**BLE Development**

You have been provided with the following documentation from ST Micro (the latter sections of the filenames are shown to make it easy to select a particular document):

* *Session 7b B-L475E-IOT01A User Manual.pdf*
* *Session 17b BlueNRG-MS SPBTLE-RF Module Data Sheet.pdf*
* *Session 17c BlueNRG-MS Datasheet.pdf*
* *Session 17d BlueNRG-MS Software Development Kit Data Brief.pdf*
* *Session 17e BlueNRG-MS Development Kits User Manual UM1870.pdf*
* *Session 17f BlueNRG-MS Stacks Programming Manual PM2037.pdf*
* *Session 17g BlueNRG-MS Profiles Application Interface User Manual UM1770.pdf*
* *Session 17h BlueNRG-MS ACI User Manual UM1865.pdf*

You have also been provided with the latest version of the BlueNRG-MS Development Kit software:

* *Session 17i BlueNRG DK 2.0.2.zip*

We have a development board that contains a BLE radio and we want to make it do something. How do we proceed?

This is a very typical scenario for an embedded developer – you are given hardware, documentation, and software, but the software was developed for different hardware, so you need to “port” it to your hardware.

The first step is to get familiar with the hardware. When a new product is being developed, the hardware engineer is likely to base it on boards from the supplier(s) of the MCU and BLE radio. The embedded development team should work in parallel with the hardware development team by using those evaluation boards (also known as “eval boards” or “demo boards”), so that when the new hardware is ready, there is some software available to bring it to life (known as “bringing up the board” or “the bring-up” or, more formally, the design verification testing or DVT).

The B-L475E-IOT01A board is our target hardware and we can imagine that it could be the basis for a hardware engineer to develop a new product using its BLE radio.

Taking a look at the user manual for the B-L475E-IOT01A board, we see that it contains the SPBTLE-RF module from ST Micro and we determine the following about how it is connected:

* The SPBTLE-RF module is connected to the MCU via the SPI3 interface, with the following GPIO pin assignments:  
    
  SCK: PC10 MOSI: PC12 MISO: PC11 CSn: PD13 IRQ: PE6 RESET: PA8
* Taking a look at the datasheet for the SPBTLE-RF module, we see that it contains the BlueNRG-MS chip from ST Micro.
* Taking a look at the datasheet for the BlueNRG-MS chip, we learn the following:  
    
  The reset pin is active low and the hold time (typ) is 1.5msec. Note that the datasheet does not provide the minimum reset hold time, only the typical reset hold time.  
    
  The maximum SPI clock rate is 8MHz.

Now that we know more about the hardware, we need to figure out how to get it to do something. Searching for “BlueNRG-MS” on the ST Micro website (st.com) we see that there is a software development kit (STSW-BLUENRG-DK) available for it, so we download it and any associated documentation we can find.

The data brief for the STSW-BLUENRG-DK doesn’t provide a lot of information but it does show its structure. Of particular interest is the *Projects* directory, under which is the *Projects\_STD\_Library* directory, under which is the *BLE\_Beacon* project. As was discussed in the previous lesson, a beacon (Broadcaster) simply transmits advertisements to an Observer – it doesn’t receive and it doesn’t connect. This looks to be about as simple as it gets, so that’s what we’ll implement first.

Digging around the ST Micro website some more we find the UM1870 user manual for BlueNRG-MS development kits. Perusing the first two chapters of this document, we learn that it doesn’t apply to our board (rats!), but Chapter 3 (*Programming with BlueNRG-MS network processor*) and Chapter 6 (*BlueNRG-MS Beacon demonstration application*) contain some good information. In particular, Chapter 3 states that the requirements for communicating with the BlueNRG-MS chip are:

* SPI interface
* Platform-dependent code to write/read from SPI
* A timer to handle SPI timeouts or to run Bluetooth LE Profiles

We already know the BlueNRG-MS chip (contained in the SPBTLE-RF module) is connected to the MCU SPI3 interface and we know which GPIO pins are used for the reset signal and the SPI chip select signal.

Taking a look at the HAL library for our board, we see that it contains a HAL version and an LL (low-level) version of functions for the SPI peripheral. That gives us two out of the three requirements list above. We know we can figure out a way to handle timeouts, so we’ll leave that for when we start to implement our beacon.

The next document we peruse is the PM0237 programming manual for the BlueNRG-MS stack. It is recommended that you read the first chapter and scan the rest of the document.

In increasing complexity is the UM1770 user manual for the BlueNRG-MS profiles application interface. We will discuss the meaning of a profile in class and you should scan this document.

The last document to be discussed today is the UM1865 user manual for the BlueNRG-MS BLE stack application command interface (ACI). Recall that ST Micro has developed a superset of the standard HCI because the BlueNRG-MS contains both the host and controller functionality. UM1865 is packed with information and you should skim it to get familiar with it. One of the odd things about some of the documentation from ST Micro (and other European silicon vendors) is that the table of contents appears at the end of the document instead of the beginning. Keep that in mind when reading these documents (it is not consistent, as you will see).

**Quiz 6 Prep**

We will review the lesson notes from Session 16 to prepare for the quiz on Wednesday.

**Class Notes**