# EE 425

# Experiment 1

# Square Wave Generation

Summer 2019

Section 1XC

Group 2

Members: Franklin Lourido, Kelvin Ma, Ryu Ohkawa

## Table of Contents

## Objective

Our goal was to produce square waves of varying frequency and duty cycle for the PIC18F4520 microcontroller. Specifically, we created square waves with half periods of 15ms and 47ms, and one with a duty cycle not equal to 50%. This task was performed primarily through the manipulation of SASM assembly code, although an oscilloscope was used to verify the output signals.

## Overview

A template .asm file was provided prior to the start of the experiment. The template file contained code that made use of subroutines LoopTime and BlinkAlive to control the frequency of outputted square waves on a pin and LED of the microcontroller. The template file used the C2 pin and A4 LED to output its signals, but in our version of the code, the A4 LED was replaced with the A2 LED. The LoopTime subroutine made use of the microcontroller's internal clock to produce delays in between lines of instructions and the BlinkAlive subroutine made use of LoopTime to toggle the appropriate LED on and off. The toggling would occur at predetermined multiples of LoopTime's delay. A variable called ALIVECNT was used to set the number of LoopTime delays the subroutine would wait in between each toggle. Loops used to call LoopTime multiple times were achieved by the use of branching instructions paired with the decrementing and reinitializing of an arbitrary counter variable.

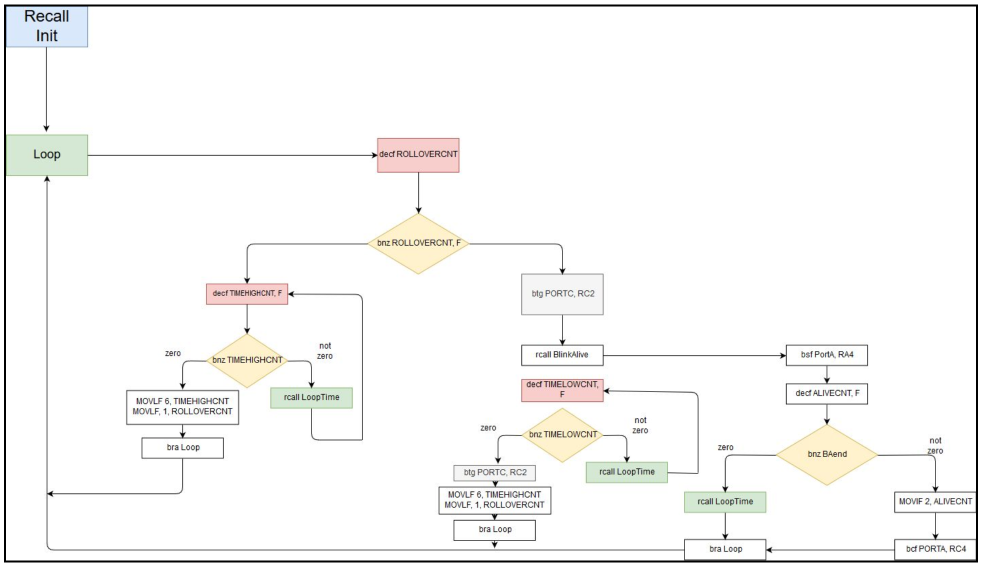
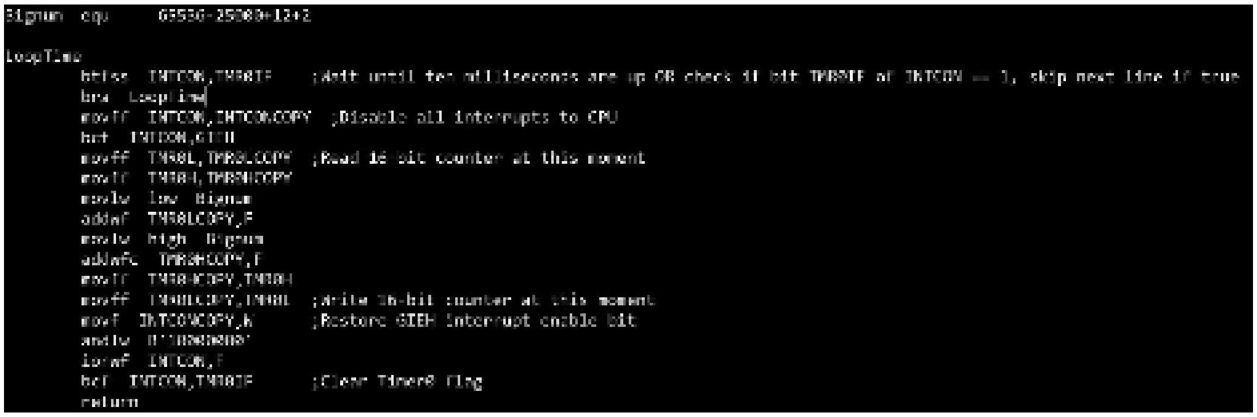
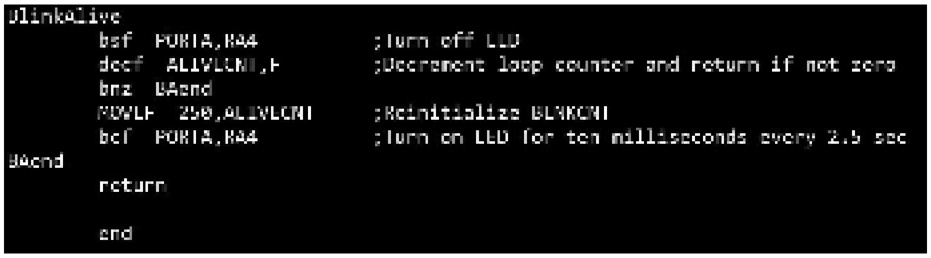


Fig. 1. Flow chart of template file.


```
Bignum  equ     65536-25000+12+2

LoopTime
        btfss  INTCON,TMR0IF     ;Wait until ten milliseconds are up OR check if bit TMR0IF of INTCON == 1, skip next line if true
        bra  LoopTime
        movff  INTCON,INTCONCOPY  ;Disable all interrupts to CPU
        bcf  INTCON,GIEH
        movff  TMR0L,TMR0LCOPY  ;Read 16-bit counter at this moment
        movff  TMR0H,TMR0HCOPY
        movlw  low  Bignum
        addwf  TMR0LCOPY,F
        movlw  high  Bignum
        addwfc  TMR0HCOPY,F
        movff  TMR0HCOPY,TMR0H
        movff  TMR0LCOPY,TMR0L  ;Write 16-bit counter at this moment
        movf  INTCONCOPY,W      ;Restore GIEH interrupt enable bit
        andlw  B'10000000'
        iorwf  INTCON,F
        bcf  INTCON,TMR0IF      ;Clear Timer0 flag
        return
```
Fig. 2. The LoopTime subroutine from the template file with Bignum set to produce a 10ms delay.

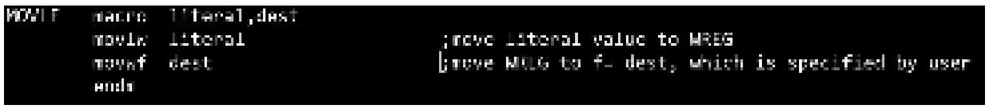


```
BlinkAlive
        bsf  PORTA,RA4          ;Turn off LED
        decf  ALIVECNT,F        ;Decrement loop counter and return if not zero
        bnz  BAend
        MOVLF  250,ALIVECNT     ;Reinitialize BLNKCNT
        bcf  PORTA,RA4          ;Turn on LED for ten milliseconds every 2.5 sec
BAend
        return

        end
```

*Fig. 3. The BlinkAlive subroutine which uses the variable ALIVECNT to determine the number of LoopTime delays in between each logic high state of the A4 LED.*

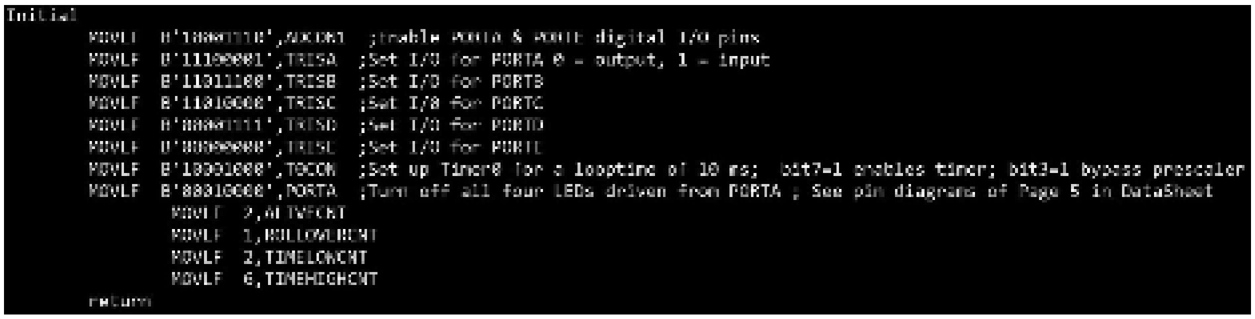
The MOVLF macro seen in BlinkAlive was used to write values directly into a register without the need to explicitly write to the working register first.  Because of the frequent need for literal addressing in this experiment, this macro was useful in reducing the number of instruction lines in our code [1].



```
MOVLF   macro  literal,dest
        movlw  literal                   ;move literal value to WREG
        movwf  dest                      ;move WREG to f- dest, which is specified by user
        endm
```

*Fig. 4. The MOVLF macro used extensively in our code.*

The Initial subroutine was responsible for initializing variable values, the designation of whether a port would be used for input or output, the setting of Timer0 to 10ms, and the turning off of all pins.  Although this subroutine varied slightly between each of our .asm files, all of the Mainline codes began with this subroutine.



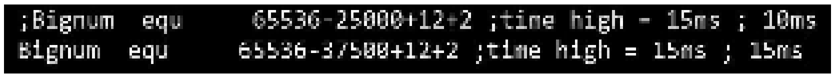
```
Initial
        MOVLF  B'10001110',ADCON1  ;Enable PORTA & PORTE digital I/O pins
        MOVLF  B'11100001',TRISA  ;Set I/O for PORTA 0 = output, 1 = input
        MOVLF  B'11011100',TRISB  ;Set I/O for PORTB
        MOVLF  B'11010000',TRISC  ;Set I/0 for PORTC
        MOVLF  B'00001111',TRISD  ;Set I/O for PORTD
        MOVLF  B'00000000',TRISE  ;Set I/O for PORTE
        MOVLF  B'10001000',T0CON  ;Set up Timer0 for a looptime of 10 ms;  bit7=1 enables timer; bit3=1 bypass prescaler
        MOVLF  B'00010000',PORTA  ;Turn off all four LEDs driven from PORTA ; See pin diagrams of Page 5 in DataSheet
                MOVLF  2,ALIVECNT
                MOVLF  1,ROLLOVERCNT
                MOVLF  2,TIMELOWCNT
                MOVLF  6,TIMEHIGHCNT
        return
```

*Fig. 5. The Initial subroutine.*

## Creating the Square Wave

### *Part 1: The 15ms Square Wave*

The 15ms period square wave was created by adjusting the Bignum value in the LoopTime subroutine.  The frequency of the internal clock was known to be 2.5MHz and therefore, the clock time was 400ns (clock_time = 1/clock_frequency).  To determine the number of clock cycles required in LoopTime, 15ms was divided by the clock time which yielded 37500 clock cycles.  Because the internal clock utilized two 8-bit counters to measure time, the default number of clock cycles for one full turn of the internal clock was 2^18 or 65536 cycles.  Looptime was designed to wait for a number of clock cycles equal to the difference between 65536 and Bignum before moving on to the next instruction line.  Bignum's value was therefore set to 65536 - 37500 + 12 + 2 in order to create a delay of 37500 clock cycles, or 15ms.   The 12 + 2 extra cycles were added to take into account the processing time of instructions.



```
;Bignum  equu     65536-25000+12+2 ;time high = 15ms ; 10ms
Bignum  equ     65536-37500+12+2 ;time high = 15ms ; 15ms
```

*Fig. 6. A comparison of Bignum for 10ms and 15ms.*

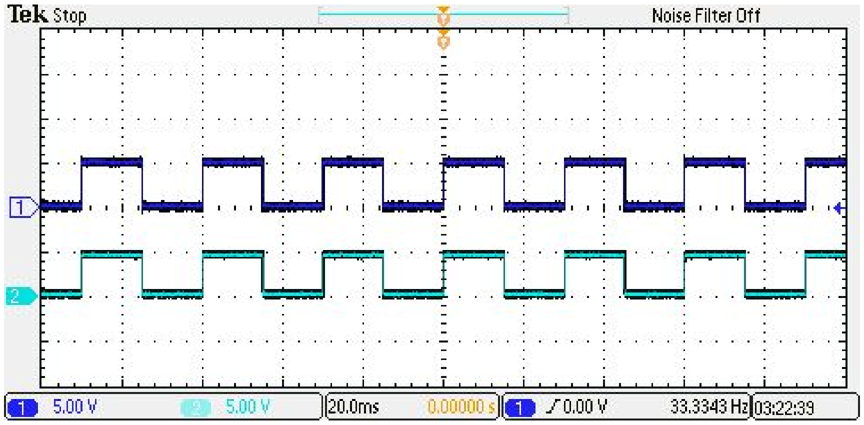
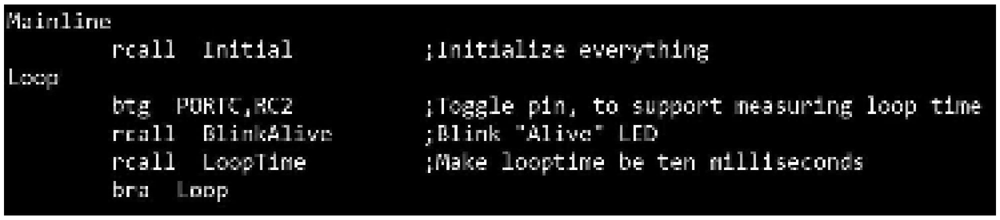


*Fig. 7. Output of pins C2 (Blue) and A2 (Teal) showing 15ms between each toggle.*

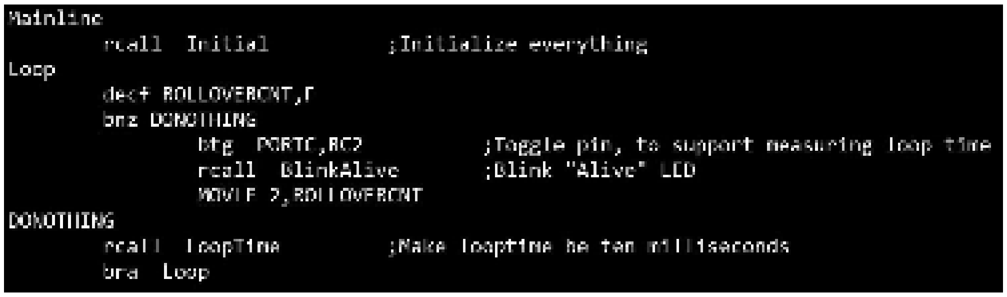
*Part 2: The 47ms Square Wave*

The maximum delay that each LoopTime could produce was limited to 26.21ms by the number of bits in the internal clock and its frequency.  In order to achieve a delay of 47ms, Bignum was set to 65536 - 58750 + 12 + 2 so that the delay of each LoopTime would be 23.5ms, or half of the desired 47ms.  A loop employing a variable called ROLLOVERCNT was then used to double the number of times LoopTime would be called in the Mainline.  ALIVECNT was also initialized to the value 3 to delay the first toggle of the LED pin by a single LoopTime delay.  This synchronized the signals in the A2 LED and C2 pins by delaying the start of the signal in the A2 LED by 3 cycles of the MainLine loop.



```
Mainline
        rcall  Initial          ;Initialize everything
Loop
        btg  PORTC,RC2          ;Toggle pin, to support measuring loop time
        rcall  BlinkAlive       ;Blink "Alive" LED
        rcall  LoopTime         ;Make looptime be ten milliseconds
        bra  Loop
```

*Fig. 8.The  Mainline from the template file toggles the C2 pin, calls BlinkAlive, and calls LoopTime unconditionally with each run of the loop.*



```
Mainline
        rcall  Initial          ;Initialize everything
Loop
        decf ROLLOVERCNT,F
        bnz DONOTHING
                btg  PORTC,RC2          ;Toggle pin, to support measuring loop time
                rcall  BlinkAlive       ;Blink "Alive" LED
                MOVLF 2,ROLLOVERCNT
DONOTHING
        rcall  LoopTime         ;Make looptime be ten milliseconds
        bra  Loop
```

*Fig. 9. The Mainline of the 47ms square wave code only performs the C2 toggle and subroutine calls when ROLLOVERCNT is equal to zero.*

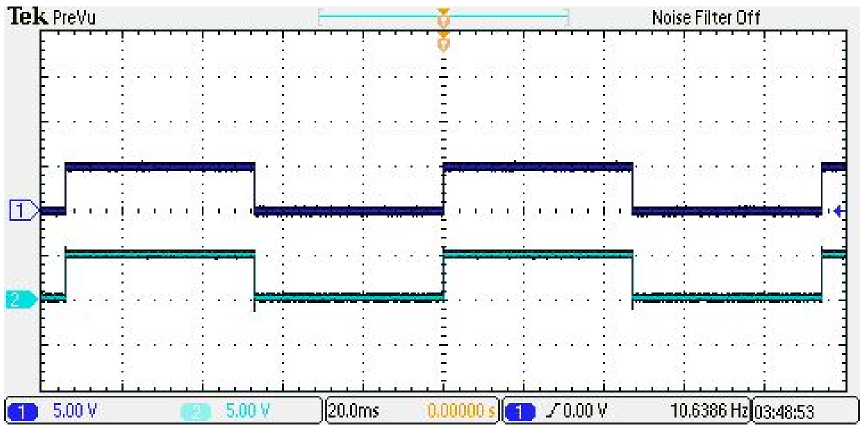
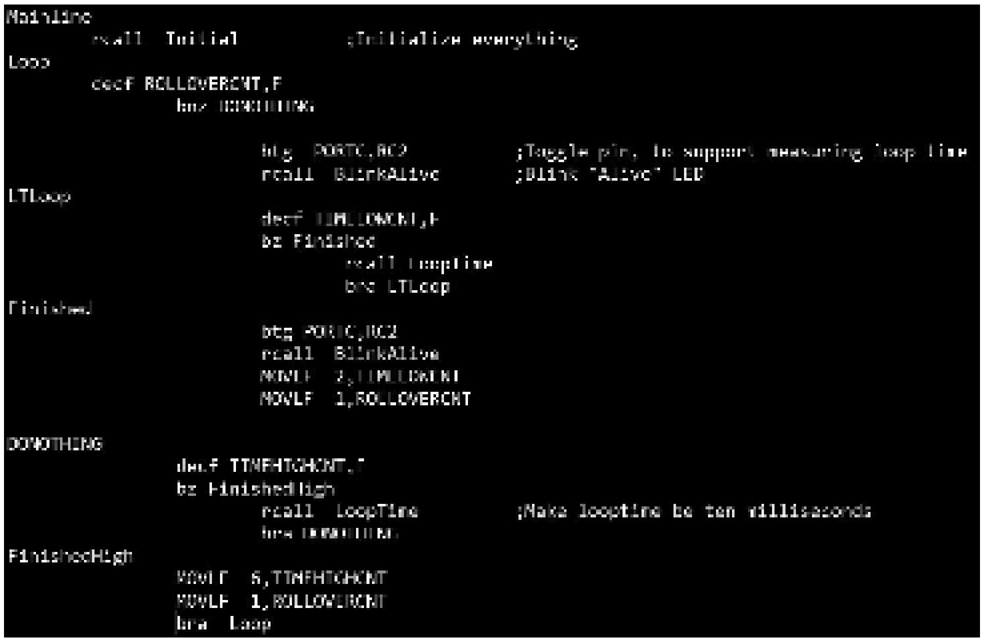


*Fig. 10. Output of pins C2 (Blue) and A2 (Teal) showing 47ms between each toggle.*

*Part 3: Duty cycle other than 50%*

A square wave with a duty cycle other than 50% was achieved with the use of two loops in the Mainline. The loops would call the LoopTime subroutine as many times as was designated by the values of their associated counter variables and would only perform the toggles to their respective pins when the counter became zero. The counter variable used to determine the number of LoopTime delays in the logic high state of pin C2 was called TIMEHIGHCNT and the variable used for the logic low time was called TIMELOWCNT. The loops would run LoopTime one less time than the value of their counter variables. For example, if TIMEHIGHCNT was reinitialized in the loop to be 3 and TIMELOWCNT to 10, then the duty cycle would be (3 - 1) / ((3 - 1) + (10 - 1))*100% or 18.2%.



```
Mainline
        rcall  Initial          ;Initialize everything
Loop
        decf ROLLOVERCNT,F
                bnz DONOTHING

                        btg  PORTC,RC2          ;Toggle pin, to support measuring loop time
                        rcall  BlinkAlive       ;Blink "Alive" LED
LTLoop
                        decf TIMELOWCNT,F
                        bz Finished
                                rcall LoopTime
                                bra LTLoop
Finished
                        btg PORTC,RC2
                        rcall  BlinkAlive
                        MOVLF  2,TIMELOWCNT
                        MOVLF  1,ROLLOVERCNT

DONOTHING
                decf TIMEHIGHCNT,F
                bz FinishedHigh
                        rcall  LoopTime         ;Make looptime be ten milliseconds
                        bra DONOTHING
FinishedHigh
                MOVLF  6,TIMEHIGHCNT
                MOVLF  1,ROLLOVERCNT
                bra  Loop
```

*Fig. 11. The 3 loops in Mainline allow the user to set the duty cycle to a desired value by adjusting the reinitialization values of TIMELOWCNT AND TIMEHIGHCNT variables.*

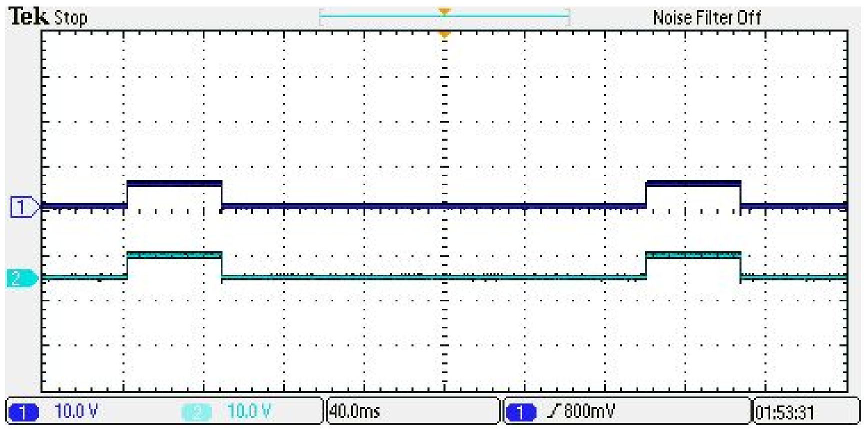


*Fig. 12. Output of pins C2 (Blue) and A2 (Teal) showing a duty cycle of approximately 18.2%.*

## Conclusion

The challenge of this experiment was familiarizing ourselves with the SASM instructions. Once we understood how to use the branch and decrement operators to build simple loops, creating square waves with half periods of some multiple of the LoopTime delay was simple. We also found that variable duty cycles could be produced by explicitly calling LoopTime multiple times in the Mainline. This method however, would become less practical when trying to achieve very small or very large duty cycles. A 1% duty cycle for example, would require 100 instructions in the Mainline dedicated to calling LoopTime alone, whereas the same result could be achieved with the editing of 2 lines of code with the adoption of our loops.

The code could likely be made more robust if the loops were written into a macro. The macro could take a value as an argument to replace TIMEHIGHCNT and use the value to run LoopTime any number of times. This macro would be written in the place of each of the two loops in the Mainline to reduce the instruction count even further.

## Works Cited

Peatman, John B. Embedded Design with the PIC18F452 Microcontroller . Upper Saddle River, N.J.: Prentice Hall, 2003. Print