CG2271: Real-Time Operating Systems

Lab 3: Interrupts

Objectives:

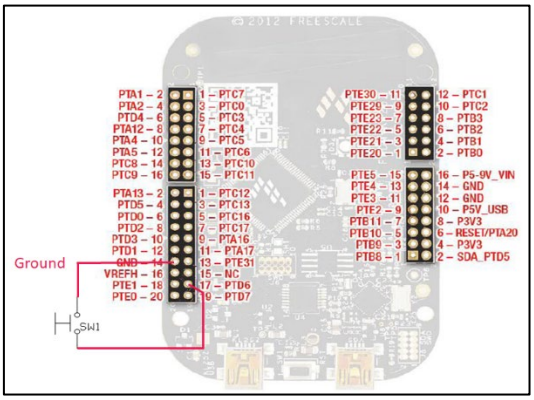
1. The objective of this lab is to understand how we can implement an Interrupt Driven System.
2. We are going to explore how to code an Interrupt Service Routine for the GPIO Line, and use it to control an output LED.
3. We are also going to look at some of the issues with the ISR and how it can better handle the switch.

Introduction

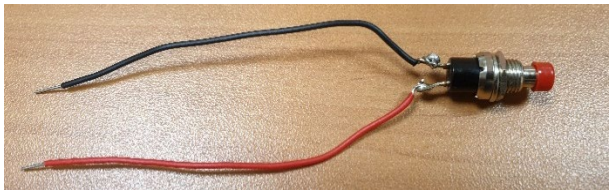
In the last lab, you dealt with GPIO lines and how to develop some simple device driver code to control the on-board RGB LEDs. In the lectures, you have been introduced to the concept of Interrupts and how they can be programmed for the Cortex M0+ that we are using. In this lab, we can going to develop a ISR for an external switch and use the Switch to trigger an Interrupt to the system.

HW Setup

The only HW we need to connect for this lab is the push-button switch to PTD6, as shown in the figure below.



For the Push-Button Switch, please solder 2 wires to both the terminals. So that it looks something like this:



SW Development

Create a new project and add a new source file to your project called myISR.c

You are going to develop the code that you went through in the Lecture Slides. Whenever the Switch at PD6 is pressed, the Interrupt will be triggered and we will jump to the **PORTD_IRQHandler()**.

The IRQ Handler is only to have THREE lines of code as shown below. The lines shown as <….> are the placeholders for the three lines of code.

```
void PORTD_IRQHandler()
{
      // Clear Pending IRQ
      <……>

      // Updating some variable / flag
      <……>

      //Clear INT Flag
      <……>
}
```

Your main code should look something like this:

```
int main(void)
{
      initSwitch();
      initLED();

      while(1)
      {
        <……>
      }
}
```

initSwitch() is to initialize the necessary PORT related functions for the Switch. You can follow the guidelines below:

```
void initSwitch(void)
{
      // enable clock for PortD
      <……>

      /* Select GPIO and enable pull-up resistors and interrupts on
      falling edges of pin connected to switch*/
      <……>

      // Set PORT D Switch bit to input
      <……>

      //Enable Interrupts
      <……>
}
```

Demonstration & Submission

Demo Requirements:

- LED Changing in Correct Sequence, One Colour at a time whenever switch is pressed.

Conclusion

Great Job! You have now gotten a good understanding of how to code ISR's for the Cortex M0+ microcontroller. You can now use this knowledge to extend the real-time nature of your applications even before we move to the RTOS! ☺