

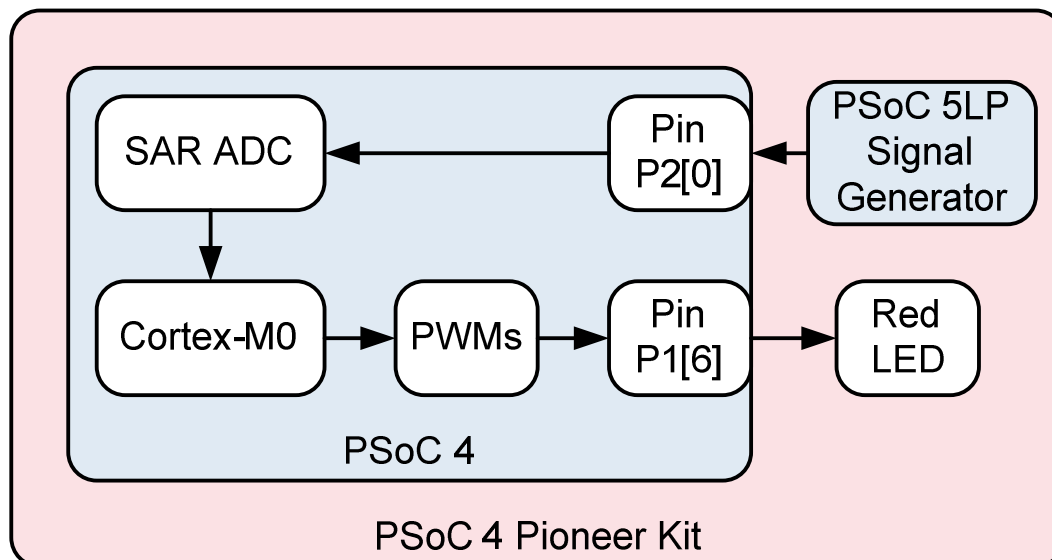
Objectives

- To learn how to use the Sequencing SAR ADC component to measure analog voltages.
- To learn how the PSoC 5 LP program and debug part can be used to implement additional behavior on the Pioneer Kit.

Requirements	Details
Hardware	CY8CKIT-042 PSoC 4 Pioneer Kit
Software	PSoC Creator 2.2 SP1
Firmware	Lab 4 Template
Components used	Sequencing SAR ADC

Block Diagram

Figure 1. Lab 4 Block Diagram



Theory

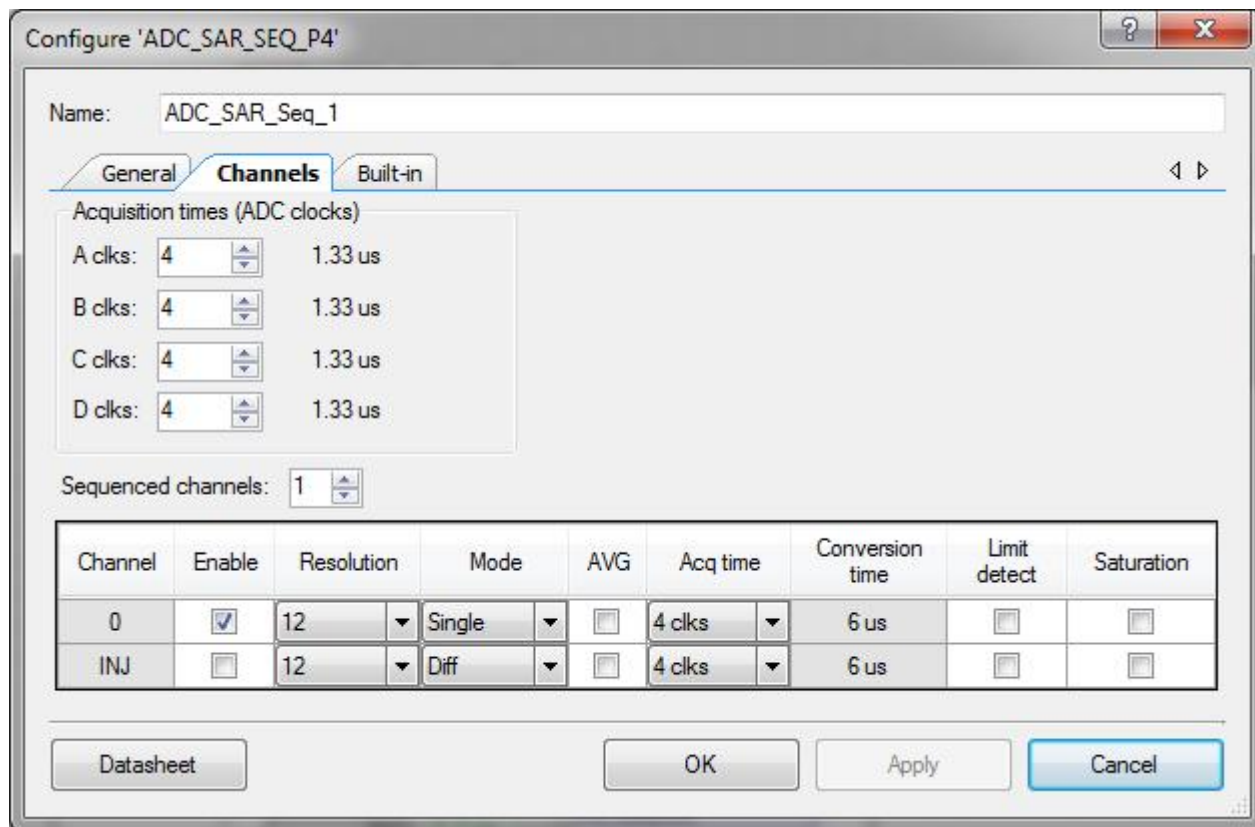
The goal of this lab is to learn how to use PSoC 4's SAR ADC measure an analog signal, and use the LEDs to display that measurement. PSoC 4's Sequencing SAR ADC is capable of making 12-bit analog measurements at a rate up to 1 megasample per second, sequencing through inputs without CPU intervention.

The lab will also demonstrate how to use the PSoC 5LP program and debug part on the Pioneer Kit to implement additional behavior. In this case, we will use a WaveDAC on the PSoC 5LP to generate two waveforms that we will read with the PSoC 4 SAR ADC.

Procedure – Firmware

- 1) Open the PSoC 4 Intro Lab Templates workspace, if it's not already open. You will be presented with a workspace in PSoC Creator containing one template project for each of the labs after Lab 1.
- 2) Set the Lab 4 template project as the active project by right-clicking on it in the Workspace Explorer, and clicking on the "Set As Active Project" option. In its initial state, this project will produce a 1 Hz sawtooth intensity on the Red LED.
- 3) Open the project's schematic by double-clicking on the "TopDesign.cysch" file in the Workspace Explorer. Note that in this schematic, we've included three PWMs and pins, along with the "LED_RGB" module to allow for easy driving of tri-colored LEDs.
- 4) In the component catalog, under the "Analog->ADC" category, select the "Sequencing SAR ADC" component, and drag it into the schematic.
- 5) Open the "ADC_SAR_Seq_1" component customizer by double-clicking on the component. Click on the "Channels" tab to configure the ADC inputs. Set the "Sequenced channels" input to 1 to remove the unused inputs. Change the "Mode" of channel 0 to "Single". The configuration window should look like the one shown in [Figure 2](#).

Figure 2. ADC SAR Sequencer Channel Configuration



Configure 'ADC_SAR_SEQ_P4'

Name: ADC_SAR_Seq_1

General Channels Built-in

Acquisition times (ADC clocks)

A clks: 4 1.33 us

B clks: 4 1.33 us

C clks: 4 1.33 us

D clks: 4 1.33 us

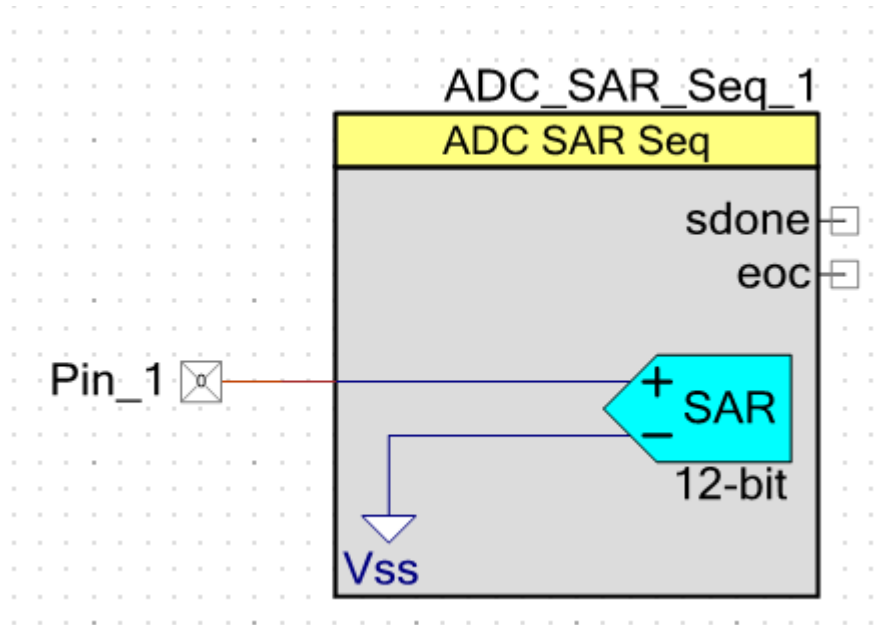
Sequenced channels: 1

Channel	Enable	Resolution	Mode	AVG	Acq time	Conversion time	Limit detect	Saturation
0	<input checked="" type="checkbox"/>	12	Single	<input type="checkbox"/>	4 clks	6 us	<input type="checkbox"/>	<input type="checkbox"/>
INJ	<input type="checkbox"/>	12	Diff	<input type="checkbox"/>	4 clks	6 us	<input type="checkbox"/>	<input type="checkbox"/>

Datasheet OK Apply Cancel

- 6) Press “OK” to close the configuration window. The ADC_SAR_Seq_1 component should now look like the one shown in [Figure 3](#).
- 7) In the component catalog, under the “Ports and Pins” category, select the “Analog Pin” component, drag it into the schematic, and connect it to the positive input of the SAR ADC. The schematic should look like the one shown in [Figure 3](#).

Figure 3. SAR ADC and Pin Appearance



- 8) Open the design wide resources by double-clicking the “Lab 4 ADC.cydwr” file. Navigate to the “Pins” tab. Using the “Port” dropdown, map the “ADC_SAR_Seq_1:Bypass” pin to P1[7], and “Pin_1” to P2[0].
- 9) Open the main.c file by double-clicking on it in the Workspace Explorer.
- 10) Replace the “Change1” line with the ADC start code, shown in [Code 1](#).

Code 1. Lab 4 “Change1” ADC Start Code

```
ADC_SAR_Seq_1_Start();
ADC_SAR_Seq_1_StartConvert();
ADC_SAR_Seq_1_IsEndConversion(
ADC_SAR_Seq_1_WAIT_FOR_RESULT);
```

- 11) Replace the “Change2” line with the ADC GetResult API, shown in [Code 2](#). The entire main.c should look like that shown in [Figure 4](#).

Code 2. Lab 4 “Change 2” ADC Get Result API

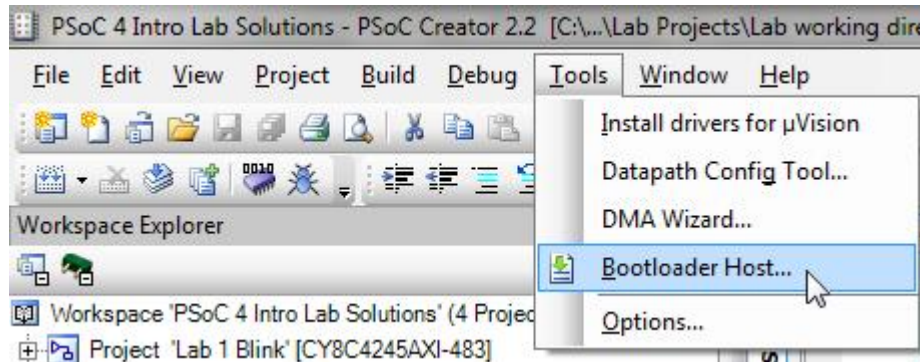
```
ADCResult = ADC_SAR_Seq_1_GetResult16(0);
```

Figure 4. Lab 4 Solution main.c

```
TopDesign.cysch  main.c
1  #include <device.h>
2  #include <LED_RGB.h>
3
4  void main()
5  {
6      /* Place your initialization/startup code here (e.g. MyInst_Start()) */
7      uint16 ADCResult;
8      LED_RGB_Start();
9
10     CyGlobalIntEnable; /* Uncomment this line to enable global interrupts. */
11
12     ADC_SAR_Seq_1_Start();
13     ADC_SAR_Seq_1_StartConvert();
14     ADC_SAR_Seq_1_IsEndConversion(ADC_SAR_Seq_1_WAIT_FOR_RESULT);
15
16     for(;;)
17     {
18         /* Place your application code here. */
19         ADCResult = ADC_SAR_Seq_1_GetResult16(0);
20         LED_RGB_SetColorRGB(ADCResult * (65535 / 2048), 0, 0);
21         CyDelay(10);
22     }
23 }
24
25 /* [] END OF FILE */
26
```

- 12) Press the “Program” button on the PSoC Creator toolbar to build the project and program your kit. After programming, The Red LED will stay at a constant intensity, because the voltage on the ADC input pin P2[0] is not varying. Next we will program the PSoC 5LP with a program to make it generate a waveform that will drive the PSoC 4 SAR ADC input.
- 13) In PSoC Creator, under the “Tools” menu, select the “Bootloader Host...” option. This is shown in [Figure 5](#).

Figure 5. PSoC Creator Bootloader Host Menu Option



- 14) Click the “Open File” button in the upper left of the Bootloader Host GUI. Select the file “PSoC5LP_WaveDAC.cyacd” in the project template directory.
- 15) In the upper right corner of the “Ports” field in the Bootloader Host GUI, press the “Filters...” button. Ensure that “Show USB Devices” is checked, and the VID and PID values are 0x04B4 and 0xF13B, respectively. Close the window by pressing the “OK” button. The window should look like the one shown in [Figure 6](#).

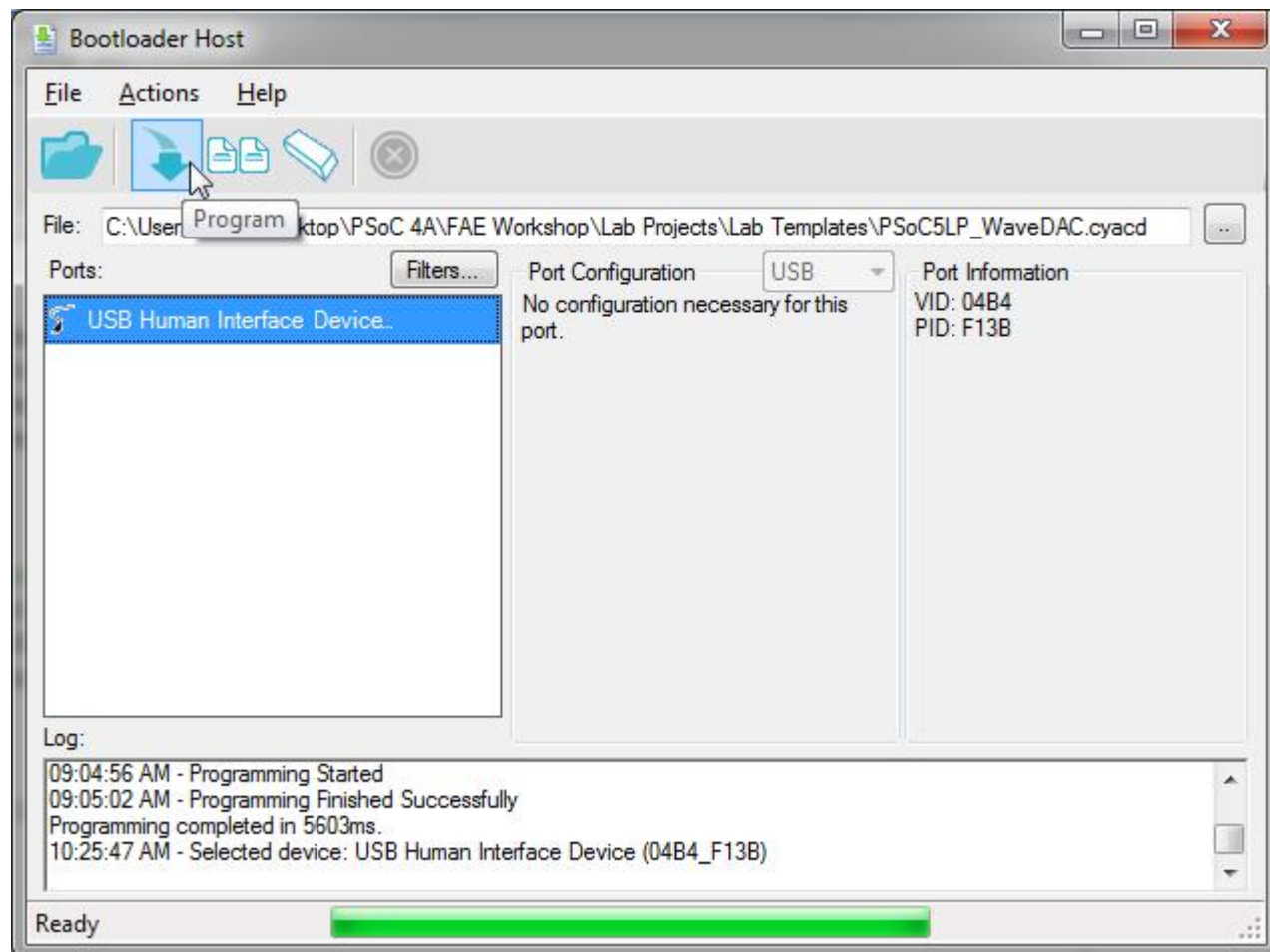
Figure 6. Bootloader Host GUI Port Filter Settings



- 16) Unplug the Pioneer Kit from USB. Hold down the reset switch, SW1, which is in the lower left corner of the board, while plugging the USB cable back in. Release the button when the status LED D1 in the upper left corner of the board starts blinking. This indicates that the board is ready to be bootloaded.

- 17) At this point, the kit should appear in the “Ports:” field in the Bootloader Host. Select it, and press the “Program” button, as shown in [Figure 7](#).

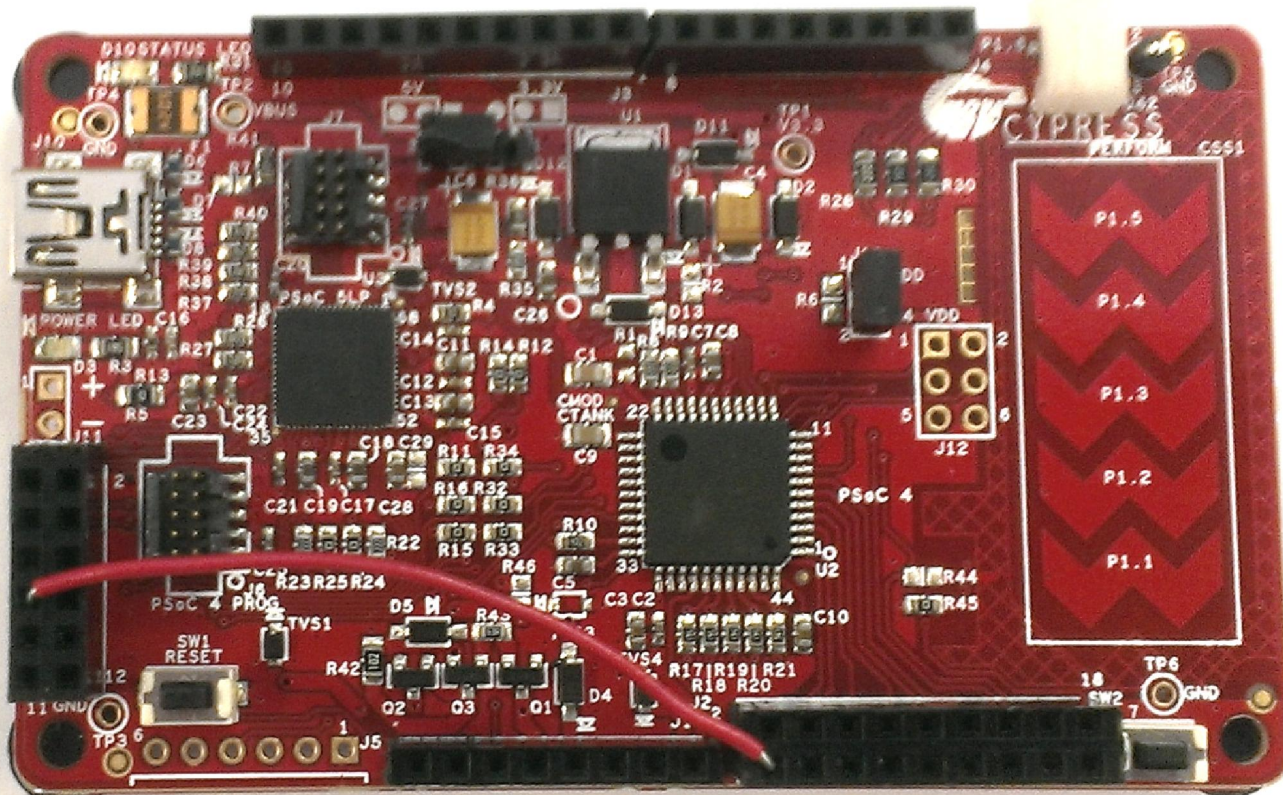
Figure 7. Bootloader Host GUI Ready for Bootloading



- 18) The results of bootloading will appear in the “Log:” at the bottom of the GUI. Our PSoC 5LP should now be generating an analog waveform. All that remains is to connect it to our PSoC 4 ADC input.

- 19) Using one of the wires shipped with the kit, connect PSoC 4 pin P2[0] (connector J2 pin 1) with PSoC 5LP pin P3[6] (connector J8 pin 7). This is shown in [Figure 8](#).

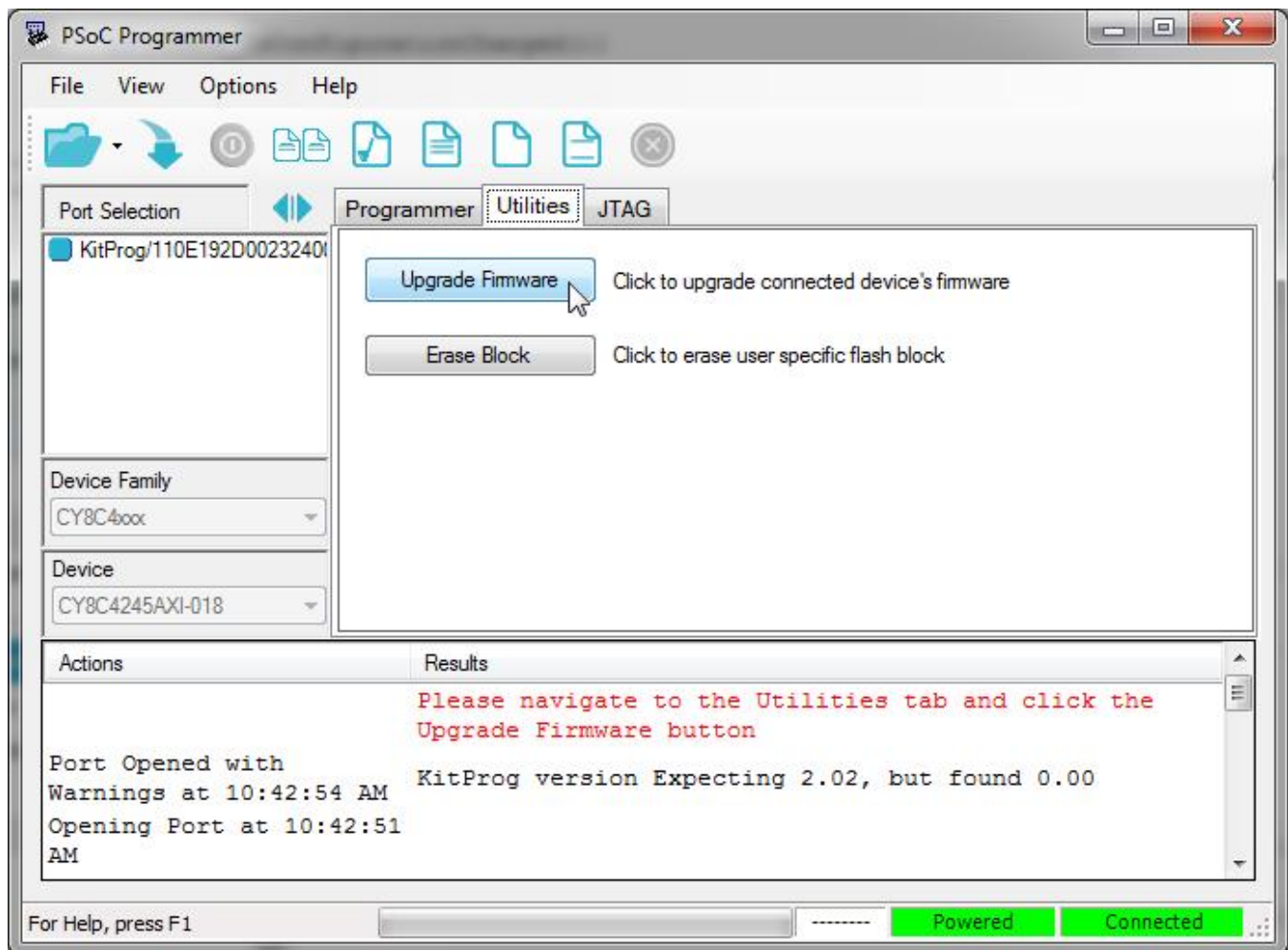
Figure 8. PSoC 4 Pioneer Kit PSoC 5LP WaveDAC Connection



- 20) At this point, the red LED should be pulsing in and out in a sine wave pattern at a frequency of 1 Hz.
- 21) Briefly press SW1 to change the WaveDAC waveform from sine wave to sawtooth wave. The red LED should now vary as a sawtooth wave. Pressing the button again will toggle between waveforms.

- 22) To restore the factory firmware to the PSoC 5 LP, open PSoC Programmer, which is located in the “Cypress->PSoC Programmer 3.18” location in the Start menu.
- 23) When PSoC Programmer is opened, it will detect the firmware installed on the PSoC 5LP and report it. It should show version “0.00” with our custom WaveDAC firmware. It will also instruct you to upgrade firmware to get to the most recent official version. This is shown in [Figure 9](#).
- 24) To restore the factory firmware, navigate to the “Utilities” tab and press the “Upgrade Firmware” button. This is shown in [Figure 9](#). The PSoC 5LP will now be programmed with the factory firmware, and won’t produce the WaveDAC outputs. To bootload the custom firmware again, repeat steps [13](#)) through [18](#)).

Figure 9. PSoC Programmer Firmware Update Interface



Conclusion

- You have successfully implemented an ADC in PSoC 4, reading analog values generated by the PSoC 5LP on the Pioneer Kit.

Stretch Goals

- Modulate another LED characteristic instead of red intensity
 - In this lab, we varied the red LED intensity based on the input to make the two waveform shapes more distinct.
 - You could use the other LED_RGB functions to vary different LED characteristics, like hue, saturation, or value. The other useful LED_RGB module functions are shown in [Code 3](#).

Code 3. LED_RGB Module Functions

```
void LED_RGB_SetColorRGB(uint16 redIntensity,  
    uint16 greenIntensity, uint16 blueIntensity);  
void LED_RGB_SetColorCircle(uint16 hue);  
void LED_RGB_SetColorCircleHSV(uint16 hue,  
    uint16 saturation, uint16 value);
```

- Transmit ADC readings from PSoC 4 back to the PC via UART
 - In this lab, we demonstrated ADC readings by driving the red LED at different intensities.
 - We could integrate some of the functionality demonstrated in lab 3, and use the UART to transmit readings back to the PC.
 - The custom PSoC 5LP WaveDAC firmware still implements a USB-UART bridge.

Document Revision History

Revision	By	Description
01	MAXK	First Release