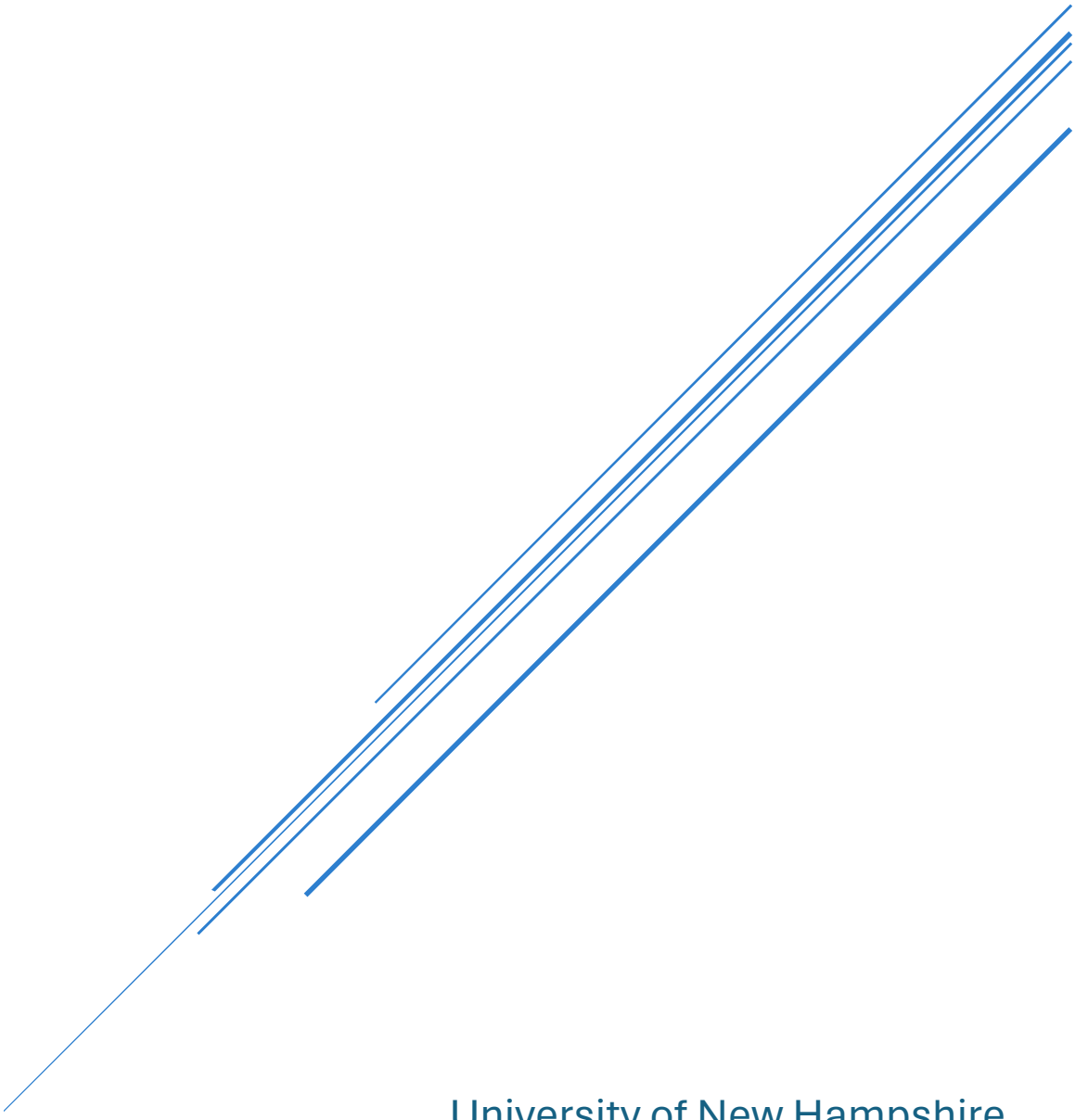
LAB 3: DIMINISHING FREQUENCY CONTROL FOR LED FLASHING

Nicholas Snyder



University of New Hampshire
ECE 649/796

Objectives:

The purpose of this lab is to learn how to use timers and achieve certain clock frequencies. This lab required the configuration of a timer module to flash the onboard LED at ever shortening intervals. This was done by lowering the TA0CCR0 register when the TA0R register reached the set value. There was freedom on how much to decrease TA0CCR0 by.

Background:

The supporting material included many lectures involving the function of timers and what each flag within a register was meant for. This also included a short homework assignment plotting the waveform of a timer.

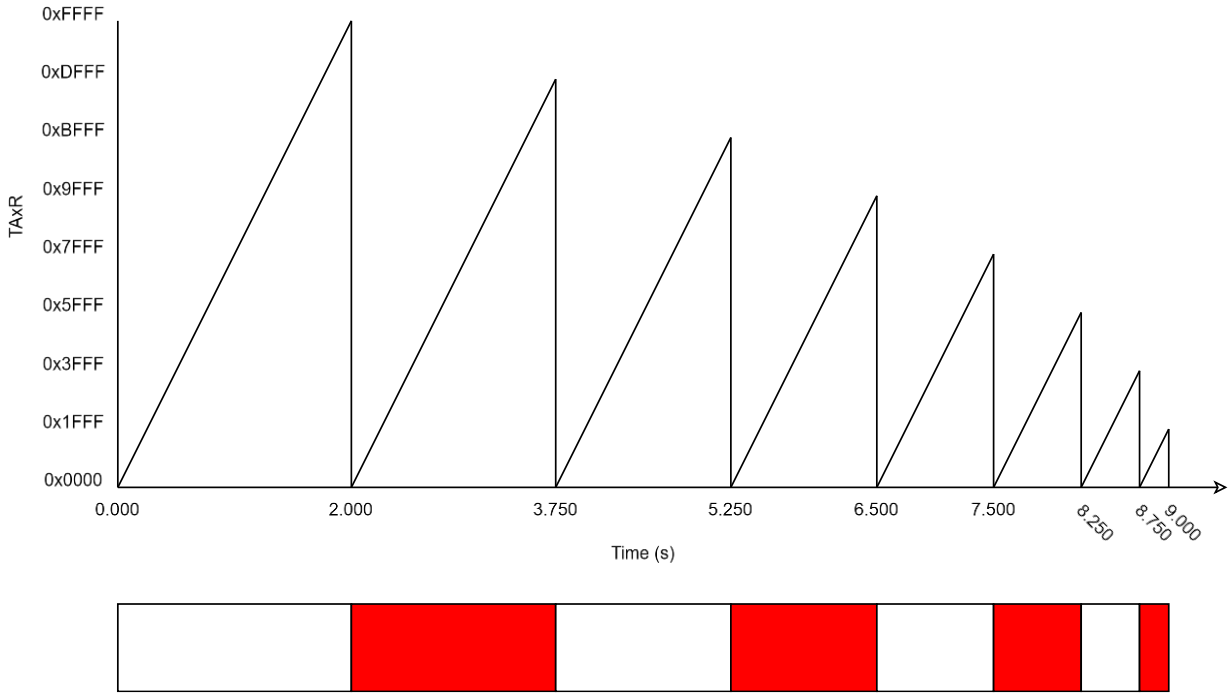
Analysis:

The lab directions included a code template that was commented to show what statements are needed and where they need to be. Filling this template was all that needed to be done to get the code to work as intended. There were also partial examples from the lecture notes of how each statement should be formatted. I chose to decrease TA0CCR0 by 0x2000 each time the interrupt function was called. With an effective frequency of 16kHz and an initial TA0CCR0 of 0xFFFF, this resulted in intervals that shortened by 250 milliseconds each time the interrupt. This behavior is shown in the waveform in the appendix.

Challenges:

While completing this lab, I ran into a simple typing error that prevented the clock from working whatsoever. This stumped both the Professor and I until it was found. This was because the error was not a spelling error but a simple “&” sign that shouldn’t have been where it was. To find this, the correct code was directly compared to my code and only then was it found.

Appendix:




```
#include <msp430fr6989.h>
#define redLED BIT0 // Red LED at P1.0

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // Stop the Watchdog timer
    PM5CTL0 &= ~LOCKLPM5; // Enable the GPIO pins

    // Direct pin as output
    P1DIR = redLED;

    // Turn LED Off
    P1OUT &= ~redLED;

    TA0CCR0 = 0xffff;
    // Configure Channel 0 for up mode with interrupt
    // Enable Channel 0 CCIE bit
    TA0CCTL0 |= CCIE;

    // Clear Channel 0 CCIFG bit
    TA0CCTL0 &= ~CCIFG;

    // Timer_A: ACLK, divide by 2, up mode, clear TAR (leaves TAIE=0)
    TA0CTL = TASSEL_1 | ID_1 | MC_1 | TACLR;
```

```
    // Enable the global interrupt bit (call an intrinsic function)
    _BIS_SR(GIE);
}

//******************************
#pragma vector = TIMER0_A0_VECTOR
__interrupt void T0A0_ISR()
{
    // Toggle the red LED
    P1OUT ^= redLED;

    // clears the flag (CCIFG in TA0CCTL0)
    TA0CCTL0 &= ~CCIFG;

    // Code for changing LED toggling frequency
    TA0CCR0 -= 0x2000;
}
```