EELE465 Lab 0

Ryan French
ryanfrench3@montana.edu

Montana State University — January 23, 2020

Introduction

The MSP430FR2355 LaunchPad™Development Kit contains two on-board LEDs which are useful for monitoring the progression of code. In this lab, one of these LEDs was used a "heartbeat indicator": a constantly-flashing signal that implies an active microcontroller. To achieve the given frequency of 0.5 Hz, three methods were implemented: a register-decrement loop, an internal timing interrupt, and a combination of the two methods (flow charts for each of these methods can be found in the appendix). This resulted in subroutines that will be useful in the future, when more complicated firmware is written to the board.

Setup

Since the LaunchPad comes fully assembled, there was no need to manually set up a circuit for the microcontroller. The MCU is flashed by a USB cable connected to a computer running Code Composer Studio. CCS is an IDE created by Texas Instruments, designed specifically for their microcontrollers. It features an integrated debugger and builder/flasher.

Solution 1: Decrement Loop

To implement this solution, it was necessary to figure out how long it took the counter decrement subroutine to complete. Several initial values were used and their frequencies recorded. These values were plotted and subjected to a cubic spline interpolation algorithm to find a factor of 0.5 Hz. This value was placed in the register for decrementing and the decrement subroutine was looped over N times, where N is the quotient of the aforementioned factor and desired frequency.

Algorithm: Decrement Loop

Input: (N, DEC): Number of Loops, Decrement Initial Value

```
set LED ;
while True do
    n = N ;
    while n ≠ 0 do
        dec = DEC ;
        while dec ≠ 0 do
            dec = dec − 1 ;
        end
        n = n − 1 ;
    end
    toggle LED ;
end
```

Solution 2: Timer Interrupt

This solution relies on one of the internal timers of the microcontroller. Initializing the MCU involves enabling the timer overflow interrupt (as well as other setup instructions). Running the code with a prescalar value of 1 reveals the base frequency of the timer overflow interrupt. This value could then be altered to output a frequency of 0.5 Hz (prescalar = 8).

Algorithm: Timer Interrupt

Input: N/A

```
set LED ;
while True do
    if interrupt then
        toggle LED ;
    end
end
```

Solution 3: Decrement and Interrupt

Here, a combination of the previous two methods was used to create a highly-accurate frequency. For this, the prescalar value for the timer was set to 1. This method was implemented by first waiting for a timer interrupt and looping over a decrement subroutine inside the interrupt before returning from the interrupt. To achieve the needed frequency, first a number of loops was chosen ($N = 20$). Then, similar to solution 1, decrement values were plotted against frequencies and a cubic spline interpolation was used to find the optimal value for the initial decrement variable. This method led to a frequency within 0.1% of 0.5 Hz.

Algorithm: Decrement and Interrupt

Input: (N, DEC): Number of Loops, Decrement Initial Value

```
set LED ;
while True do
    if interrupt then
        n = N ;
        while n ≠ 0 do
            dec = DEC ;
            while dec ≠ 0 do
                dec = dec − 1 ;
            end
            n = n − 1 ;
        end
        toggle LED ;
    end
end
```

Comments

The main hurdle encountered in this lab was learning the assembly instructions for this microcontroller, and a lot of time was spent reading through the header files looking for the defined constants (especially when setting up the timer). Another issue was determining the initial decrement value, because there is a turnover once a certain number is used (i.e., increasing the value beyond a certain value caused a discontinuity in the frequency). Once it was decided to interpolate the data, it was easy to find this value.

Appendix

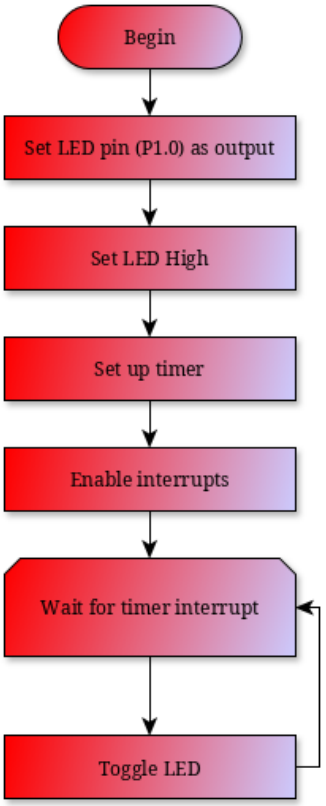

Figure 1: Decrement Loop Solution

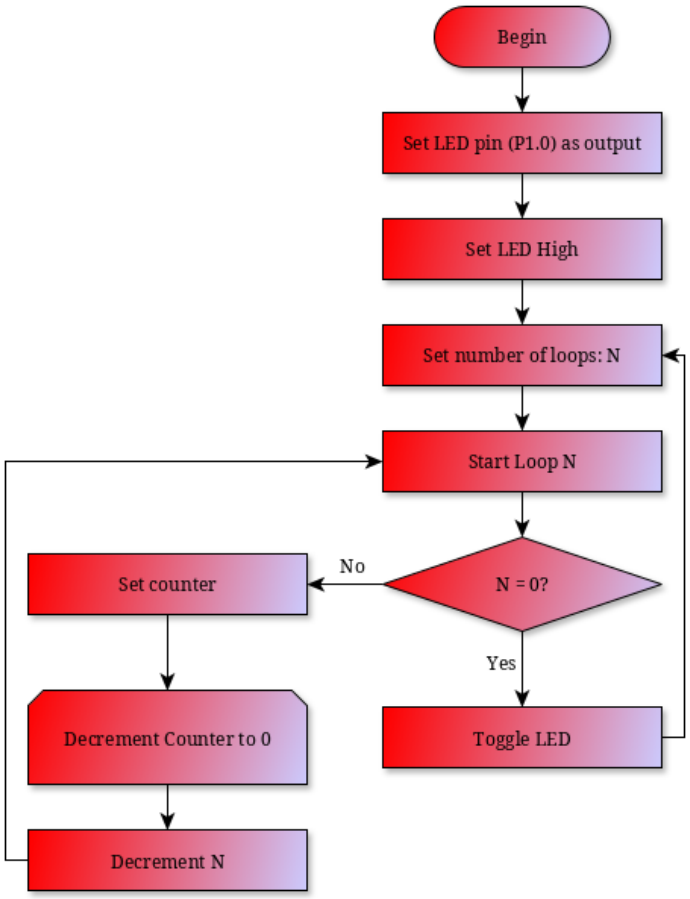

Figure 2: Timer Interrupt Solution

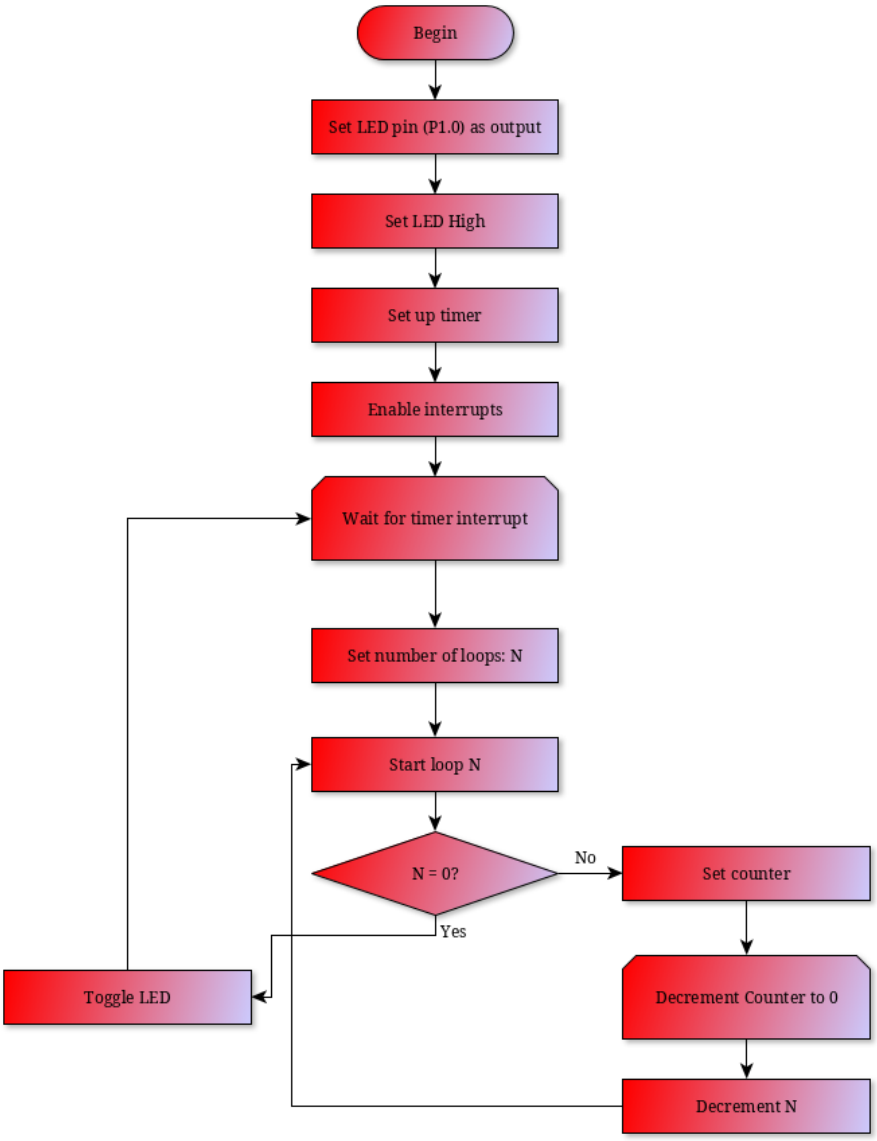

Figure 3: Combined Solution