Lesson 2-8 I2C

Welcome back to Cypress Academy. This is WICED WiFi 101. This time, we are going to use the I2C master interface to talk to an I2C slave.

The base kit does not have any I2C slaves so we are going to use the shield board [CY8CKIT-032, available Q1'18] that has a PSoC Analog Co-processor programmed as an I2C slave at address 0x42. The PSoC connects to analog sensors that measure temperature, humidity, ambient light, as well as the potentiometer. The PSoC measures the sensors and converts the values into a digital format which is stored in the I2C registers. These values will be read from the base board and then displayed to the terminal window.

The I2C register map for the PSoC has temperature stored starting at an offset address of 0x07 followed by humidity, ambient light, and the potentiometer voltage. Each value is stored as a floating-point value – that is, 4 bytes each.

First let's copy 05_interrupt to 07_i2cread and update all of the files as necessary. You know, the makefile, the make target, …

In 07_i2cread.c we will do the following:

1. We'll use a button interrupt to set a flag indicating that the I2C data should be read from the PSoC and then displayed on the UART.
2. Setup an I2C structure containing the I2C block to use. In this case, it is WICED_I2C_2, the I2C address – in this case 0x42, the address width, and the speed.
3. Initialize the I2C block with the configuration that we created.
4. Use wiced_i2c_write to send a value of 0x07 to the slave. This is used to set the register offset to the location of the temperature measurement value. This offset will be used by the PSoC during the I2C read operations to specify which data to return.
5. In the main loop, whenever the button is pressed, we use wiced_i2c_read to read the values and then display them to the terminal window. Each I2C read will be reading 16 bytes – 4 bytes each for temperature, humidity, ambient light, and POT voltage.

Let's program the board with this firmware and observe what happens on the terminal window when we push the button.

At this point we have experimented with a lot of different ways the GPIOs can be used to communicate with the outside world. In the next set of videos, we'll talk about using the RTOS functions to control more complex firmware before moving on to the Wi-Fi connectivity.

You can post your comments and questions in our WiFi developer community, or, as always, you are welcome to email me at alan_hawse@cypress.com or tweet me at @askiotexpert. Thank you.