Memo to: Mr. Cory Mettler
From: Julian Rechsteiner
Date: January 19, 2021
Regarding: EELE 465, Lab 1 – LED Heartbeat

Summary:

The purpose of this lab was to use an MSP430-FR2355 microprocessor to flash the onboard LED at 0.5 Hz. This task was accomplished using two different methods. The first method consisted of using loops of counters and the second was by using timer overflows, where the interrupt subroutine gets fired whenever an overflow occurred on the onboard timer. The goal was to produce an LED heartbeat using these two different methods.

Setup:

This project did not require much hardware to set up; the microcontroller was plugged into the computer so that it can interface with Code Composer Studio (CCS). A connection to an Analog Discovery 2 (AD2) was made via three wires, the positive channel 1 wire connecting to port 1.0 and the other two wires (negative channel 1 and AD2 ground) connecting to the onboard ground.

First Method – Delay Subroutine:

The first method implemented, Fig 1 – Fig 3, used delay loops. In order to achieve a 1s delay, two embedded loops were required. Initially, the inner loop started with a count of 0x0FFF and the outer loop with a count of 0x00FF. These initial values were improvised, so the AD2 was used to calibrate to the right values. The inner loop's count was changed to 0x6662 and the outer loop's count was replaced with 10_{10}.

Second Method – Timer Interrupts:

The second method implemented, Fig 4 – Fig 5, used timer interrupts. Equation [1] was used to calculate the initial configuration values for the interrupt. The value of $T_{overflow}$ was determined by using the frequency of the heartbeat F, 0.5 Hz. The overflow should therefore occur every 1 second. The value T was determined by the clock of the processor and the $T_{overflow}$ value. N was the number of binary counts required for the timer to overflow. By dividing the base clock T by 8 and setting the binary count to 2^{12}, a 1s timer overflow would occur, meeting the 0.5 Hz frequency goal.

$$Toverflow = T \cdot N \quad [1]$$

Conclusion:

Both methods produced similar results. The delay loop took a bit longer to perfect as it depended on the speed of the processor; it needed to be tested using an oscilloscope to get the correct value. However, it was easier to implement as it made more sense from a sequential step standpoint, and it is easy to modify the delay time in the future. On the other hand, the timer interrupt was easy to perfect as the configuration values were calculated beforehand, and the implementation code is only a couple lines long. Nonetheless, the interrupt method requires a lot of configuration setup and is not easily modifiable to another delay length for future use.

Lessons Learned:

- Learned how to call subroutines in assembly without using jmp
- Refreshed my memory on how to configure interrupts using mathematics.
- Forgot to use parenthesis when calculating the needed variables.
- Refreshed my memory on how to use the AD2
- Don't go too much into the implementation during the demo

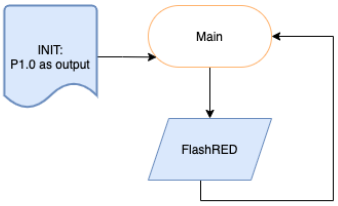

Fig 1: Main Loop

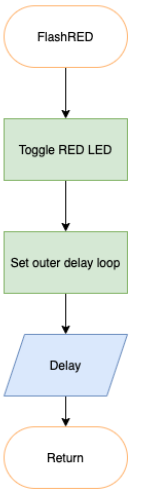

Fig 2: FlashRED Subroutine

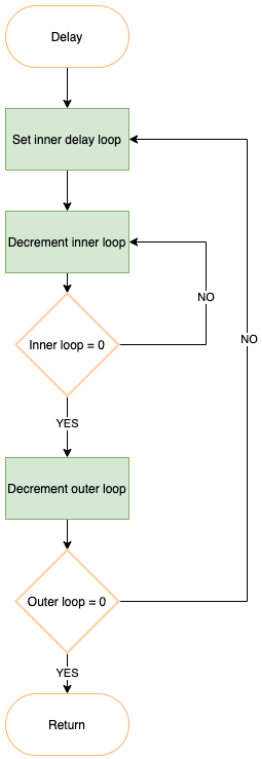

Fig 3: Delay Subroutine

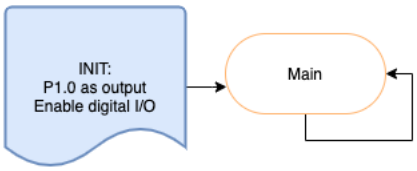

Fig 4: Main Loop

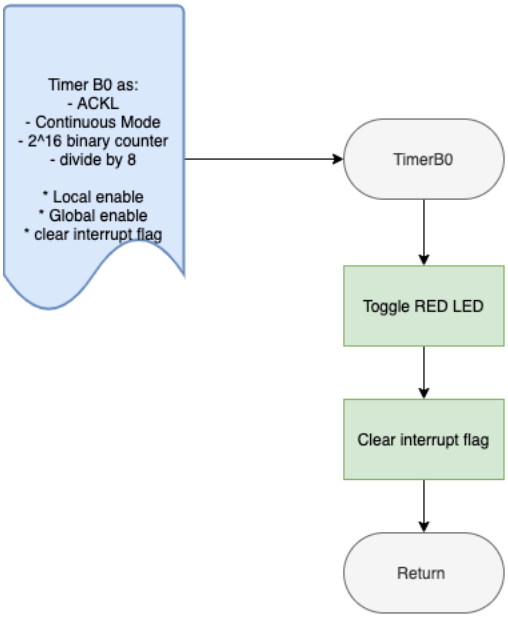

Fig 5: Timer B0 ISR