arm Education

# Embedded Systems Essentials with Arm: Getting Started

## Module 3

### SV3 (3): Module 3 Lab Project: Part 2

In the next part of the task, you will implement the same function as before, but this time it will use the BusIn and BusOut interfaces instead.

For this section you are required to add code to the function “ControlLED_BusIO”. This will primarily consist of code making use of the BusIn and BusOut interfaces, as well as a switch statement.

Start by commenting out the ‘ControlLED_DigitalIO’ function and uncomment the ‘ControlLED_BusIO’ function.

Referring back to the table containing the information about which LED should be on at a given time, a switch statement that makes a case for each column in the table would look like this.

The numbers 0 to 3 in binary are combinations of switches that set the red LED to be on. It does this by setting the bit that represents the red LED to 1, which is high. Take note that the order of pins in the constructor is the reverse order of the pins in the byte order. If you have BusIn(a,b,c,d,e,f,g,h) then the order of bits in the byte would be h, g, f, e, d, c, b, a, with ‘a’ being bit 0, ‘b’ being bit 1, ‘c’ being bit 2 and so on. So in this case, in the BusOut constructor, the RED_LED is last, but we use the value 0b0100 to set the red light on. This is because the reversal in order is equivalent to 0b0RYB

Case 4 to 5 sets the yellow LED to be on. It does this by setting the yellow LED bit to high.

Case 6 to 7 sets the red and yellow LED to be on. It does this by setting the respective LED bits to high.

Case 8 to 11 sets the yellow and blue LED to be on. It does this by setting the respective LED bits to high.

Case 12 to 14 sets the blue LED to be on. It does this by setting the blue LED bit to high.

Case 15 sets all of the LEDs to be on. It does this by setting all the respective bits to high.

Now run the code and try out different combinations of switches to test if it works!

© Arm 2020