# CY8CKIT-020

# PSoC® Development Kit Guide

Doc. # 001-56971 Rev. **
February 1, 2010

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 Data Sheets. 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

## 1.1 Kit Overview

This guide and the CY8C28 Family Processor Module gives you a practical understanding of PSoC® technology. The kit gives several example projects with step-by-step instructions to enable you to easily get started developing PSoC solutions.

To accompany the CY8CKIT-020 kit, you are required to have the CY8CKIT-001 PSoC Development Kit which provides you a common development platform where you can prototype and evaluate different solutions using any one of the PSoC 1, PSoC 3, or PSoC 5 architectures.

## 1.2 Kit Contents

The CY8CKIT-020 PSoC Development Kit includes:

- PSoC CY8C28 Family Processor Module
- Printed Documentation
  - Quick Start
  - Schematic PSoC Development Board Design
- Software CD for PSoC 1, which includes
  - PSoC® Designer™ IDE
  - PSoC® Programmer™ Software
  - CY8C28 Data Sheets
  - Example Project Files, Firmware, and Documentation

## 1.3 Installation

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

Proceed with following the PSoC Development Kit Quick Start Guide. This guide instructs you on how to install the CY8C28 Family Processor Module with the CY8CKIT-001 PSoC Development Kit. The Quick Start Guide also instructs how to install the Cypress software development tools.

### 1.3.1 Before You Begin

All Cypress software installations require administrator privileges but 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 Designer uses the 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 it for you. If you do not have Adobe Acrobat Reader, you have to manually download from Adobe website and install it.

### 1.3.3 Installing PSoC 1 Development Software

To use the CY8C28 Family Processor Module (PSoC 1) you need:

- PSoC Designer 5.0 SP6 or higher
- PSoC Programmer 3.06 or higher

If PSoC Designer 5.0 is currently installed, uninstall it. Click **Start**, click **Control Panel**, and then double click **Add or Remove Programs**.

PSoC Designer 5.0 SP6 is not presently compatible with Microsoft Internet Explorer 8 beta (any beta version). If you have Internet Explorer 8 beta, uninstall it and replace it with the released version of Microsoft Internet Explorer 8.

Insert the CY8C28 Family Processor Module Software CD and, using the menu, select **Install Software for PSoC 1**.

After installation, guides and key documents are located in the `\Documentation` subdirectory of the PSoC Designer installation directory. The default location is:
`C:\Program Files\Cypress\PSoC Designer 5\Documentation`

## 1.4 PSoC Development Board

Sold seperately, 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:

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

This full featured board incorporates three onboard linear regulators that power peripherals and PSoC modules at voltages between 1.7V and 5.5V. These regulators include a fixed 5V 1A linear regulator, a fixed 3.3V 300 mA linear regulator, and a 1.5V to 5.5V 300 mA adjustable regulator. The board also provides the ability to separate the PSoC core VCC rail into two separate rails, analog and digital. In addition, the board is able to separate the I/O VCC rails, giving the flexibility to power the I/O ports at different voltages.

The board is equipped with a 2x16 alphanumeric LCD module capable of 1.8V to 5.5V I/O. In addition, there is a mini-B full speed USB interface and a female DB9 serial communications interface. Also included is a x-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 GPIO expansion slots, allowing the I/O to expand to external boards.

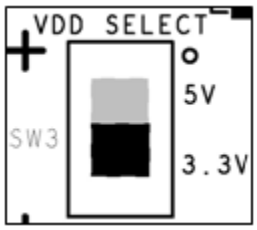
The board was designed with modularity in mind and, as a result, supports removable processor modules. This allows you to plug different PSoC modules into the board based upon the desired features of both 8-bit and 32-bit PSoC devices.

### 1.4.1 Default Switch and Jumper Settings

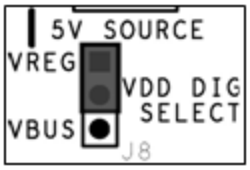
Jumpers on the CY8CKIT-001 PSoC Development Main Board have a default setting for 3.3V operation. For Default configuration, each of the jumpers must be set according to these instructions.

**Note:** All CY8C28 family processor module example projects are configured for 5V. Configure the board to 5V, before creating the example projects.

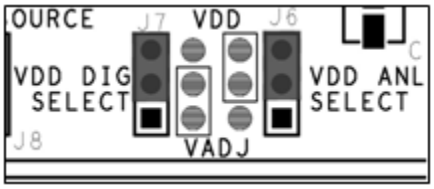
**SW3 - VDD Select.** Default Position: 3.3V (down position)



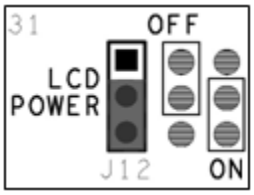

**J8 - 5V Source.** Default Position: VREG (upper two pins)



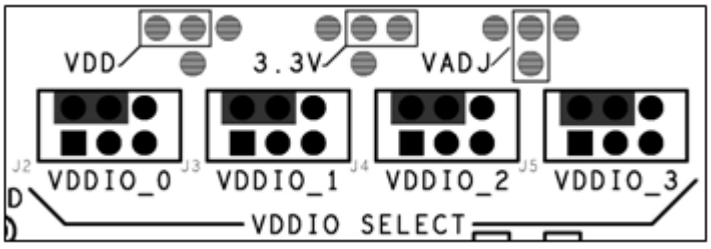

**J7, J6 - VDD Digital, VDD Analog.** Default Position: VDD (upper two pins, both headers)



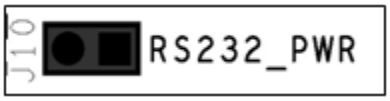

**J12 - LCD Power.** Default Position: ON (lower two pins)




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



**J10 - RS232 Power (Serial Communications).** Default Position: Installed




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


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


## 1.5 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. |

## 1.6 Document Revision History

| **Document Title: CY8CKIT-020 PSoC Development Kit Guide**<br>**Document Number: 001-56971** | | | |
|---|---|---|---|
| **Revision** | **Issue Date** | **Origin of Change** | **Description of Change** |
| ** | 11/18/09 | AESA | New Guide |

# 2. Loading My First PSoC Project

The CY8CKIT-020 PSoC Development Kit supports projects across the CY8C28 family architectures. This section walks you through the high level design process for opening, building, programming, and running your first PSoC projects using this kit.

To begin with be sure you have the CY8CKIT-001 and the CY8CKIT-020 kit. 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 6.
2. 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 were 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.
3. Configure the PSoC Development Board (jumper settings and switches) in its default configuration, as described in Default Switch and Jumper Settings on page 7.
4. Use the PSoC CY8C28 Family Processor Module for the PSoC 1 version of your first PSoC project (My First PSoC 1 Project on page 9).
5. For a PSoC 1 project, use the ISSP header on the PSoC CY8C28 Family Processor Module and connect the MiniProg3 ISSP port.
6. Power the PSoC Development Board using the 12V AC Power Adapter

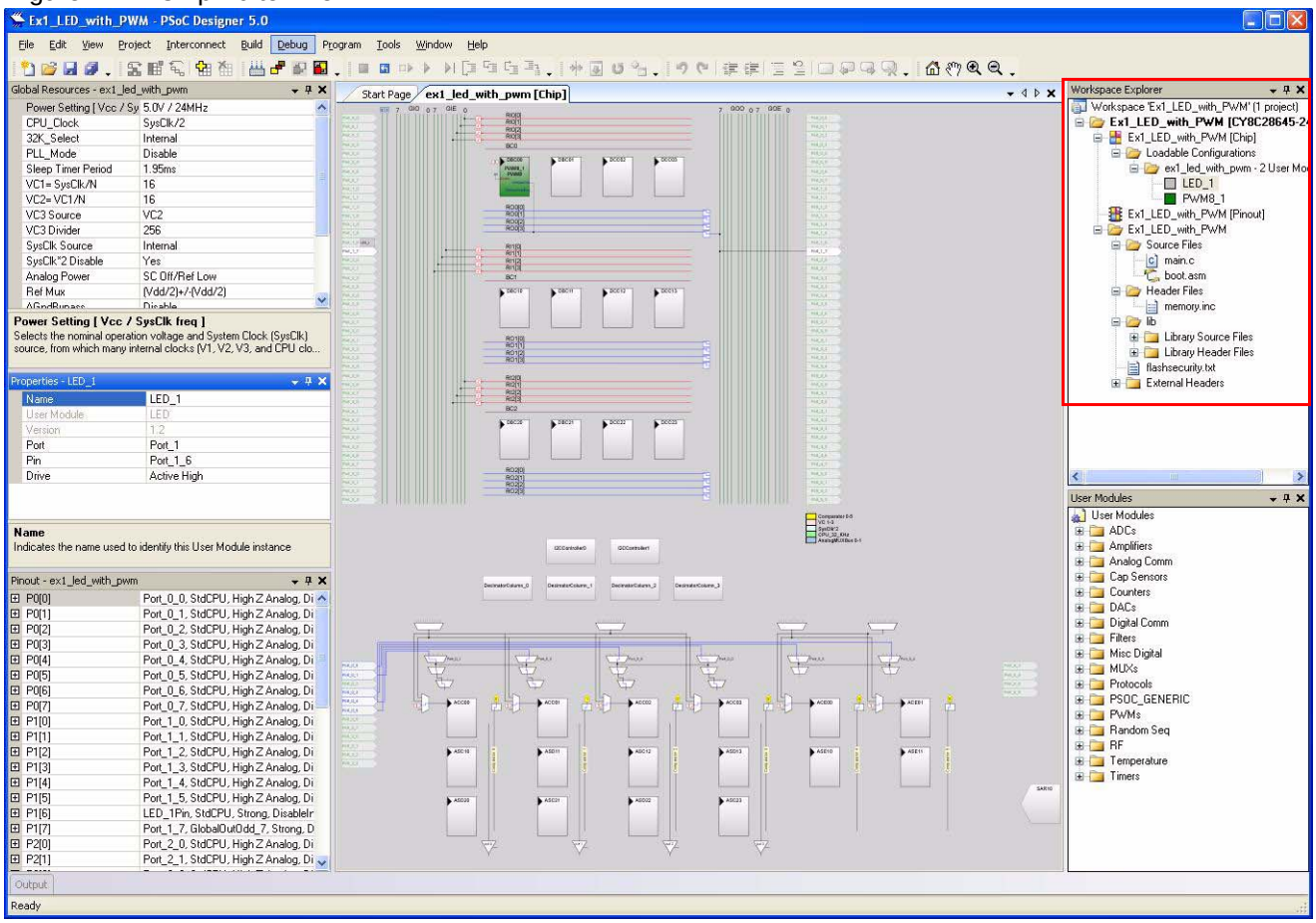
## 2.1 My First PSoC 1 Project

This is a simple PSoC 1 project using a 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 high level steps of opening, building, and programming 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 `C:\Cypress\CY8CKIT-001\CY8C28 Projects\`
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 the project files are in the **Workspace Explorer**.

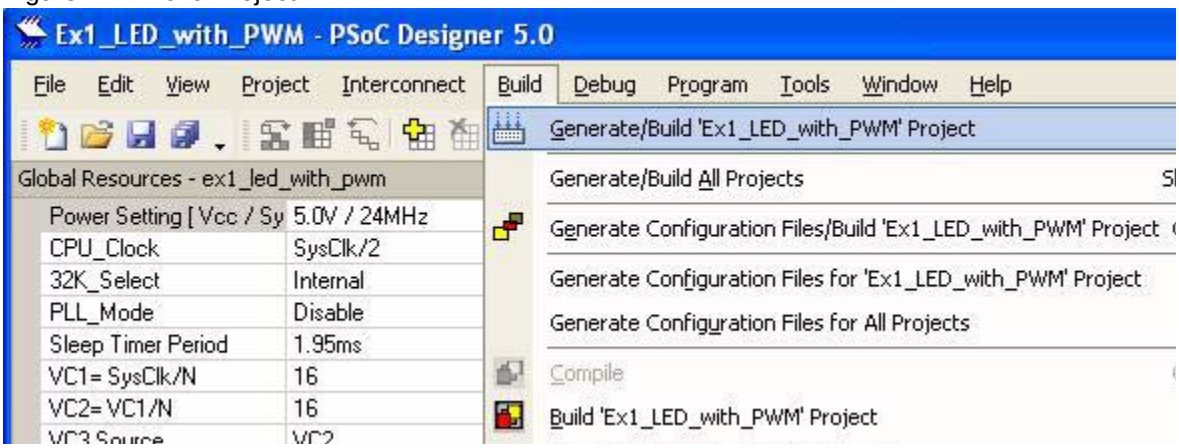
Figure 2-1. Chip Editor View




## 2.1.2 Building My First PSoC 1 Project

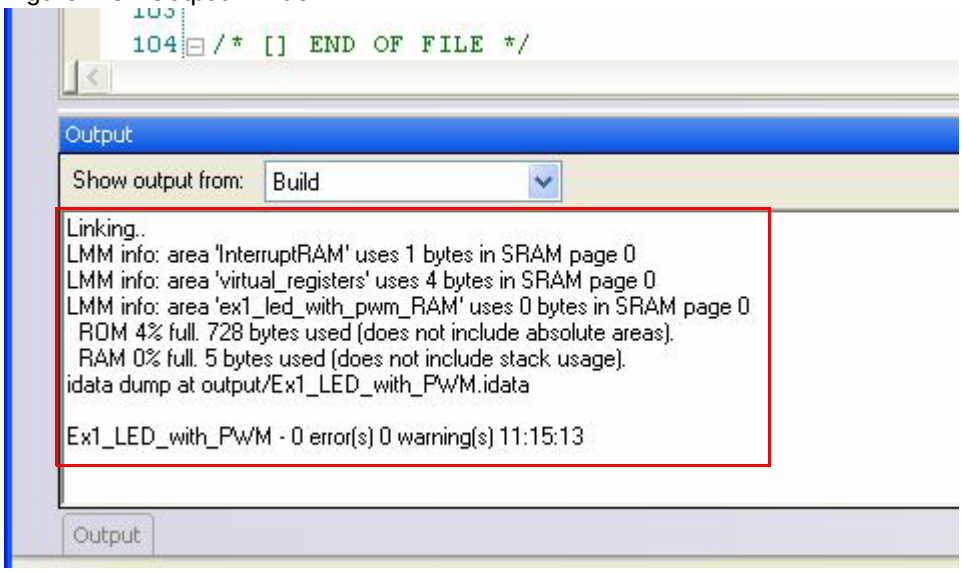
1. Select **Build** → **Generate/Build 'Ex1_LED_with_PWM' Project**.

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 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

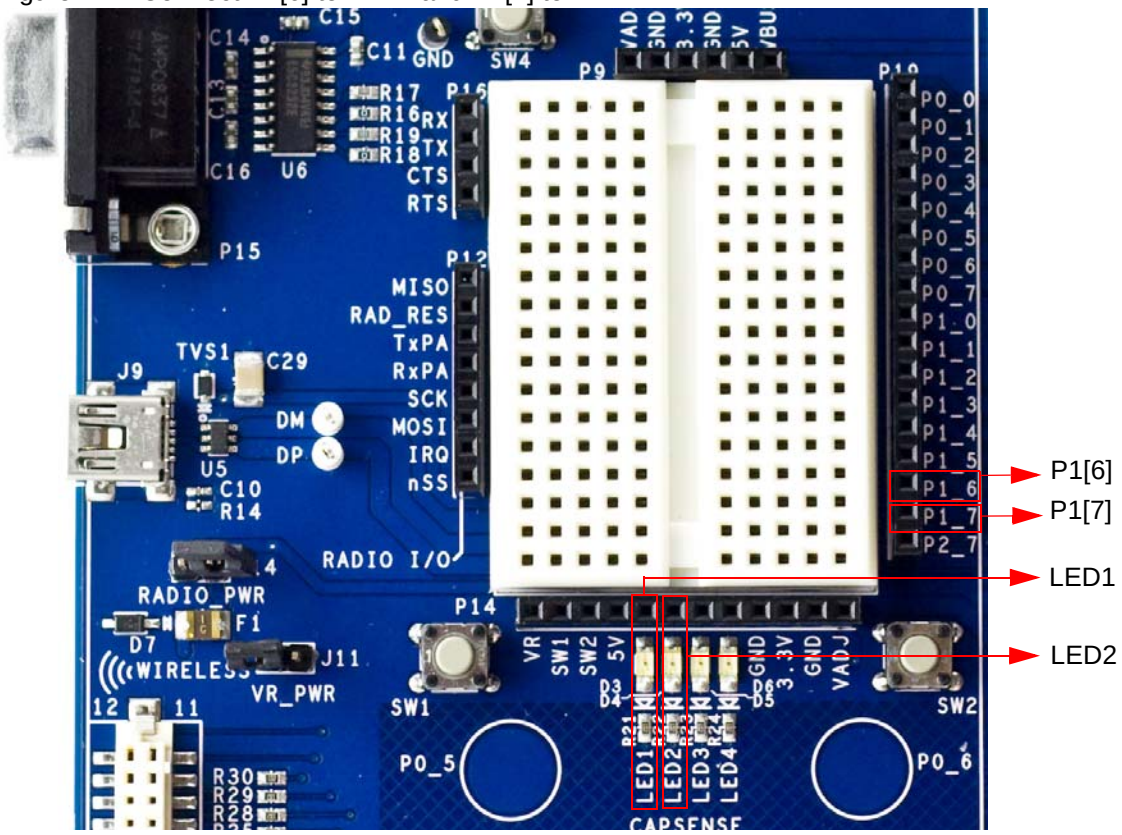
**Note** The CY8C28 Family Processor Module is designed to accommodate the use of the CY3215-DK In-Circuit Emulator (ICE Cube). When using the ICE Cube debugger, make certain that PSoC Designer is configured so that the ICE Cube DOES NOT provide power to the processor module. Within the **PSoC Designer** application, select **Project** > **Settings** and select **Debugger** from the tree list. Make sure **External only** is selected under the **Pod Power Source** section.

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 **Accquire Mode** to **Reset**.
4. In the **Program Part** window, set **Verifcation** to **On**. This ensures that downloaded checksum matches the actual checksum.
5. In the **Program Part** window, click the program arrow to program the device.
6. Wait until programming is completed to continue.

### 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 2-4. Connect P1[6] to LED1 and P1[7] to LED2




3. For more details regarding this project, see the detailed step-by-step project instructions in My First PSoC 1 Project on page 13.

# 3. Sample Projects

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

Read these precautions before you create example projects:

- All CY8C28 family processor module example projects are configured for 5V.
- Close any open project in PSoC Designer before loading or creating an example project.
- When working with example projects, use 12V wall wart power supply.
- Remove power before changing board jumpers for each example project. Reapply power after you place jumpers on the breadboard.
- When you complete each project make certain to save the project.

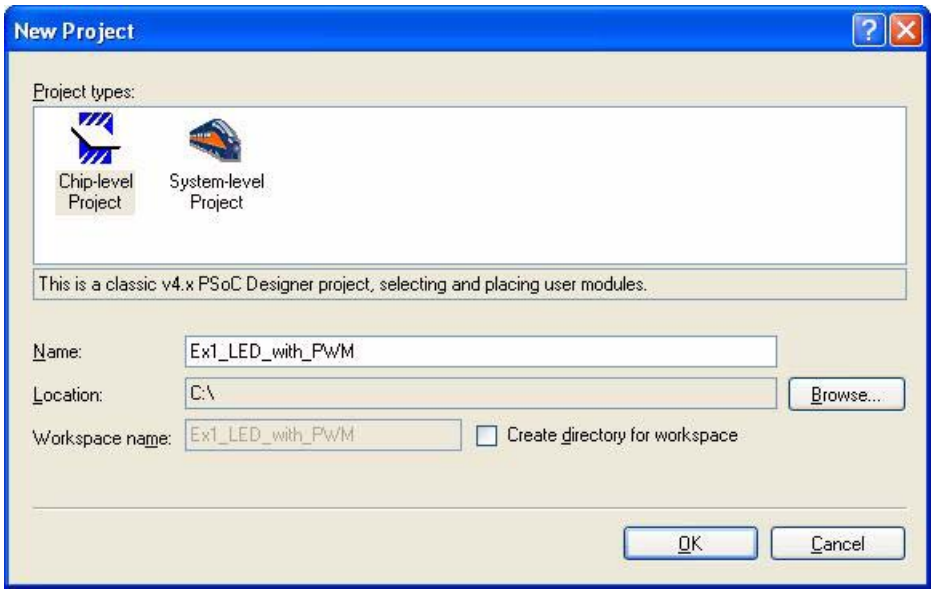
## 3.1 CY8C28 Family Processor Module Example Projects

### 3.1.1 My First PSoC 1 Project

#### *3.1.1.1 Creating My First PSoC 1 Project*

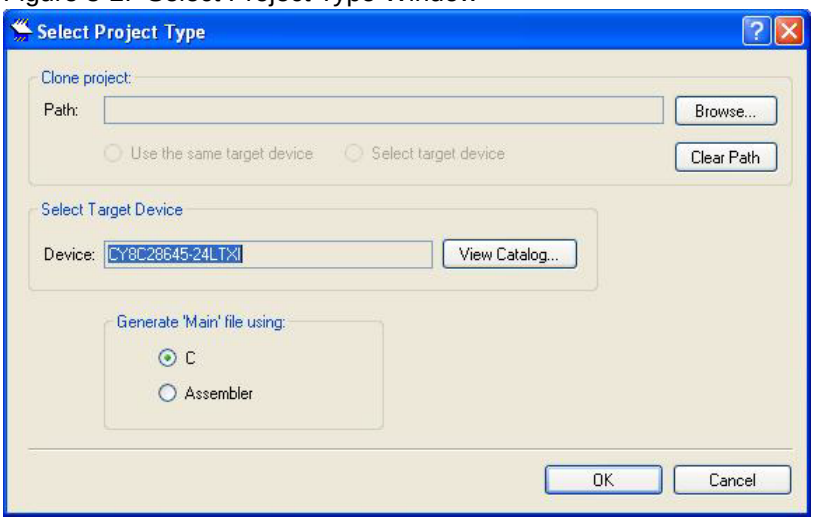
1. Open PSoC Designer 5.0
2. To create a new project, click **File** → **New Project**. The **New Project** window opens.
3. In the **New Project window**, select the **Chip-Level Project**. **Name** the project **Ex1_LED_with_PWM**.
4. In **Location**, click **Browse** and navigate to the appropriate directory.

Figure 3-1. New Project Window



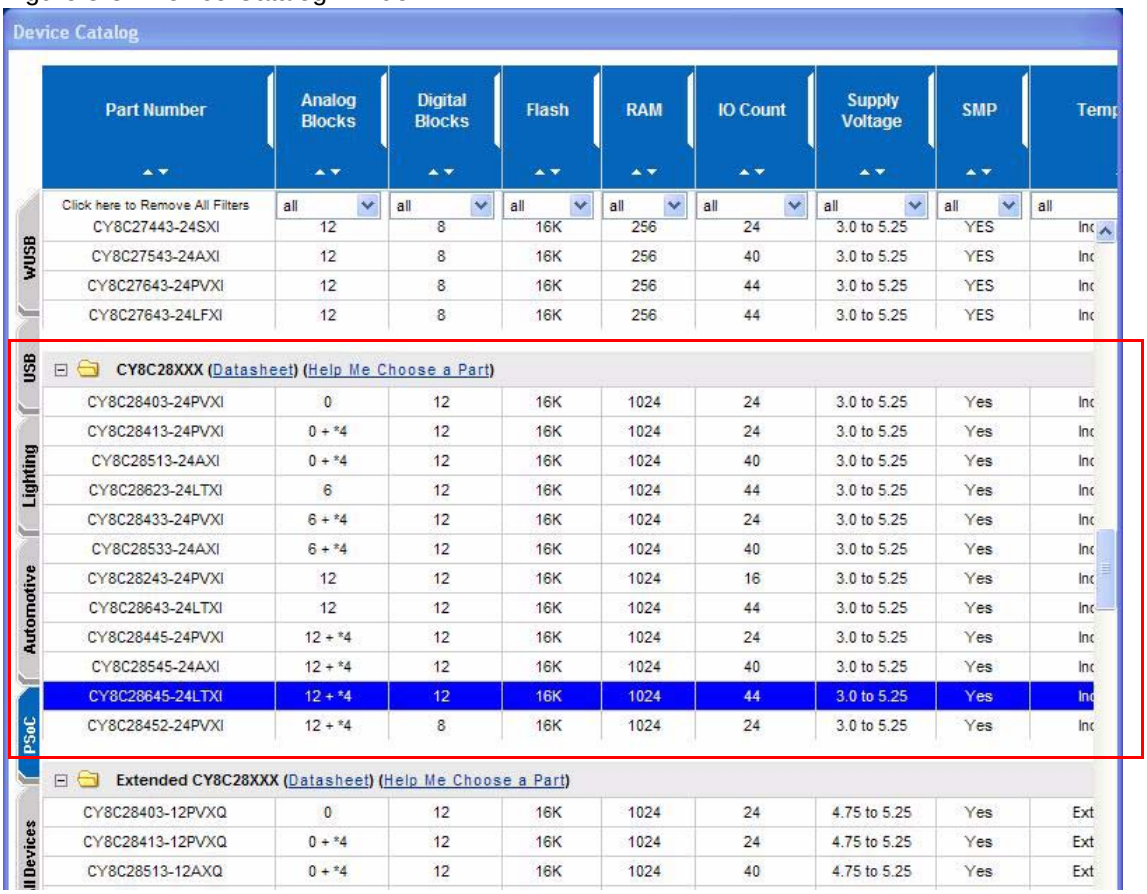
5. Click **OK**. The **Select Project Type** window opens.

Figure 3-2. Select Project Type Window



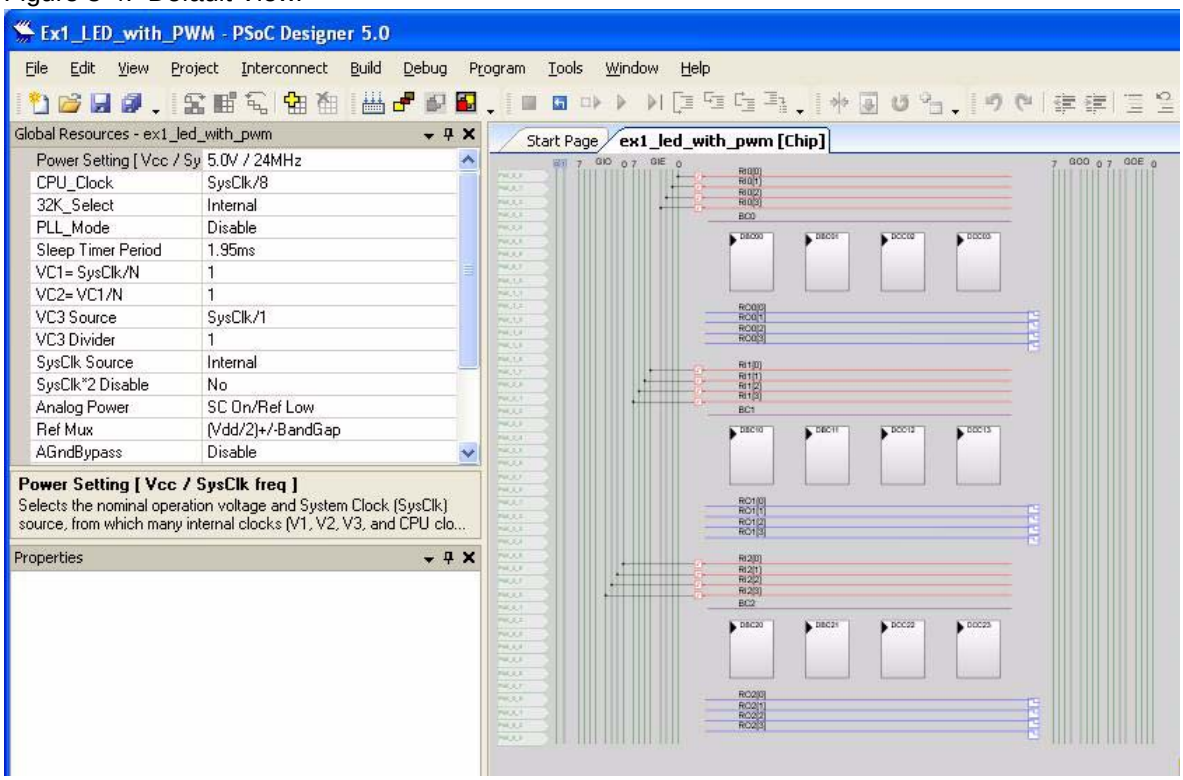

6. In this window under **Select Target Device**, click **View Catalog**.
7. The **Device Catalog** window opens. Click on the **PSoC** tab, and scroll down to the **CY8C28XXX** section.
8. For this project click CY8C28645-24LTXI device in this section and then click **Select**.

Figure 3-3. Device Catalog Window



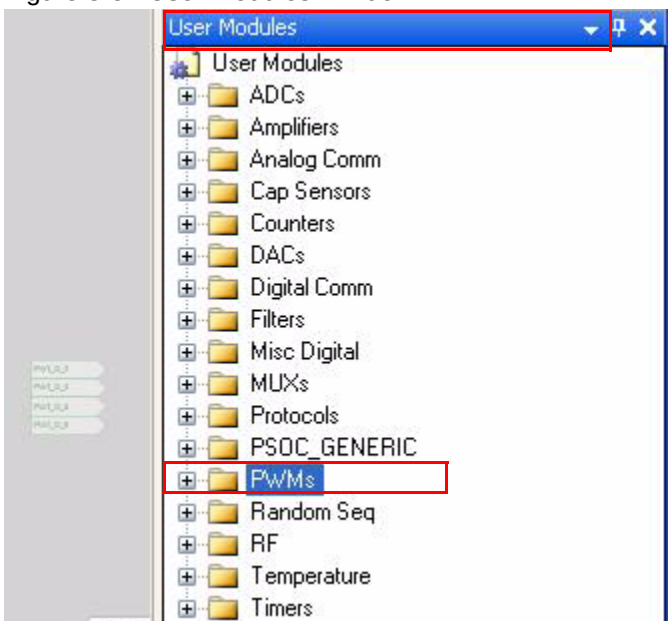
9. Under **Generate 'Main' File Using:**, for this project select **C**, then click **OK**.
10. By default, the project opens to chip view

Figure 3-4. Default View.



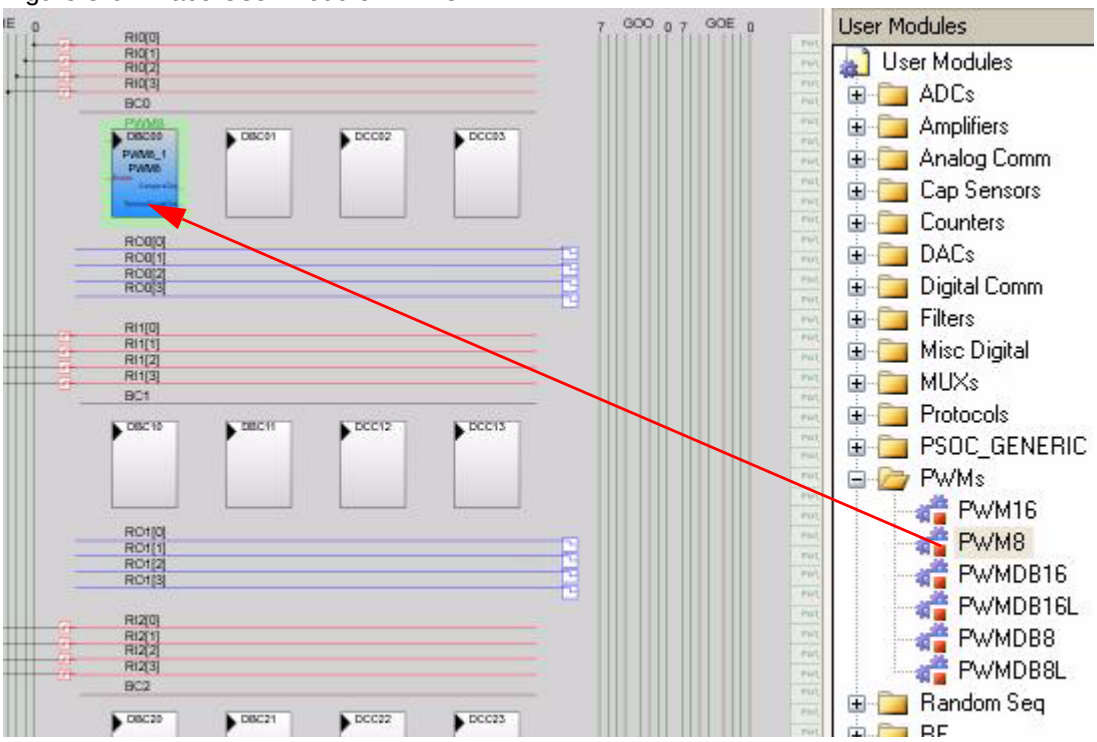

11. In the **User Modules** window, expand the **PWMs** folder.

Figure 3-5. User Modules Window




12. In this folder right click on a **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 as in the following figure. If the **Properties** window does not appear, click **View** → **Properties** Window.

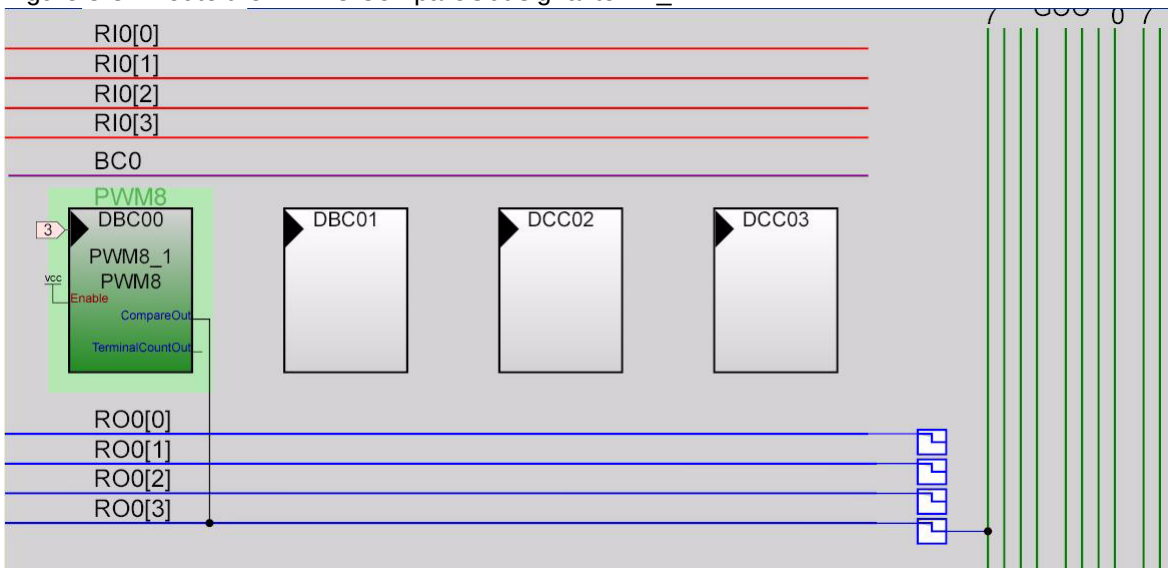
Figure 3-7. Properties Window

Properties - PWM8_1

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

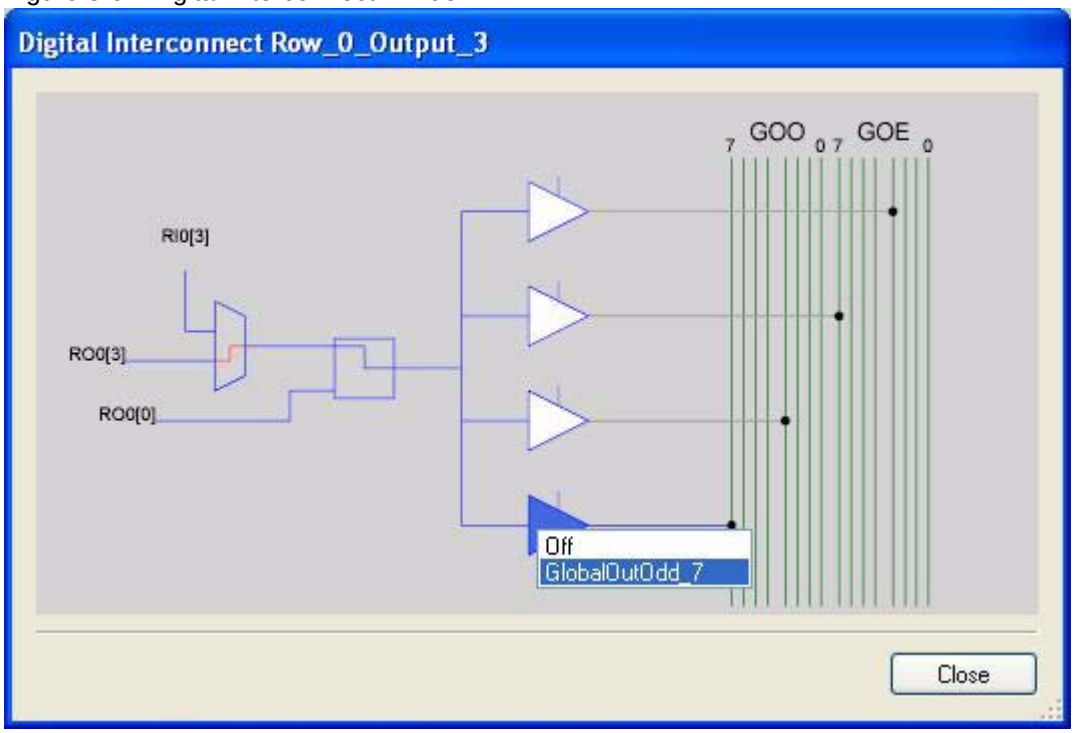
14. Next, route the PWM **CompareOut** signal to P1_7. The first step is to configure the Look Up Table (LUT) on **Row_0_Output3**.

Figure 3-8. Route the PWM8 CompareOut signal to P1_7



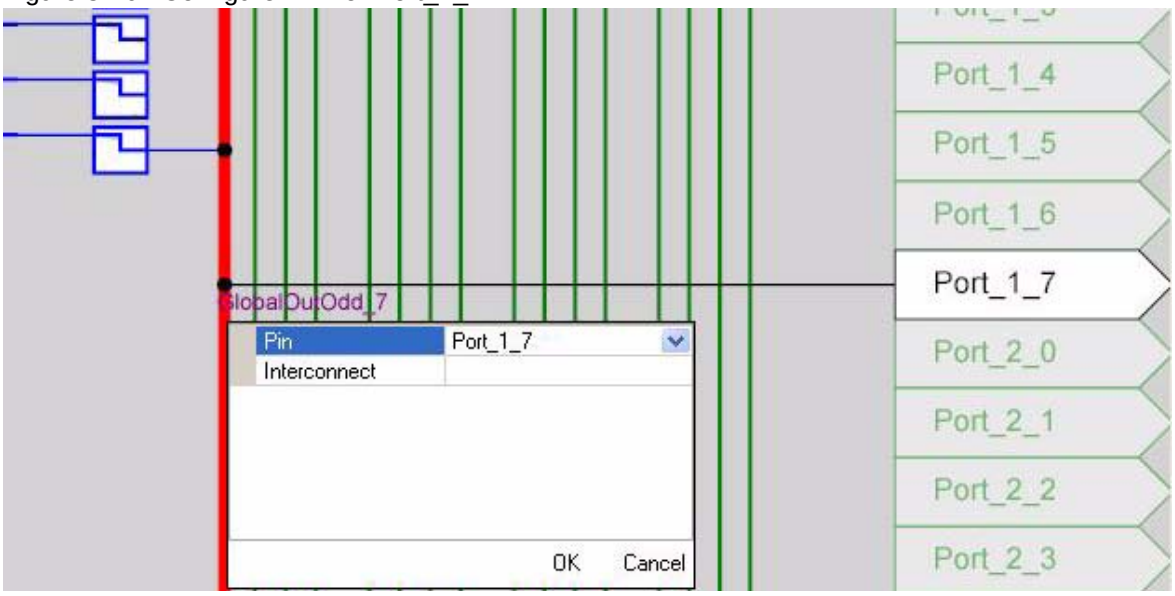

15. Double click the **LUT**, the **Digital Interconnect** window opens.
16. In this window enable **Row_0_Output_3_Drive_3** to connect to **GlobalOutOdd_7**.

Figure 3-9. Digital Interconnect Window



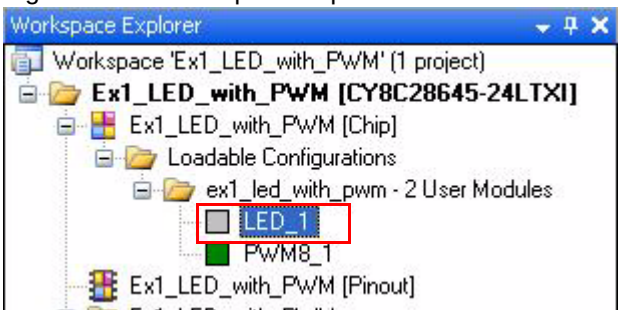

17. Click **Close**.
18. Click on **GlobalOutOdd_7**. A window appears, in this window configure **PIN** for **Port_1_7**.

Figure 3-10. Configure PIN for Port_1_7



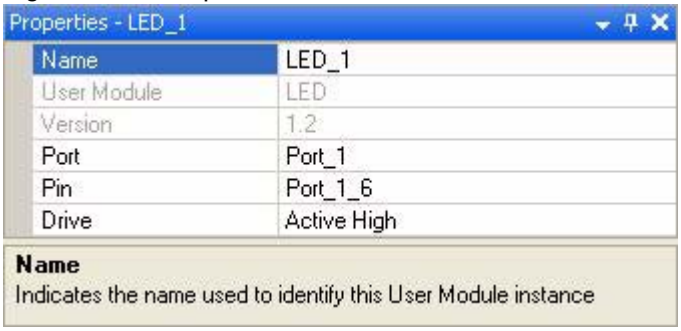

19. Click **OK** to continue.
20. In the **User Modules** window expand the **Misc Digital** folder. In this folder right click the **LED** and select place, this adds the UM to the project. This UM does not use digital or analog blocks. It appears in the **Workspace Explorer** → **Ex1_LED_with_PWM[CY8C28]** → **Ex1_LED_with_PWM[Chip]** → **Loadable Configurations** → **Ex1_LED_with_PWM - 2 User Modules**.

Figure 3-11. Workspace Explorer



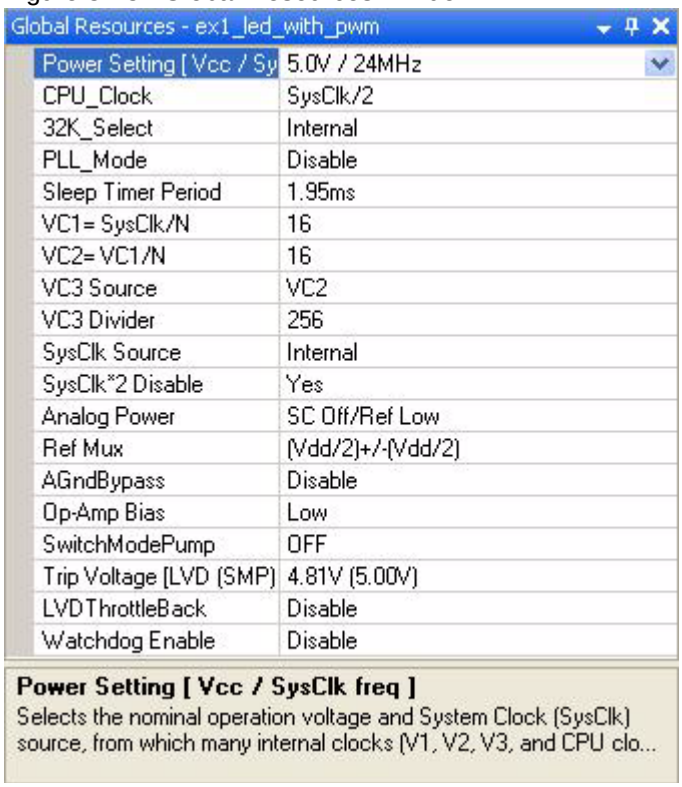

21. Click the **LED_1** UM and navigate to the **Properties** window. Configure the LED for **P1_6.**

Figure 3-12. Properties Window



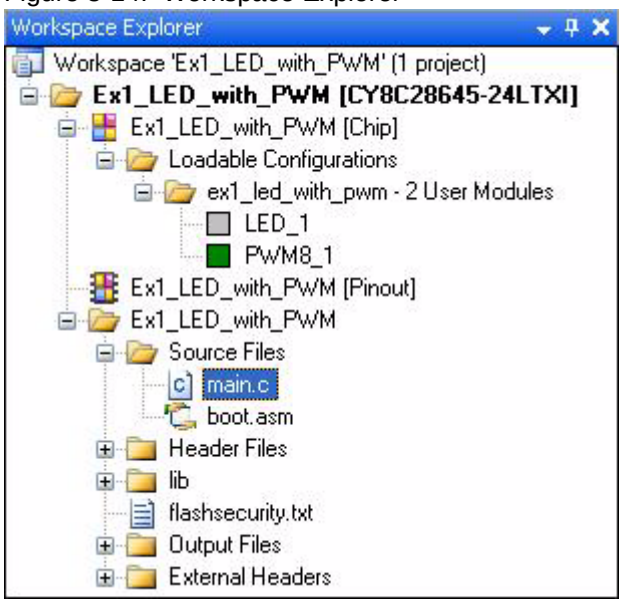

22. Configure the **Global Resources** window to match the following figure.

Figure 3-13. Global Resources Window




23. Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C28_main_Ex1.c* file, which can be found within the attachments feature of this PDF document.

Figure 3-14. Workspace Explorer




24. Save the project.
25. Build the project. **Build → Generate/Build 'Ex1_LED_with_PWM' Project**.

26. Disconnect power to the board.
27. Configure the DVK board SW3 to 5V.
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 11 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 can be found within the attachments feature of this PDF document.

**Note:** To access the embedded attachments feature in the PDF document, 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++);

        /* Switch the state of Software LED (on or off) */
        LED_1_Invert();

    } /* End of while(1) */
} /* End of main */


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

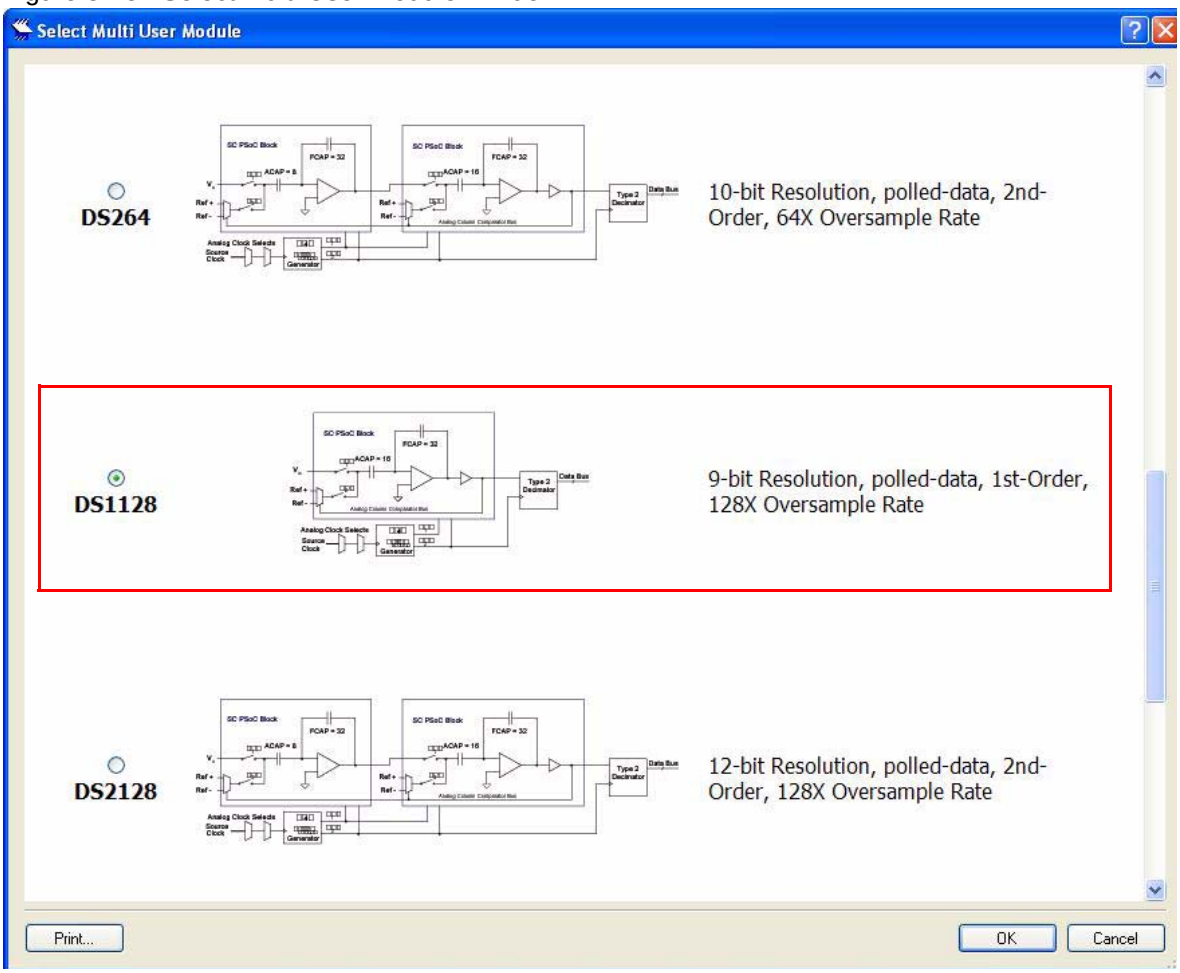
## 3.1.2 ADC to LCD Project

This project demonstrates a 9-bit Delta Sigma 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*

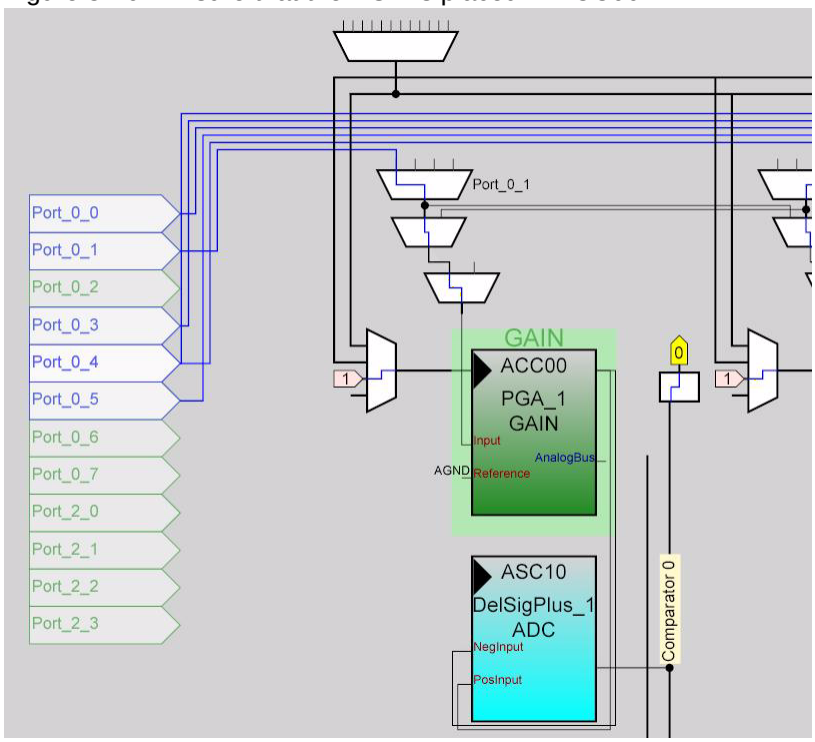
1. Follow steps 1 to 10 in section 3.1.1.1 on page 13, change the **Name** of the project to **Ex2_ADC_to_LCD**.
2. In the **User Modules** window expand the **ADCs** folder, and then right click **DelSigPlus** and choose **Place**. A window opens with multiple options for the DelSigPlus UM. In this case the **DS1128** configuration is used. Scroll down in the window to verify that this is the case.

Figure 3-15. Select Multi 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**, select place. Ensure that the **PGA** is placed in **ACC00**.

Figure 3-16. Ensure that the PGA is placed in ACC00.




6. In the **User Modules** window expand **Misc Digital**, Right click **LCD** and click place.
7. Click **PGA_1** and configure the properties to match this figure.

Figure 3-17. PGA_1 Properties

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

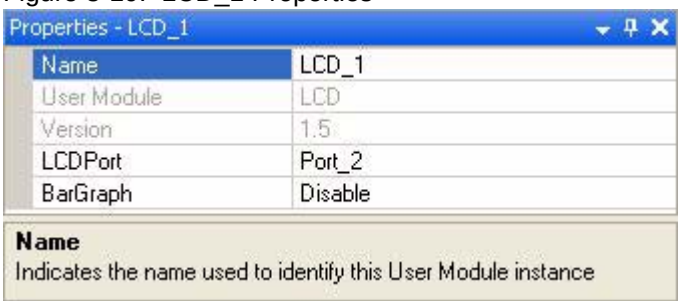
Properties - DelSigPlus_1

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

**Name**
Indicates the name used to identify this User Module instance

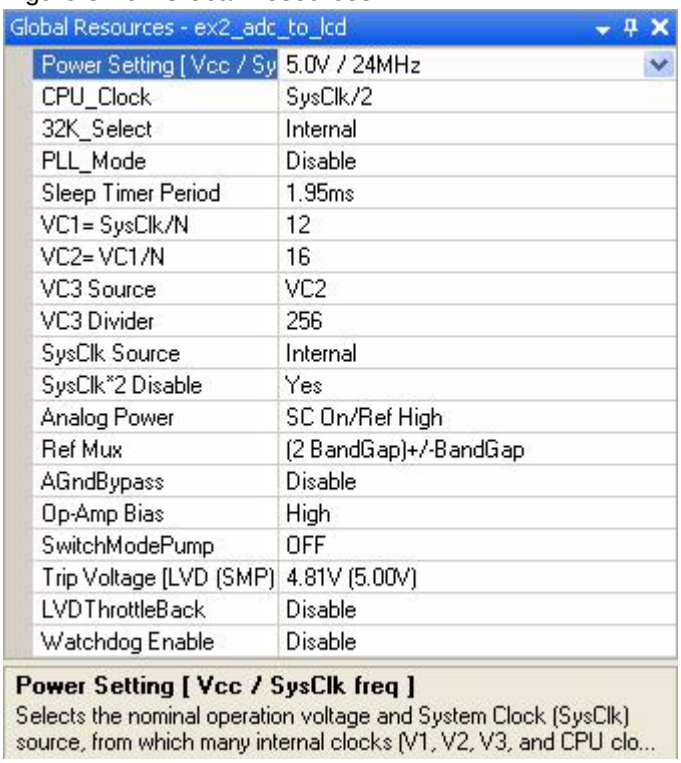
9. Click **LCD_1** and configure the properties to match this figure.

Figure 3-19. LCD_1 Properties



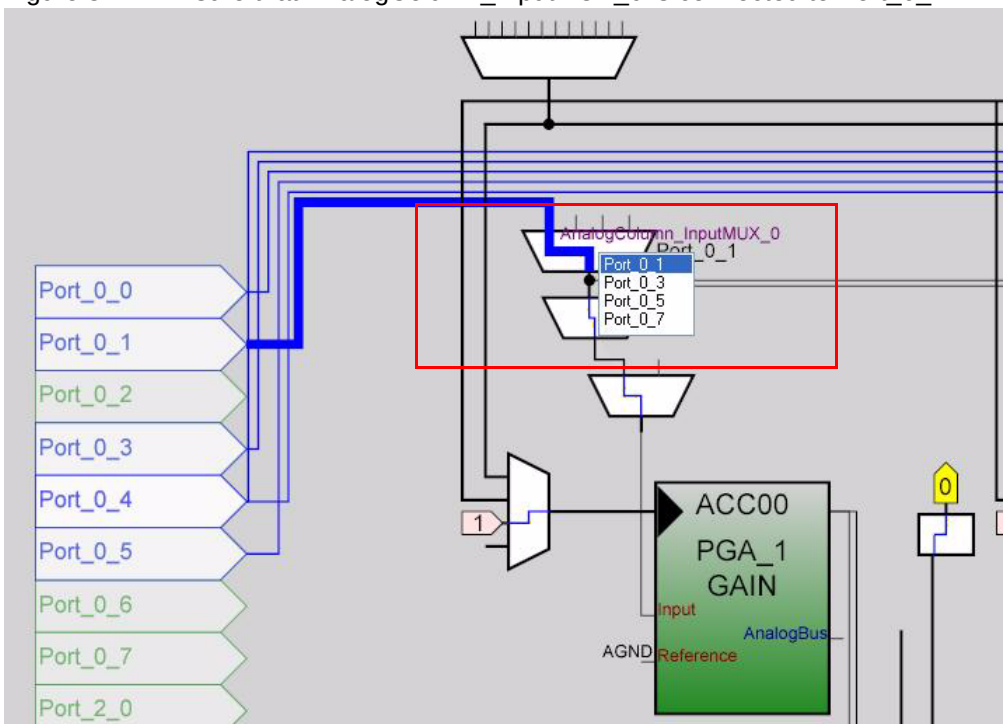

10. Configure the **Global Resources** to match the following figure.

Figure 3-20. Global Resources



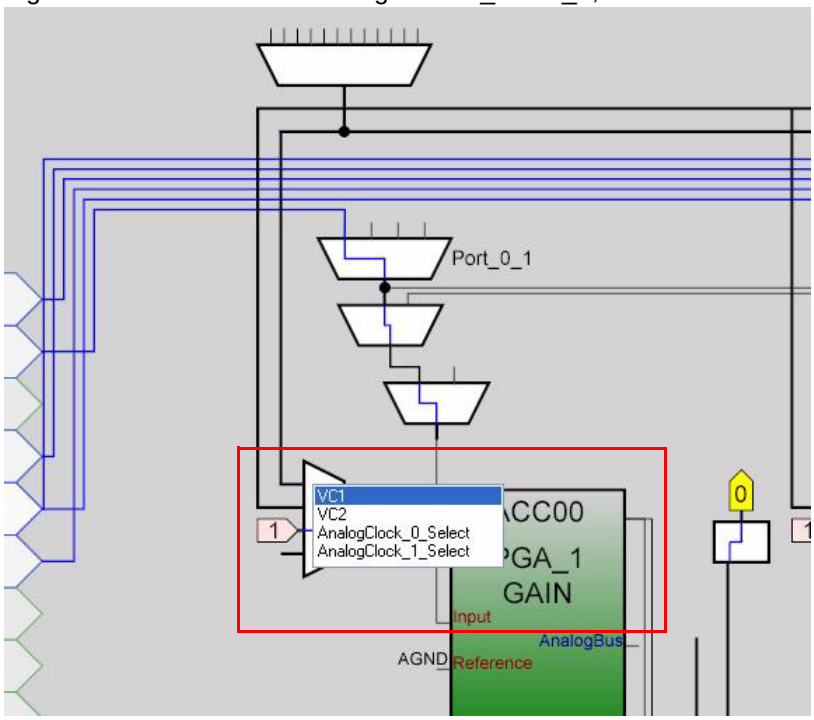

11. Ensure that **AnalogColumn_InputMUX_0** is connected to **Port_0_1**. If it is not configured for this port, double click the **MUX** and choose **Port_0_1**.

Figure 3-21. Ensure that 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 choose **VC1**.

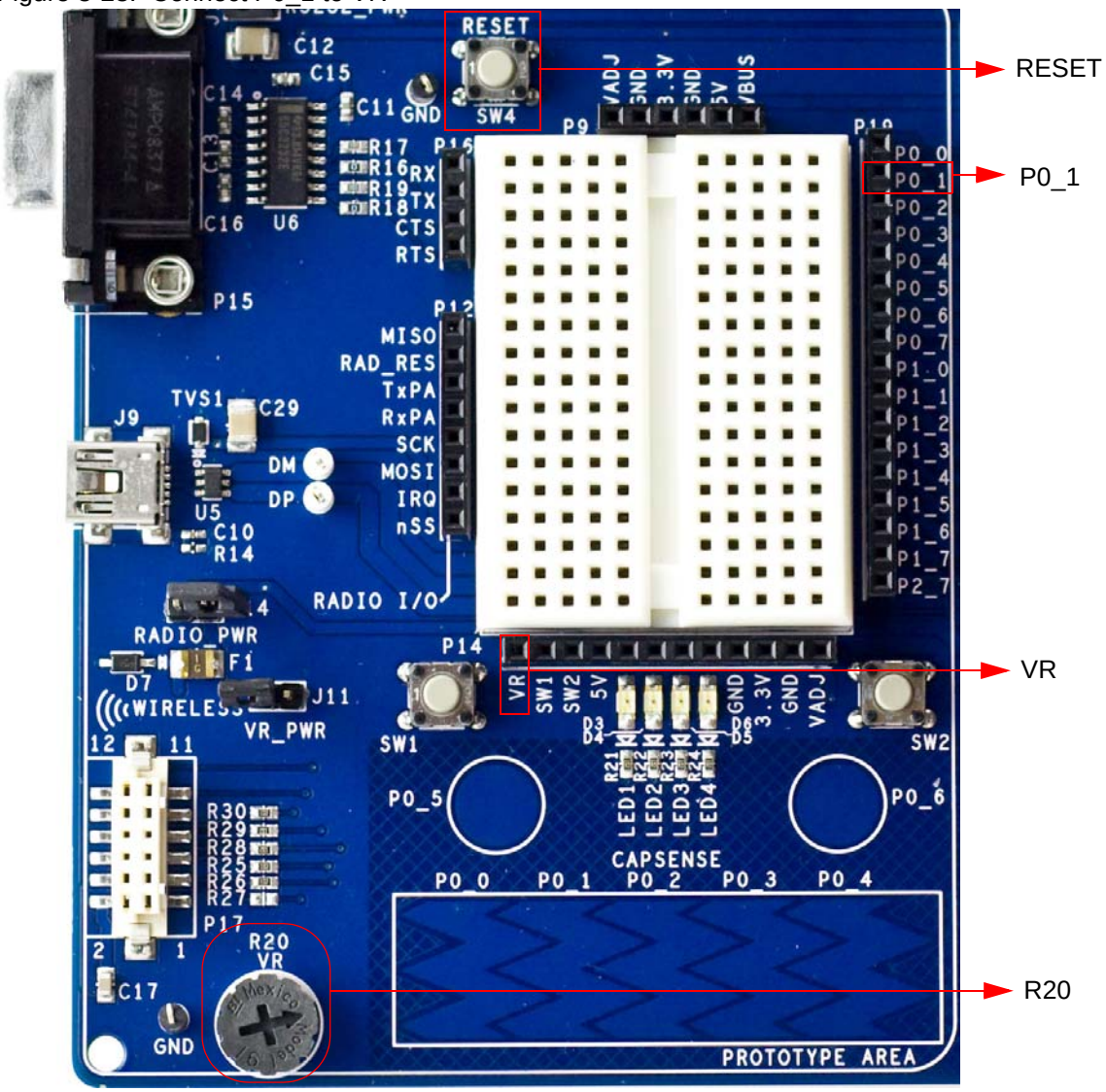
Figure 3-22. Ensure that AnalogColumn_Clock_0, is connected to VC1




13. Open the existing *main.c* file within Workspace Explorer. Replace the existing *main.c* content with the content of the embedded *CY8C28_main_Ex2.c* file, which can be found within the attachments feature of this PDF document.
14. Save the project.

15. Build the project. **Build → Generate/Build 'Ex2_ADC_to_LCD' Project**.
16. Disconnect power to the board.
17. Configure the DVK breadboard SW3 to 5V.
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 11 to program the device.
21. After programming the device, press the reset button and vary the POT (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 can be found within the attachments feature of this PDF document.

**Note:** To access the embedded attachments feature in the PDF document, 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 */
```

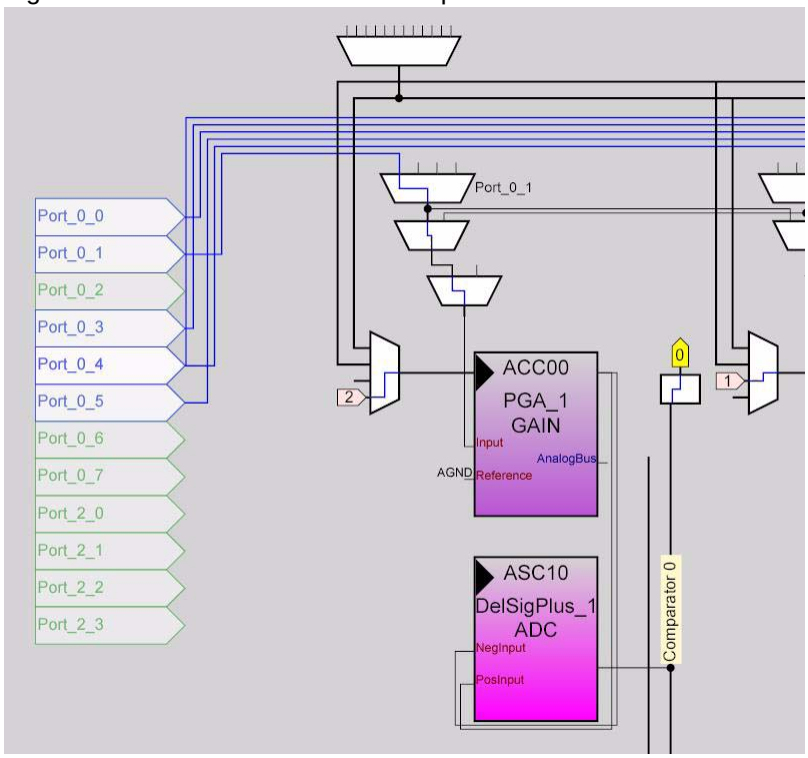
## 3.1.3 ADC to UART with DAC

This project demonstrates sine wave generation by using a 6-bit 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 similarly 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*

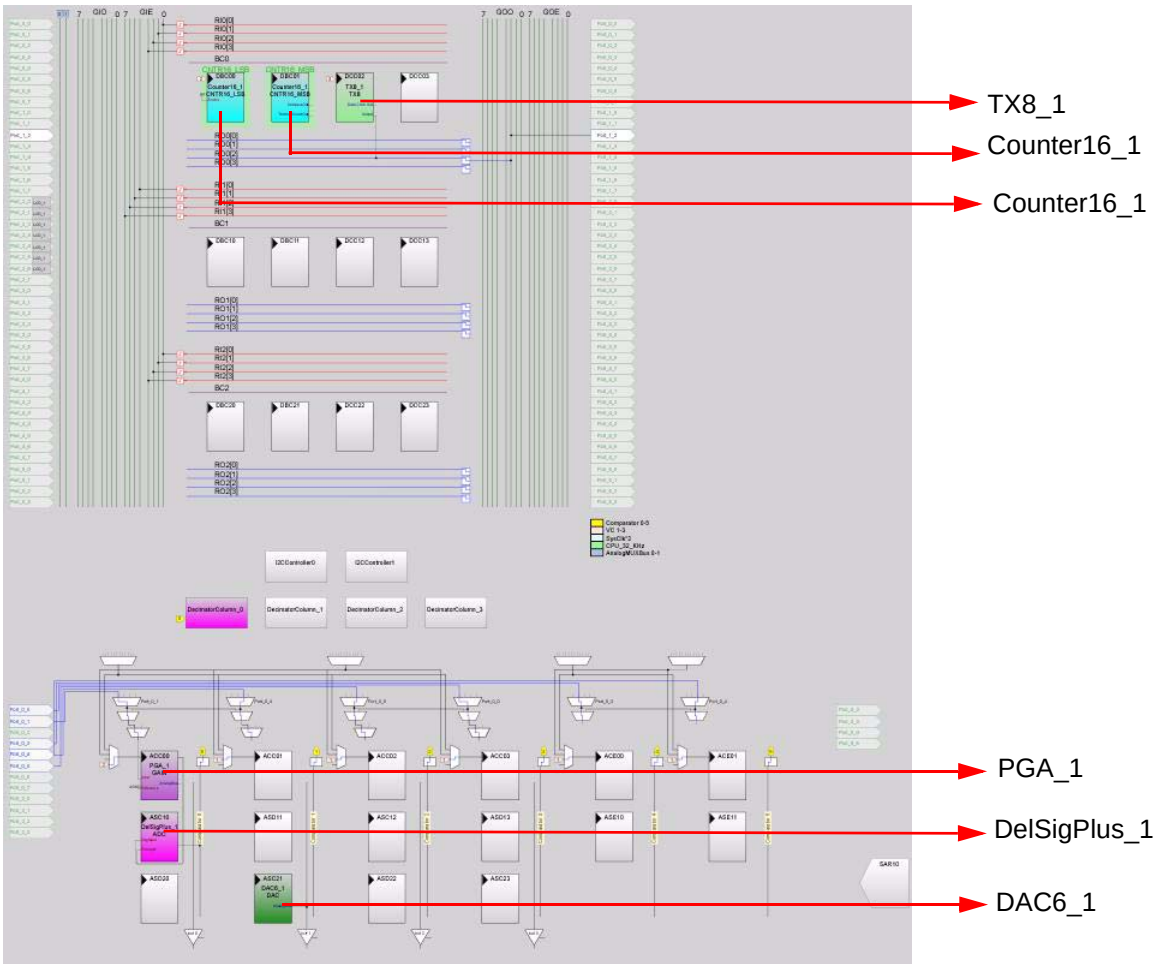
1. Follow steps 1 to 10 in section 3.1.1.1 on page 13, change the **Name** of the project to **Ex3_ADC_to_UART_with_DAC**.
2. In the **User Modules** window expand the **ADCs** folder, and then right click **DelSigPlus** and choose **Place**. A window opens with multiple options for the DelSigPlus UM. In this case 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**, select place. Ensure that the **PGA** is placed in **ACC00**.

Figure 3-24. Ensure that the PGA is placed in ACC00



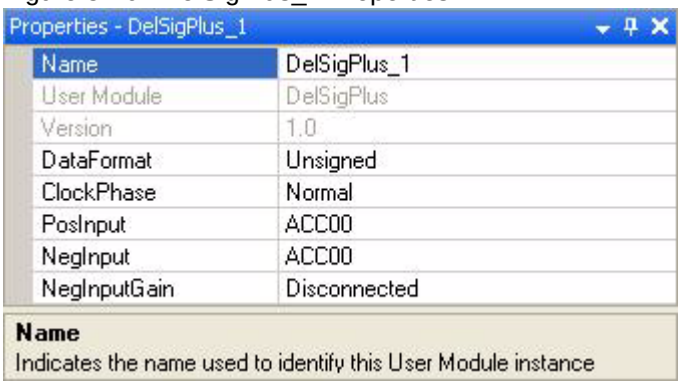

6. In the **User Modules** window expand **Misc Digital**, right click **LCD**, and click place.
7. In the **User Modules** window expand **Counters**, right click **Counter16**, and select place.
8. In the **User Modules** window expand **Digital Comm**, right click **TX8**, and click place.
9. In the **User Modules** window expand **DACs**, right click **DAC6**, and click place.
10. Move the UMs so that they match the configuration shown in Figure 3-25 on page 30.

Figure 3-25. Configure User Modules



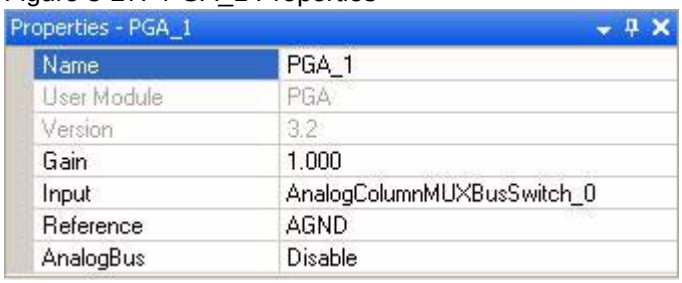

11. Click on **DelSigPlus_1** and configure it to match this figure.

Figure 3-26. DelSigPlus_1 Properties



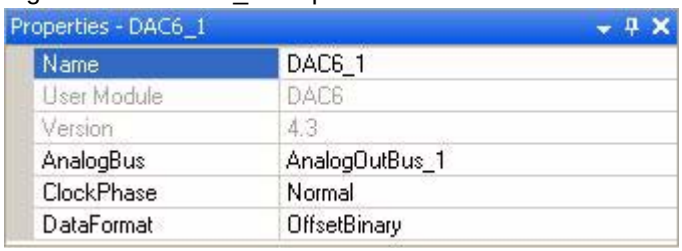
12. Click **PGA_1** and configure it to match this figure.

Figure 3-27. PGA_1 Properties



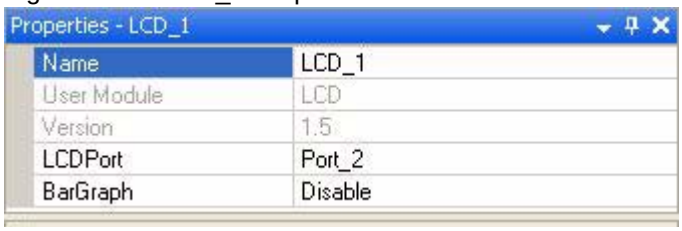

13. Click **DAC6_1** and configure it to match this figure.

Figure 3-28. DAC6_1 Properties



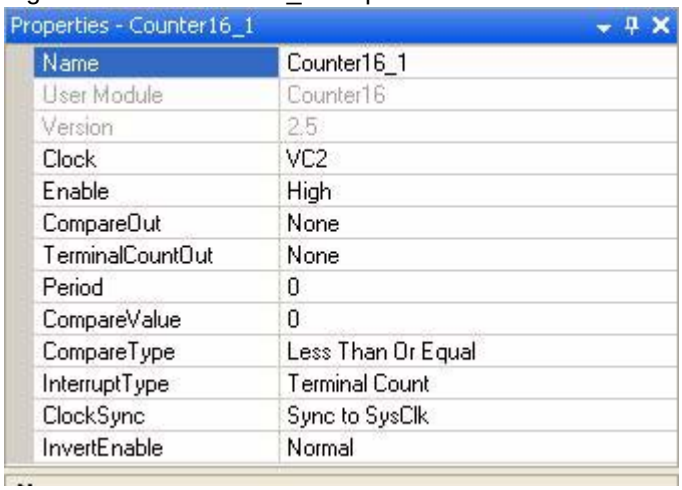

14. Click **LCD_1** and configure it to match this figure.

Figure 3-29. LCD_1 Properties



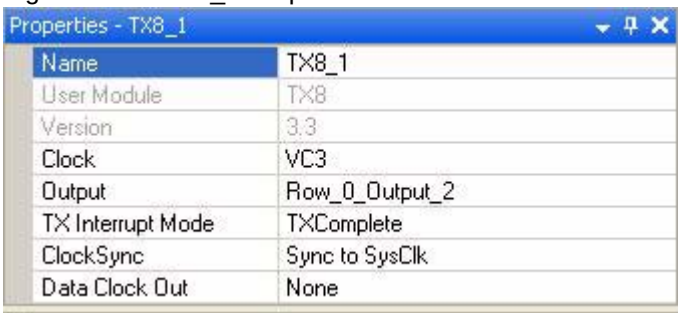

15. Click on **Counter16_1** and configure it to match this figure.

Figure 3-30. Counter16_1 Properties



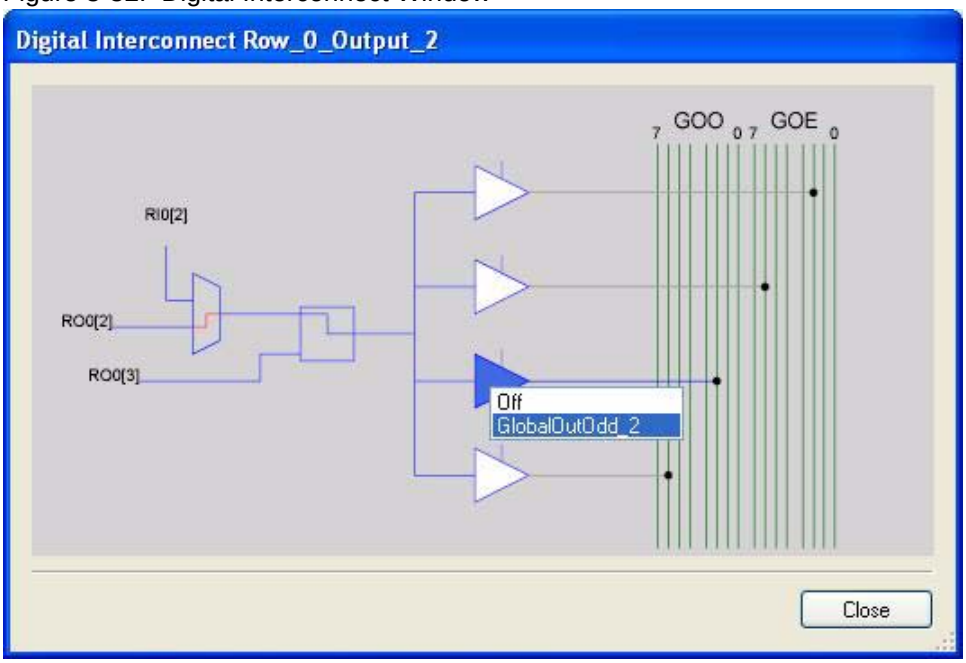
16. Click **TX8_1** and configure it to match this figure.

Figure 3-31. TX8_1 Properties



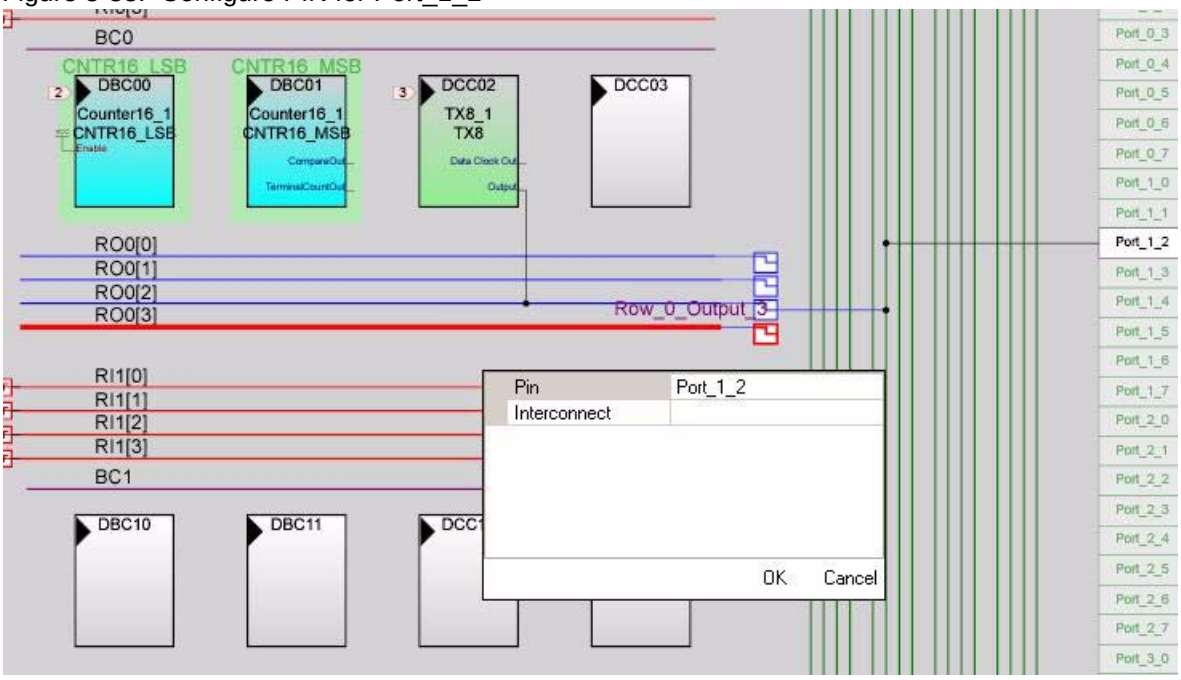

17. Click **RO0[2] LUT**, enable **Row_0_Output_2_Drive_2** to connect **GlobalOutOdd_2**.

Figure 3-32. Digital Interconnect Window



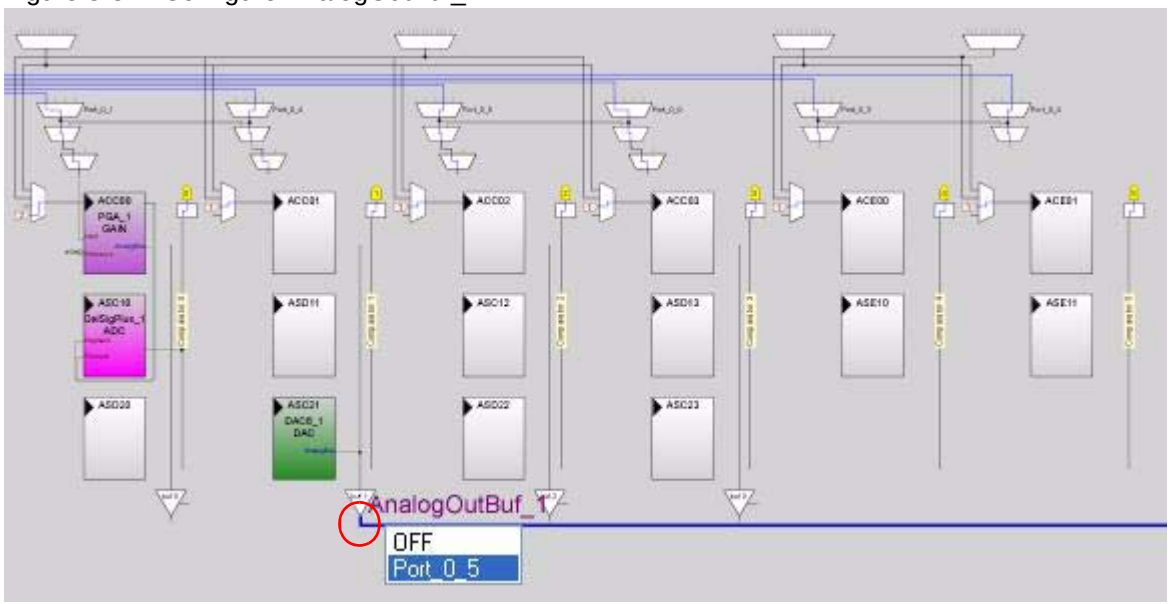
18. Click **GlobalOutOdd_2**. A window appears; in this window configure **PIN** for **Port_1_2**.

Figure 3-33. Configure PIN for Port_1_2



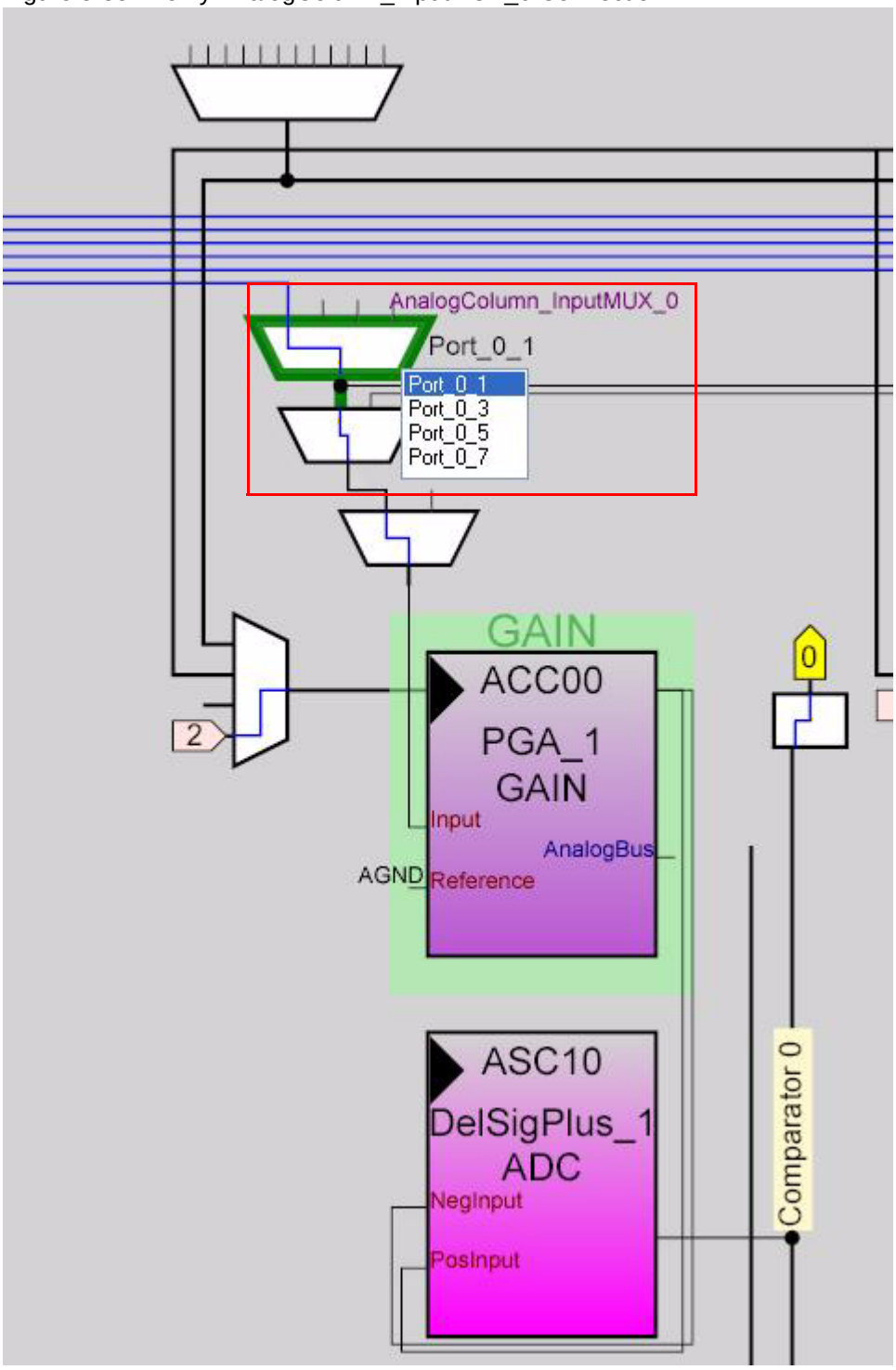

19. Click **OK** to continue.
20. Click **AnalogOutBuf_1** and configure it for **Port_0_5**.

Figure 3-34. Configure AnalogOutBuf_1



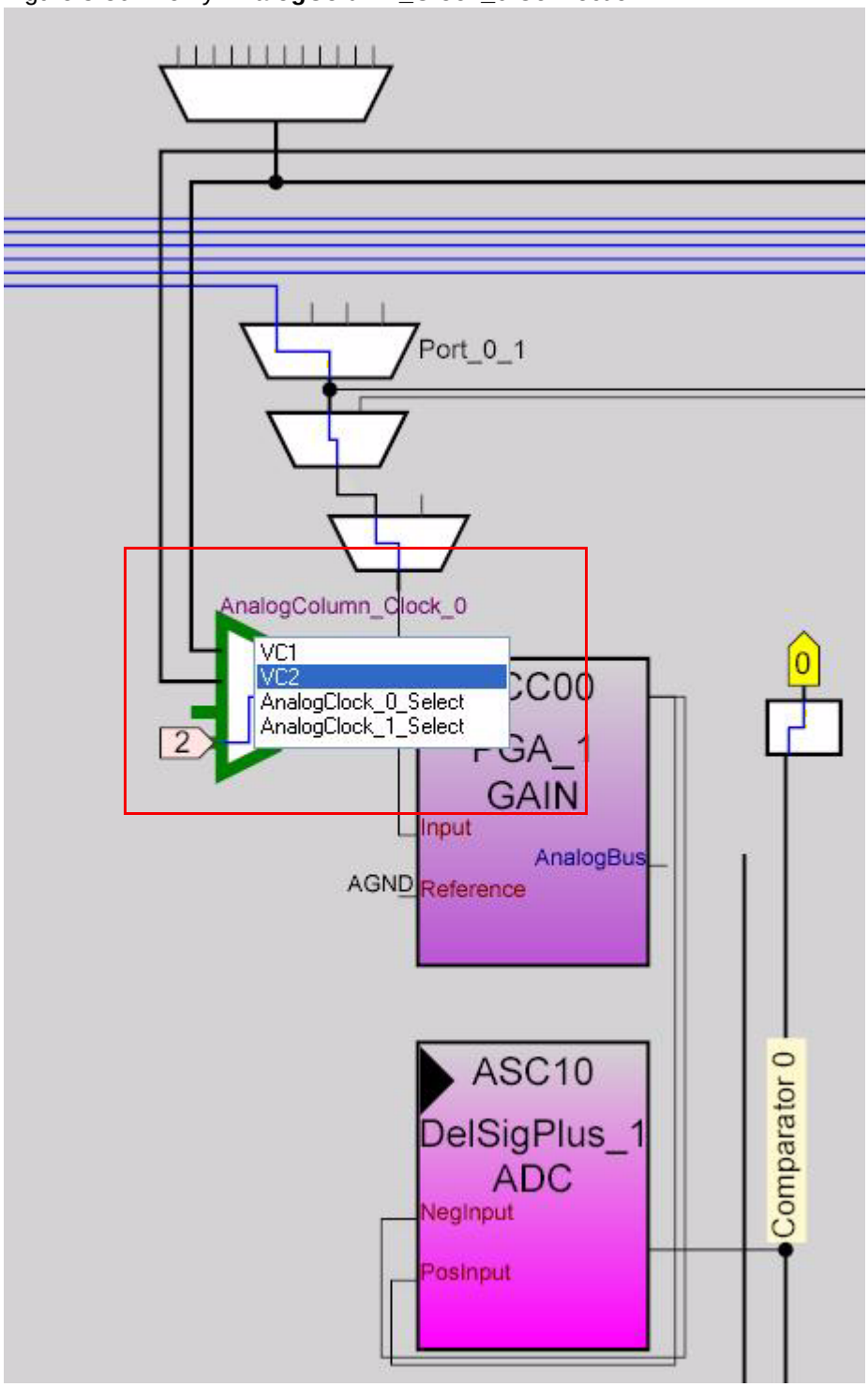

21. Verify that **AnalogColumn_InputMUX_0** is connected to **Port_0_1**. If it is not configured for this port, double click the **MUX** and choose **Port_0_1**.

Figure 3-35. Verify AnalogColumn_InputMUX_0 Connection



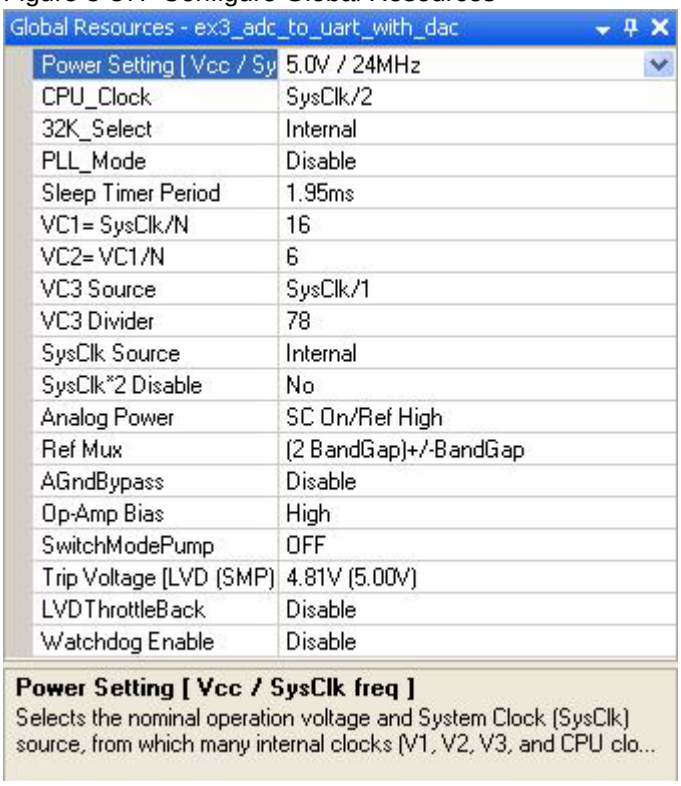
22. Verify that **AnalogColumn_Clock_0** and **AnalogColumn_Clock_1** are connected to **VC2**. If it is not, double click the **MUX** and chose **VC2**.

Figure 3-36. Verify **AnalogColumn_Clock_0** Connection



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

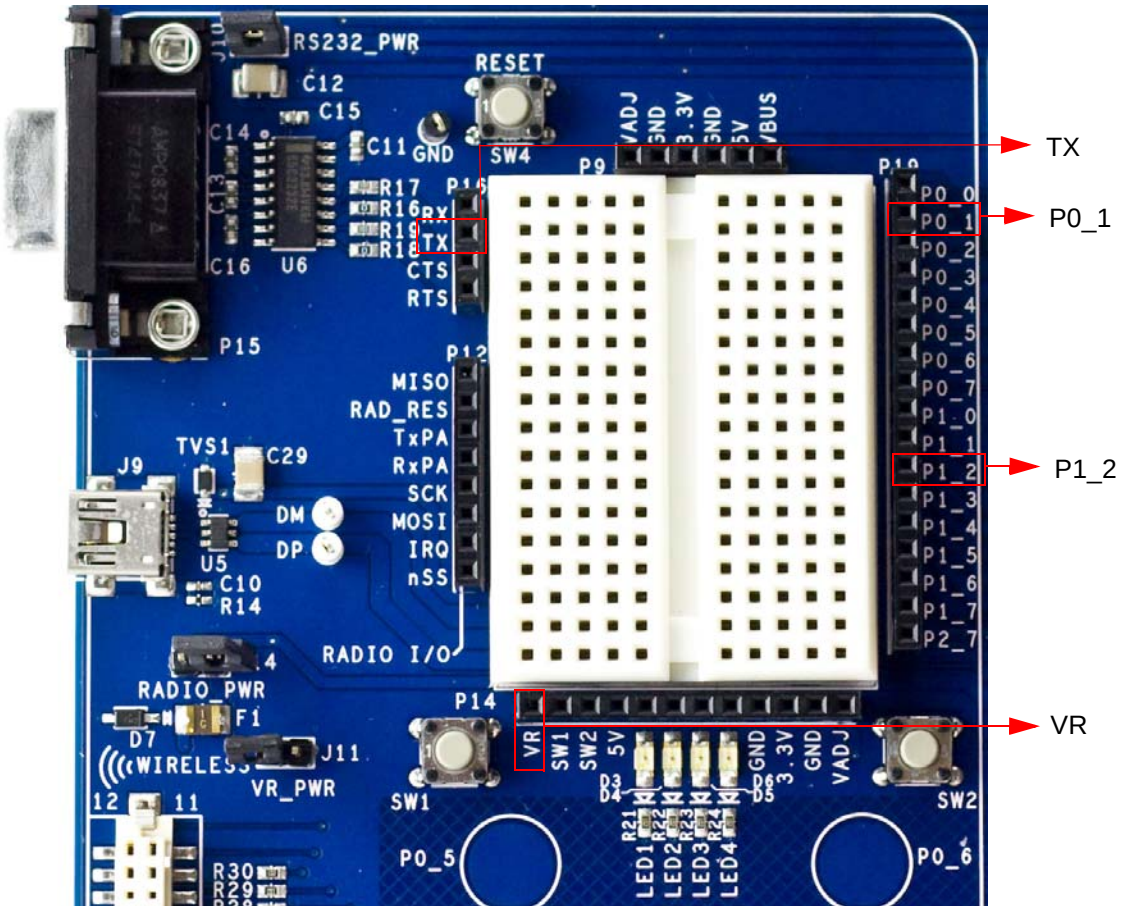
Figure 3-37. Configure Global Resources




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 can be found 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 breadboard SW3 to 5V.
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 11 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 can be found within the attachments feature of this PDF document.

**Note:** To access the embedded attachments feature in the PDF document, 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 in order 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
```

```
*
*******************************************************************************/
void Counter16_C_ISR(void)
{
    /* 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®

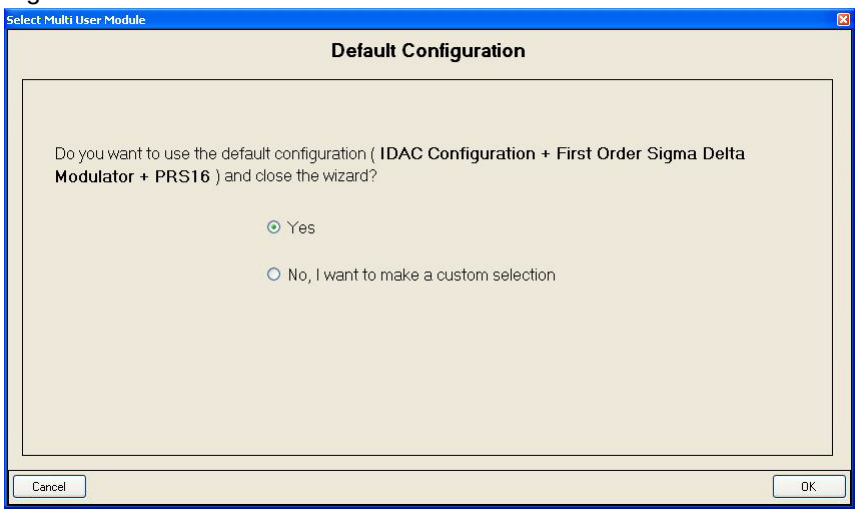
This project demonstrates CapSense. The firmware displays the CapSense button presses on the LCD (row 1) and associated LEDs. It also displays the CapSense slider position on the LCD (row 2).

Note that this project uses IDAC. But if using external Rb with CSD is desired then populate R15 (connected to P3[1]). Rb can range from 2k to 10k. Do refer to CapSense user module datasheet for more information on using Rb.

### *3.1.4.1 Creating CapSense Project*

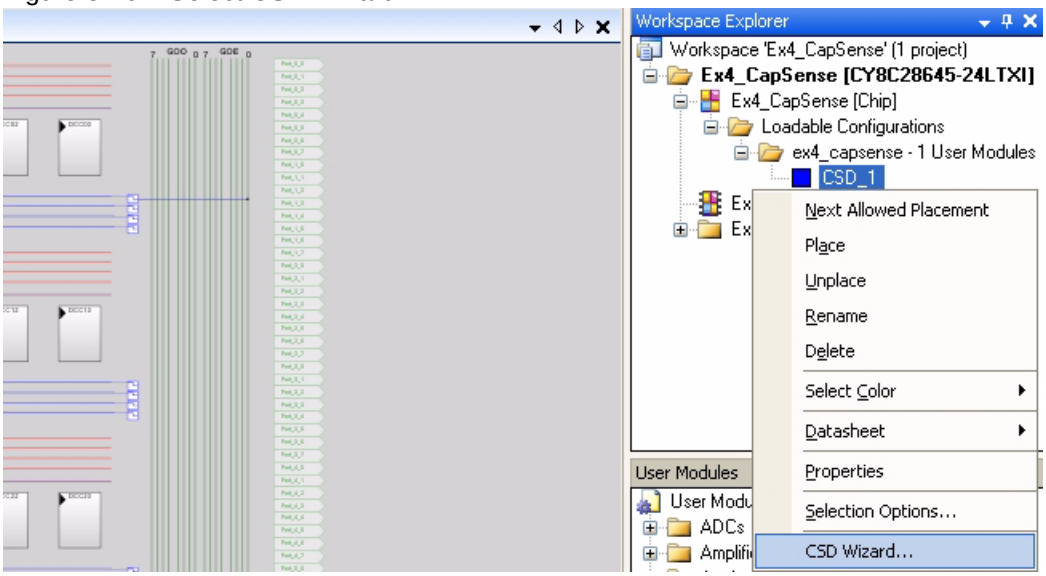
1. Follow steps 1 to 10 in section 3.1.1.1 on page 13, change the **Name** of the project to **Ex4_CapSense**.
2. In the User Modules window expand the Cap Sensors folder. Right click **CSD** and choose **Place**. A window appears with the option to use the default configuration.

Figure 3-39. Select Multi User Module Window



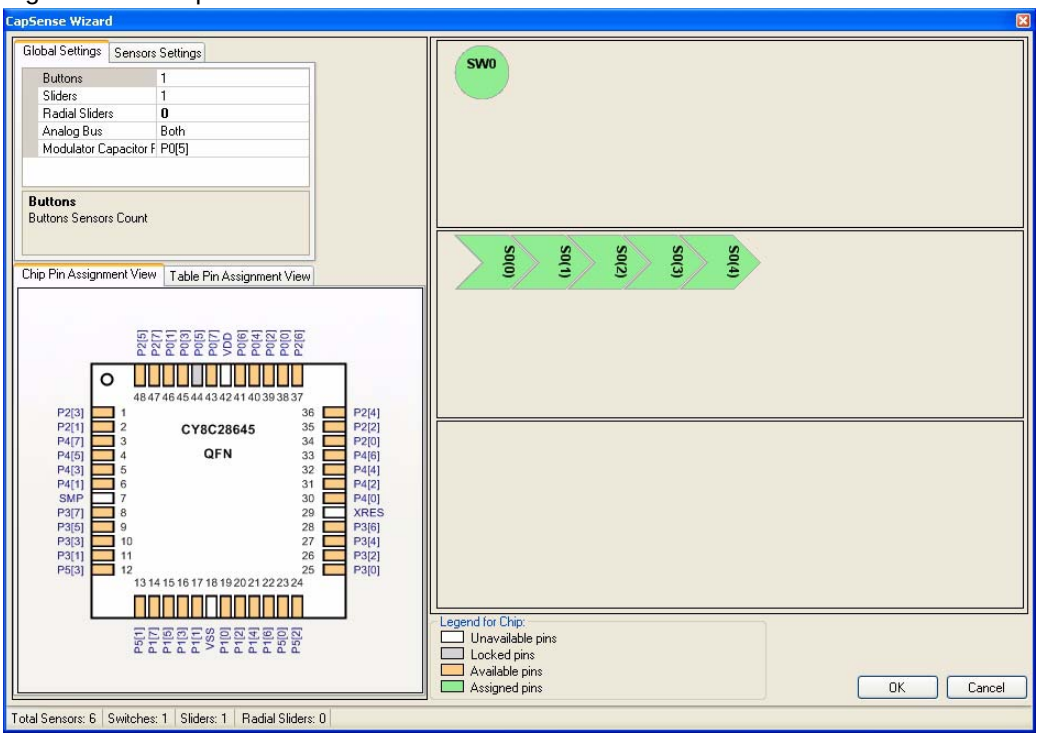

3. Select **Yes** and click **OK**.
4. Right click the **CSD** user module in the workspace explorer and select **CSD Wizard**.

Figure 3-40. Select CSD Wizard



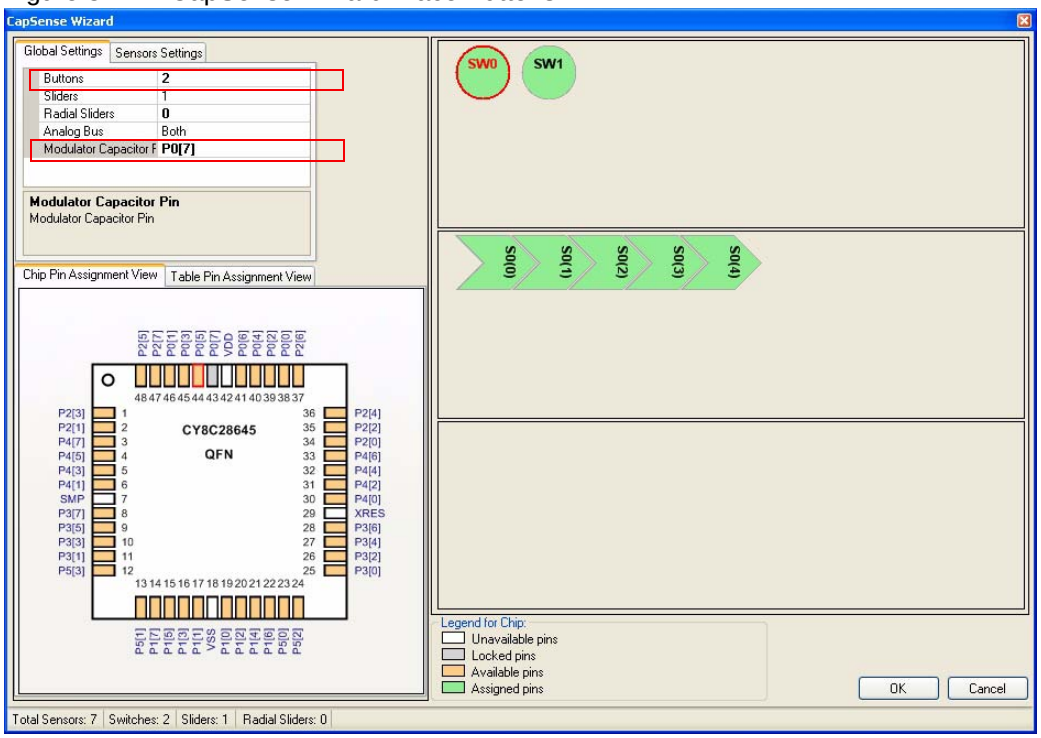

5. CapSense Wizard window opens.

Figure 3-41. CapSense Wizard



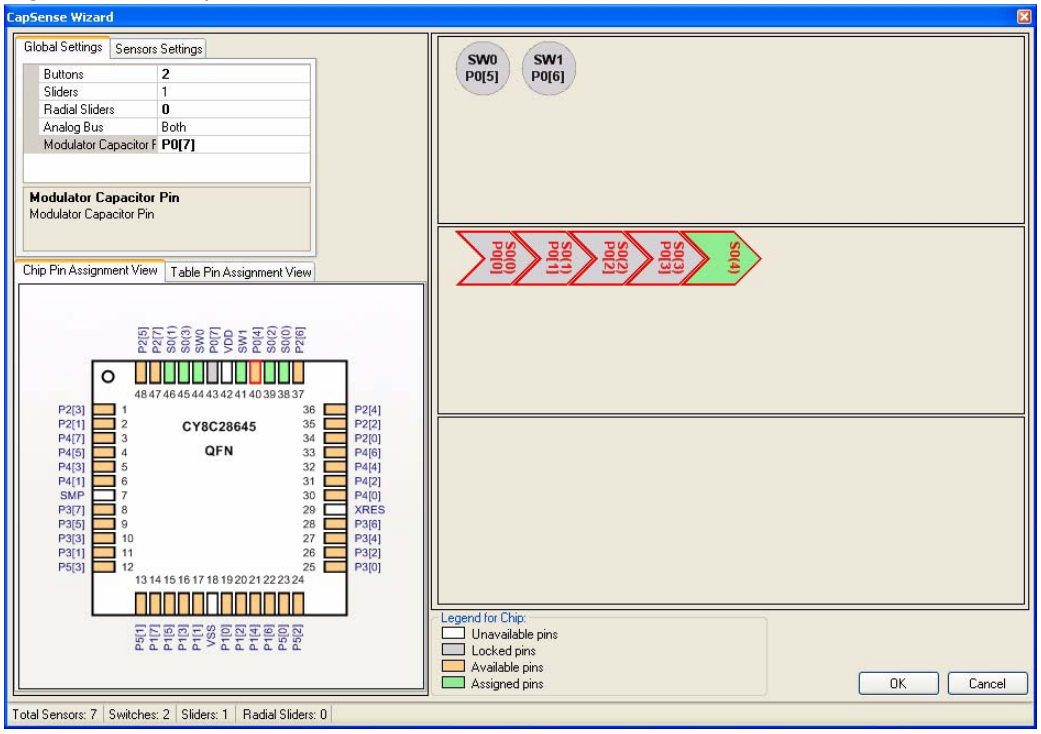

6. In the CapSense Wizard window, under the **Global Settings** Tab, set the # of buttons to 2.
7. Select P0.7 as the Modulator Capacitor Pin.

Figure 3-42. CapSense Wizard Place Buttons




8. Click and hold **SW0** and drag it to P0.5.
9. Do the same for **SW1** and drag it to P0.6

Figure 3-43. CapSense Wizard Slider Sensors

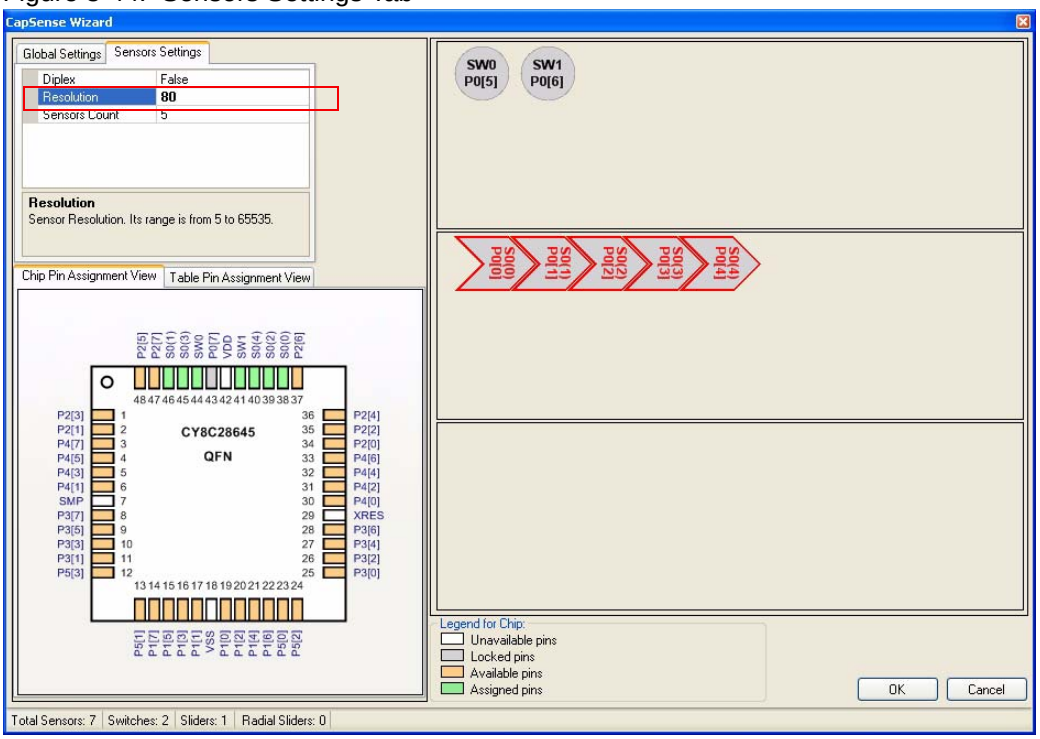



10. Do the same thing for each slider sensor and corresponding pin.

- 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

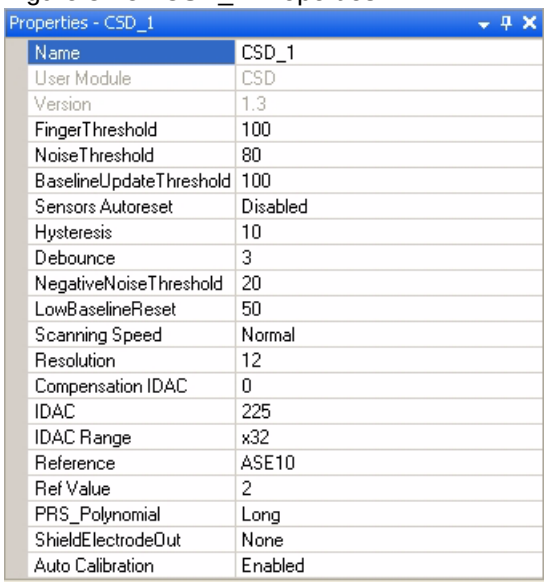
11. Select the **Sensors Settings** Tab
12. Set the **Resolution** to 80.

Figure 3-44. Sensors Settings Tab



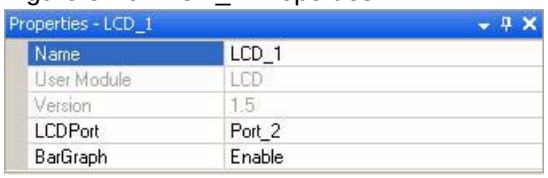

13. Click **OK**
14. In the **User Modules** window expand **Misc Digital**, right click **LCD**, and click place.
15. In the **User Modules** window expand **Misc Digital**, right click **LED**, and click place.
16. In the **User Modules** window expand **Misc Digital**, right click **LED**, and click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

Figure 3-45. CSD_1 Properties



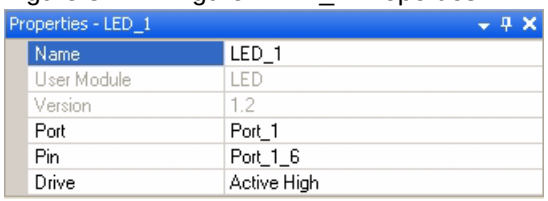

18. Click **LCD_1** and configure it to match this figure.

Figure 3-46. LCD_1 Properties



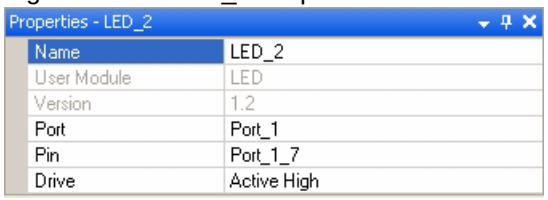

19. Click **LED_1** and configure it to match this figure.

Figure 3-47. Figure . LED_1 Properties



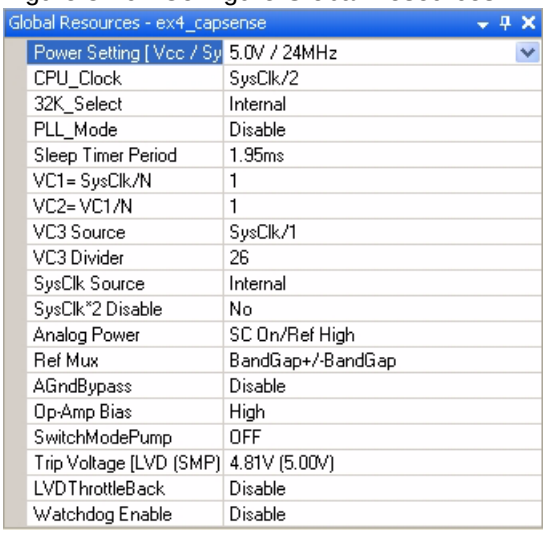

20. Click **LED_2** and configure it to match this figure.

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 can be found 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 5V.
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 13 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. Likewise, 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 Bar graph.
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 can be found within the attachments feature of this PDF document.

**Note:** To access the embedded attachments feature in the PDF document, 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 bar graph.
*
* 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 bar graph with the current finger position */
        LCD_1_DrawBG(ROW_1, COLUMN_0, NUM_CHARACTERS, value + 1);
    }
}


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

# Appendix A. Board Specifications and Layout

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

## 2.1 PSoC Development Board

### 2.1.1 Factory Default Configuration

#### *2.1.1.1 Power Supply*

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

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

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

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

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

VADJ (1.25V to 3.9V) - 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 when user wants to separate the analog power from the digital power. The source for VDD ANLG can be chosen as VDD or VADJ using the J7 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 5V, 3.3V, or VADJ using four headers. It enables the user to power the PSoC GPIOs at different voltages.

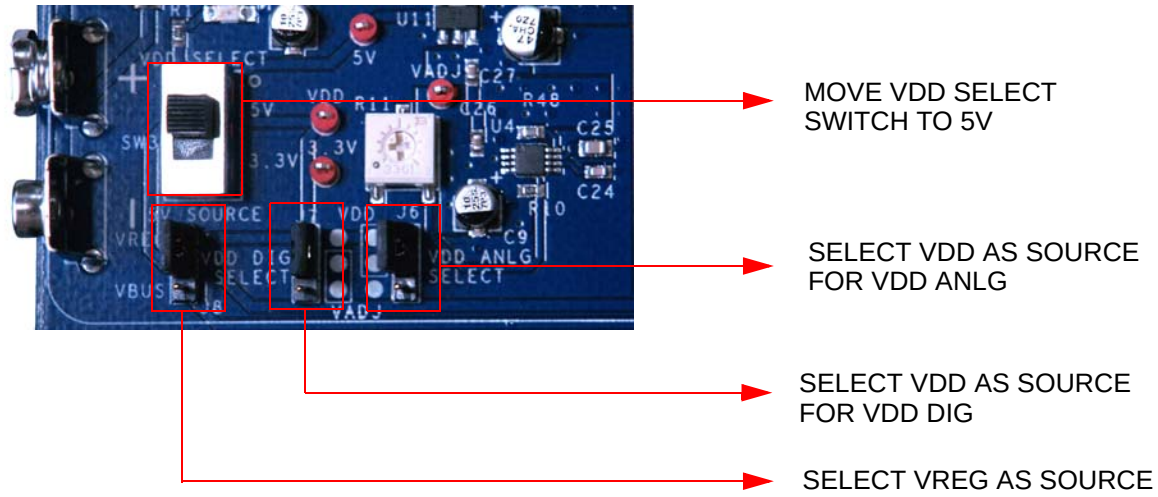
## 2.1.2 Power Supply Configuration Examples

### *2.1.2.1 Setting a 5V 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 5V.
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.

**Note** 5V operation is not recommended with ES1 Silicon. See the silicon errata for details.

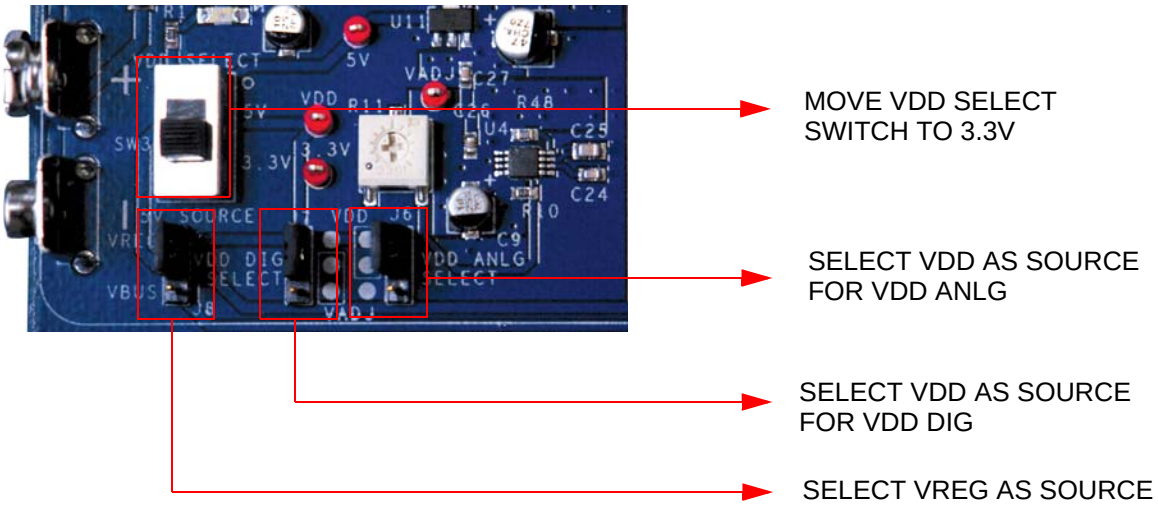
Figure 2-1. Setting a 5V Supply from VREG




### *2.1.2.2 Setting a 3.3V Supply from VREG*

1. Place the jumper on J8 header to select VREG as the source.
2. Move the VDD select switch to select 3.3V.
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.

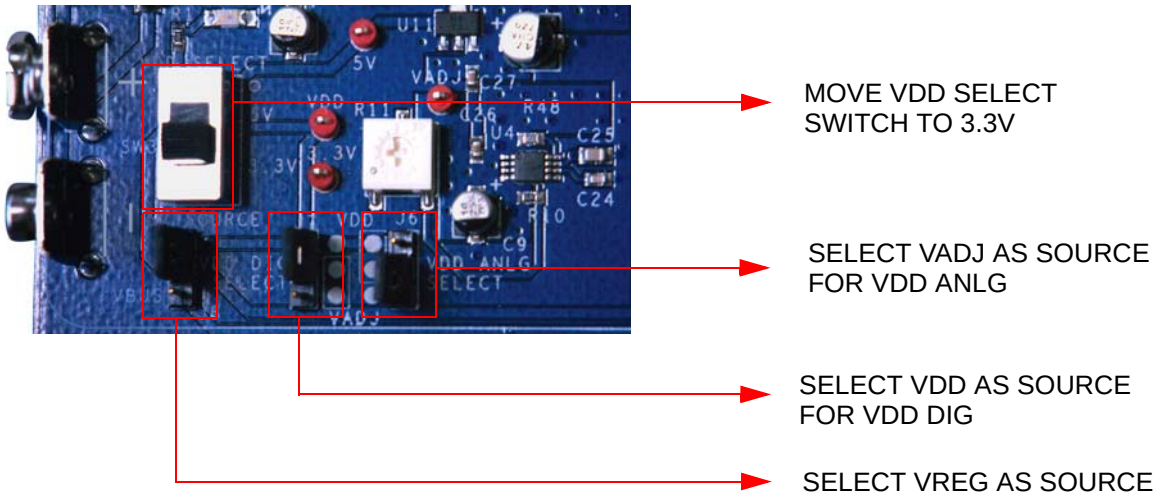
Figure 2-2. Setting a 3.3V Supply from VREG



### 2.1.2.3 *Setting VDD ANLG as VADJ and VDD DIG as VDD for VDD = 3.3V*

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

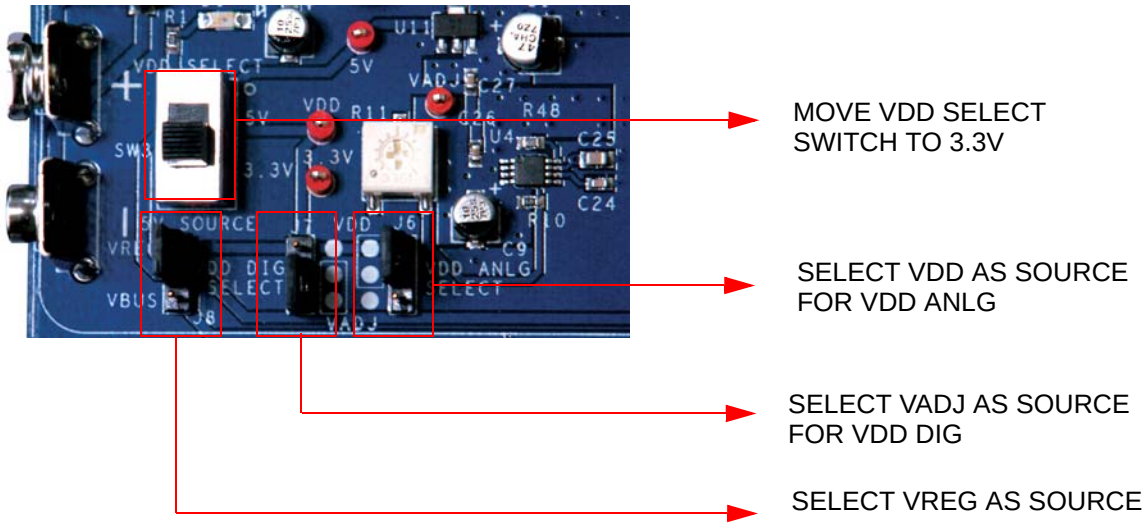



This helps to separate the analog supply from the digital supply and VDD.

### 2.1.2.4 *Setting VDD DIG as VADJ and VDD ANLG as VDD for VDD = 3.3V*

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



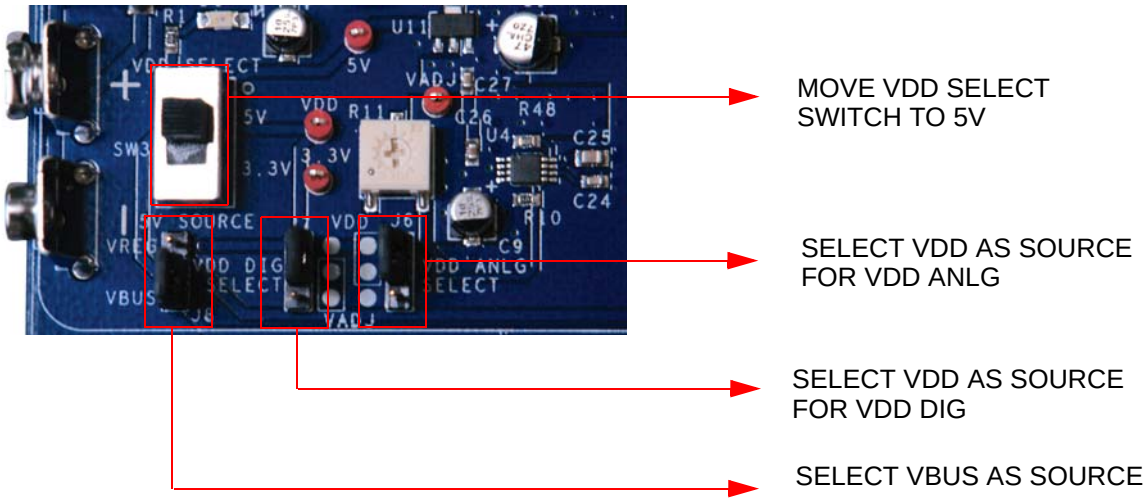

This helps to separate the digital supply from the analog supply and VDD.

### 2.1.2.5 *Setting a 5V 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 5V.
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.

**Note** 5V operation is not recommended with ES1 Silicon. See the silicon errata for details.

Figure 2-5. Setting a 5V Supply from VBUS



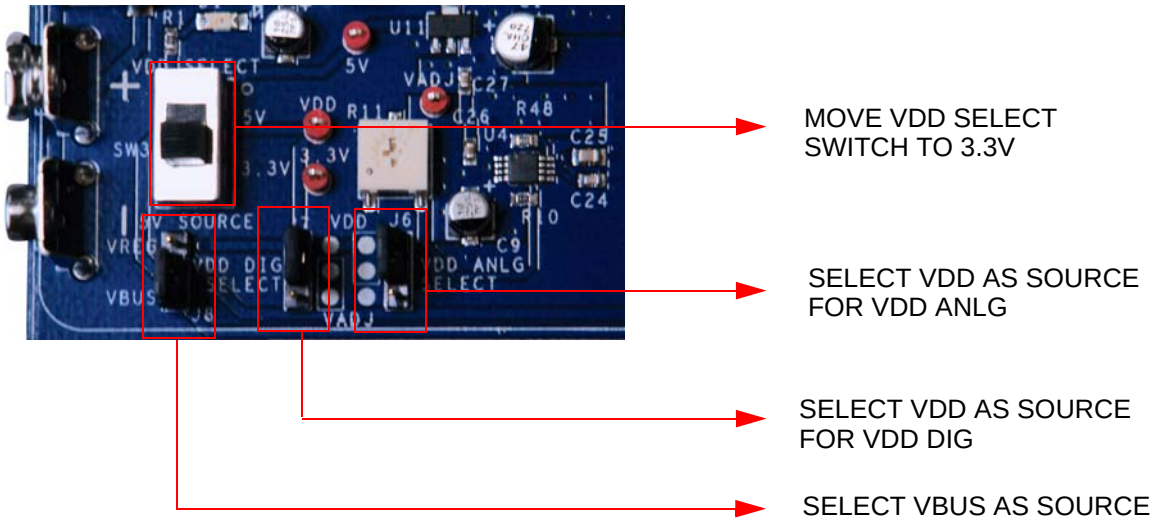

### 2.1.2.6 *Setting a 3.3V Supply from VBUS*

Due to the nature of the PSoC Development Board, powering the system from USB 'VBUS' could potentially reset other USB devices on the same hub.

By design, the PSoC Development Board is capable of drawing more than 500mA 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 10uF. As a result, plugging the PSoC Development Board into a USB hub could potentially cause other devices on the same hub to reset due to excessive inrush currents. Careful consideration should be taken when powering the PSoC Development Board from 'VBUS'. In general, it is good practice to plug the PSoC Development 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 to 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 3.3V.
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.

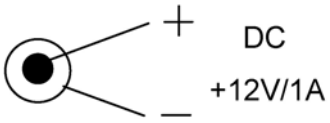
Figure 2-6. Setting a 3.3V supply from VBUS




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.

### 2.1.2.7 *J1 - DC Power Jack*

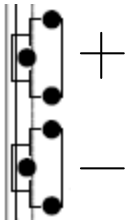
Figure 2-7. DC Power Jack




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

### 2.1.2.8 *9V Battery Terminals*

Figure 2-8. Battery Terminals



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

### 2.1.2.9 *J8 - 5V Source*

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

### 2.1.2.10 *VDD Select Switch*

This switch allows for selecting either 5V or 3.3V. VDD feeds VDD DIG, VDD ANLG and VDDIO.

#### 2.1.2.11 J7 - VDD DIG Select

This header allows the user to select the PSoC core source power. If you need to power the PSoC core at either 5V or 3.3V (based on the position of the VDD select switch), place the jumper on the upper two pins. If you need to power the PSoC core at lower voltages (1.25V to 3.9V), then 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.

#### 2.1.2.12 J6 - VDD ANLG Select

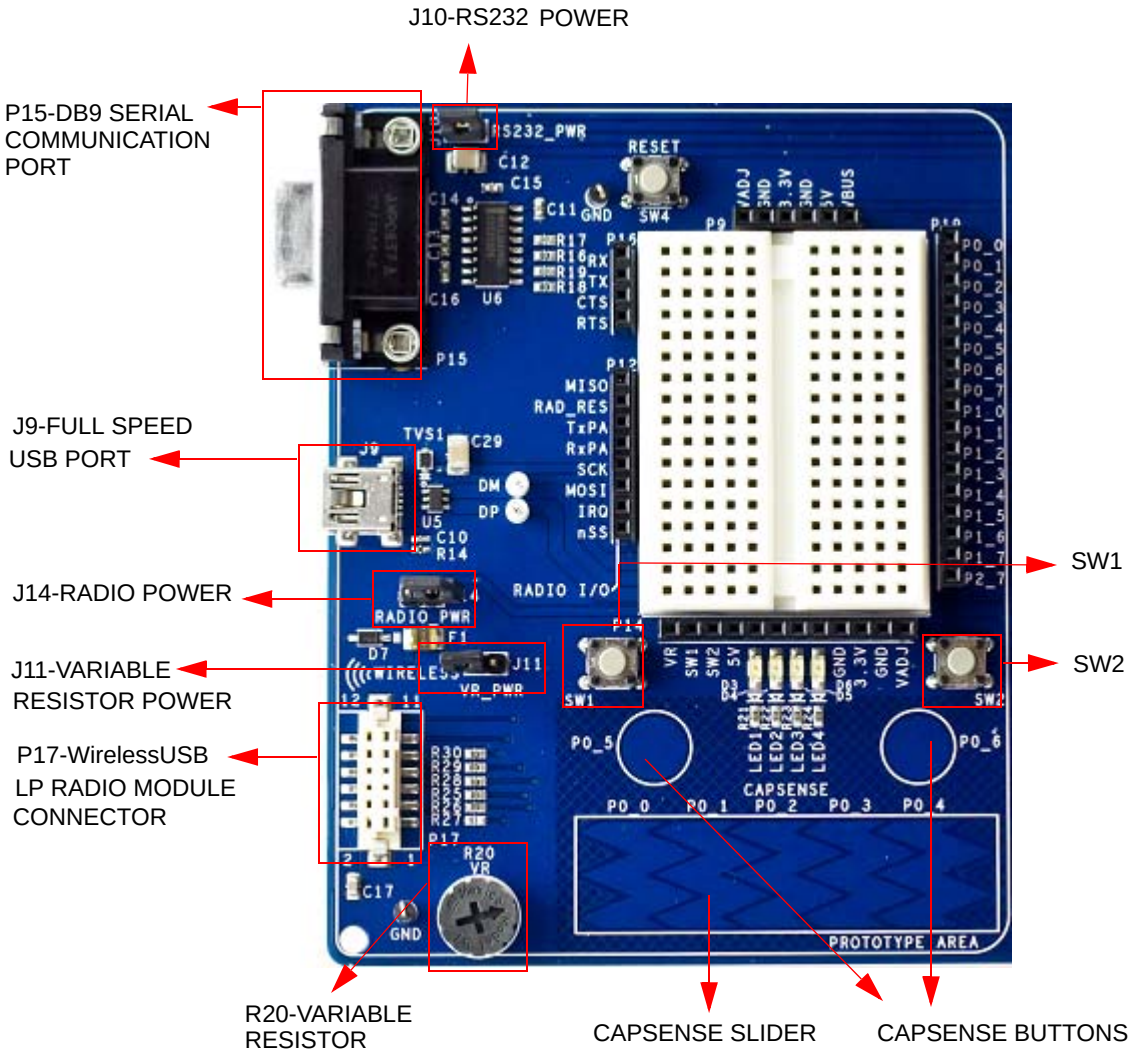
If you want to separate the analog power from the digital power, you can position the jumper on the upper two pins to source analog power at 5V or 3.3V (based on the position of the VDD select switch), or on the lower two pins to source analog power at lower voltages (1.25V to 3.9V).

#### 2.1.2.13 R11 - Adjustable Regulator Variable Resistor

This adjustable resistor is used to tune the VADJ voltage. Turning this variable resistor swings the VADJ voltage between 1.25V and 2.3V when the VDD select switch is in the 3.3V position. When the VDD select switch is in the 5V position, turning this variable resistor swings the VADJ voltage between 1.25V and 3.9V.

## 2.1.3 Prototyping Components

### *2.1.3.1 Prototyping Area*




**Note** Port1 pins are used for SWD/JTAG. The P1_6 and P1_7 pins are used for CapSense.

### *2.1.3.2 P15 - DB9 Serial Communications Port*

This is a standard female DB9 serial communications connector. Four signals are brought from the RS232 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 2-1. Connector Pin Assignments - RS232 (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) |

### *2.1.3.3 J10 - Serial Port Power*

Header J10 must be jumpered in order to use the serial communications port. Placing a jumper on J10 provides VDD power to the RS232 transceiver. This power can be either 3.3V or 5V, depending on which the position of the VDD select switch.

### *2.1.3.4 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.

### *2.1.3.5 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 4 SPI (serial peripheral interface) 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 3.3V.

Table 2-2. 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 |
| 9 | GND |
| 10 | (Empty) |
| 11 | TxPA |
| 12 | RxPA |

#### 2.1.3.6 *J14 - Wireless Radio Module Power*

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

#### 2.1.3.7 *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.

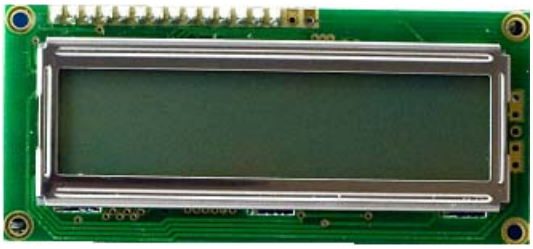
#### 2.1.3.8 *J11 - Variable Resistor Power*

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

#### 2.1.3.9 *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.

### 2.1.4 LCD Module



The board has a 2x16 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 5V. However, the I/Os have a level translator inline so that signaling may be as low as 1.8V and still be recognized by the LCD. The header J12 must be jumpered for the LCD Module to be powered. If J12 is not jumpered, it removes power from level translator. If the LCD module is removed, the receptacle pins of P18 can be used as port 2.

Table 2-3. 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 |

Table 2-3. Connector Pin Assignments - LCD Module Socket *(continued)*

| Pin Number | P18 |
|---|---|
| 13 | D6 |
| 14 | D7 |
| 15 | BACKLT LED ANODE |
| 16 | BACKLT LED CATHODE |

#### *2.1.4.1 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.

#### *2.1.4.2 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 5V to the VCC pin of the module. This 5V power is taken directly from the onboard 5V regulator.

### 2.1.5 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 560Ω should be used for achieving 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.

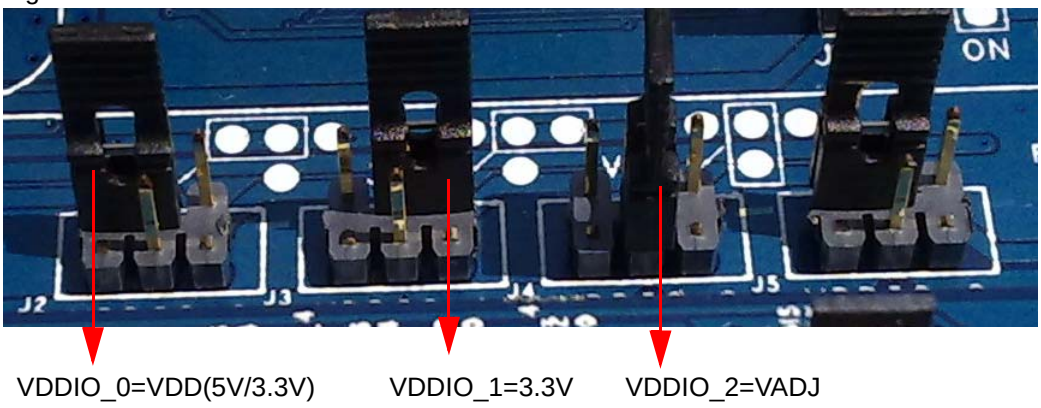
### 2.1.6 Processor Module

#### *2.1.6.1 J2, J3, J4 and J5 - VDDIO Select*

These four headers allow the user to power the PSoC GPIOs at different voltages. For instance, some of the I/O may be powered at 5V, some at 3.3V and some at 1.8V. There are four blocks of GPIO, each having its own source power. Each VDDIO header provides power to specific GPIO's and is selectable from VDD, 3.3V or VADJ. For details on which GPIOs are powered by which VDDIO header, refer to the data sheet for the PSoC device used with this board.

For example, VDDIO_0 is configured to VDD, VDDIO_1 is configured to 3.3V and VDDIO_2 is configured to VADJ by placing the jumpers in the respective positions as shown in Figure 2-9.

Figure 2-9. VDDIO Select



### 2.1.6.2 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. Doing 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.

### 2.1.6.3 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.

### 2.1.6.4 P1, P2, P3 and P4 - Processor Module Receptacles

Processor modules provide modularity to this board. Sockets P1-P4 are used to connect a processor module to the board. All supported GPIOs (including special I/Os), along with VDD DIG, VDD ANLG, 5V, 3.3V, 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.

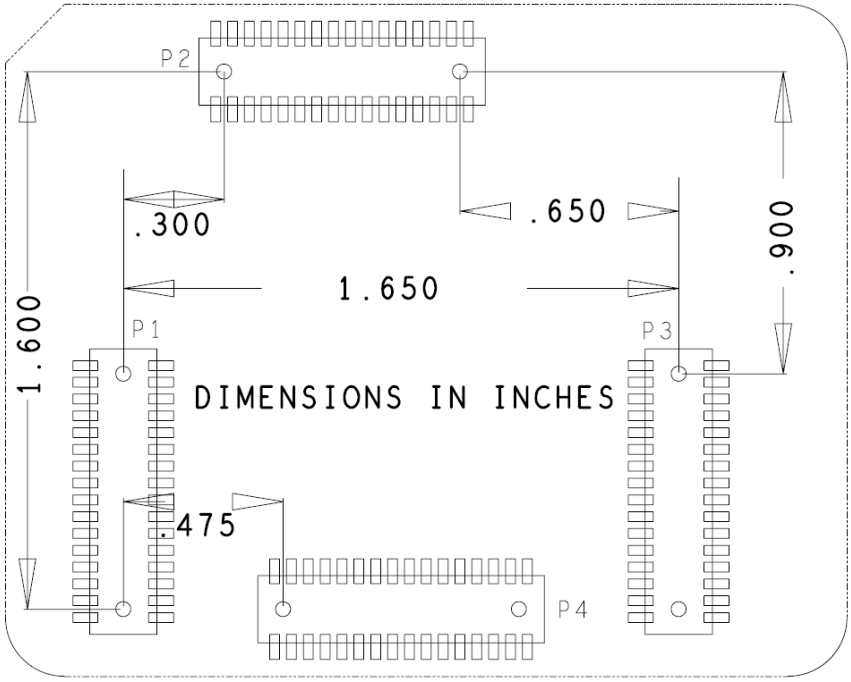
Table 2-4. Connector Pin Assignments - Processor Module Sockets

| 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 |
| 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 |

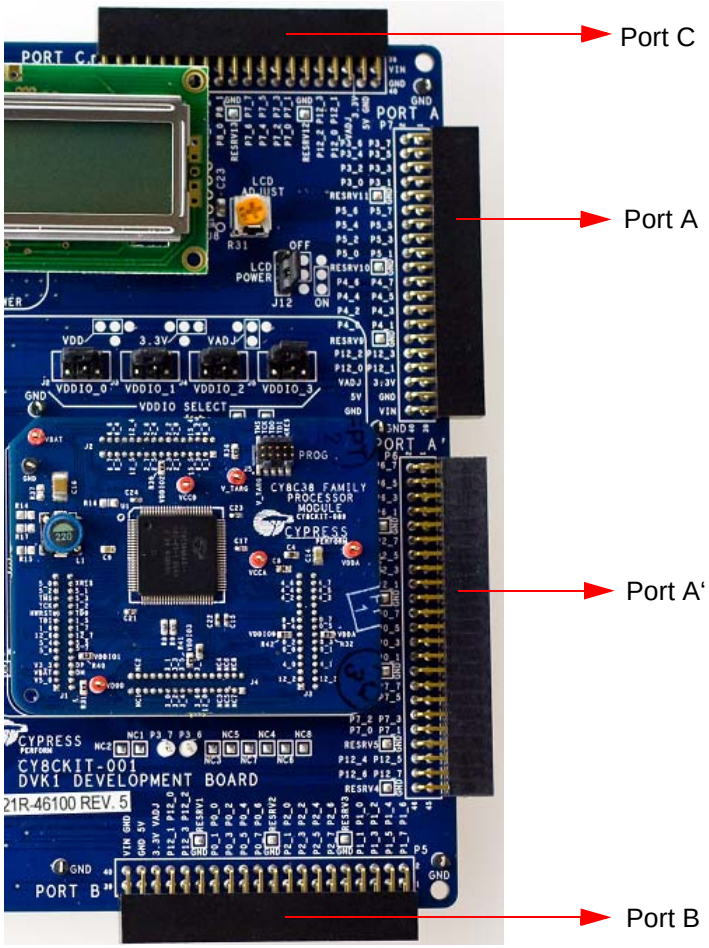
Table 2-4. Connector Pin Assignments - Processor Module Sockets *(continued)*

| Pin Number | P1 (West) | P2 (North) | P3 (East) | P4 (South) |
|---|---|---|---|---|
| 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 2-10. Mechanical Layout Details for Processor Module Connector



### 2.1.7 Expansion Ports




The board accommodates I/O expandability. Around the upper, lower and right sides of the board are 0.100" 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 2-5. 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 |

Table 2-5. Connector Pin Assignments - Expansion Port Sockets *(continued)*

| Pin Number | P5 (PORT B) | P6 (PORT A') | P7 (PORT A) | P8 (PORT C) |
|---|---|---|---|---|
| 10 | RESRV3 | RESRV8 | RESRV11 | RESRV14 |
| 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 |

#### 2.1.7.1 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.

#### 2.1.7.2 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.

#### 2.1.7.3 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.

# 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

You should not perform power cycle mode programming with PSoC Programmer on the CY8CKIT-001. This is due to the way the CY8C38 family module is designed. VTARG of the MiniProg3 is wired exclusively to VDDIO1 of the chip on the module. In order for power cycle programming to work, VTARG would need 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, 5V clamp device capable of shunting 350W of transient power. Each I/O pin is similarly protected by a 5V, 30W device.

## B.5 Level Translation

The design provides level translators that interfaces with any I/O voltage in the range of 1.2V to 5.5V without damage and function properly. There are actually 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 IO voltage levels from 1.5V to 5.5V. In addition, MiniProg3 can provide power to a simple target board, at one of four voltage levels.

Figure C-1. System Block Diagram

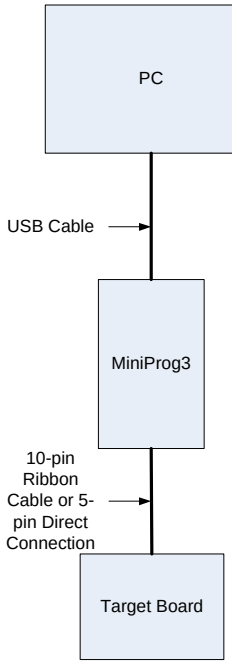



Table C-1. Connectors / Communication Protocol Support

| 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 PSoC1 family of microcontrollers. MiniProg3 supports programming PSoC1 devices through the 5-pin connector only.

For more information about the ISSP interface, refer to the PSoC1 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/ PSoC 5 families. This interface allows a daisy chain bus of multiple JTAG devices. MiniProg3 supports programming and debugging PSoC 3/ 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/ 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 uses. MiniProg3 supports programming and debugging PSoC 3/ 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/ 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/ PSoC 5 JTAG, SWD, SWV, and $I^2C$ interfaces, refer to the PSoC 3/ PSoC 5 Technical Reference Manual. For more information on PSoC 1 interfaces, refer to the PSoC 1 Technical Reference Manual.

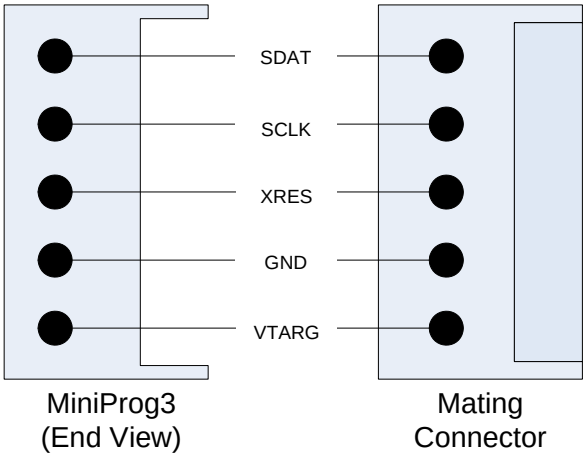
## C.2 Connectors

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

### C.2.1 5-Pin Connector

The 5-pin connector is configured as a single row with 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.

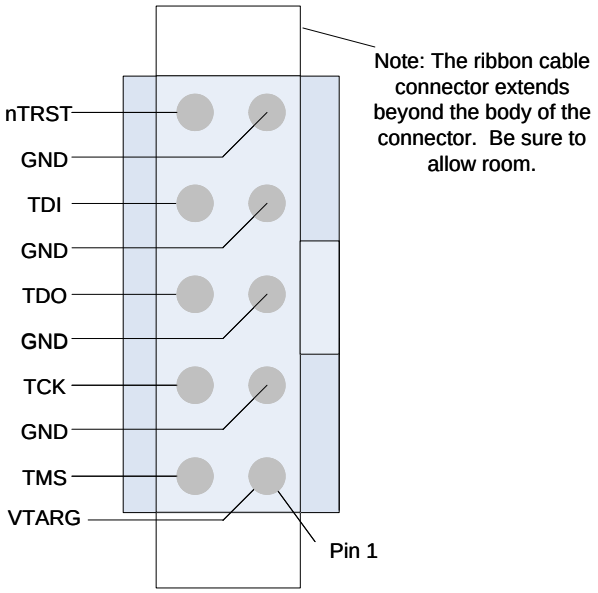
Figure C-2. 5-Pin Connector with Pin Assignments




### C.2.2 10-Pin Connector

The 10-pin connector is configured as a dual row with 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/ 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 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 1.8V, 2.5V, 3.3V, or 5V. The 5V supply may be as low as 4.25V or as high as 5.5V, as it is supplied directly from the USB port.