CECS 346 Lab 5 – Edge Triggered and SysTick Interrupts

Preparation: You will need a TI TM4C LaunchPad.

Book Reading: Textbook Sections 9.4, 9.5, 9.6

Starter Project: Lab5_Interrupts

Reference Code: Textbook Program 9.4 (in Lecture 6 slides), EdgeInterrupt, SysTick

Purpose:

The purpose of this lab is to learn how to use Edge Triggered and SysTick Interrupts, and learn how to use the Memory and Watch windows in Keil uVision simulator during on-board debugging.

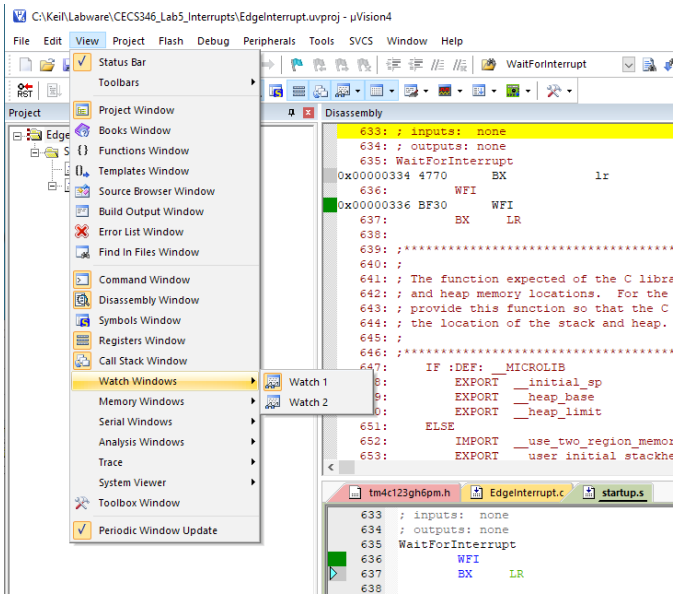
System Requirements:

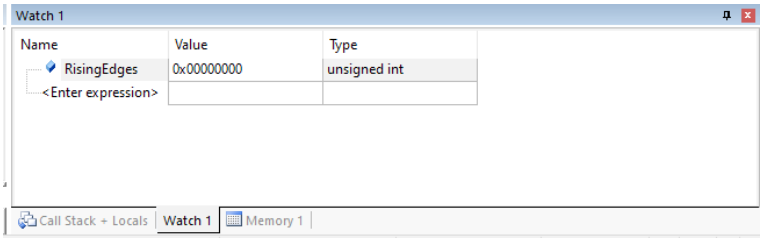
- You will write code using "friendly" initialization to initialize 3 different functionalities such that all 3 functionalities will work regardless of the order the initialization functions are called. You will code keeping in mind the CECS346 Coding Standards.
- In this lab you will flash the onboard red and blue LEDs. Instead of using a software `for` loop to generate 0.1s delay, you will use a SysTick interrupt approach to generate the same time delay.
- You will create a rising edge interrupt on PF0 and count the number of times it is triggered in a global variable. (We will use Keil to see the value of the variable.)

Procedure:

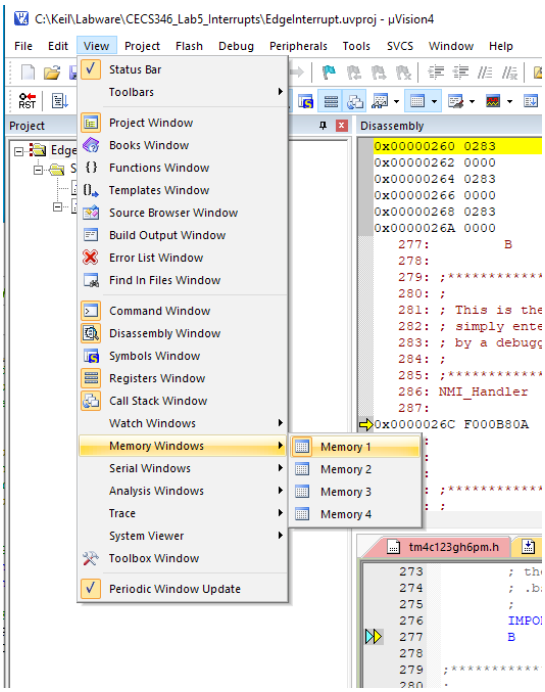
1. Download Lab5_Interrupts.zip and unzip it. Note there are 5 functions that require being implemented:
   a. PortF_LEDInit() - Initialize Port F LEDs
   b. SysTick_Init() - Initialize SysTick timer for 0.1s delay with interrupt enabled
   c. EdgeCounter_Init() - Initialize edge trigger interrupt for PF0 (SW2) rising edge
   d. GPIOPortF_Handler() - Handle GPIO Port F interrupts
   e. SysTick_Handler() - Handle SysTick generated interrupts
2. Inplement PortF_LEDInit() to initialize LEDs only (not any switches) **using friendly initialization**. See HelloLaunchPad for similar code.
3. Implement SysTick_Init() to initialize a SysTick timer with 0.1 sec delay (assuming the board is running at 16 MHz) that generates interrupts **using friendly initialization**. See textbook Program 9.7 for similar code.

4. Implement EdgeCounter_Init() to enable edge trigger interrupt for PF0 (SW2) rising edge **using friendly initialization**. See textbook Program 9.4 / EdgeInterrupt for reference.
5. Implement GPIOPortF_Handler() to do whatever is necessary and increment Rising Edges.See textbook Program 9.4 / EdgeInterrupt for reference.
6. Implement SysTick_Handler() to do whatever is necessary and toggle the red and blue LEDs.
7. Compile and simulate it.
8. View the value of RisingEdges in the Watch 1 window. To open Watch 1 window while debugging, go to View menu -> Watch Windows -> Watch 1.

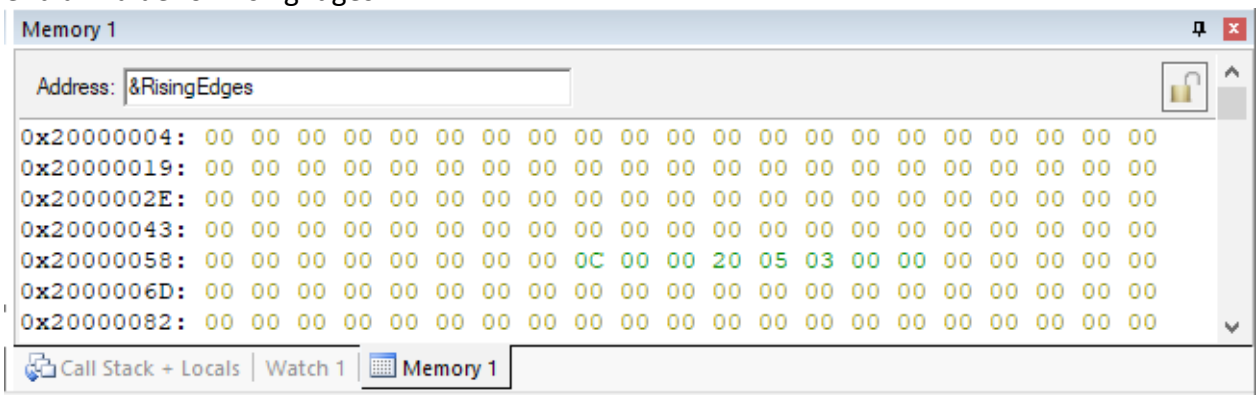




10. View the value of RisingEdges in the Memory 1 window. To open the Memory 1 window while debugging, go to View menu -> Memory Windows -> Memory 1.



11. Once Memory 1 is visible, type "&RisingEdges" in the Address box. The & represents we want to see the memory where RisingEdges is stored, not at the value of RisingEdges. (Think of &RisingEdges as the mailbox, and RisingEdges is what is in the mailbox.) Recall that the Cortex-M4 processor is little endian, so the first 4 bytes in the top left first row are the little endian value for RisingEdges.



1) Demonstrate your lab
   a. Onboard debug – Show RisingEdges incrementing when PF0 (SW2) is released.
   b. Simulator simulation: Setup Logic Analyzer to show the outputs from PF1&2.
   c. On board – Show board flashing
2) Submit to the Beachboard Dropbox
   a. Software source code (Lab5_Interrupts.c).
   b. Screenshot for Logic Analyzer outputs for PF1&2.
   c. Memory 1 screenshot, Watch 1 screenshot (could be same screenshot)