**GPIOs – Part 1**

Welcome back to Cypress Academy, WICED WiFi 101. In this video I will talk about how to read and write the 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 02\_blinkled.c from the last video. I already have the WICED include an initialization done…. But don’t 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 it as an input, output, pull up, pull down, etc. To do that we would use the wiced\_gpio\_init function 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 from the 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 can you can use “right click open declaration” functionality to help understand what is happening

To drive the LED I can use “OUTPUT\_PUSH\_PULL”.

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 is done for when you want to use a pin that isn’t already configured in the platform.c.

Next, I need to add the function calls to turn the LED on and off with delays in between. That is done as you can see here…

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. In the solution projects, we used WICED\_SH\_LED1, which is the name given to one of the LEDs on the shield as specified in the platform files for the kit/shield combination.

If you want to use the solution projects but don’t have a shield like the one we used, you will need to change the name from WICED\_SH\_LED1 to WICED\_LED1. Since the platform files handle all of the pin mapping for you, that is the only change needed.

The delay function wiced\_rtos\_delay\_milliseconds is part of the RTOS (real-time operating system). It causes the thread to suspend for the specified time – in this case 250 milliseconds. I will talk more about RTOSes in later videos.

Now I have everything needed to make the LED blink. Double click on the make target that you created in the previous video and if everything is typed correctly, you will see the project build, program, and run. Once that is done, the LED will start to blink!

If anything went wrong, carefully check the following items:

1. The project folder name and make file name are EXACTLY the same.
2. The make file has a unique application NAME and there are NO spaces at the end of the name.
3. The make file 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 make file name or that you have a spelling error in the C source file name in the make file.

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

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 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 question in our Wifi developer community or as always you are welcome to email me at alan\_hawse@cypress.com or tweet me at @askioexpert with your comments, suggestions, criticisms and questions.