ECE 8448 Homework Assignment #5

Working with interrupts & without Linux
(For two weeks)

- Reading: Study the sections related to Timer Modules (Sections 2.4 & 2.5.6-10, 16) and Exceptions and Interrupts (Section 3) of the *DE1-SoC Computer System with ARM CORTEX-A9* and *Using ARM GIC* (covered in the lecture).
- Hardware: Use the DE1-SoC Computer.
- Software: Intel Monitor Program (remember to remove the Linux SD card and set the MSEL(0-4) switches to 010011).

Assignment:

Using the GIC_example code (partially discussed in the lecture and can be found under DE1-SoC_Computer folder) as reference, perform the following two steps.

- **Part I**

  Consider the main program shown in Figure 1. The code is required to set up the ARM stack pointer for interrupt mode, initialize some devices, and then enable interrupts. The subroutine *config_GIC( )* configures the GIC to send interrupts to the ARM processor from two sources: HPS Timer 0, and the pushbutton KEYs port. The main program calls the subroutines *config_HPS_timer( )* and *config_KEYS( )* to set up the two ports. You are to write these two subroutines. Set up HPS Timer 0 to generate one interrupt every 0.25 seconds. Other functions can be found under the DE1-SoC Computer folder or in the lecture slides.

  In Figure 1 the main program executes an endless loop writing the value of the global variable *count* to the red lights LEDR. In the interrupt service routine for HPS Timer 0 you are to increment the variable *count* by the value of the *run* global variable, which should be either 1 or 0. You are to toggle the value of the *run* global variable in the interrupt service routine for the pushbutton KEYs, each time a KEY is pressed. When $run = 0$, the main program will display a static count on the red lights, and when $run = 1$, the count shown on the red lights will increment every 0.25 seconds.

  Make a new Monitor Program project for this part, and compile, download, and test your code.

```
int count = 0;                          // global counter for red lights
int run = 1;                            // global, used to increment/not the count variable

int main(void)
{
    volatile int * LEDR_ptr = (int *) 0xFF200000;

    set_A9_IRQ_stack ();                // initialize the stack pointer for IRQ mode
    config_GIC ();                      // configure the general interrupt controller
    config_HPS_timer ();                // configure HPS Timer 0
    config_KEYs ();                     // configure pushbutton KEYs to generate interrupts

    enable_A9_interrupts ();            // enable interrupts in the A9 processor

    while (1)                           // wait for an interrupt
        *LEDR_ptr = count;
}
```

Figure 1: Main program for Part I.

- **Part II**
  Modify your program from Part I so that you can vary the speed at which the counter displayed on the red lights is incremented. All of your changes for this part should be made in the interrupt service routine for the pushbutton KEYs. The main program and the rest of your code should not be changed.

  Implement the following behavior. When KEY_0 is pressed, the value of the *run* variable should be toggled. Hence, pressing KEY_0 stops/runs the incrementing of the *count* variable. When KEY_1 is pressed, the rate at which the *count* variable is incremented should be doubled, and when KEY_2 is pressed the rate should be halved. You should implement this feature by stopping HPS Timer 0 within the pushbutton KEYs interrupt service routine, modifying the load value used in the timer, and then restarting the timer.

What & How to Submit:

- Zip the Monitor project folder, and send it through https://elearning.villanova.edu as message attachments (select ECE 8448 and send a message to me). Please include your last name when naming the zip file.