arm Education

Embedded Systems Essentials with Arm: Getting Started

Module 4

SV2 (4): Module 4 Lab Project: Part 1

Now let's start looking at the exercise. For this lab, you will be using the Mbed simulator, a simulation of a microcontroller and hardware peripherals.

Here you can see a diagram mapping out the design of the simulated board. The pin descriptions can also be seen.

As previously mentioned, in this lab you will make use of interrupts to toggle some LEDs, making use of the Mbed API where possible.

Begin by opening up the simulator and loading the "Module 4 – Part 1 – Skeleton" lab demo.

We will now go through the code required for completing this project.

In the first part of the exercise, you need to write code that will enable the toggling of LEDs through interrupts. This will be achieved using a mixture of Mbed APIs such as DigitalIn/Out and InterruptIn.

To build upon the skeleton code, the first thing we do is define our outputs. These will be our LEDs are we create three – one for the red LED, yellow LED, and blue LED. This is done using the DigitalOut constructor.

Next, we define our interrupt input objects. These will be buttons and we will create four of them. This is done using the InterruptIn constructor.

Now we can add some code to our main function. First, we set our LED output objects to a default value of 0. This will ensure they are off at the start of the program.

Then we link our button input objects to their respective interrupt handlers, so that when they are pressed the correct response can be carried out. In this case, we have four buttons and create a function that handles each of them.

Finally, in the main function we simulate the processor being put into sleep mode once exiting from the interrupt service routine.

Make note that with physical boards – sleep mode can be used in order to reduce the power consumption of the application. For example, in order to use the 'sleep-on-exit' feature, you can use the 'wait-for-interrupt' operation "underscore W F I".

However, the wait-for-interrupt operation is not implemented in the simulation so we can use "wait ms" instead.

Once this is done, we can populate the handler functions. The first three functions simply set the corresponding LED to the opposite of its current state when the button is pressed. So, if the LED is on – it is turned off, and vice versa.

The fourth handler function simply sets all of the LEDs to be on.

Now add the components and try running the code to test if it works!