# Chapter 9: WICED Academy Shield

## Description

In order to add inputs and outputs that can be measured and controlled by the WICED device, we have created a shield board that can be connected to the WICED kit. It includes a PSoC analog co-processor as

## Features

### PSoC4 (Analog Coprocessor)

The heart of the shield is a PSoC 4 Analog Coprocessor (CY8C4A45AZI-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, we use:

Resistance sensing for measuring temperature

Current sensing for measuring ambient light

Voltage sensing for measuring a potentiometer

Capacitance sensing for measuring humidity

### LEDs

There are six LEDs on the shield. Four are associated with the CapSense buttons (although they can be controlled independently via I2C if desired as you will see later). The other two LEDs are controlled by the base board. All LEDs are active low, but the inversion is handled by the PSoC using the SmartIO block so that they appear to be active high to the WICED base board.

The two independently controlled LEDs are connected to the Arduino header as follows:

|  |  |
| --- | --- |
| **LED** | **Header Pin** |
| LED0 | D10 |
| LED1 | D9 |

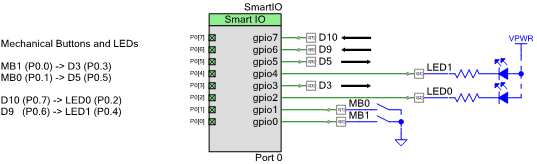
### Mechanical Buttons

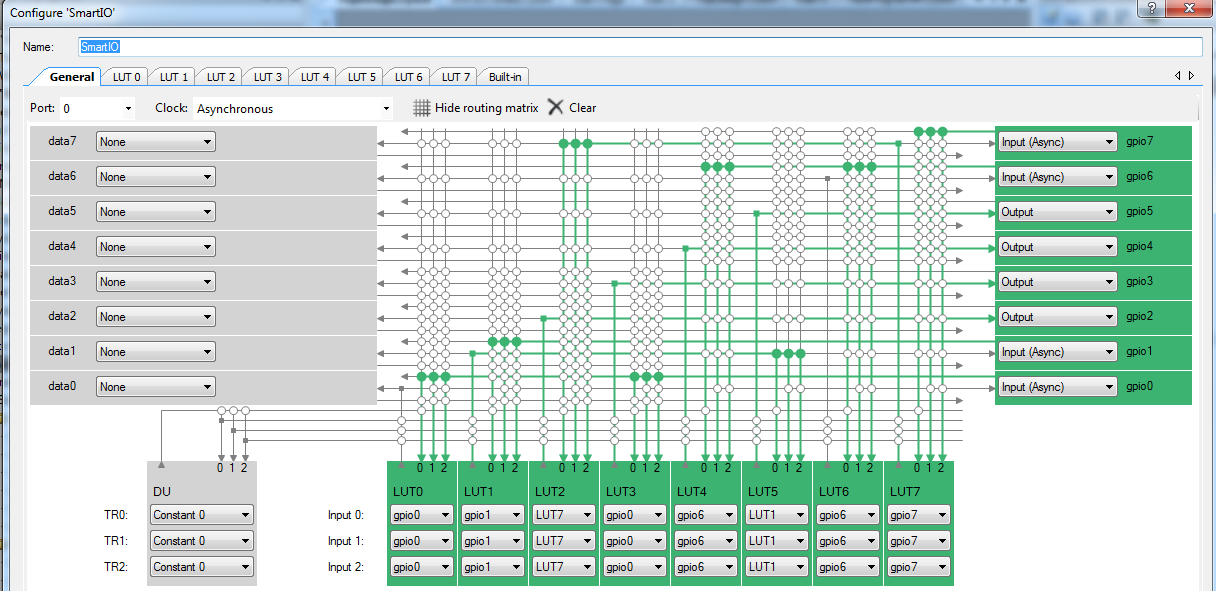
There are two mechanical buttons on the shield. These are active low and a pullup resistor is included for each of them on the shield. They are connected to the Arduino header as follows:

|  |  |
| --- | --- |
| **LED** | **Header Pin** |
| MB0 | D5 |
| MB1 | D3 |

The state of the mechanical buttons can also be read via I2C as you will see later.

The two independent LEDs and the mechanical buttons are all controlled by the SmartIO block in PSoC. The schematic and the SmartIO configuration are shown below.





### CapSense Buttons

There are four CapSense buttons. These buttons are scanned by the PSoC and their state is reported via an I2C register (see the I2C section later).

By default, each CapSense button will light an LED when it is touched. The LEDs can be “decoupled” from the CapSense buttons if desired by setting bit 0 in the I2C LED Control register. Once that is done, the LED Value register can be used to control the LEDs independently from the CapSense buttons.

### Proximity

There is a proximity sensor that runs around the outer edge of the board. The proximity sensor state is reported over I2C (see the I2C section later).

### Thermistor

The temperature is calculated by measuring voltage across a thermistor. The actual temperature calculation is handled by a PSoC component called “Thermistor” which greatly simplifies the coding required. 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) and is reported in degrees Celsius.

### Ambient Light Sensor

The ambient light is calculated by measuring current through a photo-transistor. The schematic and firmware are based on code example CE211252. The light value can be read over the I2C interface (see I2C section below for details) and is reported in Lux. In addition, the raw value at the output of the TIA can be measured at Arduino pin A0.

### Potentiometer

The voltage of the potentiometer is measured and can be read over the I2C interface (see I2C section below for details) and is reported in Volts. The ADC range is limited to 0 – 2.4V. In addition, the raw POT voltage is available at Arduino pin A2.

The schematic for the Thermistor, Ambient Light Sensor, and Potentiometer is shown below.



### 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) and is reported as a percentage.



### DAC Output

A voltage DAC output is available on Arduino pin A1. The DAC voltage can be set in the range from 0V - 2.4V over the I2C interface (see I2C section below for details).



### 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 0x42 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** | **Format** | **Details** |
| --- | --- | --- | --- |
| 00 | DAC Voltage | 4 Byte Float | Desired DAC voltage in Volts |
| 04 | LED Value | 1 Byte | CapSense LED values if LED Control bit 1 is 1. Mapping is:  Bit 0: CSLED0  Bit 1: CSLED1  Bit2: CSLED2  Bit3: CSLED3 |
| 05 | LED Control | 1 Byte | Bit 0 sets how the 4 CapSense LEDs are controlled:  0 = CapSense Control  1 = Base Board Control via the LED Value Register |
| 06 | Button State | 1 Byte | State of CapSense, Proximity, and Mechanical Buttons:  Bit 0: CapSense B0  Bit 1: CapSense B1  Bit 2: CapSense B2  Bit 3: CapSense B3  Bit 4: Mechanical Button MB0  Bit 5: Mechanical Button MB1  Bit 6: Proximity |
| 07 | Temperature | 4 Byte Float | Temperature reported in °C |
| 0B | Humidity | 4 Byte Float | Humidity reported in % |
| 0F | Ambient Light | 4 Byte Float | Ambient light reported in Lux |
| 13 | Potentiometer | 4 Byte Float | Potentiometer reported in Volts |

### U8G OLED Display

The shield contains a U8G OLED display with an I2C interface. By default, the display is driven by the PSoC analog co-processor. In order to use the I2C bus from the base board – either to drive the OLED display or for other I2C operations - it is necessary to stop the PSoC from driving the display so that the I2C bus is available for the base board. To do this, hold button MB0 down until the display clears (5-10 seconds). The PSoC OLED display can be re-enabled by holding button MB1 down until the display comes back on (5-10 seconds).

### Arduino pins

The Arduino pin connections between the shield and the base board are shown below.

| **Arduino** | **WICED Pins** | **Shield Function** |
| --- | --- | --- |
| A0 | N/A\* | Ambient Light TIA Output |
| A1 | N/A\* | DAC Voltage |
| A2 | N/A\* | Potentiometer |
| D3 | WICED\_SH\_MB1  WICED\_GPIO\_3 | Mechanical Button MB1 |
| D5 | WICED\_SH\_MB0  WICED\_GPIO\_12 | Mechanical Button MB0 |
| D9 | WICED\_SH\_LED0  WICED\_GPIO\_16  WICED\_PWM\_4 | LED1 |
| D10 | WICED\_SH\_LED1  WICED\_GPIO\_7 | LED0 |
| D14 | WICED\_GPIO\_48 | I2C\_SDA |
| D15 | WICED\_GPIO\_49 | I2C\_SCL |

\* The analog pins do not connect directly to the WICED device for the board we are using. Instead, they connect to a separate ADC on the base board. The ADC can be read using I2C.

## Appendix: Programming the PSoC Analog Co-processor

The analog coprocessor on the shield 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\ww101-shield\firmware\WW101-Shield.cywrk*

To open the workspace in PSoC Creator, double-click on the workspace (cywrk) file. Note, you must have PSoC Creator 4.1 or later installed to open the project.

### Bootloading

The project contains an I2C bootloader. You can bootload the project by connecting the shield to any PSoC Pioneer kit whose Kitprog I2C pins connect to Arduino header pins D14 and D15. For example, the PSoC 4M Pioneer kit (CY8CKIT-044) will work for this purpose.

To put the PSoC analog co-processor in bootloader mode, hold down MB0 and MB1 simultaneously until LED0 and LED1 flash in an alternating pattern. Then, you can use the PSoC Bootloader Host to load the new firmware. The I2C address is 0x42.

Note: If you are using the CY8CKIT-044 with the test program that is included in the workspace, you must also put that kit into bypass mode to bootload the shield. To do that, hold down SW2 (about 5 seconds) until the red LED in the tri-color LED begins to flash. This disables the LCD update from the CY8CKIT-044 which frees up the I2C bus so that the KitProg can use it for bootloading.

The Bootloadable firmware file can be found in the workspace at:

*WA-101 Files\ww101-shield\firmwarwe\WW101-Shield.cydsn\CortexM0p\ARM\_GCC\_541\Release\WW101-Shield.cyacd*