Lab 5: ARM A9 Timer Interrupts

Reading: Study the ARM HPS Timer (Sections 2.4) and Exceptions and Interrupts (Section 3.2) of the *DE10-Standard Computer System with ARM CORTEX-A9* manual.

Part I

In Lab 4, you implemented a counter in the main () function and used interrupt-driven KEYs to change its behavior. In this lab, you will use a HPS Timer to update your counter every 0.25 seconds when the timer generates an interrupt. Most of the code you had in main () of Lab 4 will be implemented in an ISR for the timer in Lab 5. To accomplish this, you will need to add two functions, config_HPS_timer () and HPS_timer_ISR (), for the Timer and modify various functions. Specifically,

1. You will use four global variables: reset, run, direction, and blink, controlled by the four pushbuttons, respectively. Their purposes are described in the HPS_timer_ISR () and main () section below. The counter variable is also global because both the main () and the HPS_timer_ISR () will need to access it.
2. config_HPS_timer (): Configures the HPS Timer 0 to generate one interrupt every 0.25 seconds. An example for configuring a 1-second timer is shown on Page 52 of the *DE10-Standard Computer System with ARM CORTEX-A9* manual. Calculate the initial Timer value for the 0.25 period used in this lab.
3. HPS_timer_ISR (): Every time the Timer generates an interrupt, increment/decrement the counter value based on the global variables reset, run, and direction.
   a. reset = 1 (KEY0 pressed), the counter restarts from 0 when direction = 0, or 0x3FF when direction is 1. Then set reset to 0.
   b. run = 1, the counter increments or decrements by one depending on the direction variable (0: increment; 1: decrement). When run = 0, the counter keeps its value until run becomes 1.
   c. direction = 0, the counter value increments by one; direction = 1 decrements the counter by one. When the counter value reaches the maximum value that can be represented by 10 LEDRs, i.e. 0x3FF, it restarts from 0. When it reaches 0 when counting down, it restarts from 0x3FF.
4. pushbutton_ISR(): Performs the following depending on the KEY pressed:
   a. KEY0: set the global variable reset to 1; Show 0 on the HEX0;
   b. KEY1: toggles the global variable run; Show 1 on the HEX0;
   c. KEY2: toggles the global variable direction; Show 2 on the HEX0;
   d. KEY3: toggles the global variable blink; show 3 on the HEX0;
5. Add an additional call to config_interrupt with the correct interrupt ID in main () to configure the HPS Timer 0 interrupt.
6. Modify __cs3_isr_irq () in exceptions.c to call the HPS_timer_ISR() when the interrupt ID matches the HPS Timer 0 interrupt ID.
7. main (): Remove the previous code for the counter. Add a call to the function config_HPS_timer (). The only task in the while (1) loop now is to show the current counter value on the LEDRs if the global variable blink is 1 or turn the LEDRs off if blink is 0.

Part II

Modify your pushbutton_ISR () from Part I so that you can vary the speed at which the counter displayed on the LEDRs is incremented. KEY0 & KEY3 keep their current functions, i.e. setting reset & blink. When KEY 1 is pressed, the rate at which the counter 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_ISR(), reading and modifying the initial load value used in the Timer, and then restarting the Timer.

What to submit:

Zip the entire project folders (including source code, project file, and executable code) for Parts I and II, and submit the zip files in Bb.