**Lab 02 – Real-Time Clock**

**ECEn 427**

main.c  
debouncer.h  
debouncer.c  
clock.h  
clock.c

**The Parker Brothers**  
*James Parker and Parker Lusk*  
14 September 2015

**Time Spent**  
10 hours

**Enabling Interrupts**  
In this lab, we must enable interrupts by doing the following:

* Telling the Microblaze to allow hardware interrupts.
* Telling the interrupt controller to
  + interrupt the Microblaze when the interrupt controller gets an interrupt
  + allow downstream peripherals to interrupt the controller
* Enabling interrupts on the GPIO block, so that when a button is pressed, it will interrupt the interrupt controller

To do this, we had to enable the GPIO’s global enable and allow interrupts on all pins. Then, an ISR dispatcher had to be registered to the Microblazes one hardware interrupt. After this function is registered, we enable the two downstream peripherals (FIT and GPIO) and enable the interrupt controller’s master enable. The last thing to be done is enabling the Microblaze hardware interrupt.

**Interrupt Handler Steps**  
The ISR dispatcher’s job is to figure out which downstream peripheral triggered an interrupt. It does the following:

1. Get the interrupt status of the interrupt controller.
2. Check the status against the peripheral masks to see which one needs to be serviced.
3. Call the appropriate handler (timer\_interrupt\_handler or pb\_interrupt\_handler) to do its specialized task.

The pb\_interrupt\_handler kicks of the button debouncer and takes care of incrementing the clock when the time is right.

The timer\_interrupt\_handler is the heart of the system, as from it we can keep time and know when the buttons are debounced. It also takes care of refreshing the clock every 200ms.