Lab 3: Interrupts

Kyle Swanson

March 31, 2015

For this lab, the main focus was on interrupts and how the ARM architecture handles them. We continued using the vector table, learned what interrupts were, and how they related to hardware buttons. We kept learning about the different registers, and how they may be affected by interrupts.

The vector table again played a major part in this lab. For completeness, the vector table addresses were listed in the *statup.s* file, see the example below. Make sure to note the SysTick_Handler as it comes up later in the program.

```
DC32    Stack                   ; 0x00000000   0-Stack Pointer
DC32    __iar_program_start     ; 0x00000004   1-Reset Handler
DC32    NMI_Handler             ; 0x00000008   2-NMI Handler
DC32    HardFault_Handler       ; 0x0000000C   3-Hard Fault Handler
DC32    MemManage_Handler       ; 0x00000010   4-MPU Fault Handler
DC32    BusFault_Handler        ; 0x00000014   5-Bus Fault Handler
DC32    UsageFault_Handler      ; 0x00000018   6-Usage Fault Handler
DC32    SVC_Handler             ; 0x0000002C   11-SVCall Handler
DC32    DebugMon_Handler        ; 0x00000030   12-Debug Monitor Handler
DC32    PendSV_Handler          ; 0x00000038   14-PendSV Handler
DC32    SysTick_Handler         ; 0x0000003C   15-SysTick Handler
```

Figure 1: A selection of the vector table values.

To start, we had a program that simply looped for ever.

```
B       .                ; Pretend the processor is gainfully occupied.
```

As the comment implies, this is to simulate a computer performing actions, like running another program, or receiving input from a keyboard.

Before the infinite loop, the program did a few other tasks, like initializing the GPIO ports, and the interrupts associated with those ports.

To see how ports are initialized, lets inspect the *GPIOF_Init* branch. It starts by using the *LDR* instruction to move SYSCTL_GPIOHBCTL_R into R0.

Continuing through the rest of this branch, it continues to set different values which are essentially defaults to set up the Input Output controller for the switch we want.

Next, we need the Interrupts for our button to be configured. An interrupt is [TODO!!!!!]. The configuration for our interrupt happens in the *GPIOF_Interrupt_Init* branch.