# Chapter 9: WICED Academy Shield

## Description

In order to add a few additional features for demonstrating the capabilities of the WICED device, we are using a CY8CKIT-048 PSoC 4 Analog Coprocessor Pioneer kit as a shield in concert with the BCM4343W\_AVN starter kit.

Complete documentation on that kit can be found at:

<http://www.cypress.com/documentation/development-kitsboards/cy8ckit-048-psoc-analog-coprocessor-pioneer-kit>

The Analog Coprocessor Pioneer kit is powered from the base board so a separate USB cable is NOT required.

## Features

### PSoC4 (Analog Coprocessor)

The heart of the kit is a PSoC 4 Analog Coprocessor (CY8C4A45LQI-483). This PSoC combines flexible Analog Front Ends, programmable Analog Filters, and high-resolution Analog-to-Digital converters along with an efficient-yet-powerful ARM® Cortex®-M0+ signal processing engine. The data from the PSoC can be accessed over I2C, UART, or SPI. In our case, I2C is used as the communication mechanism.

The PSoC 4 is capable of sensing voltage, current, resistance, inductance and capacitance. For our purposes, resistance sensing is used for measuring temperature via a thermistor, while capacitance sensing is used for measuring humidity.

### LEDs

There is a tri-color LED on the kit. The LEDs are active low but the inversion is handled by the PSoC 4 such that they will appear to be active high to the WICED kit. The PSoC pins are configured as “Open drain, drives low”.

Two of these LEDs (red and blue) are controlled (i.e. driven) by the PSoC 4. They can be turned on/off remotely using the I2C interface (see the I2C description below). These LEDs also connect to the Arduino pins A3 and A4. Because they are configured in the PSoC as Open drain, drives Low, they can be driven directly from the WICED device as GPIOs. In that case, keep in mind that the LEDs are active low so that pulling the pin low will turn them on.

The third LED (green) is controlled directly from Arduino pin A1 (it is actually routed through the PSoC 4 to allow it to be mapped to a pin that can be controlled by a PWM from the WICED processor but to the WICED device it appears to be directly connected).



### Thermistor

The temperature is calculated by measuring voltage across a thermistor using the ADC. The schematic and firmware are based on code example CE211321. The temperature value can be read over the I2C interface (see I2C section below for details). The temperature is reported in hundredths of a degree Celsius (i.e. temperature \* 100).



### Humidity

The humidity is calculated by measuring capacitance of a humidity sensor using the CapSense block. The schematic and firmware are based on code example CE211322. The humidity value can be read over the I2C interface (see I2C section below for details). The humidity is reported in tenths of a percent (i.e. % humidity \* 10).



### I2C

The I2C interface is an EZI2C slave. That is, the first byte of a write into the slave is an offset to the set of I2C registers. The remaining bytes (if any) are the data to be written starting at the offset. For I2C reads from the slave, the offset is whatever was set in the previous write.

The I2C slave is assigned to 7-bit address 0x08 and is configured for a speed of 100 kHz. It is connected to Arduino pins D14 (SDA) and D15 (SCL).

The I2C register map is as follows:

| Offset | Description | Details |
| --- | --- | --- |
| 00 | Red LED | 0 = OFF, non-zero = ON |
| 01 | Blue LED | 0 = OFF, non-zero = ON |
| 02 | Temperature | LSB of temperature \* 100 |
| 03 | Temperature | MSB of temperature \* 100 |
| 04 | Humidity | LSB of humidity \* 10 |
| 05 | Humidity | LSB of humidity \* 10 |



### Arduino pins

The Arduino pin connections between the shield and the base board are shown below. This mapping was created by looking at the schematic for the CY8CKIT-048, the schematic for the BCM94343W\_AVN in the platform/schematics directory, the comments at the top of platform.h, and the constants in platform.c.

| **Arduino** | **Kit Header Name** | **Module Pin** | **WICED Pins** | **Shield Function** |
| --- | --- | --- | --- | --- |
| A0 | ADC\_IN1 | MICRO\_ADC\_IN1 | WICED\_GPIO\_2  WICED\_ADC\_1  WICED\_PWM\_3 | Inductive Sensor |
| A1 | ADC\_IN2 | MICRO\_ADC\_IN2 | WICED\_GPIO\_3  WICED\_ADC\_2  WICED\_PWM\_4 | Button  Green LED\* |
| A2 | ADC\_IN3 | MICRO\_ADC\_IN3 | WICED\_GPIO\_4  WICED\_ADC\_3  WICED\_PWM\_5 | N/C |
| A3 | MICRO\_WAKEUP | MICRO\_WKUP | WICED\_GPIO\_1 | Blue LED |
| A4 | I2C2\_SDA | MICRO\_I2C2\_SDA | WICED\_GPIO\_21 | Red LED |
| A5 | I2C2\_SCL | MICRO\_I2C2\_SCL | WICED\_GPIO\_20 | VSSA |
| D0 | UART\_RX | MICRO\_UART\_RX | WICED\_GPIO\_10 | Analog Co-processor UART TX† |
| D1 | UART\_TX | MICRO\_UART\_TX | WICED\_GPIO\_09 | Analog Co-processor UART RX† |
| D2 | UART\_CTS | MICRO\_UART\_CTS | WICED\_GPIO\_15 | N/C |
| D3 | UART\_RTS | MICRO\_UART\_RTS | WICED\_GPIO\_16 | N/C |
| D4 | GPIO\_D4 | MICRO\_GPIO\_2 | WICED\_GPIO\_26 | N/C |
| D5 | GPIO\_D5 | MICRO\_GPIO\_3 | WICED\_GPIO\_27 | N/C |
| D6 | GPIO\_D6 | MICRO\_GPIO\_4 | WICED\_GPIO\_28 | N/C |
| D7 | GPIO\_D7 | MICRO\_GPIO\_5 | WICED\_GPIO\_17 | N/C |
| D8 | UART6\_RX | USART6\_RX | WICED\_GPIO\_14 | N/C |
| D9 | UART6\_TX | USART\_TX | WICED\_GPIO\_13 | N/C |
| D10 | SPI\_SS | MICRO\_SPI2\_SSN | WICED\_GPIO\_22 | N/C |
| D11 | SPI\_MOSI | MICRO\_SPI2\_MOSI | WICED\_GPIO\_25 | N/C |
| D12 | SPI\_MISO | MICRO\_SPI2\_MISO | WICED\_GPIO\_24 | N/C |
| D13 | SPI\_SCK | MICRO\_SPI2\_SCK | WICED\_GPIO\_23 | N/C |
| D14 | I2C1\_SDA | MICRO\_I2C1\_SDA | WICED\_GPIO\_12  WICED\_PWM\_2 | I2C\_SDA for Analog Co-processor and FRAM (slaves) and KitProg2 (master) |
| D15 | I2C1\_SCL | MICRO\_I2C1\_SCL | WICED\_GPIO\_11  WICED\_PWM\_1 | I2C\_SCL for Analog Co-processor and FRAM (slaves) and KitProg2 (master) |

\* The Green LED on the analog shield does not connect directly to A1. It is routed through the PSoC. This is done so that a PWM can be used to drive the LED in a later exercise.

† The analog co-processor chip has its Tx connected to the Rx of both the base board and KitProg2 and vice versa so that it can communicate via UART to either the base board or to the KitProg2. Therefore, the base board cannot communicate over UART to the KitProg2 since the Tx/Rx lines would be reversed.

## Programming the CY8CKIT-048

The analog coprocessor on the CY8CKIT-048 board is pre-programmed with the firmware that contains the functionality described above. If, for some reason, you want to modify that functionality or you need to re-program the firmware into the kit, please refer to the following sections.

### PSoC Creator Project

The project workspace is included with the class files at:

*WA-101 Files\projects\PSoC\WA101\_AnalogCoProcessor\WA101\_AnalogCoProcessor.cywrk*

To open the workspace in PSoC Creator, double-click on the workspace (cywrk) file. Note, you must have PSoC Creator 4.0 or later installed to open the project.

### Project Hex File

The project’s hex file is included with the class files at:

*WA-101 Files\projects\PSoC\WA101\_AnalogCoProcessor.hex*

To program the hex file to the CY8CKIT-048:

1. Connect a USB-mini B cable to the connector on the CY8CKIT-048 and connect the other end to your PC.
2. Open PSoC Programmer.
3. Click on the *File Load* button and navigate to the hex file.
4. Verify the settings as shown.
5. Click the *Program* button
6. Once programming is complete (PASS is indicated in PSoC Programmer), remove the cable from the CY8CKIT-048.

