**THE CITY COLLEGE OF NEW YORK**

160 Convent Avenue, New York, NY 10031

*The City University of New York*



**EE 42500 Computer Engineering Lab**

**Experiment 3**

**Pulse Train Synchronization**

**Prof. Hakan Peckan**

Jonathan Martinez

Summer 2020

## Introduction

In this lab we will be implementing our knowledge about programing the microcontroller and we will be creating a pulse train synchronization. We will use the first program as a template and add some subroutines that will solve the tasks

## Analysis

The oscilloscope images below has time divisions of 0.1ms each, meaning that we will create a LoopTime delay of 0.1ms. We will use our formula once again:

Therefore we can set Bignum now:

Bignum = 65536-250+12+2



Figure 1 shows how the pins should look like on port C. We can see that we can break down the pulse train into 4 sections, in between each section we will have a delay of 0.1ms, and the code looks as follows. We will implement the usage of a counter to keep track of the current section within the code execution.

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

MOVLF B'11111111',TMR0H ;ADDED by AC

MOVLF B'00000000',TMR0L ;ADDED by AC

MOVLF B'00000000',PORTC

MOVLF B'00001000',TEMP8

MOVLF B'00000111',TEMP7

MOVLF B'00000110',TEMP6

MOVLF B'00000101',TEMP5

MOVLF B'00000100',TEMP4

MOVLF B'00000011',TEMP3

MOVLF B'00000010',TEMP2

MOVLF B'00000001',TEMP1

MOVLF B'00000100',COUNTER

return

Here we created new TEMP variables with a number next to it, this will act as constants for later usage. We now implement the code to create the pulse train.

PulseTrain25

movf COUNTER, W ;copy COUNTER into working register

cpfseq TEMP4 ;compare TEMP4 to working register, skip line if W=4

bra JMP25\_1 ;w =!4

bsf PORTC, RC0 ;set/clr if w=4

bcf PORTC, RC3

bra DECRE25

JMP25\_1

cpfseq TEMP3 ;compare TEMP4 to working register, skip line if W=3

bra JMP25\_2 ;w =!3

bsf PORTC, RC1 ;set/clr if w=3

bcf PORTC, RC0

bra DECRE25

JMP25\_2

cpfseq TEMP2 ;compare TEMP4 to working register, skip line if W=2

bra JMP25\_3 ;w =!2

bsf PORTC, RC2 ;set/clr if w=2

bcf PORTC, RC1

bra DECRE25

JMP25\_3

cpfseq TEMP1 ;compare TEMP4 to working register, skip line if W=1

bra JMP25\_4 ;w =!1

bcf PORTC, RC2 ;set/clr if w=1

bsf PORTC, RC3

bra DECRE25

JMP25\_4

DECRE25 decf COUNTER,F ;Decrement loop counter and return if not zero

bnz PT25end

MOVLF 4,COUNTER ;Reinitialize COUNTER

PT25end

Return

We now have the subroutine for task 1, note that the counter starts at 4 and is decreased. During the subroutine we are checking what the value of the counter is. If the counter matches the expected value the corresponding ports are changed and then a branch is performed to DECRE25. DECREE25 is the name of the label that points to the decrement portion of the subroutine. Note that the subroutine is labeled PulseTrain25, due to a 25% duty cycle. We now move onto the pulse train of 50% duty cycle and the same ideas are applied. The code is as follows.



Figure 2 shows what the pins should look like on an oscilloscope, the code to implement this is as follows:

PulseTrain50

movf COUNTER, W ;copy COUNTER into working register

cpfseq TEMP4 ;compare TEMP4 to working register, skip line if W=4

bra JMP50\_1 ;w =!4

bsf PORTC, RC0 ;set/clr if w=4

bcf PORTC, RC2

bra DECRE50

JMP50\_1

cpfseq TEMP3 ;compare TEMP4 to working register, skip line if W=3

bra JMP50\_2 ;w =!3

bsf PORTC, RC1 ;set/clr if w=3

bcf PORTC, RC3

bra DECRE50

JMP50\_2

cpfseq TEMP2 ;compare TEMP4 to working register, skip line if W=2

bra JMP50\_3 ;w =!2

bsf PORTC, RC2 ;set/clr if w=2

bcf PORTC, RC0

bra DECRE50

JMP50\_3

cpfseq TEMP1 ;compare TEMP4 to working register, skip line if W=1

bra JMP50\_4 ;w =!1

bsf PORTC, RC3 ;set/clr if w=1

bcf PORTC, RC1

bra DECRE50

JMP50\_4

DECRE50

decf COUNTER,F ;Decrement loop counter and return if not zero

bnz PT50end

MOVLF 4,COUNTER ;Reinitialize COUNTER

PT50end

Return

Once again the counter starts at 4 and is decreased at the end of the subroutine. During the subroutine we are checking what the value of the counter. If the counter matches the expected the corresponding ports are changed and then a branch is performed to DECRE50.



For task 3 we need to reduce our rollover period by half, this is because the pulse trains have a shift by half of a division, therefore a new rollover period is halved of that from the previous. We can also see that the overall period remains the same, but in order to create this signal we double the amount of delays. Halved rollover period but doubled the number of delays means same period.

Bignum equ 65536-125+12+2 ; task 1,2 cycles=250 and task 3 cycles=125

The code is as follows

PulseTrain37

movf COUNTER, W ;copy COUNTER into working register

cpfseq TEMP8 ;compare TEMP4 to working register, skip line if W=8

bra JMP37\_1 ;w =!8

bsf PORTC, RC0 ;if w=8 set RC0

bra DECRE37

JMP37\_1

cpfseq TEMP7 ;compare TEMP4 to working register, skip line if W=7

bra JMP37\_2 ;w =!7

bcf PORTC, RC3 ;if w=7 clr RC3

bra DECRE37

JMP37\_2

cpfseq TEMP6 ;compare TEMP4 to working register, skip line if W=6

bra JMP37\_3 ;w =!6

bsf PORTC, RC1 ;if w=6 set RC1

bra DECRE37

JMP37\_3

cpfseq TEMP5 ;compare TEMP4 to working register, skip line if W=5

bra JMP37\_4 ;w =!5

bcf PORTC, RC0 ; if w=5 clr RC0

bra DECRE37

JMP37\_4

cpfseq TEMP4 ;compare TEMP4 to working register, skip line if W=4

bra JMP37\_5 ;w =!4

bsf PORTC, RC2 ; if w=4 set RC2

bra DECRE37

JMP37\_5

cpfseq TEMP3 ;compare TEMP4 to working register, skip line if W=3

bra JMP37\_6 ;w =!3

bcf PORTC, RC1 ; if w=3 clr RC1

bra DECRE37

JMP37\_6

cpfseq TEMP2 ;compare TEMP4 to working register, skip line if W=2

bra JMP37\_7 ;w =!2

bsf PORTC, RC3 ; if w=2 set RC3

bra DECRE37

JMP37\_7

cpfseq TEMP1 ;compare TEMP4 to working register, skip line if W=1

bra JMP37\_4 ;w =!1

bcf PORTC, RC2 ; if w=1 clr RC2

bra DECRE37

DECRE37

decf COUNTER,F ;Decrement loop counter and return if not zero

bnz PT37end

MOVLF 8,COUNTER ;Reinitialize COUNTER

PT37end

Return

Throughout the subroutine code of PulseTrain37 we have several labels that helps in finding the value of the working register, this working register value determines whether or not the bits are set or cleared. Once again at the end of the subroutine we see that the counter is reinitialized to 8. One way to visualize the approach used for this solution, as with previous solutions, is as follows.

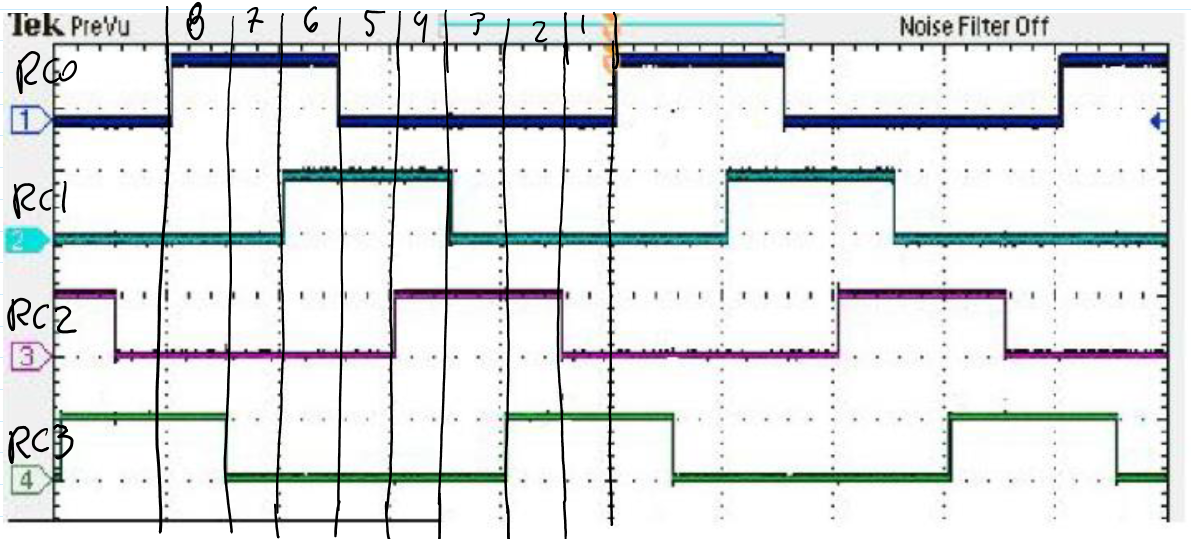


Figure 4: Pulse train divided into 8 sections

Here we make some remarks: on 8 RC0 is set, on 7 RC3 is cleared, 6 RC1 is set, 5 RC0 is cleared, 4 RC2 is set, 3 RC1 cleared, 2 RC3 is set 1 RC2 is cleared.

## Code Execution

Here we run the PulsTrain25 subroutine and analyze the pins on a simulated logic analyzer:

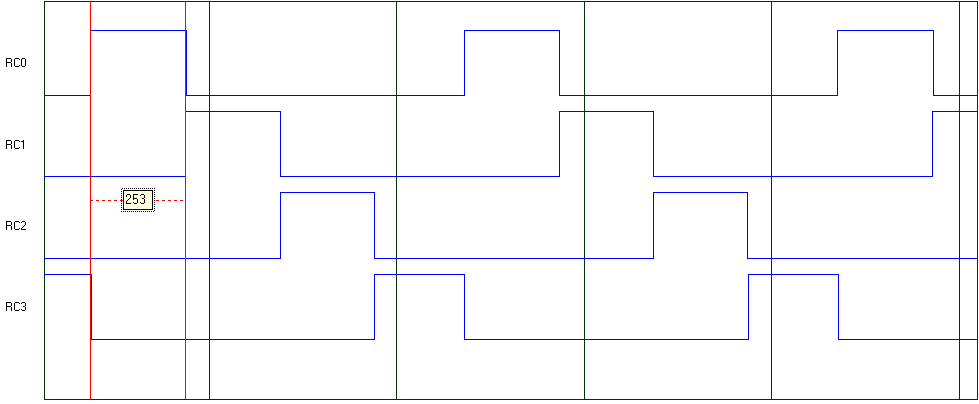


Figure 5

From figure 5 we see that the rollover period takes ~250 cycles which translates to 0.1ms. We have then have a period of 0.4ms. This result also matches our expected values from figure 1. We now run PulseTrain50 and see the result.

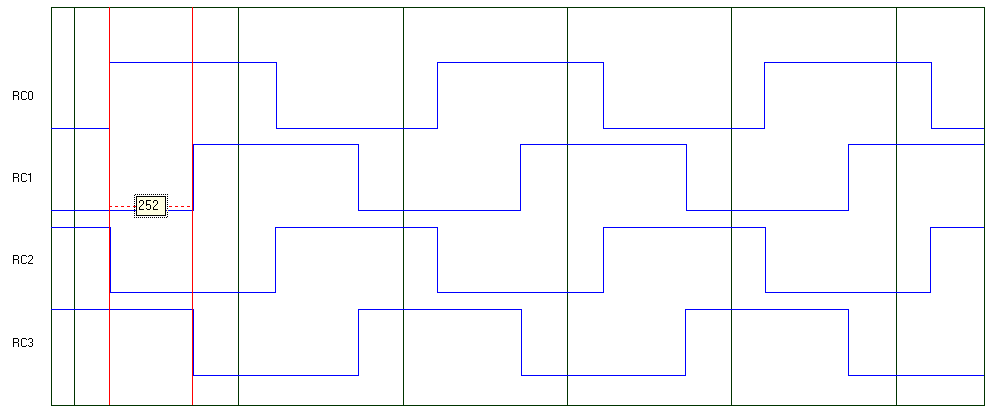


Figure 6

Here we have the result from PulseTrain50, we see a duty cycle of 50%. Once again a rollover period of ~250 cycles which puts us at a period of 0.4ms. The pulse train works as intended. We now run Pulsetrain37 and monitor the logic analyzer.

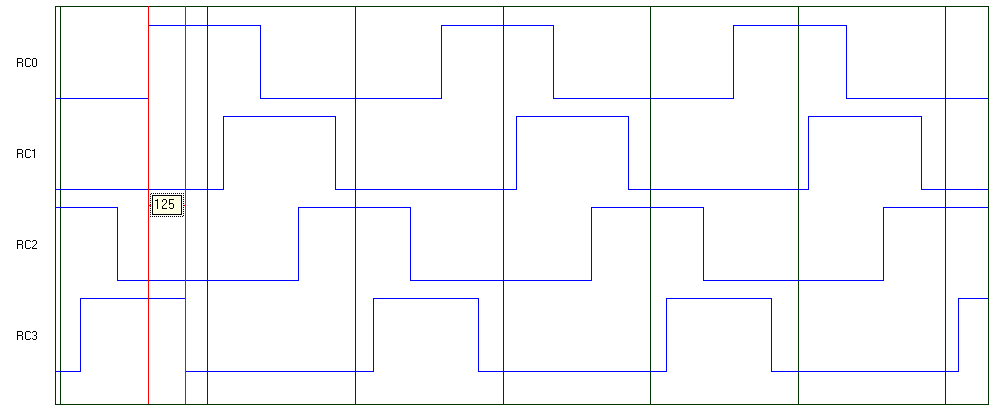


Figure 7

In figure 7 we see a new rollover period of 125 cycles, this equates to 0.05ms of rollover period, however we do have twice the number of rollover periods. Therefore our overall period is still 0.4ms (0.05ms \* 8). We also see the new duty cycle of 37.5%. The logic analyzer shows what we expected to see.

## Conclusion

It is important to note that the value of Bignum changes based on the subroutine used. The subroutine PulseTrain37 needs a different value of Bignum due to the unique rollover period that is required, this is where the two expressions come into play.

For task 1 and 2:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Rollover # / Iteration (each at 0.1ms)** | | | | | | | | | |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Variable** | RC0(state) | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |  |
| RC1(state) | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |  |
| RC2(state) | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |  |
| RC3(state) | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |  |
| COUNTER | 3 | 2 | 1 | 4 | 3 | 2 | 1 | 4 | 3 |  |

Table 1: Task 1

Given that the counter is decremented immediately after the setting/clearing of the pins we will consider the counter to begin at 1 count less than initial.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Rollover # / Iteration (each at 0.1ms)** | | | | | | | | | |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Variable** | RC0(state) | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |  |
| RC1(state) | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |  |
| RC2(state) | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |  |
| RC3(state) | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |  |
| COUNTER | 3 | 2 | 1 | 4 | 3 | 2 | 1 | 4 | 3 |  |

Table 2: Task 2

With the tables now we can visualize how the counter works in conjunction with the pulse train synchronizer. I will show one more table for task 3

For task 3:

However in both cases we have the same overall periods.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Rollover # / Iteration (each at 0.05ms)** | | | | | | | | | |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Variable** | RC0(state) | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |  |
| RC1(state) | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |  |
| RC2(state) | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |  |
| RC3(state) | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |  |
| COUNTER | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 8 | 7 |  |

Table 3: Task 3

Here in table 3 we see the pins are set and counter is decremented and the loop is performed for the next rollover. Once again the rollover period is halved but rollover number is doubled, therefore we maintain the same overall period for the waveforms.

## Code Used

;;;;;;; P1 for QwikFlash board ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Use 10 MHz crystal frequency.

; Use Timer0 for ten millisecond looptime.

; Blink "Alive" LED every two and a half seconds.

; Toggle C2 output every ten milliseconds for measuring looptime precisely.

;

;;;;;;; Program hierarchy ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Mainline

; Initial

; PulseTrain25, PulseTrain50, PulseTrain37

; LoopTime

;

;;;;;;; Assembler directives ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

list P=PIC18F4520, F=INHX32, C=160, N=0, ST=OFF, MM=OFF, R=DEC, X=ON

#include <P18F4520.inc>

\_\_CONFIG \_CONFIG1H, \_OSC\_HS\_1H ;HS oscillator

\_\_CONFIG \_CONFIG2L, \_PWRT\_ON\_2L & \_BOREN\_ON\_2L & \_BORV\_2\_2L ;Reset

\_\_CONFIG \_CONFIG2H, \_WDT\_OFF\_2H ;Watchdog timer disabled

\_\_CONFIG \_CONFIG3H, \_CCP2MX\_PORTC\_3H ;CCP2 to RC1 (rather than to RB3)

\_\_CONFIG \_CONFIG4L, \_LVP\_OFF\_4L & \_XINST\_OFF\_4L ;RB5 enabled for I/O

errorlevel -314, -315 ;Ignore lfsr messages

;;;;;;; Variables ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

cblock 0x000 ;Beginning of Access RAM

TMR0LCOPY ;Copy of sixteen-bit Timer0 used by LoopTime

TMR0HCOPY

INTCONCOPY ;Copy of INTCON for LoopTime subroutine

COUNTER ;Counter for blinking "Alive" LED

TEMP8

TEMP7

TEMP6

TEMP5

TEMP4

TEMP3

TEMP2

TEMP1

endc

;;;;;;; Macro definitions ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MOVLF macro literal,dest

movlw literal ;move literal value to WREG

movwf dest ;move WREG to f= dest, which is specified by user

endm

;;;;;;; Vectors ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

org 0x0000 ;Reset vector, READ Section 5.7

nop

goto Mainline ;goes to Mainline; thus skipping the interrupts below

org 0x0008 ;High priority interrupt vector

goto $ ;Trap

org 0x0018 ;Low priority interrupt vector

goto $ ;Trap

;;;;;;; Mainline program ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Mainline

rcall Initial ;Initialize everything

Loop

rcall PulseTrain37 ;possible subroutines: PulseTrain25, PulseTrain50, PulseTrain37

rcall LoopTime ; looptime

bra Loop

;;;;;;; Initial subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; This subroutine performs all initializations of variables and registers.

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'00000000',PORTA ;Turn off all four LEDs driven from PORTA ; See pin diagrams of Page 5 in DataSheet

MOVLF B'11111111',TMR0H ;ADDED by AC

MOVLF B'00000000',TMR0L ;ADDED by AC

MOVLF B'00000000',PORTC

MOVLF B'00001000',TEMP8

MOVLF B'00000111',TEMP7

MOVLF B'00000110',TEMP6

MOVLF B'00000101',TEMP5

MOVLF B'00000100',TEMP4

MOVLF B'00000011',TEMP3

MOVLF B'00000010',TEMP2

MOVLF B'00000001',TEMP1

MOVLF B'00000100',COUNTER

Return

;;;;;;; LoopTime subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; This subroutine waits for Timer0 to complete its ten millisecond count

; sequence. It does so by waiting for sixteen-bit Timer0 to roll over. To obtain

; a period of precisely 10000/0.4 = 25000 clock periods, it needs to remove

; 65536-25000 or 40536 counts from the sixteen-bit count sequence. The

; algorithm below first copies Timer0 to RAM, adds "Bignum" to the copy ,and

; then writes the result back to Timer0. It actually needs to add somewhat more

; counts to Timer0 than 40536. The extra number of 12+2 counts added into

; "Bignum" makes the precise correction.

Bignum equ 65536-125+12+2 ;task 1,2=250 and task 3=125

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

;;;;;;; Subroutines ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PulseTrain25

movf COUNTER, W ;copy COUNTER into working register

cpfseq TEMP4 ;compare TEMP4 to working register, skip line if W=4

bra JMP25\_1 ;w =!4

bsf PORTC, RC0 ;set/clr if w=4

bcf PORTC, RC3

bra DECRE25

JMP25\_1

cpfseq TEMP3 ;compare TEMP4 to working register, skip line if W=3

bra JMP25\_2 ;w =!3

bsf PORTC, RC1 ;set/clr if w=3

bcf PORTC, RC0

bra DECRE25

JMP25\_2

cpfseq TEMP2 ;compare TEMP4 to working register, skip line if W=2

bra JMP25\_3 ;w =!2

bsf PORTC, RC2 ;set/clr if w=2

bcf PORTC, RC1

bra DECRE25

JMP25\_3

cpfseq TEMP1 ;compare TEMP4 to working register, skip line if W=1

bra JMP25\_4 ;w =!1

bcf PORTC, RC2 ;set/clr if w=1

bsf PORTC, RC3

bra DECRE25

JMP25\_4

DECRE25

decf COUNTER,F ;Decrement loop counter and return if not zero

bnz PT25end

MOVLF 4,COUNTER ;Reinitialize COUNTER

PT25end

return

PulseTrain50

movf COUNTER, W ;copy COUNTER into working register

cpfseq TEMP4 ;compare TEMP4 to working register, skip line if W=4

bra JMP50\_1 ;w =!4

bsf PORTC, RC0 ;set/clr if w=4

bcf PORTC, RC2

bra DECRE50

JMP50\_1

cpfseq TEMP3 ;compare TEMP4 to working register, skip line if W=3

bra JMP50\_2 ;w =!3

bsf PORTC, RC1 ;set/clr if w=3

bcf PORTC, RC3

bra DECRE50

JMP50\_2

cpfseq TEMP2 ;compare TEMP4 to working register, skip line if W=2

bra JMP50\_3 ;w =!2

bsf PORTC, RC2 ;set/clr if w=2

bcf PORTC, RC0

bra DECRE50

JMP50\_3

cpfseq TEMP1 ;compare TEMP4 to working register, skip line if W=1

bra JMP50\_4 ;w =!1

bsf PORTC, RC3 ;set/clr if w=1

bcf PORTC, RC1

bra DECRE50

JMP50\_4

DECRE50

decf COUNTER,F ;Decrement loop counter and return if not zero

bnz PT50end

MOVLF 4,COUNTER ;Reinitialize COUNTER

PT50end

return

PulseTrain37

movf COUNTER, W ;copy COUNTER into working register

cpfseq TEMP8 ;compare TEMP4 to working register, skip line if W=8

bra JMP37\_1 ;w =!8

bsf PORTC, RC0 ;if w=8 set RC0

bra DECRE37

JMP37\_1

cpfseq TEMP7 ;compare TEMP4 to working register, skip line if W=7

bra JMP37\_2 ;w =!7

bcf PORTC, RC3 ;if w=7 clr RC3

bra DECRE37

JMP37\_2

cpfseq TEMP6 ;compare TEMP4 to working register, skip line if W=6

bra JMP37\_3 ;w =!6

bsf PORTC, RC1 ;if w=6 set RC1

bra DECRE37

JMP37\_3

cpfseq TEMP5 ;compare TEMP4 to working register, skip line if W=5

bra JMP37\_4 ;w =!5

bcf PORTC, RC0 ; if w=5 clr RC0

bra DECRE37

JMP37\_4

cpfseq TEMP4 ;compare TEMP4 to working register, skip line if W=4

bra JMP37\_5 ;w =!4

bsf PORTC, RC2 ; if w=4 set RC2

bra DECRE37

JMP37\_5

cpfseq TEMP3 ;compare TEMP4 to working register, skip line if W=3

bra JMP37\_6 ;w =!3

bcf PORTC, RC1 ; if w=3 clr RC1

bra DECRE37

JMP37\_6

cpfseq TEMP2 ;compare TEMP4 to working register, skip line if W=2

bra JMP37\_7 ;w =!2

bsf PORTC, RC3 ; if w=2 set RC3

bra DECRE37

JMP37\_7

cpfseq TEMP1 ;compare TEMP4 to working register, skip line if W=1

bra JMP37\_4 ;w =!1

bcf PORTC, RC2 ; if w=1 clr RC2

bra DECRE37

DECRE37

decf COUNTER,F ;Decrement loop counter and return if not zero

bnz PT37end

MOVLF 8,COUNTER ;Reinitialize COUNTER

PT37end

return

end