Hi, I’m Alan Hawse. I’m Senior Vice President of Technical Staff for Solutions and Software at Cypress Semiconductor. We previously talked about the basics of Wi-Fi networking. Now let’s put that information to use and see how to use the WICED SDK to connect our kit to WiFi.

Every WICED Wi-Fi chip has a section of flash called the Device Configuration Table, or DCT. This table holds lots of information about the device including Wi-Fi information such as the SSID, channel, security type, etc.

To preconfigure the Wi-Fi section of the DCT for your project, you need a file called wifi\_config.h. You can get a template for the file from the SDK under include/default\_wifi\_config\_dct.h.

You should copy this file to your project’s directory and rename it to just wifi\_config.h

The default wifi\_config.h file looks like this.

There are 3 sections to this file.

The first contains information used if you want your device to act as an access point during configuration. That is, you want it to be an AP during initial device setup so that you can connect to it from a different device to configure it.

The second contains information for when your device will be an AP during normal operation. In fact it is possible to use WICED devices as an access point and a station simultaneously

The third section contains information needed if your device is going to be a station that connects to an access point. For the majority of this class, we will be using this third section.

To include your DCT information when the firmware is built, you must add the following line in the make file:

WIFI\_CONFIG\_DCT\_H := wifi\_config.h

From the default DCT file, you can right click on the definitions and select open declaration to see the available choices. For example, here are all the types of security that WICED supports.

The DCT information is mapped into flash by the SDK. You normally don’t need to know about this since you can set everything up in the wiced\_config\_dct.h file. However, if you want to read or write the DCT on the fly (for example to change the access point settings during configuration) you will need to know some of the details.

The DCT is divided into sections. The section called wifi\_config is contains guess what? All of the Wi-Fi information, imagine that.

If we right click on wifi\_config we can see the different values available. The second entry – stored\_ap\_list – is an array of available access points that the device knows about. The first element in that array is the default AP that the device will auotmatically connect to.

If we right click again, we see that each AP has details, a key length, and a key. Right clicking on the details shows where the AP’s SSID and other information are stored.

The manual explains how to read and write various sections of the DCT. We won’t cover that in detail here, but feel free to read about it on your own.

Once the settings are in the DCT, it is very simple to connect to the network – we just use a single function call of wiced\_network\_up.

This is where you should standup and applaud. I think that the wiced\_network\_up function is remarkable because in one simple API call, you are connected to the network. This is the result an unbelievable amount of research and development. There is a massive amount of things that need to go right for you to be reliably connected to the network and WICED take care of all of that for you.

Back to the code, the wiced\_network\_up function takes three arguments:

First is the type of interface. In our case, this will be WICED\_STA\_INTERFACE since we want to connect as a station.

Second is the method to specify the IP settings such as the IP address, gateway address, and netmask. In our case, we will use WICED\_USE\_EXTERNAL\_DHCP\_SERVER so that the DHCP server on the network is used.

The final argument is not needed in this case since we are not specifying the IP settings so we can just enter NULL. However, if you want to use a static IP, this is where you would define those settings.

Note that many of the WICED functions return a value of type wiced\_result\_t. This can be very useful for debugging purposes. It is often a good idea to capture this value and print it to the debug UART.

The wiced\_result\_t data type is a giant nested enumeration that specifies what happened during the function call.

If you right click and open declaration, you can see the different possible return value buckets. If we select the top level generic bucket, you will see that a return value of 0 means WICED\_SUCCESS. Which is what I hope always happens. However, in the real world other things can happen that our outside of the control of the wiced chip, and this list gives you a clue about what went wrong.

Earlier I mentioned that the WICED device can act as an access point during initial setup so that you can connect to it and specify the Wi-Fi settings like SSID, security type, etc. for the AP that you want your IoT device to connect to. This is one of the ways an IoT device can be “introduced” to your network. Other methods include using Bluetooth to configure the Wi-Fi settings, connecting via USB, or hard-coding the AP settings. We will use hard-coded settings for this class but there are examples of the other types of configuration in the SDK.

At this point we know everything we need know to attach to an AP from our WICED kit.

Lets make a project

The project is called “05” (since this is chapter 5) / 01\_attach\_wpa2.

In the makefile we need to specify the name of the wifi configuration file.

The configuration file tells the firmware to connect to an AP called WW101WPA with WPA2 PSK encryption and the passphrase as shown. When you run the solution project, you will have to modify these to match your network.

In the C file, we just call wiced\_network\_up after wiced\_init and then look at the result. If the result is equal to WICED\_SUCCESS then we turn on an LED. If the connection did NOT work, then we blink the LED to indicate something went wrong.

Now let’s program the project and see what happens. The LED turned on, so that means that we are now connected to the Wi-Fi network.

You can also open a terminal window and then reset the kit to see the debug information that is printed by default when the device boots and connects to the network.

Now let’s expand that project a bit to print out some additional information about the network.

First, we will copy 01\_attach\_wpa2 to 03\_wpa2\_info and make the necessary changes.

In the C file we will add in function calls to get and then print:

the IP address that was assigned to our device

the netmask

the gateway IP address

the IP address of www.cypress.com

and finally, our device’s MAC address.

Then just program the project into the kit and look at the terminal window for the new information.

That’s all there is to connecting to a Wi-Fi network and getting information about it using WICED.

In the solution projects, we provide an example that shows you how to read and write the WiFi configuration in the DCT so feel free to explore that as well.

Now that we are connected to the network we want to start sending and receiving data. That will be the topic of our next chapter.

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