

## Getting Started with PSoC® Analog Coprocessor

**Author:** Rajiv Badiger

**Associated Part Family:** CY8C4Axxx

**Associated Code Example:** [CE211283](#)

**Related Application Notes:** [AN211294](#)

### More code examples? We heard you.

To access an ever-growing list of hundreds of PSoC code examples, please visit our [code examples web page](#). You can also explore the PSoC video library [here](#).

AN211293 introduces you to the PSoC® Analog Coprocessor and takes you through your first design. The PSoC Analog Coprocessor is a single chip solution that integrates Analog Front Ends (AFEs), ADCs and DACs with a Signal Processing Engine and a host processor communication interface. This application note provides you an overview of additional resources to accelerate your learning.

## Contents

|     |                                              |   |     |                                                   |    |
|-----|----------------------------------------------|---|-----|---------------------------------------------------|----|
| 1   | Introduction.....                            | 2 | 6.2 | About the Design .....                            | 8  |
| 2   | PSoC Resources .....                         | 3 | 6.3 | Part 1: Create the Design .....                   | 9  |
| 3   | PSoC Creator .....                           | 3 | 6.4 | Part 2: Program the Device .....                  | 22 |
| 3.1 | PSoC Creator Help .....                      | 4 | 6.5 | Part 3: Test the Design .....                     | 24 |
| 3.2 | Technical Support.....                       | 4 | 7   | Summary .....                                     | 24 |
| 4   | Code Examples .....                          | 5 | 8   | Related Application Notes and Code Examples ..... | 25 |
| 5   | PSoC Analog Coprocessor Feature Set .....    | 6 |     | Document History.....                             | 27 |
| 5.1 | The Concept of PSoC Creator Components....   | 7 |     | Worldwide Sales and Design Support.....           | 28 |
| 6   | My First PSoC Analog Coprocessor Design..... | 8 |     |                                                   |    |
| 6.1 | Before You Begin.....                        | 8 |     |                                                   |    |

# 1 Introduction

The PSoC Analog Coprocessor simplifies the design of sensor-based systems by delivering a scalable and reconfigurable architecture that integrates programmable analog front ends (AFEs). A signal processing engine (32-bit ARM® Cortex®-M0+) can calibrate and tune the AFE in software.

Additionally, the PSoC Analog Coprocessor enables designs to send aggregated, pre-processed, and formatted sensor data over serial communication interfaces to host processors.

Analog sensors generally come in five different types, depending on their output electrical signal – voltage, current, resistance, capacitance, or inductance. Each sensor type requires a specific AFE design. For example:

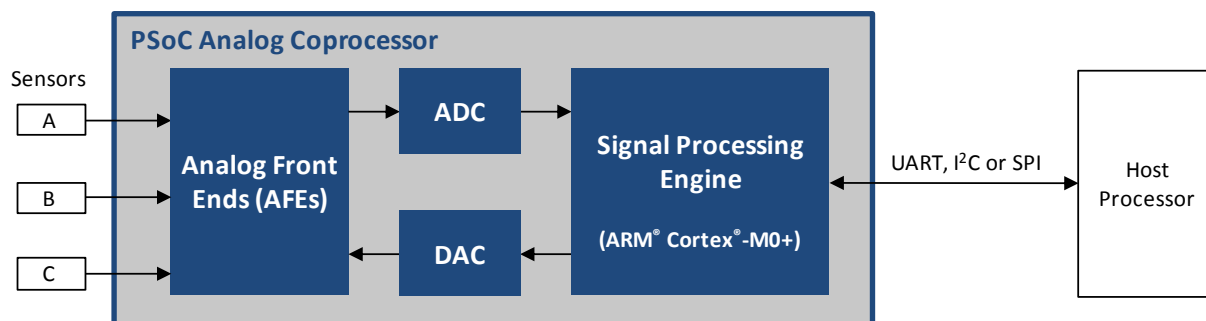
- A thermocouple, which is a voltage-output temperature sensor, requires an instrumentation amplifier
- An ambient light sensor, which is a current-output sensor, requires a trans-impedance amplifier (TIA)

Systems that use multiple analog sensors usually require multiple specialized ICs to implement the AFE, which increases BOM cost and PCB size. Systems designed for IoT applications must combine data from multiple sensors to enable new sensing capabilities, commonly known as sensor fusion. Sensor fusion solutions often require custom AFEs. The PSoC Analog Coprocessor reduces the need for specialized ICs, offering the ability to create custom AFEs in a single-chip solution.

Figure 1 shows a generic block diagram of a sensor-based system. It includes:

1. An analog front end (AFE) to condition the sensor outputs by amplifying and filtering the signals.
2. An analog-to-digital converter (ADC) or a comparator (not shown) to convert the conditioned sensor output into digital data.
3. A programmable signal processing engine with a serial communication interface, to format the sensor data and send it to the host processor.

Figure 1. Sensor-Based System



This application note introduces you to the capabilities of the PSoC Analog Coprocessor and gets you started with a simple design. The design is also available as code example [CE211283](#), for Cypress kit [CY8CKIT-048](#).

The [Related Application Notes and Code Examples](#) section provides a rich set of documents to accelerate your learning. This includes an advanced application note [AN211294, AFE Implementation Using PSoC Analog Coprocessor](#).

## 2 PSoC Resources

Cypress provides a wealth of information data at [www.cypress.com](http://www.cypress.com) to help you to select the right PSoC device for your design, and quickly and effectively integrate the device into your design. For a comprehensive list of resources, see [KBA86521](#), [How to Design with PSoC 3](#), [PSoC 4](#), [PSoC 5LP](#) and [PSoC Analog Coprocessor](#). The following is an abbreviated list:

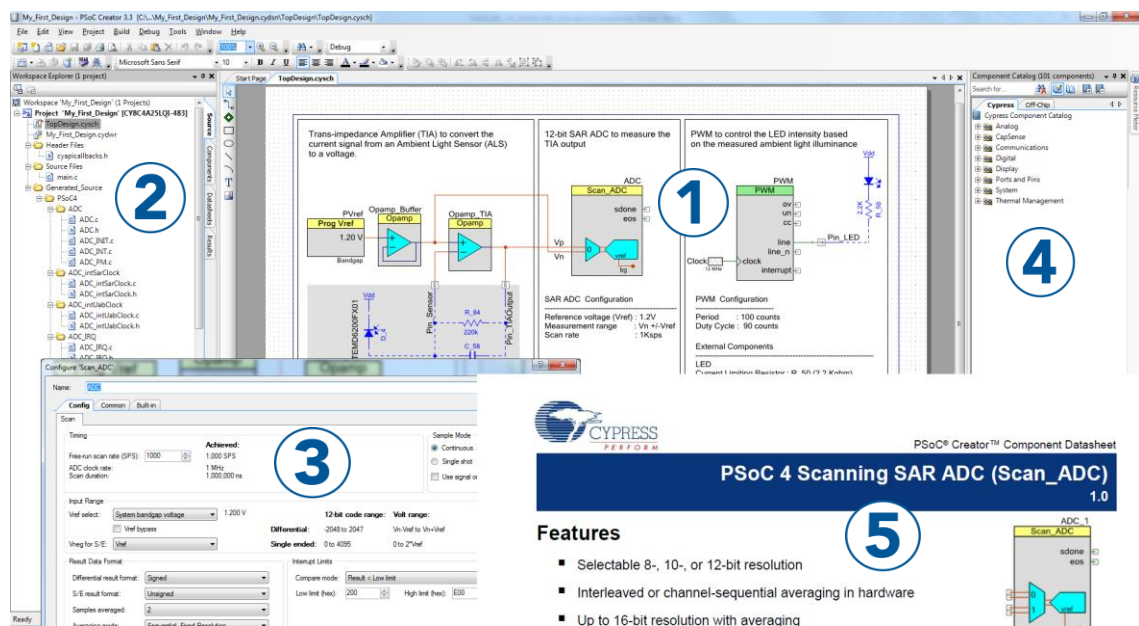
- **Overview:** [PSoC Portfolio](#), [PSoC Roadmap](#)
- **Product Selectors:** [PSoC 1](#), [PSoC 3](#), [PSoC 4](#), [PSoC 5LP](#) or [PSoC Analog Coprocessor](#). In addition, [PSoC Creator](#) includes a device selection tool.
- **Datasheets** describe and provide electrical specifications for the PSoC 3, PSoC 4, PSoC 5LP and PSoC Analog Coprocessor device families.
- **CapSense® Design Guides:** Learn how to design capacitive touch-sensing applications with the [PSoC 3](#), [PSoC 4](#), [PSoC 5LP](#) and [PSoC Analog Coprocessor](#) families of devices.
- **Application Notes** and **Code Examples** cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples.
- **Technical Reference Manuals (TRM)** provide detailed descriptions of the architecture and registers in each of the PSoC 3, PSoC 4, PSoC 5LP, and PSoC Analog Coprocessor device families.
- **Development Kits:** The [CY8CKIT-048](#) PSoC Analog Coprocessor Pioneer Kit is an easy-to-use and inexpensive development platform. This kit can function as a standalone kit or as an Arduino shield.

## 3 PSoC Creator

[PSoC Creator](#) is a free Windows-based Integrated Design Environment (IDE). It enables concurrent hardware and firmware design of systems based on PSoC 3, PSoC 4, PSoC 5LP and PSoC Analog Coprocessor. See [Figure 2](#) – with PSoC Creator, you can:

1. Drag and drop Components to build your hardware system design in the main design workspace
2. Codesign your application firmware with the PSoC hardware
3. Configure Components using configuration tools
4. Explore the library of 100+ Components
5. Review Component datasheets

Figure 2. PSoC Creator Features



### 3.1 PSoC Creator Help

Visit the [PSoC Creator home page](#) to download the latest version of PSoC Creator. Then, launch PSoC Creator and navigate to the following items:

- **Quick Start Guide:** Choose **Help > Documentation > Quick Start Guide**. This guide gives you the basics for developing PSoC Creator projects.
- **System Reference Guide:** Choose **Help > System Reference > System Reference Guide**. This guide lists and describes the system functions provided by PSoC Creator.
- **Component datasheets:** Right-click a Component and select **Open Datasheet**. Visit the [PSoC Analog Coprocessor Component Datasheets page](#) for a list of all PSoC Analog Coprocessor Component datasheets.
- **Document Manager:** PSoC Creator provides a document manager to help you to easily find and review document resources. To open the document manager, choose the menu item **Help > Document Manager**.

### 3.2 Technical Support

If you have any questions, our technical support team is happy to assist you. You can create a support request on the [Cypress Technical Support](#) page.

If you are in the United States, you can talk to our technical support team by calling our toll-free number: +1-800-541-4736. Select option 8 at the prompt.

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

- [Self-help](#)
- [Local sales office locations](#)

## 4 Code Examples

PSoC Creator includes a large number of code example projects. These projects are available from the PSoC Creator Start Page, as Figure 3 shows.

Example projects can speed up your design process by starting you off with a complete design, instead of a blank page. The example projects also show how PSoC Creator Components are used in various applications. Code examples, datasheets (**Documentation** tab) and sample code are included, as Figure 4 shows.

In the Find Example Project dialog shown in Figure 4, you have several options:

- Filter for examples based on architecture or device family, i.e., PSoC 3, PSoC 4, PSoC 5LP or PSoC Analog Coprocessor; category; or keyword
- Select from the menu of examples offered based on the Filter Options
- Review the datasheet for the selection (on the **Documentation** tab)
- Review the code example for the selection. You can copy and paste code from this window to your project, which can help speed up code development, or
- Create a new project (and a new workspace if needed) based on the selection. This can speed up your design process by starting you off with a complete, basic design. You can then adapt that design to your application.

Figure 3. Code Examples in PSoC Creator

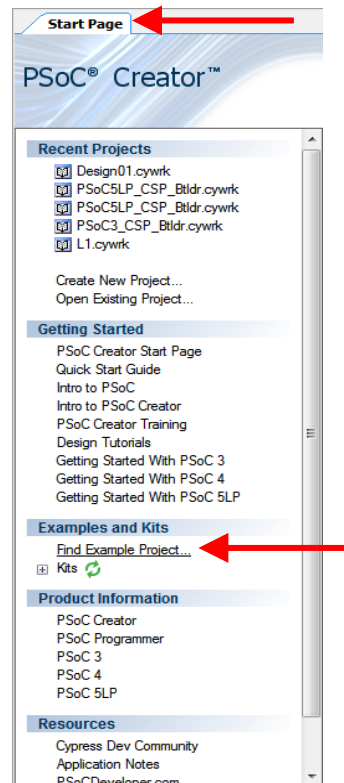
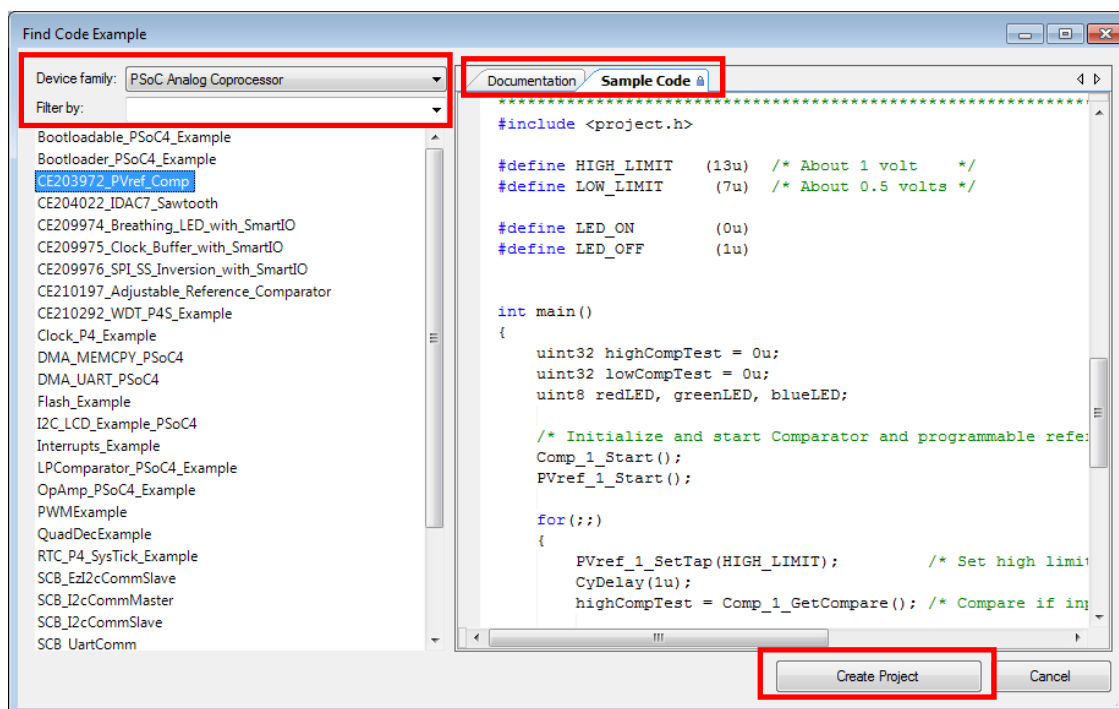


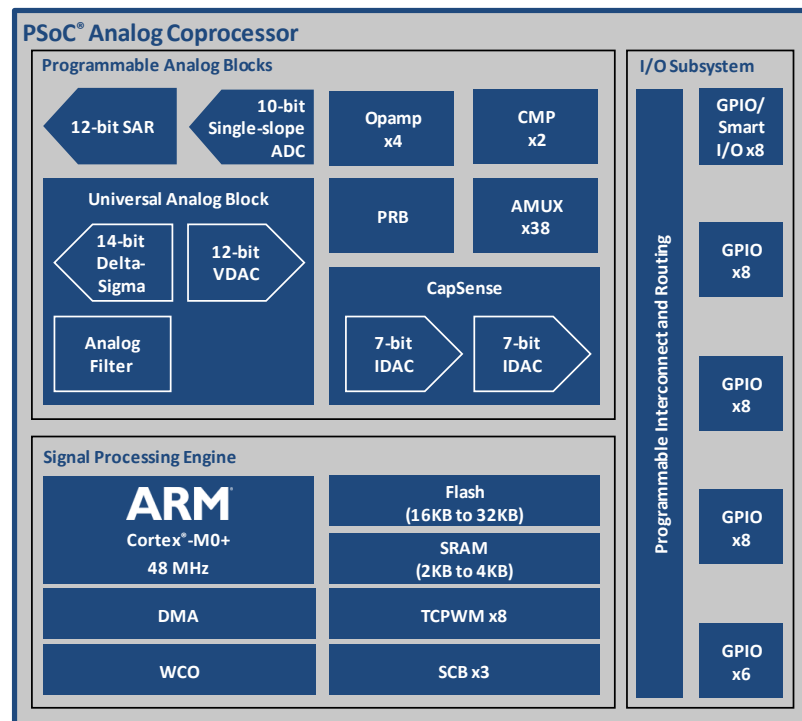
Figure 4. Code Example Projects, with Sample Code



## 5 PSoC Analog Coprocessor Feature Set

The PSoC Analog Coprocessor has an extensive set of analog features, and other resources, as [Figure 5](#) shows:

Figure 5. PSoC Analog Coprocessor Block Diagram



Below is a list of major features of the PSoC Analog Coprocessor. For more information, see the [device datasheet](#), [Technical Reference Manual \(TRM\)](#), and [Related Application Notes and Code Examples](#).

### Operating Range and Low-Power Modes

- Device operating voltage 1.71 V to 5.5 V
- Sleep mode gates off clocks to the CPU. 3.1 mA typical current at 12 MHz.
- Deep-Sleep mode with operational analog. 2.5  $\mu$ A typical current.

### Programmable Analog Blocks

- Universal Analog Block (UAB)

The UAB can be configured as one of the following:

- 12-bit buffered voltage DAC (VDAC), with a sample rate of 500 kHz
- 2<sup>nd</sup>-order bi-quad filter, as a low-pass, high-pass, band-pass, or notch filter
- 12-bit delta-sigma ADC, with a sample rate of 7.8 Ksps and a DNL of  $\pm 1$  LSB<sup>1</sup>
- 14-bit incremental delta-sigma ADC, with a sample rate of 100 sps and a DNL of  $\pm 2$  LSB<sup>1</sup>

- Four programmable Opamps
  - 90 dB open-loop gain, rail-to-rail operation
  - Can be used with external components to form standard Opamp circuits
  - Can use an internal resistor array to form a programmable gain amplifier (PGA) with gain up to 32
  - 6 MHz gain-bandwidth when driving external I/Os, with up to 10 mA drive
  - 8 MHz gain-bandwidth when driving internal nodes such as the SAR ADC
  - $\pm 1$  mV input offset voltage
  - 15  $\mu$ A operating current in Deep-Sleep mode
- Two low power comparators (CMP)
  - Wake up the device from low-power modes

<sup>1</sup> Component support for feature to be made available 2nd half 2016.

- 12-bit SAR ADC
  - Sample rate up to 1 Msps
  - Selectable resolution 8-, 10-, or 12-bit
  - Automated hardware sequencer with 16 input channels
  - Each channel can be differential or single-ended
  - Integrated hardware averaging per channel
  - Programmable input channels, for example external pins, Opamps, and UAB
- Single-Slope ADC
  - Selectable 8- or 10-bit resolution
  - Sample rate up to 11.6 ksps with 10-bit resolution
  - Input measurement range from  $V_{SS}$  to  $V_{DDA}$  on any GPIO pin
  - Implemented in the CapSense block
- Programmable Reference Block (PRB)
  - Four voltage references, independently adjustable in 16 steps, from  $V_{DDA}$  to  $V_{SS}$ , or 1.2 V to  $V_{SS}$
  - References can be routed to internal high-impedance analog resources: ADC, VDAC, comparator, Opamps.
  - References can also be routed to a GPIO if buffered through an Opamp
- CapSense®
  - Measures capacitance; can be used with capacitive sensors, for example liquid level or touch sensing applications
  - Self and mutual capacitive sensing methods
  - Improved electromagnetic interference (EMI) using spread spectrum clock and programmable slew rate control
- IDACs
  - Two 7-bit current DACs (IDACs) for use with CapSense or for general purpose applications
  - A single 8-bit IDAC can be created by combining the two IDACs in parallel
  - 37.5 nA LSB current, for precise capacitance measurements
  - Six output current ranges (4.76  $\mu$ A to 609  $\mu$ A), in source or sink configuration

### 32-bit Signal Processing Engine

- ARM Cortex-M0+ CPU, operating at up to 48 MHz
- Up to 32 KB of flash, with read accelerator
- Up to 4 KB of SRAM
- Eight-channel direct memory access (DMA) controller
- Watch crystal oscillator (WCO) for real-time clock (RTC) applications
- Three serial communication blocks (SCBs), each configurable as SPI, I<sup>2</sup>C, or UART
- Eight 16-bit timer / counter / pulse-width modulator (TCPWM) blocks

### I/O Subsystem

- Up to 38 GPIOs that can be used for analog, digital, CapSense, or Segment LCD functions
- Programmable drive modes and slew rates
- Eight Smart I/Os, that can implement boolean operations on pin signals

## 5.1 The Concept of PSoC Creator Components

The key to successful PSoC designs is the [PSoC Creator IDE](#). PSoC Creator encapsulates PSoC peripherals and other resources as graphical elements called Components. Components are dragged and dropped onto a schematic, and wired together, making the hardware design process fast and easy. Design changes can be quickly made with just a few mouse clicks.

The following Components are provided for designs using the Programmable Analog Blocks:

- Programmable Gain Amplifier (PGA) and Opamp
- Voltage DAC (VDAC), implemented in the UAB
- Scanning SAR ADC, implemented with the SAR ADC
- Programmable Vref (PVref), implemented in the PRB
- CapSense ADC, implemented with single-slope ADC

Each Component has a datasheet; for more information on a Component, refer to its datasheet. Visit the [PSoC Analog Coprocessor Component Datasheets page](#) for a list of all PSoC Analog Coprocessor Component datasheets.

## 6 My First PSoC Analog Coprocessor Design

This section helps you to build a simple sensor-based AFE design and program it into a development kit. It provides detailed steps that make it easy to learn PSoC design techniques using the PSoC Creator IDE.

### 6.1 Before You Begin

#### 6.1.1 Have you installed PSoC Creator?

Download and install PSoC Creator from the [PSoC Creator home page](#). Support for the PSoC Analog Coprocessor family is available in PSoC Creator 3.3 SP2 and later revisions.

#### 6.1.2 Do you have a development kit?

This design is developed for the [CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit](#). This kit offers footprint-compatibility with Arduino™ shields and baseboards. It features five onboard sensors, an RGB LED, two push-button switches, a Cypress F-RAM™, and KitProg2 – an onboard programmer/debugger and USB-UART/I2C bridge.

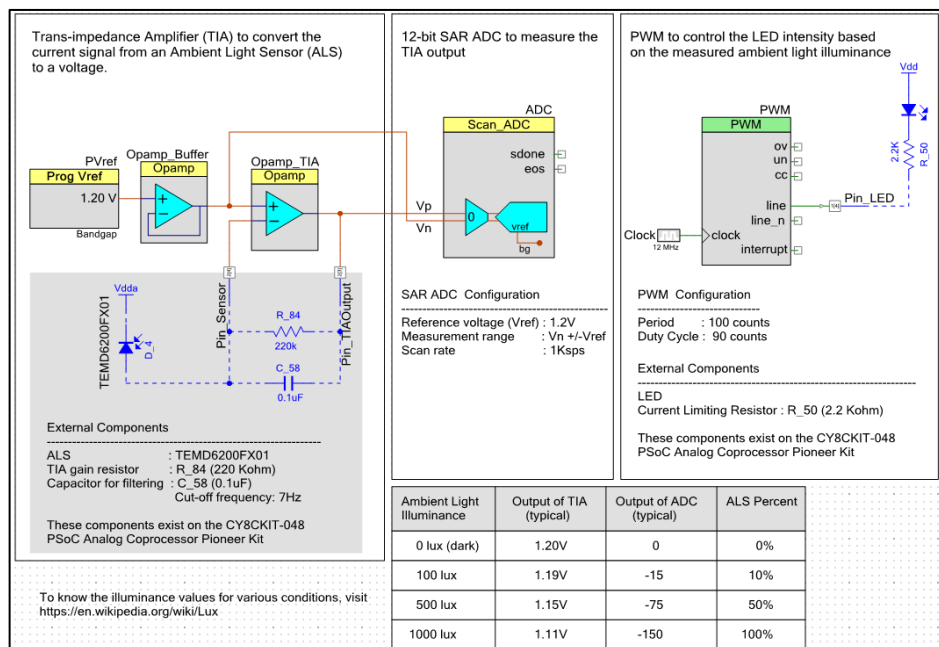
#### 6.1.3 Want to see the project in action?

If you don't want to go through the design process, you can get the completed code example project at <http://www.cypress.com/CE211283>. You can then jump to the [Build and Program](#) steps.

### 6.2 About the Design

This design implements a simple analog conditioning circuit for an ambient light sensor. The output from an ambient light sensor is a weak current signal that is proportional to the ambient light illuminance. [Figure 6](#) shows the PSoC Creator schematic to condition and measure this current signal.

Figure 6. My First PSoC Analog Coprocessor Design



The current output from the ambient light sensor is converted to a voltage signal using a trans-impedance amplifier (TIA), made using one of the Opamps and external passive components. The reference voltage for the TIA is set to 1.20 V from a PRB block. The output of the TIA is measured using the 12-bit SAR ADC, and calibrated in terms of percentage for an ambient light illuminance range of 0 to 1 Klux. This percentage value is used as the duty cycle of the PWM. An LED connected to the output of the PWM shows the variation in brightness in proportion to the ambient light illuminance.

**Note:** The CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit has the required ambient light sensor on the board, connected to the PSoC Analog Coprocessor. No extra components are required for testing the design.

### 6.3 Part 1: Create the Design

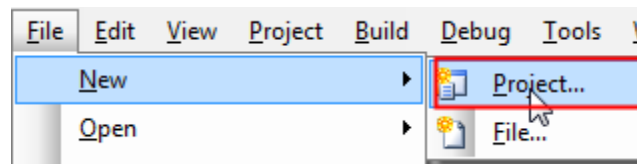
This section takes you step by step through the design process. It guides you through both hardware and firmware design entry.

**Note:** These instructions assume that you are using PSoC Creator 3.3 SP2 or higher. The overall development process is the same for subsequent versions of PSoC Creator; however, some of the dialog boxes may be different.

1. Create a new PSoC Creator project.
  - a. Start PSoC Creator.
  - b. Select menu item **File > New > Project**, as [Figure 7](#) shows.

A Create Project window is displayed.

Figure 7. Create a New PSoC Creator Project



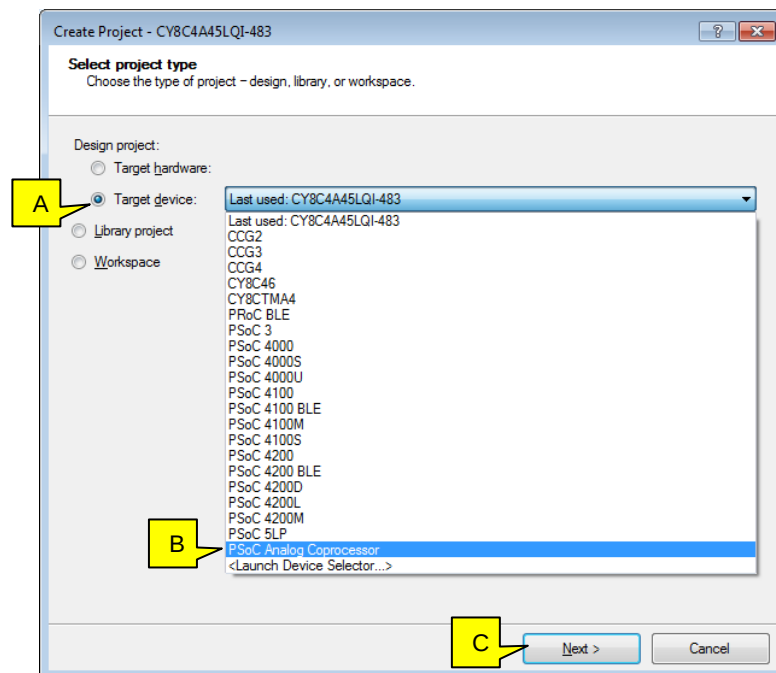
2. Select **PSoC Analog Coprocessor** as the target device, as [Figure 8](#) shows.

PSoC Creator can speed up the development process by automatically setting various project options for specified development kits or target devices.

- A. Click **Target device**.
- B. In the pull down menu, select **PSoC Analog Coprocessor**.
- C. Click **Next**.

PSoC Creator selects CY8C4A45LQI-483 as the default device in the PSoC Analog Coprocessor family. This device is mounted on the CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit.

Figure 8. Selecting Target Device



3. Choose the Empty Schematic, as Figure 9 shows.
  - A. Click **Empty Schematic**.
  - B. Click **Next**.
  - C. In the next dialog, enter text for a **Workspace name**, as Figure 10 shows. A workspace name is a container for one or more projects.
  - D. Specify the **Location** of your workspace.
  - E. Enter text for a **Project name**. The project and workspace names can be same or different.
  - F. Click **Finish**.

Figure 9. Selecting Schematic Template

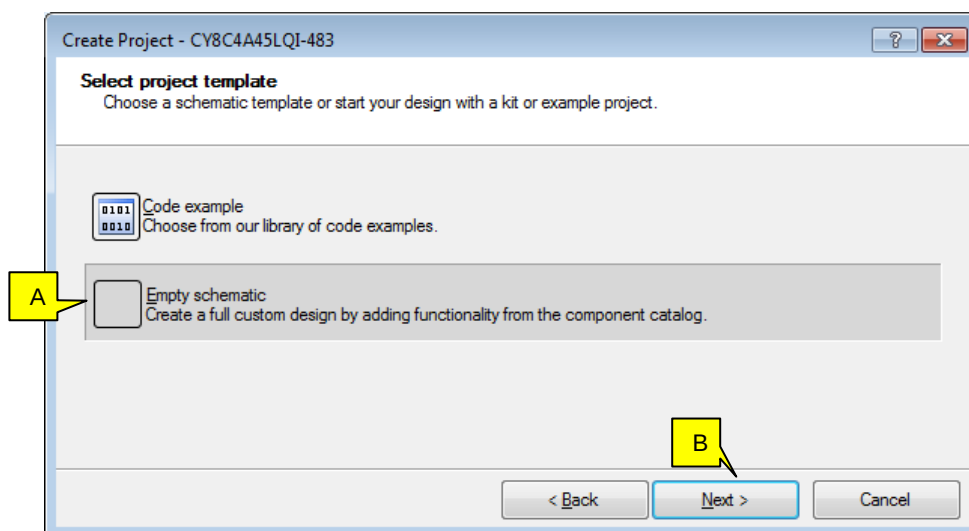
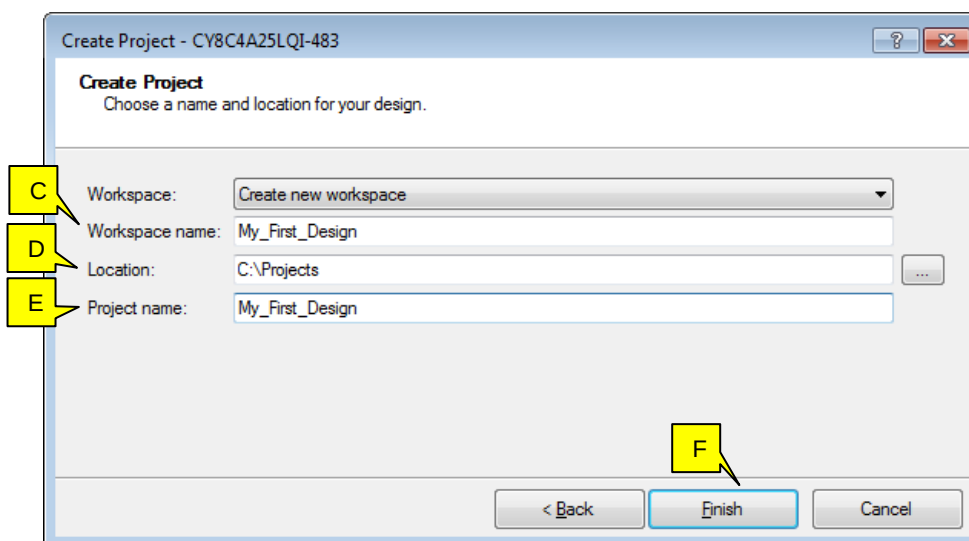
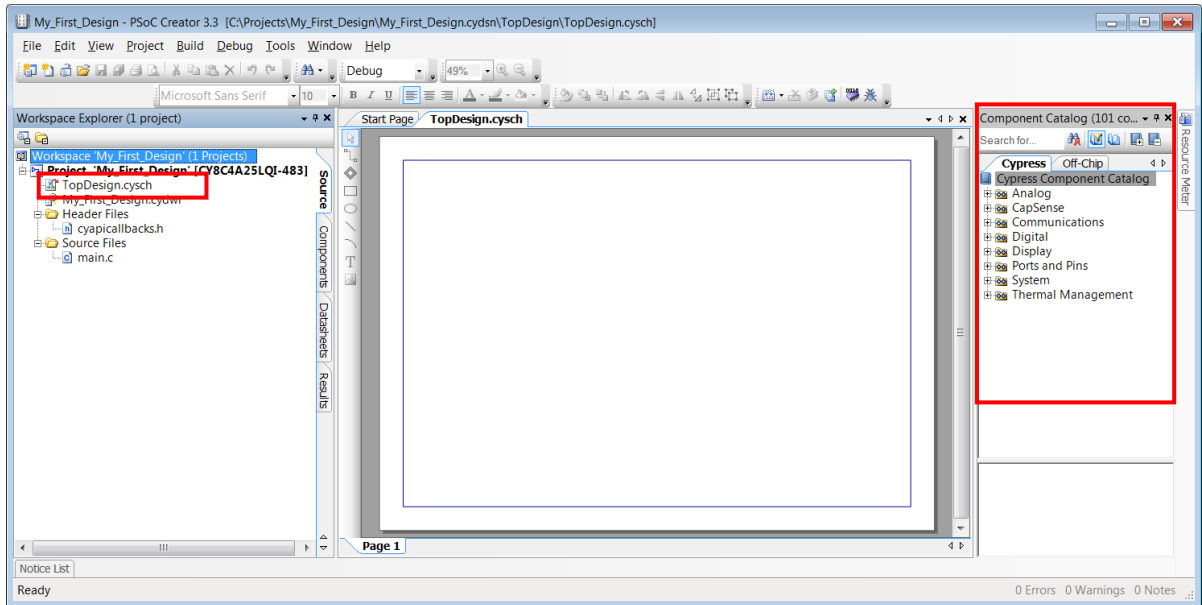


Figure 10. Project Naming and Location



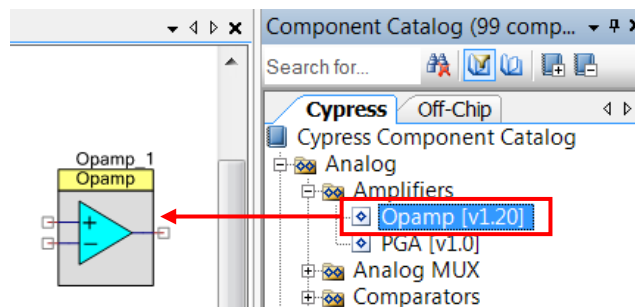
- The schematic of the project opens by default, as [Figure 11](#) shows. This is where the hardware design schematic is created. Note the associated file *TopDesign.cysch* in the **Workspace Explorer** window. If the schematic is not displayed, double-click the file to open the schematic.

Figure 11. Schematic



- Because “Empty Schematic” was selected in step 3, the Top Design is blank. The Component catalog is on the right side of the window, as [Figure 11](#) shows. If it is not displayed, open it from menu item **View > Component Catalog**. You are now ready to start placing Components.
- Start by creating a trans-impedance amplifier (TIA) in the schematic. Select and place (drag and drop) an Opamp Component from the Component Catalog, as [Figure 12](#) shows. Find the Opamp Component in the Analog group, Amplifiers subgroup.

Figure 12. Opamp Component Selection



By default, the instance name of the Component is Opamp\_1. The Component is configured with default properties, for example, medium power setting. In the next step, we change the configuration for this application.

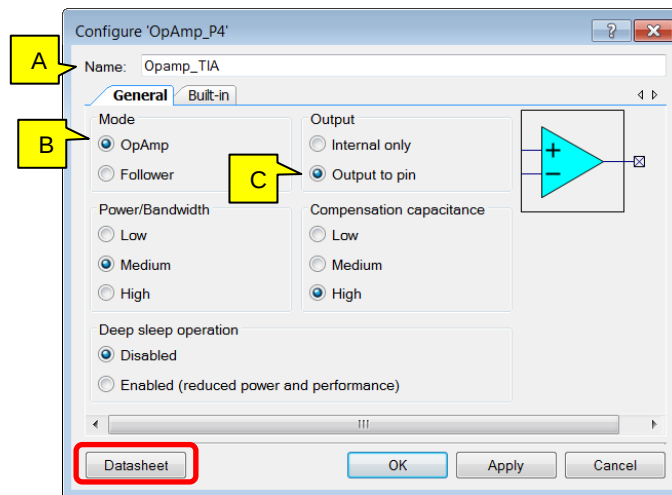
7. Double-click the Opamp Component to configure it. Configure the Component as [Figure 13](#) shows.
  - A. Name the component “Opamp\_TIA”.
  - B. Ensure that **OpAmp** mode is selected (default setting).
  - C. Enable the **Output to pin** option, to route the Opamp output to a pin.

Each Component has an associated datasheet that can be accessed from the configuration window, as [Figure 13](#) shows. The Component datasheet provides more information on the Component configuration, the application programming interface (API), and the electrical specifications.

Leave the rest of the settings at their default values. Refer to the Component datasheet to learn the significance of each setting.

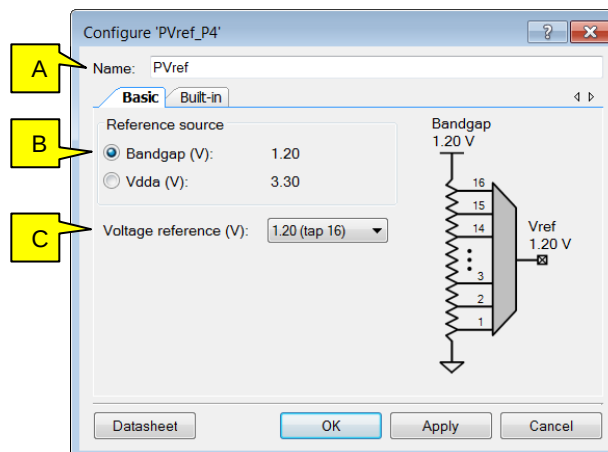
Click **OK** to apply the settings and close this window.

Figure 13. Opamp Component Configuration



8. Similarly, place a PVref Component (available in the Analog group in the Component Catalog) and configure its parameters, as [Figure 14](#) shows. This Component sets the reference of the TIA and the negative input of the SAR ADC to 1.20V.
  - A. Name the Component “PVref”.
  - B. Set the Reference source to **Bandgap (V) 1.20** (default).
  - C. Set the Voltage reference (V) to **1.20 (tap 16)** (default).

Figure 14. Programmable Reference Configuration

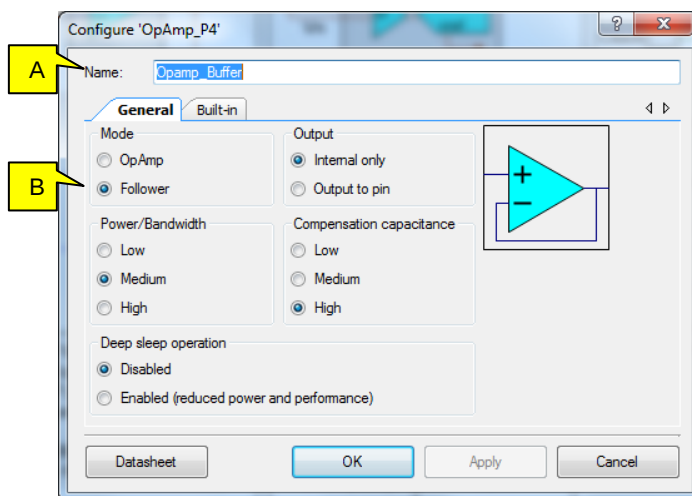


9. Place an Opamp Component (available in the Analog group, Amplifiers subgroup in the Component Catalog) and configure its parameters, as Figure 15 shows. This Component is used to buffer the reference voltage generated by the PVref Component.

- A. Name the component “Opamp\_Buffer”.

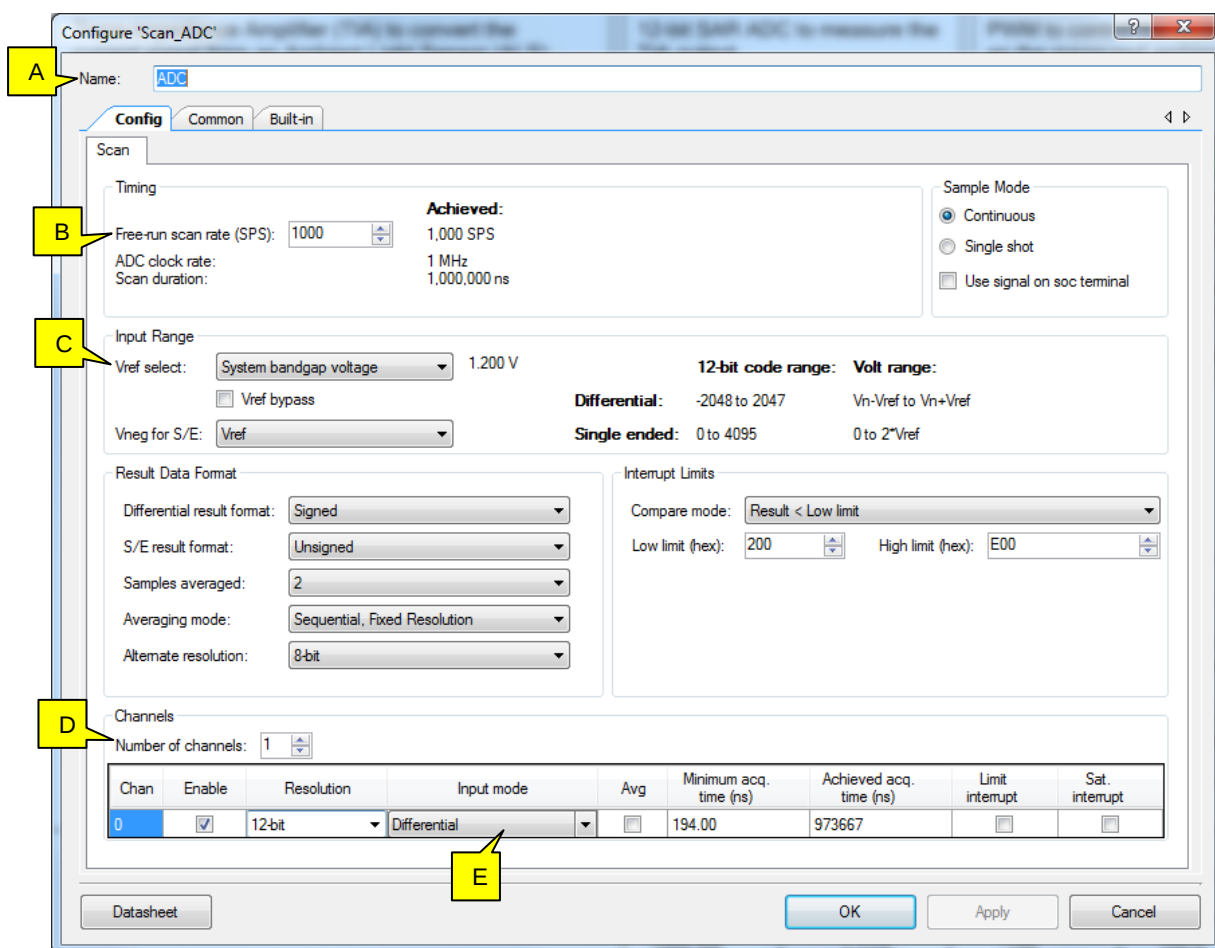
- B. Set the Mode to **Follower**.

Figure 15. Opamp\_Buffer Configuration



10. Place a Scanning SAR ADC (available in the Analog group, ADC subgroup in the Component Catalog), and configure the parameters as Figure 16 shows.
  - A. Name the Component “ADC”.
  - B. Set the Free-run scan rate (SPS) to **1000**.
  - C. Set the Vref select to **System bandgap voltage (1.200 V)**. The measurement range is  $\pm 1.2$  V relative to the voltage at the negative input of the SAR ADC.
  - D. Set the Number of channels to **1**, because only the TIA output needs to be measured.
  - E. Set the Input mode to **Differential** (default).

Figure 16. SAR ADC Configuration - Configuration Tab



Configure 'Scan\_ADC'

Name: **ADC**

**Config** Common Built-in

**Scan**

**Timing**

Free-run scan rate (SPS): **1000** **Achieved:** 1,000 SPS

ADC clock rate: 1 MHz

Scan duration: 1,000,000 ns

**Sample Mode**

☒ Continuous

☐ Single shot

☐ Use signal on soc terminal

**Input Range**

Vref select: **System bandgap voltage** 1.200 V

☐ Vref bypass

Vneg for S/E: **Vref**

**12-bit code range:** **Differential:** -2048 to 2047 **Volt range:** Vn-Vref to Vn+Vref

**Single ended:** 0 to 4095 0 to 2\*Vref

**Result Data Format**

Differential result format: **Signed**

S/E result format: **Unsigned**

Samples averaged: **2**

Averaging mode: **Sequential, Fixed Resolution**

Alternate resolution: **8-bit**

**Interrupt Limits**

Compare mode: **Result < Low limit**

Low limit (hex): **200** High limit (hex): **E00**

**Channels**

Number of channels: **1**

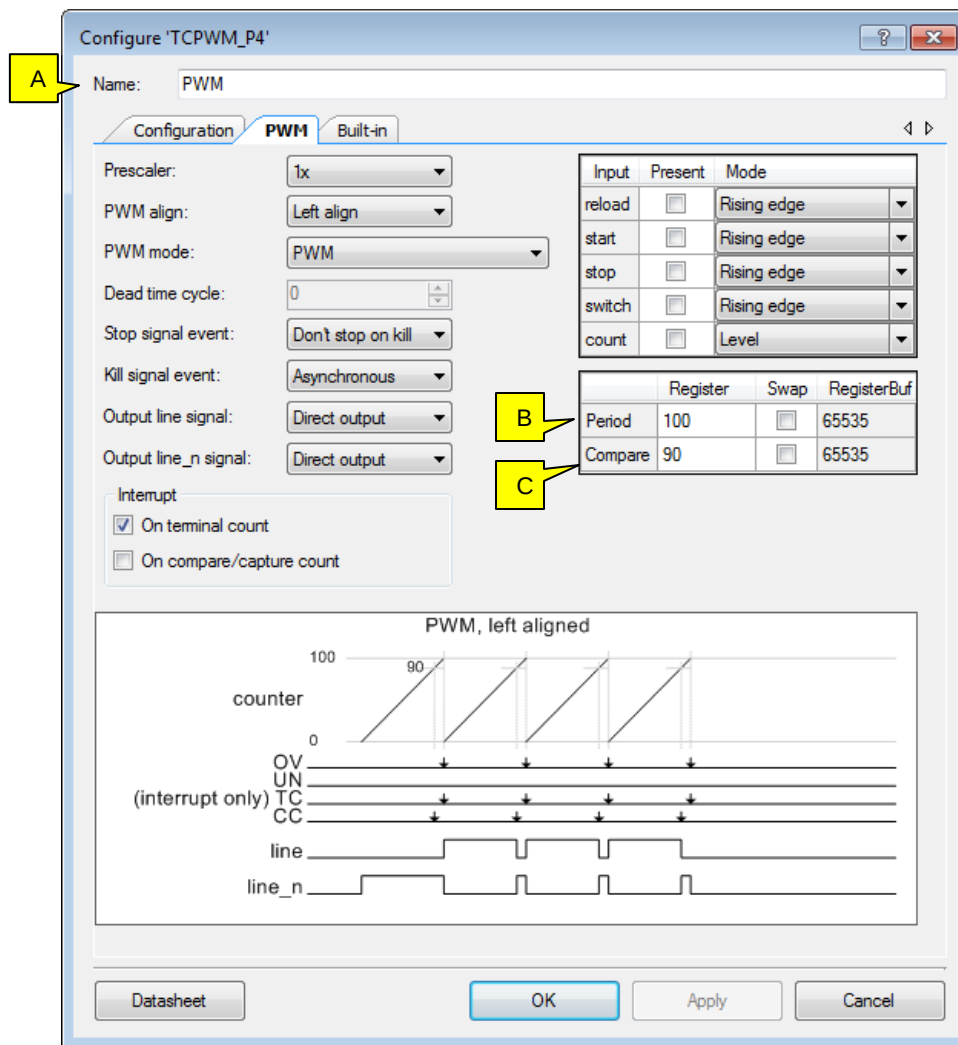
| Chan | Enable                              | Resolution | Input mode          | Avg                      | Minimum acq. time (ns) | Achieved acq. time (ns) | Limit interrupt          | Sat. interrupt           |
|------|-------------------------------------|------------|---------------------|--------------------------|------------------------|-------------------------|--------------------------|--------------------------|
| 0    | <input checked="" type="checkbox"/> | 12-bit     | <b>Differential</b> | <input type="checkbox"/> | 194.00                 | 973667                  | <input type="checkbox"/> | <input type="checkbox"/> |

Datasheet OK Apply Cancel

11. Place a PWM Component, available in the Digital group, Functions subgroup in the Component Catalog. This Component is used to drive the LED. Configure the Component as Figure 17 shows.
  - A. Name the Component “PWM”.
  - B. Set the Period value to **100**. This allows the PWM output duty cycle to be controlled directly in percentage units, ranging from 0 to 100%.
  - C. Set the Compare value to **90**, for an initial output duty cycle of 90%. The value is changed in firmware at run time.

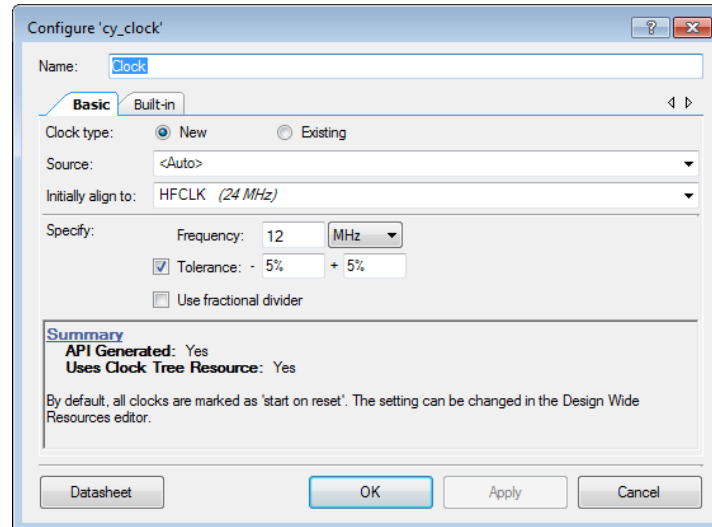
The PWM output frequency depends on the Clock input which is configured in the next step.

Figure 17. PWM Component Configuration



12. Place a Clock Component, available in the System group in the Component Catalog. This Component is used to drive the clock to the PWM Component. Name the Component “Clock”, and leave the default frequency setting at 12 MHz, as [Figure 18](#) shows. With the PWM period set to 100, as configured in the previous step, the output PWM frequency is 120 kHz.

Figure 18. Clock Component Configuration



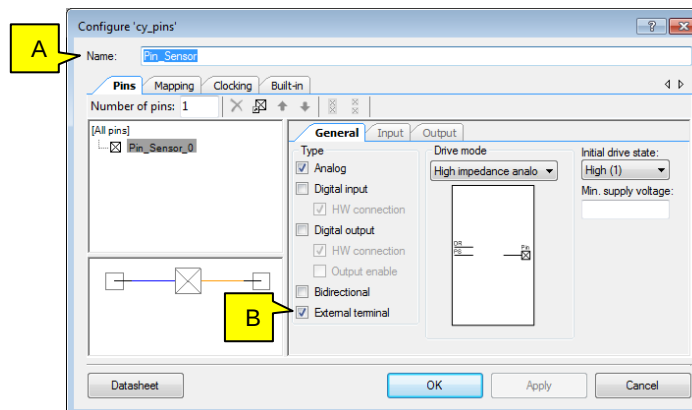
13. Place and configure pins:

First, place and configure a pin for the TIA input. As it is an analog input, select an Analog Pin Component, which is available in the Ports and Pins group in the Component Catalog.

- A. Name the component “Pin\_Sensor”
- B. Enable **External terminal**. This allows you to place and connect external (Off-Chip) components. External components on the schematic are included for clarification; they have no effect on the design.

[Figure 19](#) shows the Pin\_Sensor configuration.

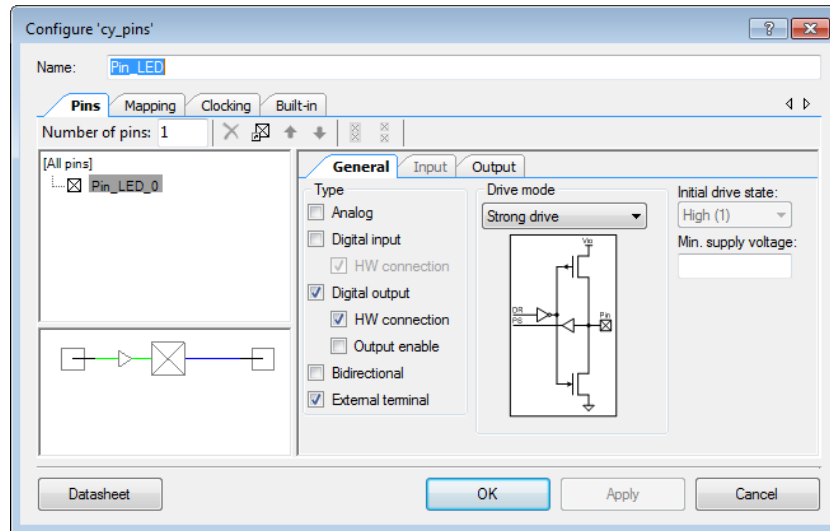
Figure 19. Pin\_Sensor Component Configuration



Next, place another Analog Pin Component for the TIA Output, similar to Pin\_Sensor. Name the Component “Pin\_TIAOutput”, and enable the **External terminal**.

Place a digital output pin for driving the LED. Name the Pin “Pin\_LED”, and enable the **External terminal**, as Figure 20 shows. Leave the other settings at their default values.

Figure 20. Pin\_LED Configuration



14. As explained in the previous step, the Off-Chip Components are optional on the schematic. They are available from the Off-Chip tab in the Component Catalog, as Figure 21 shows. If you want to include Off-Chip components, place the Components and configure their values as Table 1 shows.

Figure 21. Off-Chip Components

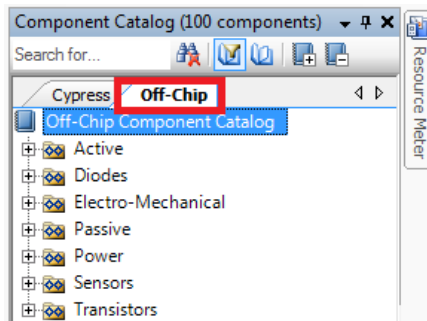


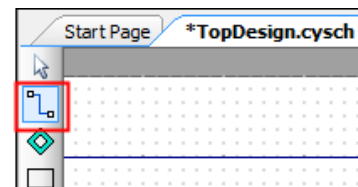
Table 1. Pin and External Components Configuration

| Component   | Component Catalog |         |             |
|-------------|-------------------|---------|-------------|
|             | Tab               | Group   | Value       |
| Photo Diode | Off-Chip          | Diodes  | -           |
| Resistor    | Off-Chip          | Passive | 220 K       |
| Capacitor   | Off-Chip          | Passive | 0.1 $\mu$ F |
| Resistor    | Off-Chip          | Passive | 2.2 K       |
| LED         | Off-Chip          | Diodes  | -           |
| Vdd         | Off-Chip          | Power   | -           |
| Ground      | Off-Chip          | Power   | -           |

This completes Component placement and configuration. The next step is to connect the Components together.

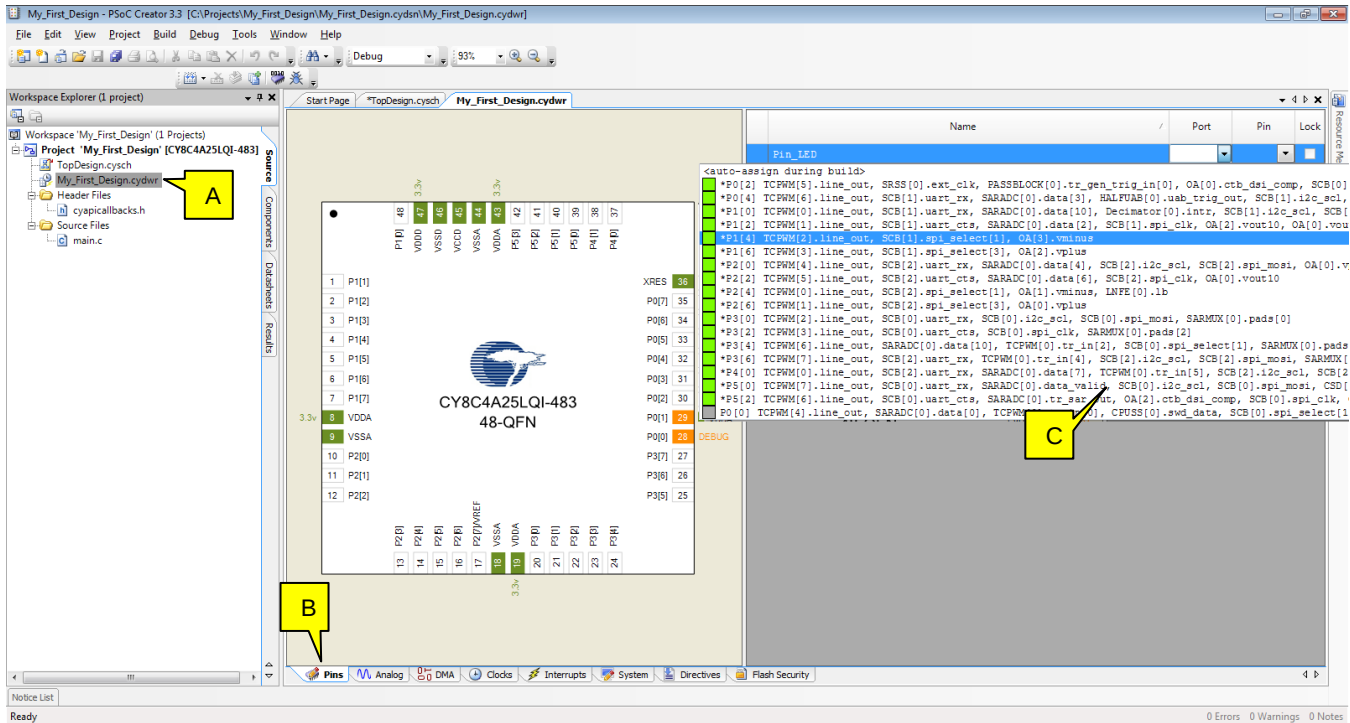
15. Select the wire tool (Figure 22) (also available by pressing ‘w’ as a shortcut). Wire the Component terminals together, as Figure 6 shows.

Figure 22. Wire Tool



16. At this point, the hardware design is complete; however, the Pin Components must still be associated with physical pins. Choose the physical pin for the development kit that you are using.
  - A. In the Workspace Explorer window, double-click the file *My\_First\_Design.cydwr*.
  - B. Select the **Pins** tab. The pins used in the design appear in the list.
  - C. Select the desired physical pin for each Pin Component used in the design, as Figure 23 shows.

Figure 23. Pin Assignment



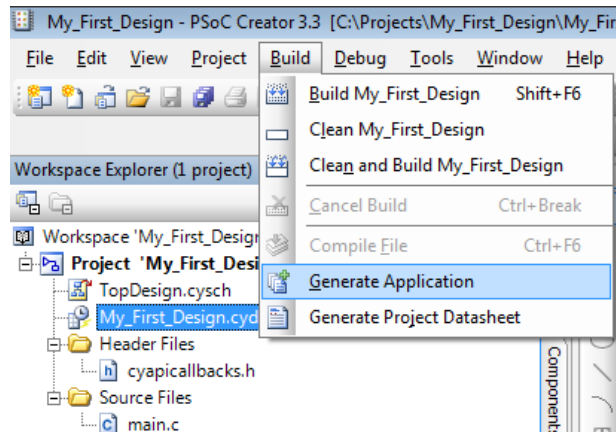
If you are using the CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit, the pins should be set as per Table 2:

Table 2. Physical Pin Assignments for CY8CKIT-048

| Pin Component Name | Physical Pin |
|--------------------|--------------|
| Pin_Sensor         | P2[4]        |
| Pin_TIAOutput      | P2[3]        |
| Pin_LED            | P1[4]        |

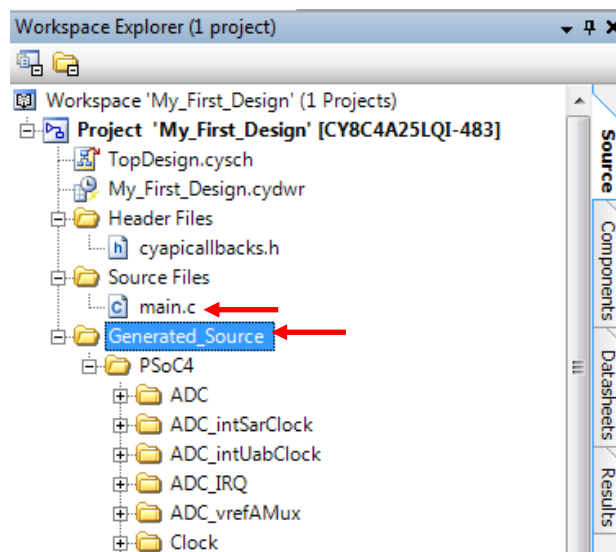
17. Now that the hardware configuration settings are done, it is time to add the firmware code. However, before doing so, it is best to have PSoC Creator generate the API code that is associated with the Components. Select **Build > Generate Application**, as [Figure 24](#) shows. If there are no errors, PSoC Creator generates several code files in the folder *Generated\_Source*, as [Figure 25](#) shows.

Figure 24. Generate Application



Open the auto-generated file *main.c* from the workspace, as [Figure 25](#) shows.

Figure 25. Generated Source Files



Add the code in [Code 1](#) to the *main.c* file in your project. The code does the following tasks:

- Initializes the Components using the Component API functions, for example, `Opamp_TIA_Start()`
- Reads and filters the ADC conversion result
- Converts the filtered ADC result into a percentage value
- Updates the PWM duty cycle with the calculated percentage value

Code 1. *main.c*

```
#include <project.h>
#include "stdio.h"

/* IIR Filter Coefficient */
#define FILTER_COEFFICIENT_ALS 8

/* ADC Channel for ALS - Channel 0 */
#define ALS_CHANNEL 0x00

/* ADC Counts at 1K lux illuminance, calculated based on the sensor TEMD6200FX01 datasheet */
#define ADCCOUNTS_1K_LUX 0x96

int main()
{
    /* Variables */
    int ADCResult, ADCFiltOut=0, ALSPercent;

    /* Enable global interrupt */
    CyGlobalIntEnable;

    /* Start the TIA */
    Opamp_TIA_Start();

    /* Start the Reference Voltage */
    PVref_Start();

    /* Start the Opamp buffer */
    Opamp_Buffer_Start();

    /* Start the SAR ADC; continuous conversions */
    ADC_Start();
    ADC_StartConvert();

    /* Start the PWM */
    PWM_Start();

    for(;;)
    {
        /* ADC scan rate is set to 1ksps */
        /* Check if the ADC result is ready */
        if(ADC_IsEndConversion(ADC_RETURN_STATUS))
        {
            /* Get the sign extended 16 bit result with 11 bits of magnitude */
            ADCResult = ADC_GetResult16(ALS_CHANNEL);

            /* IIR Filter - sample rate: 1 ksps */
            /* Weight on the new sample is 1/8, and weight on the previous filter output is 7/8 */
            ADCFiltOut = (ADCResult + (FILTER_COEFFICIENT_ALS - 1) * ADCFiltOut) /
                FILTER_COEFFICIENT_ALS;

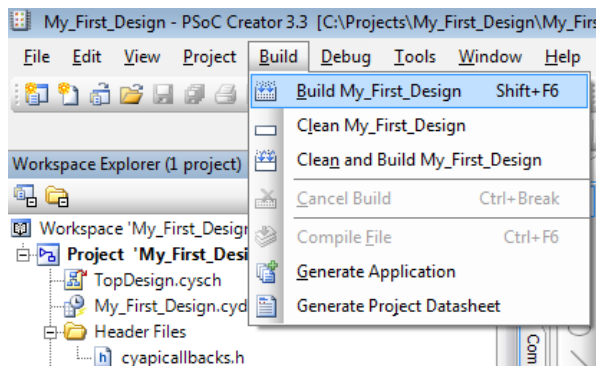
            /* Convert filtered data into percentage. Filter output is multiplied by -1, so that
             the percentage value is directly proportional to the ambient light illuminance */
            ALSPercent = (-1*ADCFiltOut*100) / ADCCOUNTS_1K_LUX;

            /* Limit the values between 0 to 100 to express the ambient light
             illuminance in percentage for a given operating window (0 to 1 Klux)*/
            ALSPercent = (ALSPercent > 100) ? 100 : ((ALSPercent < 0) ? 0 : ALSPercent);

            /* Update PWM duty cycle. The LED has an active low connection on the CY8CKIT-048 kit. */
            PWM_WriteCompare(PWM_PWM_PERIOD_VALUE - ALSPercent);
        }
    }
}
```

18. If you skipped to this step without going through the design process, do the following:
  - A. Download the code example file *CE211283.zip* from <http://www.cypress.com/CE211283>, and extract it to a convenient location in your computer.
  - B. Download and install PSoC Creator as described in section 6.1.1.
  - C. Open the *CE211283.cywrk* file.
  - D. Confirm that the project pin assignments match your development kit (DVK). See Table 2 for CY8CKIT-048.
  - E. Select **Build > Build <project name>**, as Figure 26 shows. If there are no errors, the project is built and ready to program into the target DVK.

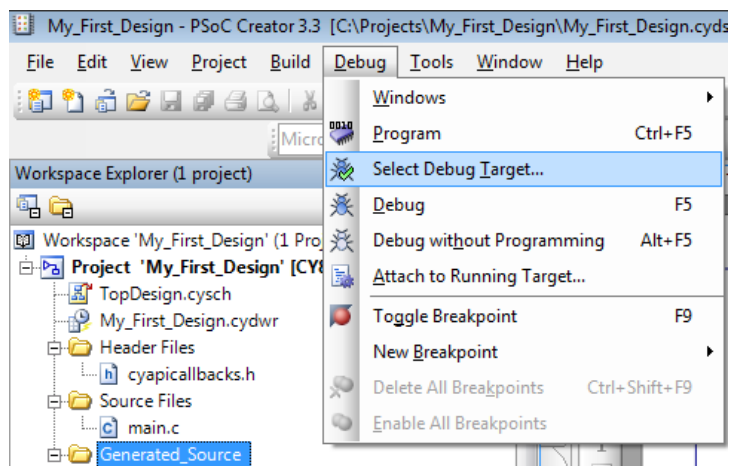
Figure 26. Build the Project



## 6.4 Part 2: Program the Device

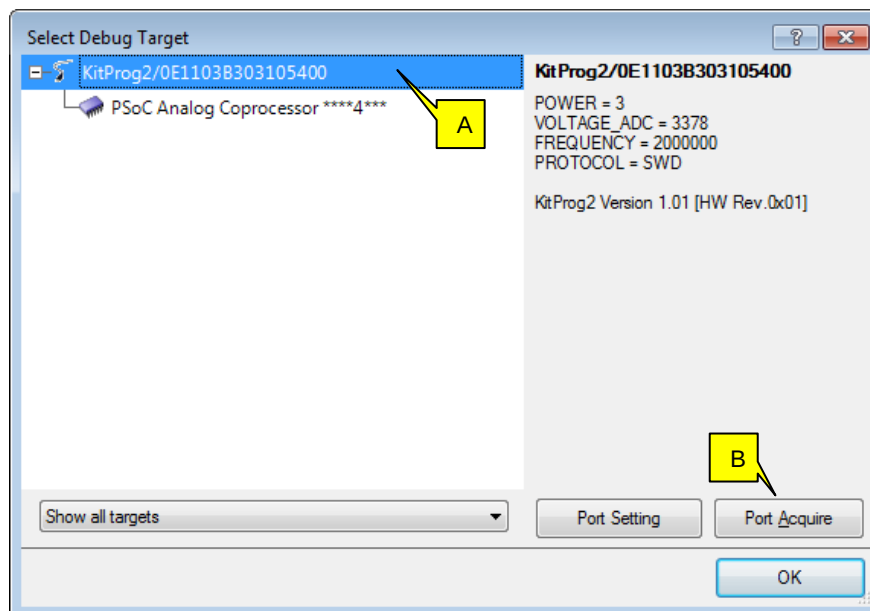
1. Connect the DVK to the USB port of your computer.
2. Confirm the connection between PSoC Creator and your DVK. To do this, first select PSoC Creator menu item **Debug > Select Debug Target**, as Figure 27 shows.

Figure 27. Select Debug Target



- A. A “Select Debug Target” dialog is displayed, as Figure 28 shows. Click on your target DVK (PSoC Creator supports multiple DVK connections)
- B. Click **Port Acquire**.

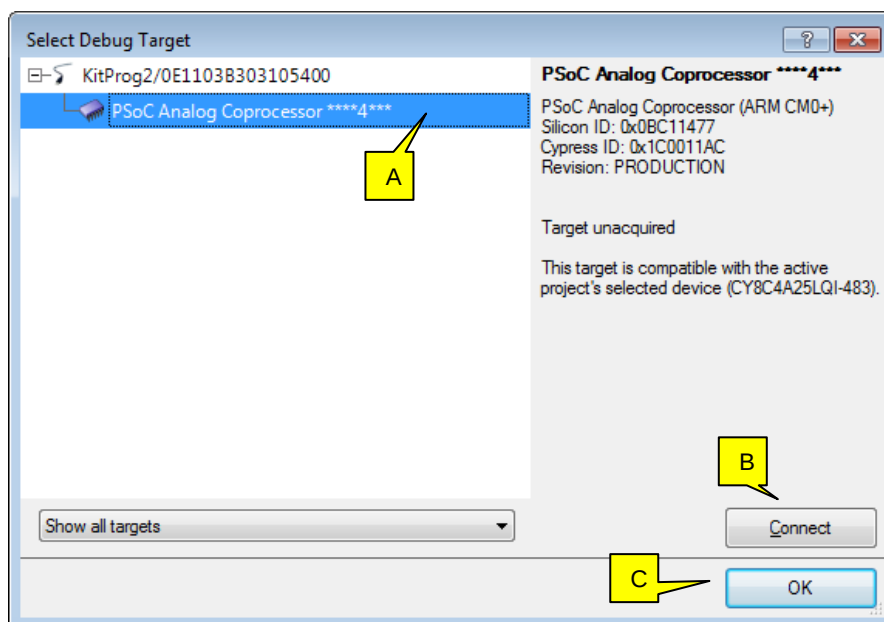
Figure 28. Configure the Target



3. Connect to the PSoC on your target DVK. See [Figure 29](#).
  - A. Click **PSoC Analog Coprocessor**.
  - B. Click **Connect**. The “Target unacquired” message changes to “Target acquired”, and the button label changes to “Disconnect”.
  - C. Click **OK** to close the dialog.

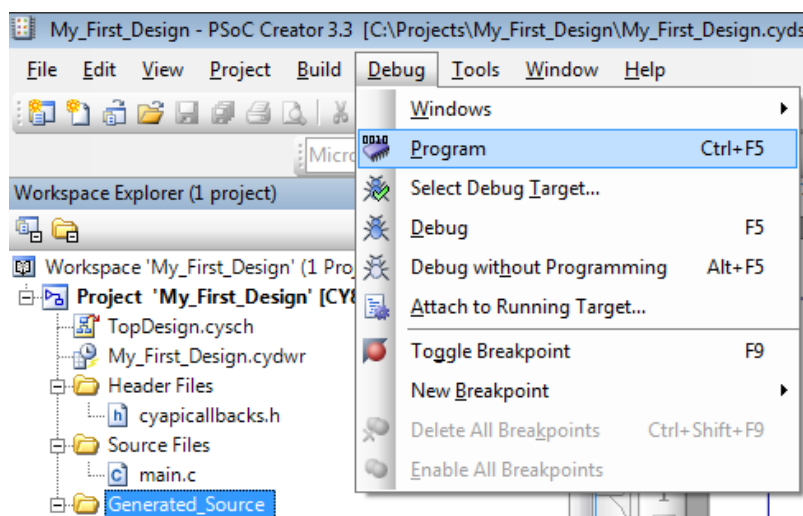
PSoC Creator is now connected to the target DVK and PSoC; you can now program the PSoC device.

Figure 29. Connect to PSoC Analog Coprocessor



4. To program the PSoC Analog Coprocessor, select **Debug > Program**, as [Figure 30](#) shows.

Figure 30. Programming PSoC



- Programming begins; the programming status is displayed in the PSoC Creator status bar (the lower-left corner of the window, as Figure 31 shows).

**Note:** You may see a warning message “This programmer is currently out of date”. See the KitProg User Guide in your kit documentation for information on how to upgrade your programmer firmware.

Figure 31. Programming Status

```

Programming started for device: 'PSoC Analog Coprocessor ****4****'.
Programming CY8C4A25LQI-483 image into CY8C4A45LQI-483 device.
Device ID Check
Erasing...
Programming of Flash Starting...
  
```

Notice List

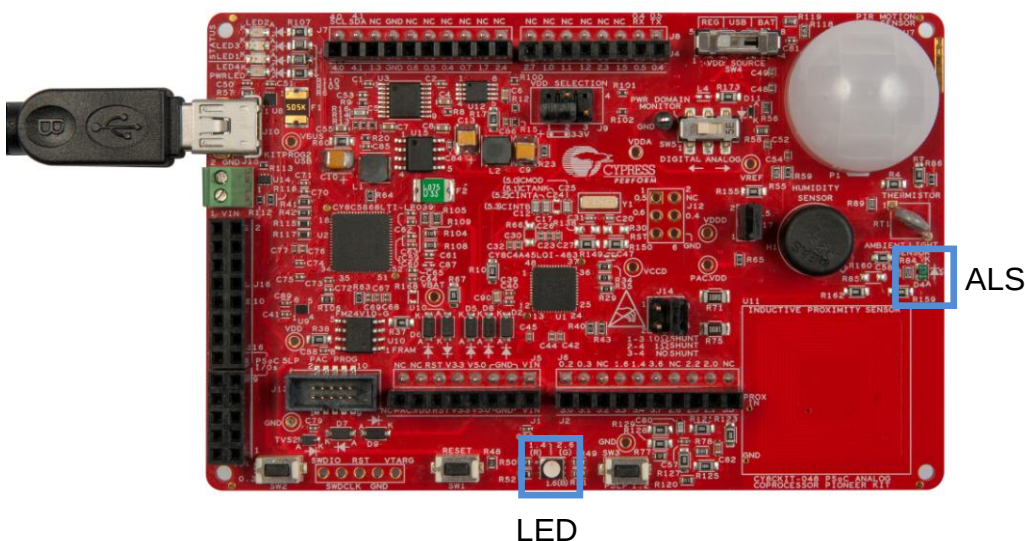
Programming - 46 of 256 blocks

## 6.5 Part 3: Test the Design

Follow the steps below to test the design:

- Connect the USB port of the CY8CKIT-048 DVK to the PC, as Figure 32 shows.

Figure 32. CY8CKIT-048 DVK



- Move your hand over the ambient light sensor (ALS), and observe that the red LED intensity varies accordingly. You may need to move the test setup to a location with sufficient ambient light.

## 7 Summary

This application note introduced you to the PSoC Analog Coprocessor device and helped you create your first design. This application note provided you an overview of additional resources to accelerate your learning.

## 8 Related Application Notes and Code Examples

Table 3 lists the selected system-level and general application notes that are recommended for the next steps in learning about PSoC and PSoC Creator.

Table 3. General and System-Level Application Notes

| Document                | Document Name                                                                     |
|-------------------------|-----------------------------------------------------------------------------------|
| <a href="#">AN86233</a> | PSoC 4 and PSoC Analog Coprocessor Low-Power Modes and Power Reduction Techniques |
| <a href="#">AN88619</a> | PSoC 4 Hardware Design Considerations                                             |
| <a href="#">AN73854</a> | PSoC 3, PSoC 4, and PSoC 5LP Introduction to Bootloaders                          |
| <a href="#">AN89056</a> | PSoC 4 – IEC 60730 Class B and IEC 61508 SIL Safety Software Library              |

Table 4 lists the application notes (AN) and code examples (CE) for specific peripherals and applications of the device.

Table 4. Documents Related to PSoC Analog Coprocessor Features

| Document                   | Document Name                                                           |
|----------------------------|-------------------------------------------------------------------------|
| <b>Programmable Analog</b> |                                                                         |
| <a href="#">AN211294</a>   | AFE Implementation Using PSoC Analog Coprocessor                        |
| <a href="#">AN60590</a>    | PSoC 3, PSoC 4 and PSoC 5LP - Temperature Measurement with a Diode      |
| <a href="#">AN70698</a>    | PSoC 3, PSoC 4 and PSoC 5LP – Temperature Measurement with an RTD       |
| <a href="#">AN66477</a>    | PSoC 3, PSoC 4 and PSoC 5LP – Temperature Measurement with a Thermistor |
| <a href="#">CE211252</a>   | Interfacing PSoC Analog Coprocessor with Ambient Light Sensor           |
| <a href="#">CE211301</a>   | Interfacing PSoC Analog Coprocessor with PIR Motion Sensor              |
| <a href="#">CE211305</a>   | Interfacing PSoC Analog Coprocessor with Inductive Proximity Sensor     |
| <a href="#">CE211321</a>   | Interfacing PSoC Analog Coprocessor with Thermistor                     |
| <a href="#">CE211322</a>   | Interfacing PSoC Analog Coprocessor with Humidity Sensor                |
| <b>CPU and Interrupts</b>  |                                                                         |
| <a href="#">AN89610</a>    | PSoC 4 and PSoC 5LP ARM Cortex Code Optimization                        |
| <a href="#">AN90799</a>    | PSoC 4 Interrupts                                                       |
| <b>I/O</b>                 |                                                                         |
| <a href="#">AN86439</a>    | PSoC 4 and PSoC Analog Coprocessor - Using GPIO Pins                    |
| <b>CapSense</b>            |                                                                         |
| <a href="#">AN85951</a>    | PSoC 4 and PSoC Analog Coprocessor CapSense® Design Guide               |
| <a href="#">AN92239</a>    | Proximity Sensing with CapSense                                         |
| <b>Bootloader</b>          |                                                                         |
| <a href="#">AN86526</a>    | PSoC 4 and PSoC Analog Coprocessor I2C Bootloader                       |
| <a href="#">AN68272</a>    | PSoC 3, PSoC 4, PSoC 5LP and PSoC Analog Coprocessor UART Bootloader    |
| <b>Segment LCD</b>         |                                                                         |
| <a href="#">AN87391</a>    | PSoC 4 Segment LCD Drive                                                |
| <b>Programming</b>         |                                                                         |
| <a href="#">AN84858</a>    | PSoC 4 Programming Using an External Microcontroller (HSSP)             |

---

## About the Author

Name: Rajiv Badiger  
Title: Applications Engineer Staff  
Background: Bachelor of Engineering in Electronics and Communication

## Document History

Document Title: AN211293 - Getting Started with PSoC® Analog Coprocessor

Document Number: 002-11293

| Revision | ECN     | Orig. of Change | Submission Date | Description of Change       |
|----------|---------|-----------------|-----------------|-----------------------------|
| **       | 5155799 | RJVB            | 03/23/2016      | New application note        |
| *A       | 5732670 | AESATP12        | 05/16/2017      | Updated logo and copyright. |

## Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at [Cypress Locations](#).

## Products

|                               |                                                                    |
|-------------------------------|--------------------------------------------------------------------|
| ARM® Cortex® Microcontrollers | <a href="http://cypress.com/arm">cypress.com/arm</a>               |
| Automotive                    | <a href="http://cypress.com/automotive">cypress.com/automotive</a> |
| Clocks & Buffers              | <a href="http://cypress.com/clocks">cypress.com/clocks</a>         |
| Interface                     | <a href="http://cypress.com/interface">cypress.com/interface</a>   |
| Internet of Things            | <a href="http://cypress.com/iot">cypress.com/iot</a>               |
| Memory                        | <a href="http://cypress.com/memory">cypress.com/memory</a>         |
| Microcontrollers              | <a href="http://cypress.com/mcu">cypress.com/mcu</a>               |
| PSoC                          | <a href="http://cypress.com/psoc">cypress.com/psoc</a>             |
| Power Management ICs          | <a href="http://cypress.com/pmic">cypress.com/pmic</a>             |
| Touch Sensing                 | <a href="http://cypress.com/touch">cypress.com/touch</a>           |
| USB Controllers               | <a href="http://cypress.com/usb">cypress.com/usb</a>               |
| Wireless Connectivity         | <a href="http://cypress.com/wireless">cypress.com/wireless</a>     |

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

## PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6](#)

## Cypress Developer Community

[Forums](#) | [WICED IOT Forums](#) | [Projects](#) | [Videos](#) | [Blogs](#) | [Training](#) | [Components](#)

## Technical Support

[cypress.com/support](http://cypress.com/support)



Cypress Semiconductor  
198 Champion Court  
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2016-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit [cypress.com](http://cypress.com). Other names and brands may be claimed as property of their respective owners.