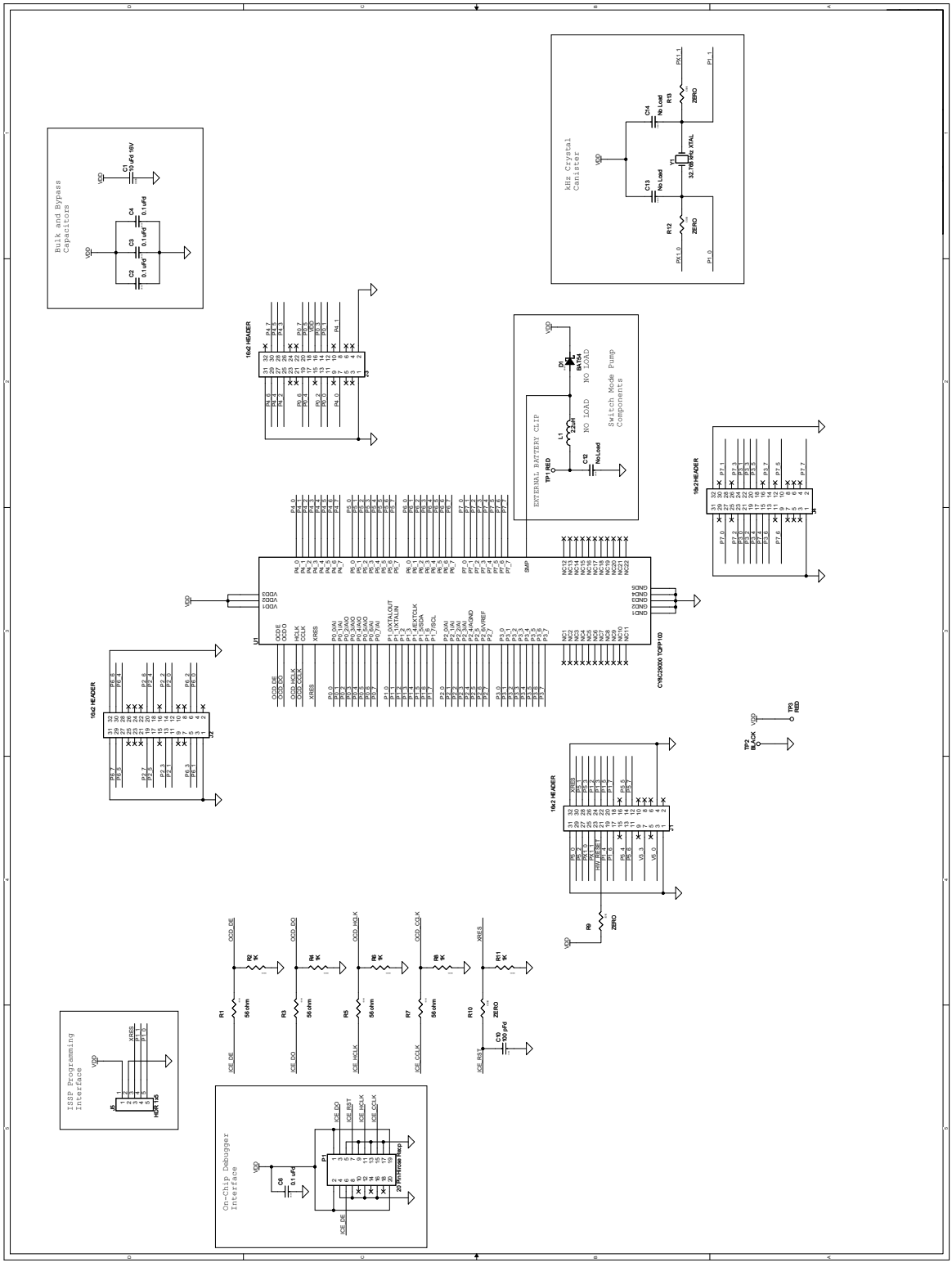
A.2.1 CY8CKIT-001 PSoC Development Board



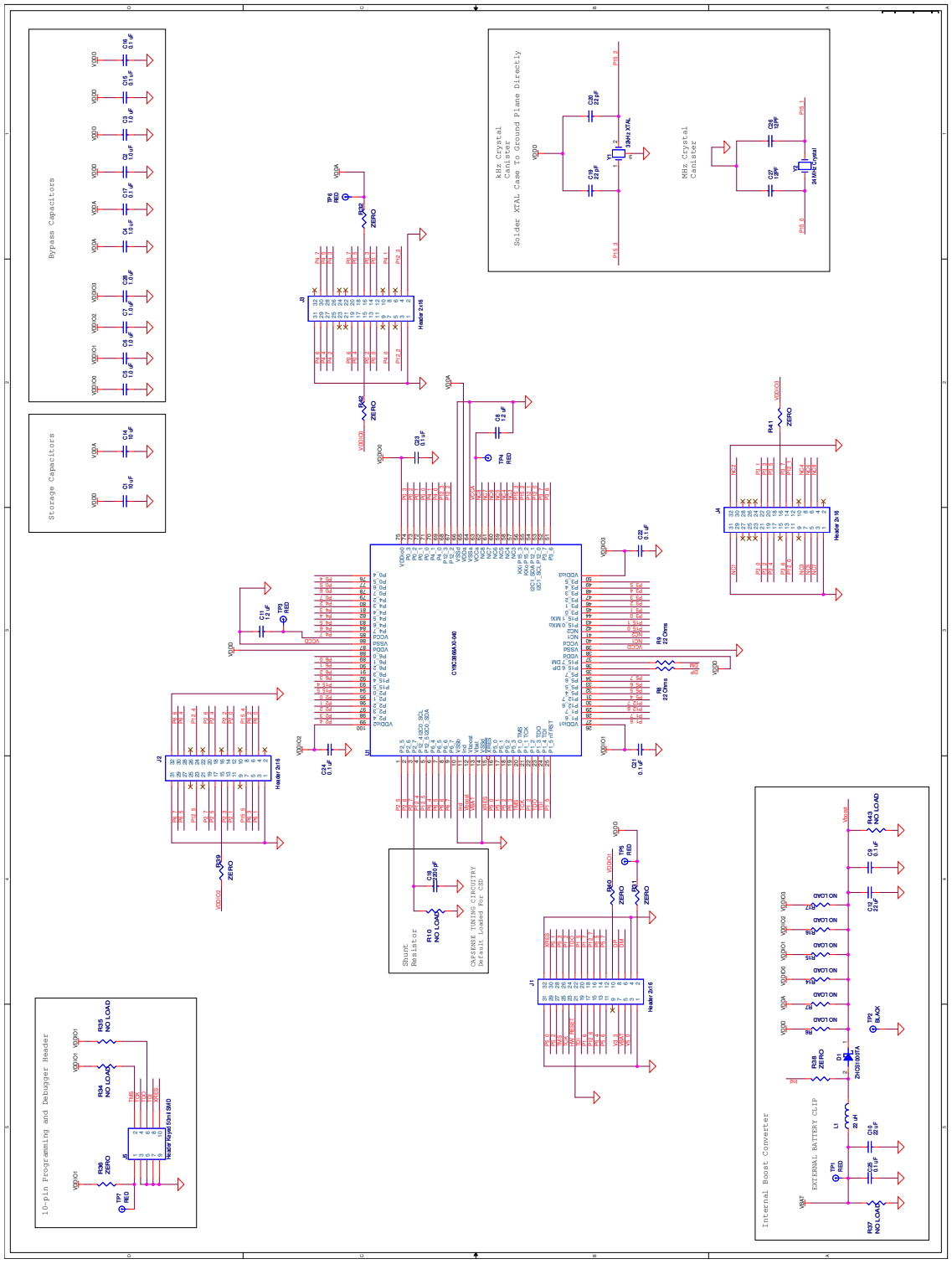
A.2.2 CY8C28 Family Processor Module



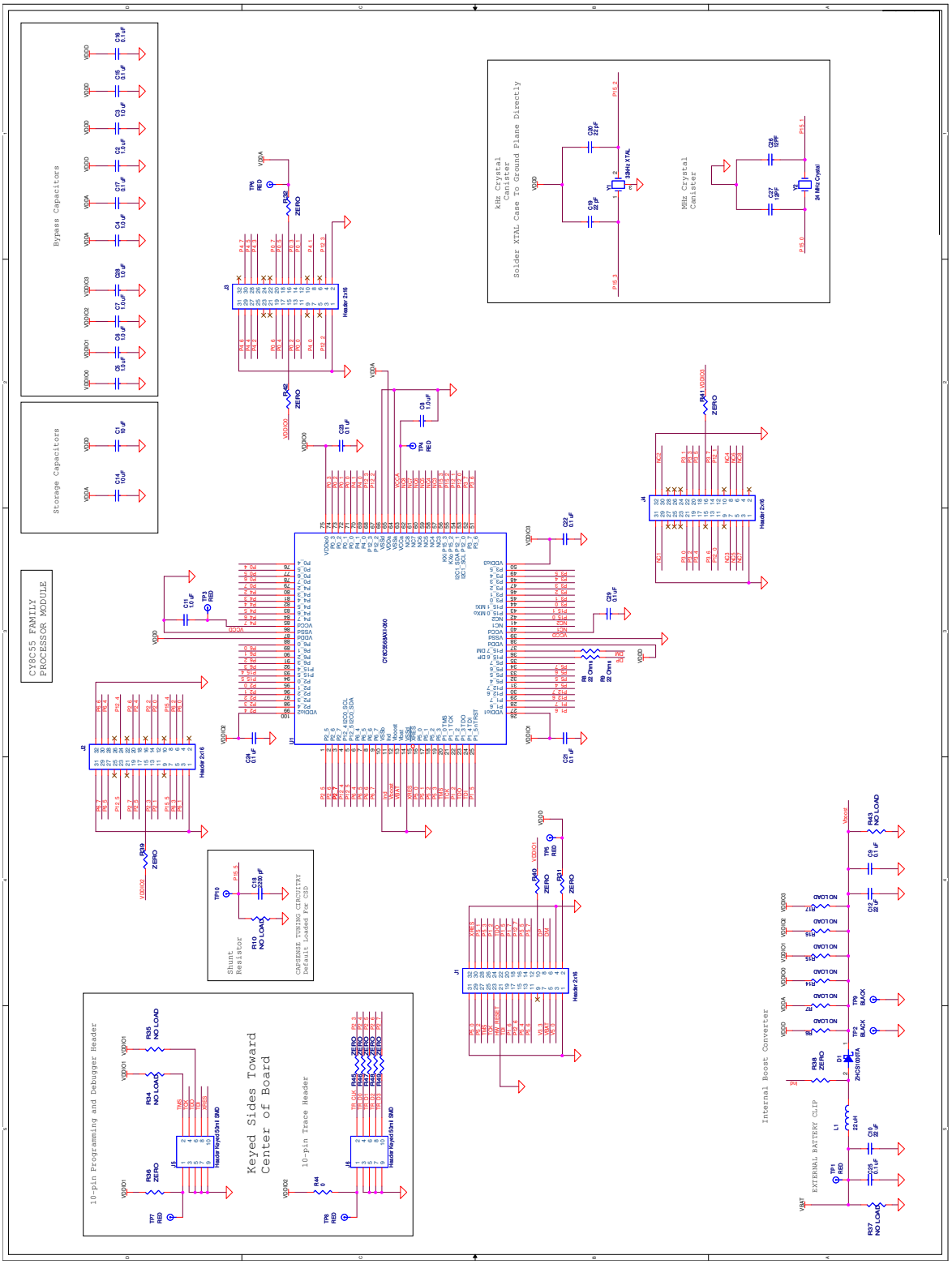
A.2.3 CY8C29 Family Processor Module



A.2.4 CY8C38 Family Processor Module



A.2.5 CY8C55 Family Processor Module



A.2.6 Enabling Boost Component in PSoC 3 and PSoC 5 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

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
1	1	PCB	PRINTED CIRCUIT BOARD	Cypress Semiconductor	PDCR-9461 REV **
2	1	BH1	BATTERY HOLDER 9V Male PC MT	Keystone Electronics	593
3	1	BH2	BATTERY HOLDER 9V Female PC MT	Keystone Electronics	594
4	4	C1,C3,C9,C30	CAP ELECT 10UF 25V VS SMD size B	Panasonic - ECG	EEE-1EA100WR
5	1	C2	CAP ELECT 330UF 25V FK SMD	Panasonic - ECG	EEE-FK1E331P
6	2	C4,C25	CAP CERAMIC 1.0UF 25V X5R 0603 10%	Taiyo Yuden	TMK107BJ105KA-T
7	1	C10	CAP 10000PF 16V CERAMIC X7R 0402	Yageo America	04022R103K7B20D
8	10	C11,C13,C14,C15,C16,C21,C22,C26,C27,C28	CAP .10UF 16V CERAMIC X7R 0603	Kemet	C0603C104J4RACTU
9	2	C12,C29	CAP 10UF 16V CERAMIC X5R 1210	Panasonic - ECG	ECJ-4YB1C106K
10	1	C17	CAP .33UF 16V CERAMIC X7R 0805	Panasonic - ECG	ECJ-2YB1C334K
11	2	C20,C23	CAP .1UF 50V CERAMIC X7R 0805	Panasonic - ECG	ECJ-2YB1H104K
12	1	C24	CAP 33nF 50V CERAMIC X8R 0603	TDK Corporation	C1608X8R1H333K
13	1	D1	LED GREEN CLEAR 1206 SMD	Chicago Miniature Lamp, Inc	CMD15-21VGC/TR8
14	1	D2	DIODE SCHOTTKY 40V 1.5A SMA	Vishay IR	10MQ040NTRPBF
15	4	D3,D4,D5,D6	LED HI EFF RED CLEAR 1206 SMD	Chicago Miniature Lamp, Inc	CMD15-21VRC/TR8
16	1	D7	DIODE ZENER 3.6V 500MW SOD123	ON Semiconductor	MMSZ4685T1G
17	1	F1	FUSE RESETTABLE .10A 30V HLD SMD	Bourns	MF-USMF010-2
18	1	J1	CONN JACK POWER 2.1mm PCB RA	CUI	PJ-102A
19	4	J2,J3,J4,J5	CONN HEADER 6POS .100 STR 15AU	FCI	67996-206HLF
20	4	J6,J7,J8,J12	CONN HEADR BRKWAY .100 03POS STR	Tyco Electronics/Amp	9-146280-0-03
21	1	J9	CONN USB MINI B SMT RIGHT ANGLE	Tyco	1734035-2
22	3	J10,J11,J14	CONN HEADR BRKWAY .100 02POS STR	Tyco Electronics	9-146280-0-02
23	4	P1,P2,P3,P4	CONN FMALE 32POS DL .050 SMT GOLD	Samtec	RSM-116-02-S-D-LC
24	3	P5,P7,P8	CONN FMALE 40POS DL .100 R/A GOLD	Sullins Electronics Corp.	PPPC202LJBN-RC
25	1	P6	CONN FMALE 46POS DL .100 R/A GOLD	Sullins Electronics Corp.	PPPC232LJBN-RC
26	1	P9	CONN RECT 6POS .100 VERT	3M	929850-01-06-RA
27	1	P11	SOLDERLESS BREADBOARD 1.8x1.35	3M	923273-I
28	1	P12	CONN RECT 8POS .100 VERT	3M	929850-01-08-RA
29	1	P14	CONN RECT 12POS .100 VERT	3M	929850-01-12-RA
30	1	P15	CONN D-SUB RCPT R/A 9POS 30GOLD	AMP Division of TYCO	5747844-4
31	1	P16	CONN RECEPT 4POS .100 VERT GOLD	3M	929850-01-04-RA
32	1	P17	CONN RECEPT 12POS 2mm SMD TIN	Hirose Electric Co. LTD.	DF11Z-12DS-2V(20)
33	1	P18	CONN REC .100 14POS for LCM-S01602DSR/A	3M	929850-01-14-RA

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
34	1	P19	CONN RECT 17POS .100 VERT	3M	929850-01-17-RA
35	10	R1,R21,R22, R23,R24,R25,R26,R28,R29,R30	RES 1.0K OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ102V
36	1	R10	RES 120 OHM 1/10W 5% 0603 SMD	Panasonic-ECG	ERJ-3GEYJ121V
37	1	R11	TRIMPOT 500 OHM 6mm SQ SMD	Bourns Inc.	3361P-1-501GLF
38	1	R14	RES 100K OHM 1/16W 5% 0402 SMD	Panasonic - ECG	ERJ-2GEJ104X
39	1	R15	RES 200 OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ201V
40	9	R16,R18,R41,R42,R43,R44,R45, R46,R47	RES ZERO OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V
41	2	R17,R19	RES 100 OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ101V
42	1	R20	10K OHM THUMBWH CERM ST POT	BI Technologies	91AR10KLF
43	1	R31	POT 10K CARBON LAYDOWN (103)	Panasonic - ECG	EVN-D8AA03B14
44	7	R32,R33,R34,R35,R36,R37,R38	RES 6.04K OHM 1/10W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF6041V
45	1	R39	RES 200K OHM 1/10W 5% 0603 SMD	Panasonic-ECG	ERJ-3GEYJ204V
46	1	R48	RES 100K OHM 1/10W 1% 0603 SMD	Yageo	RC0603FR-07100KL
47	3	SW1,SW2,SW4	SWITCH TACT 6mm MOM 150GF	Omron	B3F-1022
48	1	SW3	SWITCH SLIDE MINI SPDT PCMNT SLV	C&K Components	1101M2S3CQE2
49	5	TP1,TP2,TP3,TP4,TP14	TEST POINT 43 HOLE 65 PLATED RED	Keystone Electronics	5000
50	4	TP18,TP19,TP56,TP58	TEST POINT 43 HOLE 65 PLATED WHITE	Keystone Electronics	5002
51	9	TP30,TP32,TP33,TP34,TP35,TP36,TP37,TP38,TP39	TEST POINT 43 HOLE 65 PLATED BLACK	Keystone Electronics	5001
52	1	TVS1	TVS 5.0 VOLT 350 WATT SOD-323	Semtech	SD05.TCT
53	1	U2	IC REG LDO 1.0A 5.0V TO-252	Diodes Inc	AP1117D50L-13
54	1	U4	IC REG LDO 0.3A ADJ 8MSOP	National Semiconductor	LP3982IMM-ADJ/NOPB
55	1	U5	IC SINGLE USB PORT TVS SOT-23-6	Texas Instruments	SN65220DBV
56	1	U6	IC LINE DRVR/RCVR RS-232 16-SOIC	Texas Insturments	SN65C3232ED
57	1	U7	IC XLATR 8BIT LV 20-TSSOP	NXP Semiconductors	GTL2003PW
58	1	U8	IC SOCKET 8PIN MS TIN/TIN .300	Mill-Max Manufacturing	110-44-308-41-001000
59	1	U11	IC REG LDO 300mA 3.3 V SOT89R	Diodes Inc	AP130-33YRL-13
60	1	NA	5V LCD Module 16POS w/14 pin header installed	Lumex	LCM-S01602DSR/A
61	5	NA	BUMPER WHITE .500X.23 SQUARE	Richco Plastics Co.	RBS-3R
62	11	NA	SHUNT GOLD W/HANDLE, BLACK	Kobiconn	151-8030-E
No Load Components					
63	2	R27,R49	RES NO LOAD 0603 SMD	NA	NA
64	1	R40	RES NO LOAD 0805 SMD	NA	NA

A.3.2 CY8C28 Family Processor Module

Item	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 Electronic 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,R7	RES 56 OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ560V
10	5	R2,R4,R6,R8,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 Load Components					
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 Corporation	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

Item	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,C6	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 Electronic 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,R7	RES 56 OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEYJ560V
8	5	R2,R4,R6,R8,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 Load Components					
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	Qty	Reference	Description	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	PCB	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,C5,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,C17,C21,C22,C23,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 Electronic 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,R36,R38,R39,R40,R41,R42	RES ZERO OHM 1/16W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V
20	6	TP1,TP3,TP4,TP5,TP6,TP7	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
Do Not Install					
26	6	R6,R7,R14,R15,R16,R17	RES NO LOAD 0805 SMD	NA	NA
27	5	R10,R34,R35,R37,R43	RES NO LOAD 0603 SMD	NA	NA

A.3.5 CY8C55 Family Processor Module

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
1	1	N/A	Schematic	N/A	REF-15051
2	1	N/A	Assembly Drawing	N/A	121R-54600 REV **
3	1	N/A	Fab Drawing	N/A	N/A
4	1	N/A	Assembly Adhesive Label	N/A	#REF!
5	1	N/A	PCB	N/A	#REF!
6	2	C1,C14	CAP, CER, 10 uF, 16 V, 5%, X5R, 0805, SMD	Murata Electronics North America	GRM21BR61C106KE15L
7	7	C2,C3,C4,C5,C6,C7,C28	CAP, CER, 1.0 uF, 10 V, 10%, X5R, 0603, SMD	Kemet	C0603C105K8PACTU
8	2	C8,C11	CAP, CER, 1.0 uF, 10 V, 10%, X5R, 0805, SMD	Murata	GRM219R61A105KC01D
9	2	C9,C25	CAP, CER, 0.1 uF, 16 V, 5%, X7R, 0603, SMD	Kemet	C0603C104J4RACTU
10	2	C10,C12	CAP, CER, 22 uF, 10 V, 10%, X5R, 1210, SMD	Kemet	C1210C226K8PACTU
11	8	C15,C16,C17,C21,C22,C23,C24,C29	CAP, CER, 0.1 uF, 16 V, 10%, X7R, 0402, SMD	Kemet	C0402C104K4RACTU
12	1	C18	CAP, CER, 2200 pF, 50V, 5%, COG, 0603, SMD	Murata	GRM1885C1H222JA01D
13	2	C19,C20	CAP, CER, 22 pF, 50V, 5%, COG, 0603, SMD	Panasonic - ECG	ECJ-1VC1H220J
14	1	D1	DIODE, SCHOTTKY, 40 V, 1 A, ZHCS1000TA, SOT-23, SMD	Zetex	ZHCS1000TA
15	4	J1,J2,J3,J4	CONN, HDR, 2x16, 0.05", GOLD, TH	Centronic Precision Electronic Co.	HHLHS32GB1
16	2	J5,J6	CONN, HDR, KEYED, 2x5, 0.050", Gold, SMD	Digilent	161-026
17	1	L1	IND, FIXED, 22 uH, 20%, .960 A, 7032, SMD	TDK Corporation	SLF7032T-220MR96-2-PF
18	2	R8,R9	RES 22 OHM 1/16W 1% 0603 SMD	Panasonic - ECG	ERJ-3EKF22R0V
19	13	R31,R32,R36,R38,R39,R40,R41,R42,R45,R46,R47,R48,R49	RES ZERO OHM 1/10W 5% 0603 SMD	Panasonic - ECG	ERJ-3GEY0R00V
20	1	R44	RES, 0.0 Ohms, 1/8 W, 5%, 0805, SMD	Panasonic - ECG	ERJ-6GEY0R00V
21	7	TP1,TP3,TP4,TP5,TP6,TP7,TP8	CONN, TEST POINT, RED, TH	Keystone Electronics	5000
22	2	TP2,TP9	CONN, TEST POINT, BLACK, TH	Keystone Electronics	5001
23	1	U1	IC, PSoC5,CY8C5568AXI-060, TQFP-100, SMD	Cypress Semiconductor	#REF!
24	1	Y1	CRYSTAL 32.768 KHZ CYL 12.5PF CFS308	Citizen America Corporation	CFS206 32.768KDZF-UB
25	2	C26,C27	CAP, CER, 12 pF, 50V, 5%, COG, 0603, SMD	Murata Electronics North America	GRM1885C1H120JA01D
26	1	Y2	CRYSTAL, 24 MHz, 30 ppm, HC49, SMD	ECS Inc.	ECS-240-12-5PX-TR
DO NOT INSTALL					
27	6	R6,R7,R14,R15,R16,R17	RES NO LOAD 0805 SMD	NA	NA
28	5	R10,R34,R35,R37,R43	RES NO LOAD 0603 SMD	NA	NA
29	1	TP10	TEST POINT 43 HOLE 65 PLATED WHITE	NA	NA

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	TCK	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

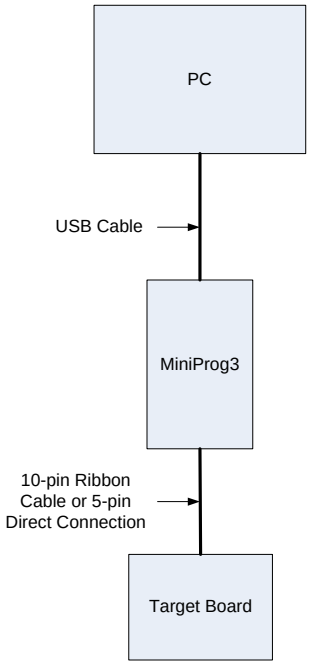


Table C-1. Connectors / Communication Protocol Support

Connector	ISSP	JTAG	SWD and SWV[a]	I^2C
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 5 families. This interface allows a daisy chain bus of multiple JTAG devices. MiniProg3 supports programming and debugging PSoC 3 and PSoC 5 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 5 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 5 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 5 firmware, using SWV, through the 10-pin connector and in conjunction with SWD only.

C.1.4 I^2C™

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 5 JTAG, SWD, SWV, and I^2C interfaces, see the PSoC 3 and PSoC 5 Technical Reference Manual. For more information on PSoC 1 interfaces, see the PSoC 1 Technical Reference Manual.

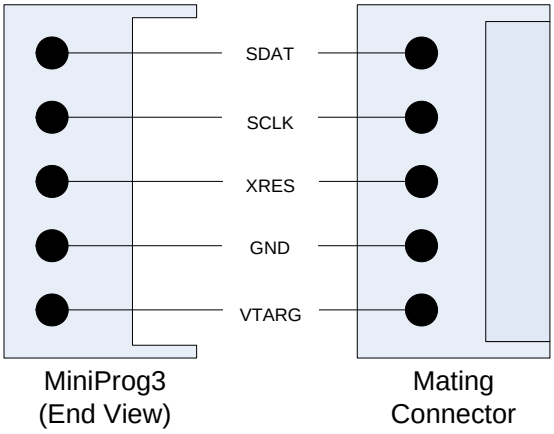
C.2 Connectors

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.

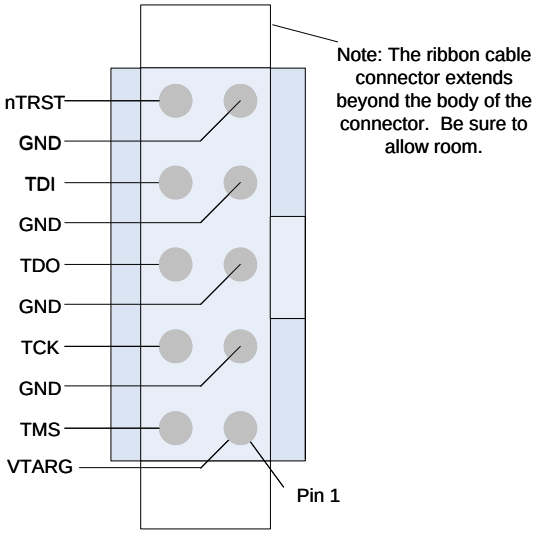
Figure C-2. 5-Pin Connector with Pin Assignments



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.

Table Appendix C-2. Communication Protocol Pin Assignments

Protocol	Signal	5-Pin	10-Pin
ISSP	SCLK	4	
	SDAT	5	
	XRES	3	
JTAG	TMS		2
	TCK		4
	TDO		6
	TDI		8
	XRES		10
SWD / SWV	SDIO	5	2
	SCK	4	4
	SWV[a]		6
	XRES	3	10
I2C	SCK	4	
	SDA	5	

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 5, 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.

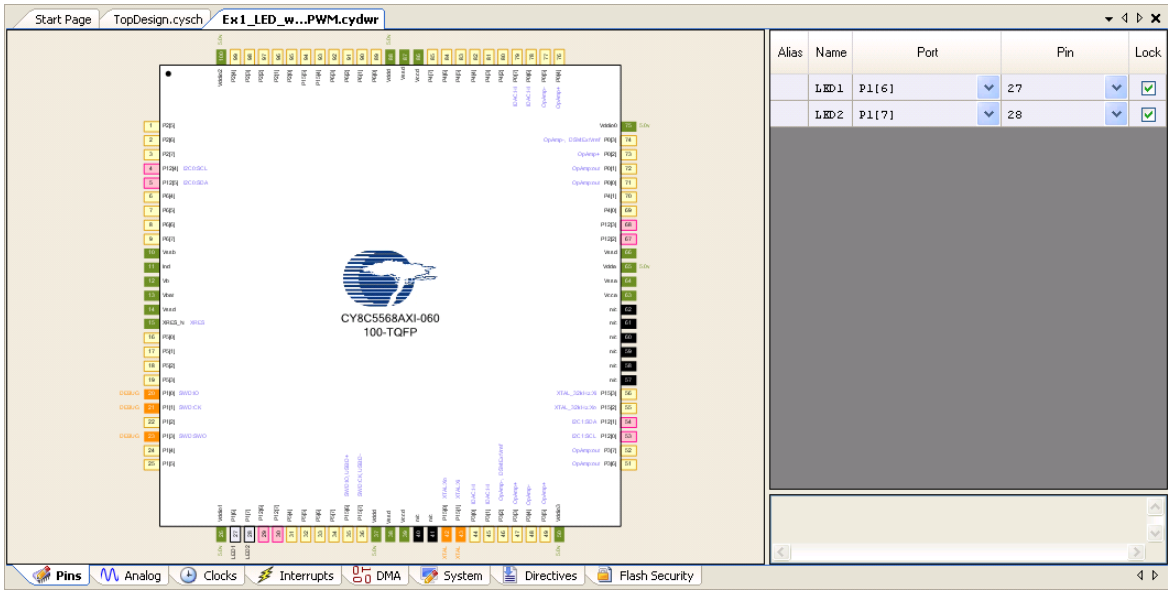
Appendix D. PSoC Creator DWR

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.

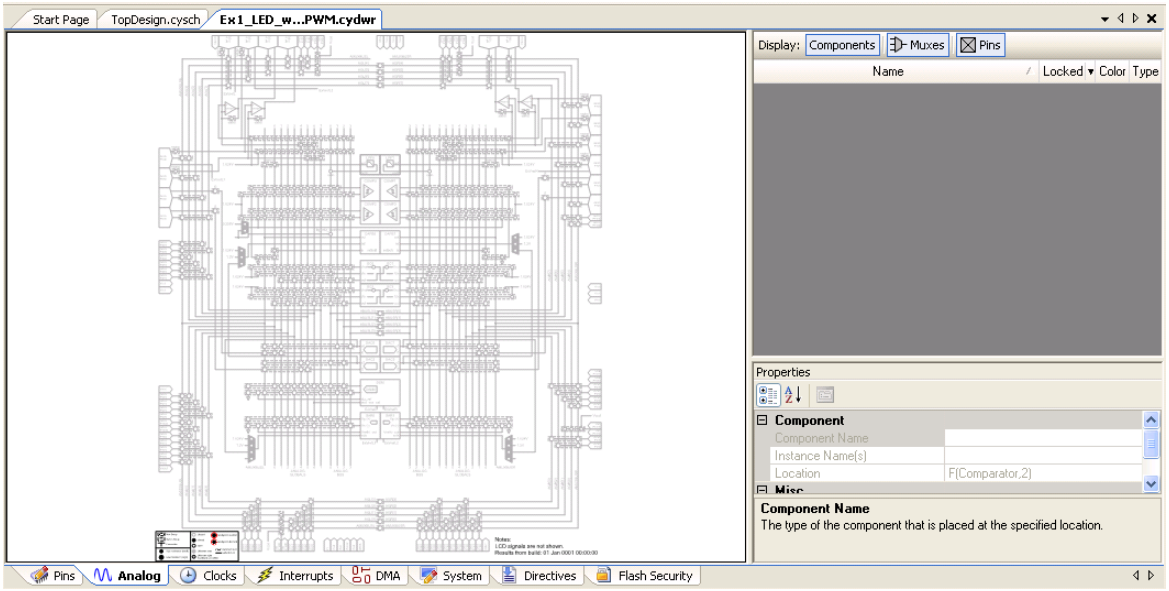
The Pins tab of the DWR file or the Pin Editor allows you to manually assign the pins used in the schematic to the PSoC.

Figure Appendix D-1. DWR File - Pin Editor



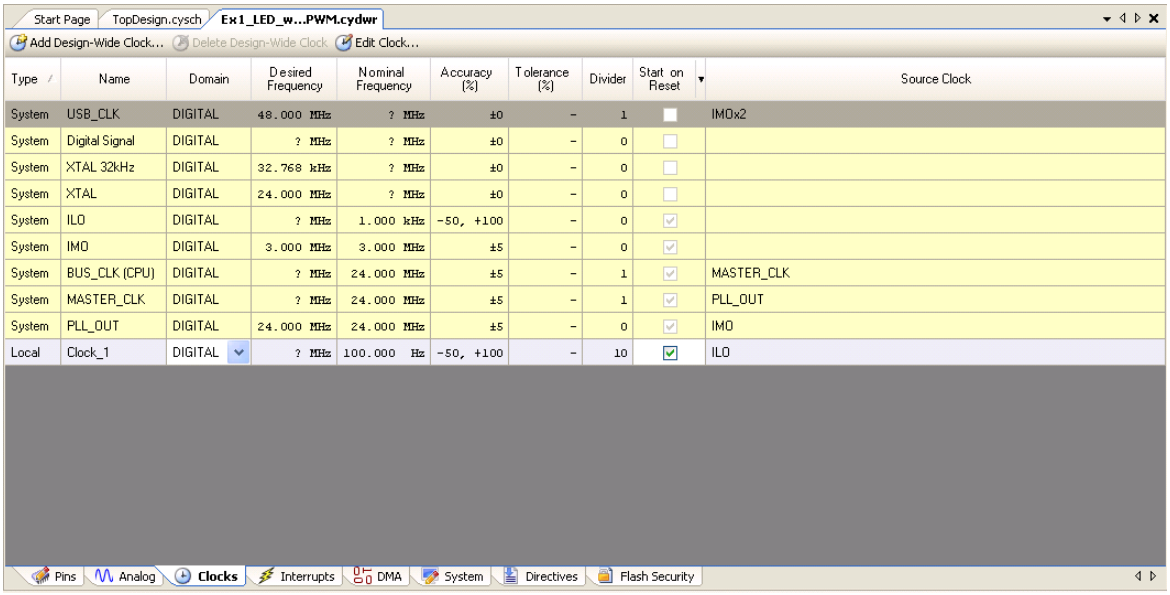
The Analog tab or the Analog Device Editor provides an interconnect view of the PSoC 3 and PSoC 5 devices along with place-and-route results for a particular design. The editor also allows for manual place-and-route with the ability to lock-down all or some of the results.

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



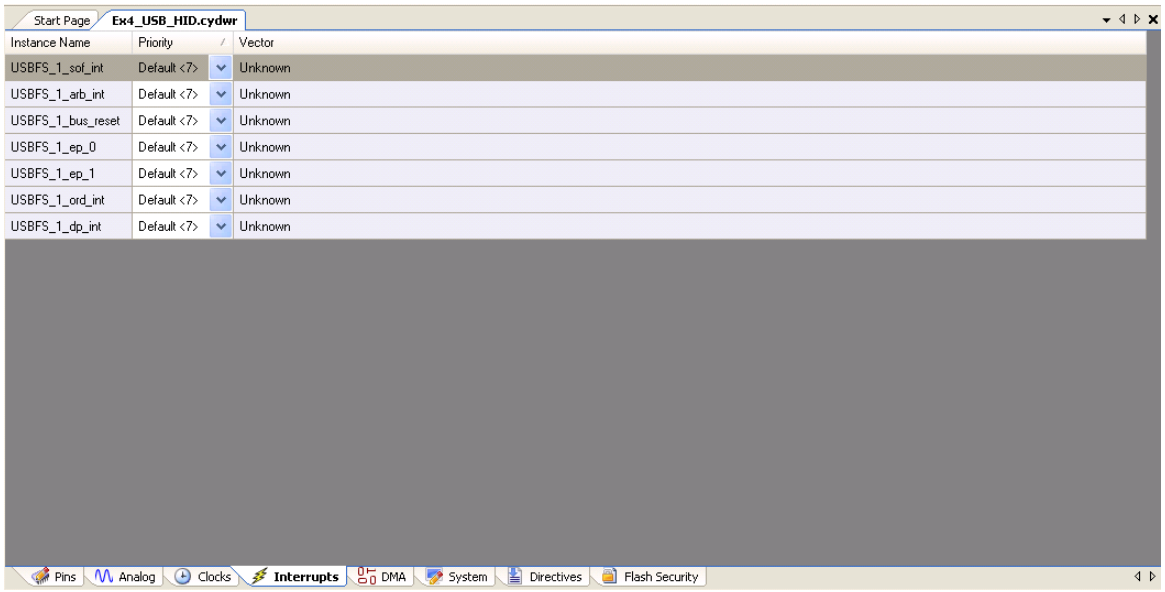
The Clocks tab or the Clock Editor is a design-wide resources tool to create and edit clocks. This tool allows you to view all clocks, add and delete design-wide clocks, as well as edit design-wide and system clocks.

Figure Appendix D-3. DWR File - Clock Editor



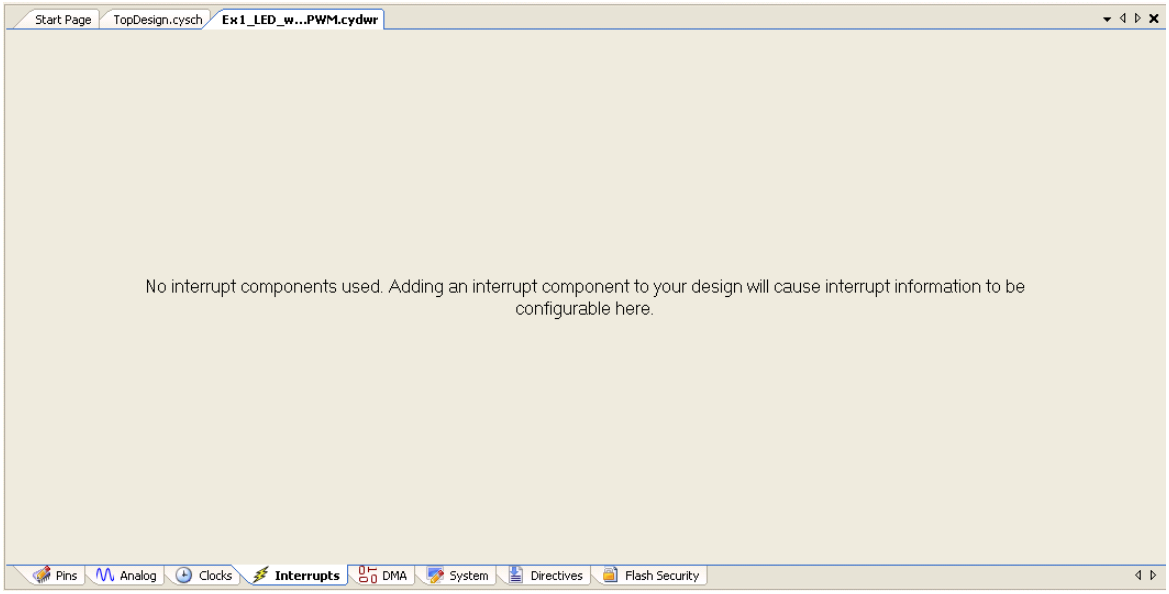
The Interrupts tab or the Interrupt Editor allows you to change the priority of interrupt service routines (ISRs) in your design.

Figure Appendix D-4. DWR File - Interrupt Editor



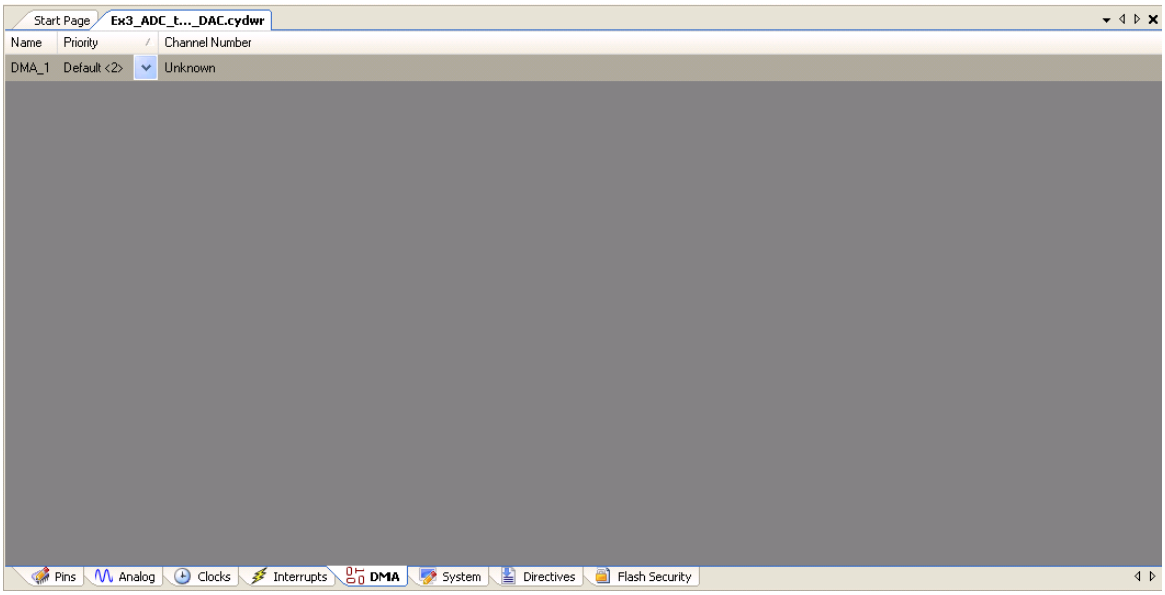
Note If no interrupts are used in your design, the Interrupt Editor gives the message, as shown in the following figure.

Figure Appendix D-5. Message for No Interrupts



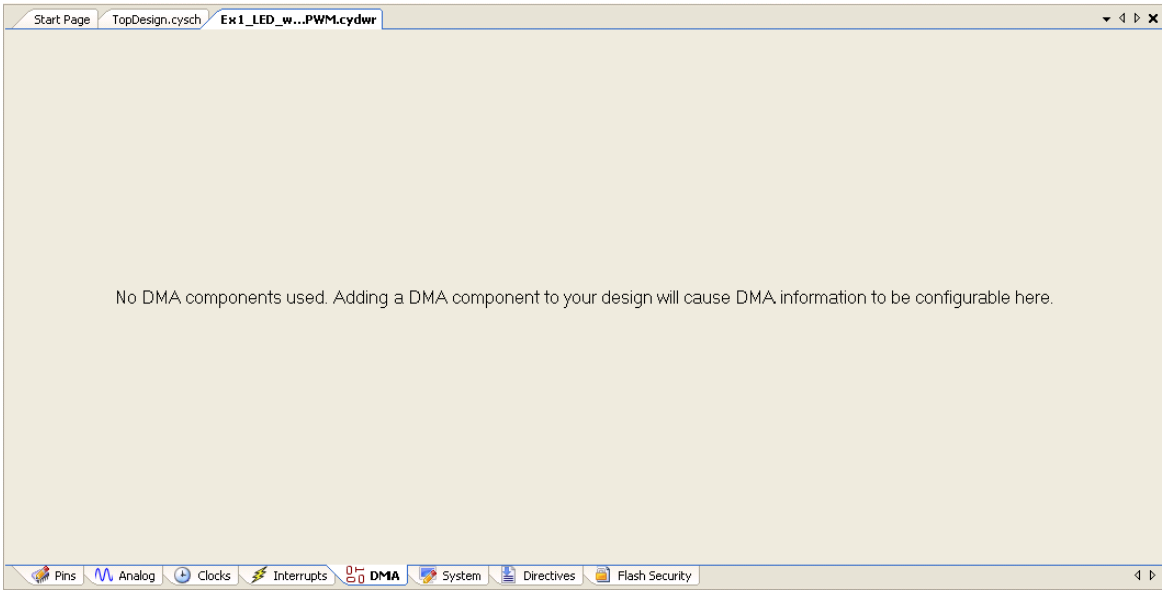
The DMA tab or the DMA Editor displays all the direct memory access (DMA) components that have been directly placed in the design, as well as all the DMA components "inside" placed components.

Figure Appendix D-6. DWR File - DMA Editor



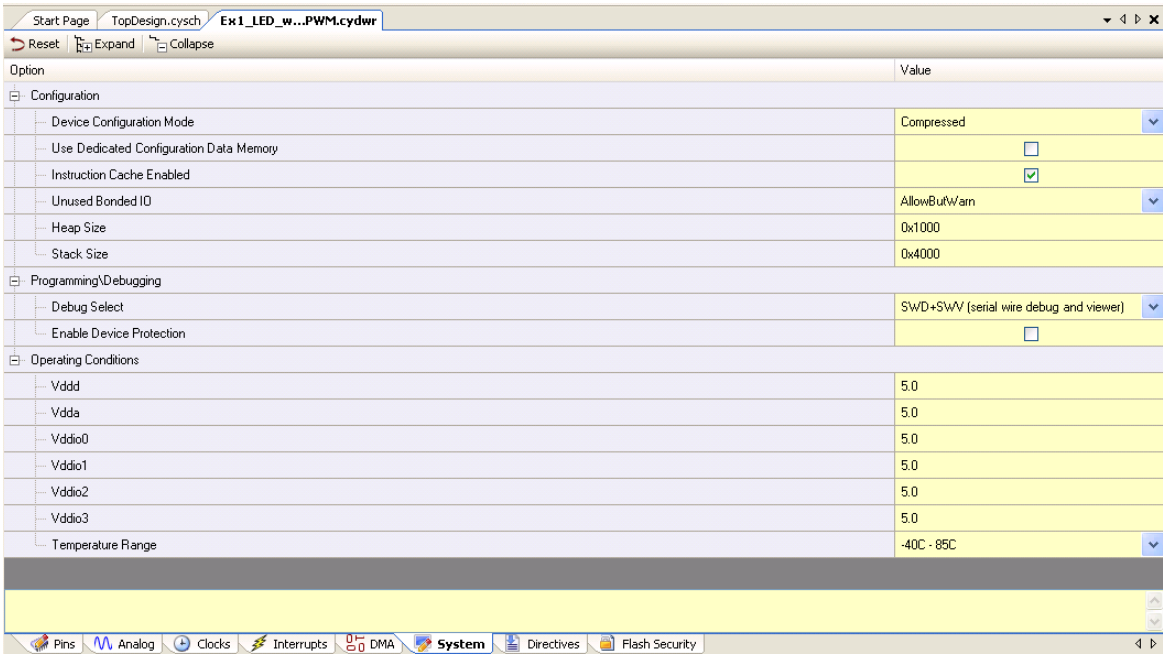
Similar to the Interrupt editor, if there is no DMA component used in the design DMA editor shows the message that there is no DMA component being used.

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.

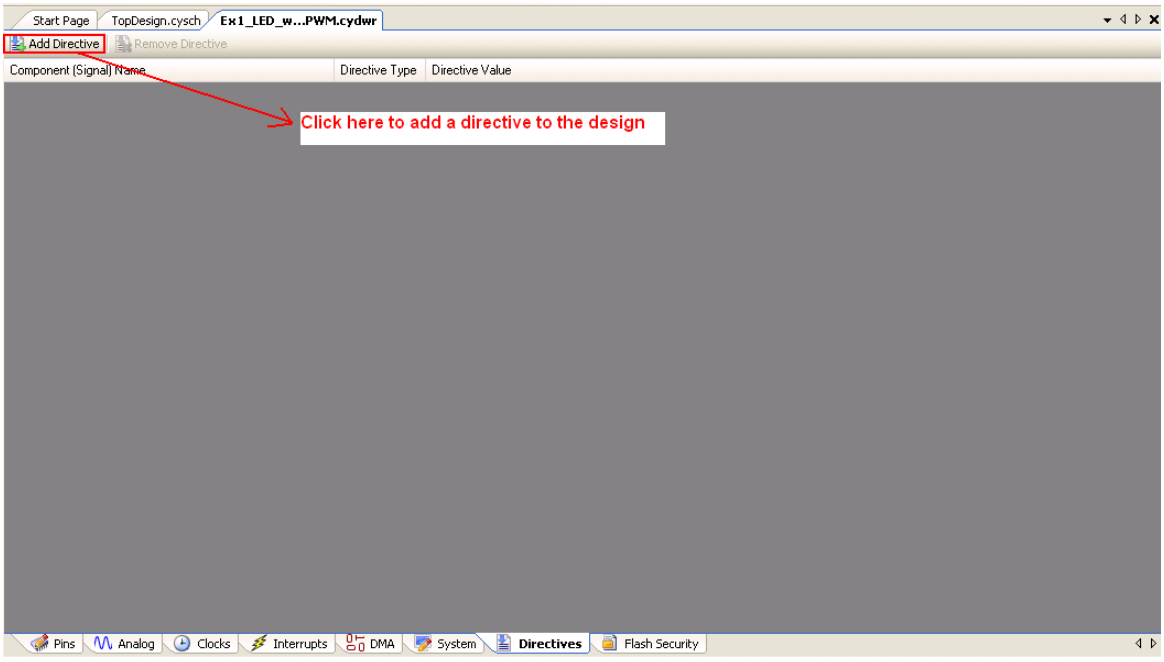
Figure Appendix D-8. DWR File - System Editor



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.

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