Lesson 2-3 GPIOs and LEDs – Part 1

Hi, I'm Alan Hawse. Welcome back to Cypress Academy, WICED WiFi 101. In this video I will talk about how to read and write the WICED GPIOs.

First, I will take the empty project that I created in the last video and add code to blink an LED by using a GPIO as an output.

Let's start by opening WICED Studio, and then opening the file called 02_blinkled.c that I created in the last video. I already have the WICED include done, as well as the WICED initialization done. But in your future projects, don't forget to call wiced_init.

Normally, you would have to initialize a GPIO before using it … that is to say you need to configure the GPIO as either an input, an output, pull up, pull down, etc. To do that we would use the wiced_gpio_init function just like this…

In order to see the different possible configurations, you can right click on the function name and select "Open Declaration". Then, do the same thing for a parameter type and the type name. This will bring you to the pin configuration datatype which includes all of the valid selections along with a comment about what each one does.

In general most of the source code for WICED is available to you, and you can use "right click open declaration" functionality to help understand what is happening.

To drive the LED, I can use the "OUTPUT_PUSH_PULL" selection.

For the LEDs on this platform, the initialization is already done in the platform.c file so you don't actually have to do it in application_start, but I wanted to show you how it's done for when you want to use a pin that isn't already configured by your platform.c.

Next, I need to add the function calls to turn the LED on and off with delays in between.

The name I am using here is WICED_LED1 which is one of the two LEDs on the base board. This is defined in the platform files.

The delay function wiced_rtos_delay_milliseconds is part of the RTOS (real-time operating system) that we deliver as part of the WICED SDK. That function call causes the thread to suspend for the specified time – in this case 250 milliseconds. I will talk more about RTOSes in the later videos, specifically in chapter 3.

Now I have everything needed to make the LED blink. So I'll double click on the make target that you created in the previous video and if you typed correctly, you will see the project build, then program, and then start running. Once all that's done, the LED will start to blink! [Yay, the LED blinks]

If anything went wrong, carefully check the following items:

1. The project folder name and the makefile name are EXACTLY the same.
2. The makefile has a unique application NAME, and there are NO spaces at the end of the name.
3. The makefile has the correct name for the C source code file.
4. The make target has the correct names, paths, and spelling.

If you see an error that says “No rule to make target…” it usually means you have a mismatch between the project folder name and the makefile name, or that you have a spelling error in the C source file name in the makefile.

If you see an error that says “empty variable name” it usually means that you have a space after the application name in the makefile.

If you see an error that says “Unknown component…” it usually means that your make target has an error.

If you see an error that says “recipe for target download_dct failed” it usually means that your kit is not connected, or the device drivers for your kit are not installed.

In the next video, I’ll show you how to read the state of a button and how to use an interrupt.

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.