

Appendix D:

Source Codes

Accelerometer Acquisition program (MPLAB)

```
List    p=16F877a
        include "p16f877a.inc"
        __config _cp_off & _wdt_off & _xt_osc & _pwrtc_on

;Reading Accelerometer duty cycle value

;This subroutine collects and calculates T1X, T1Y and T2
;T1X is represented by registers T1XHi and T1Xlo
;T1Y is represented by registers T1YHi and T1Ylo
;T2 is represented by registers T2Hi and T2lo

T1XEndlo    equ    46h
T1XEndHi    equ    47h
T1Ybeginlo  equ    48h
T1YbeginHi  equ    49h
T1YEndlo    equ    50h
T1YEndHi    equ    55h
T1YHi       equ    56h
T1YLo       equ    81h
T1XHi       equ    82h
T1XLo       equ    83h
T2Hi        equ    84h
T2Lo        equ    85h
ZXcalHi     equ    86h
ZXcalLo     equ    87h
ZXActualHi  equ    88h
ZXActualLo  equ    89h
u_term_lo_acce    equ    95h
u_term_hi_acce    equ    96h
KHi         equ    97h
KLo         equ    98h

; Start at the reset vector
org 0x000
goto start

org 0x0004

        incf    Timer1H
        bcf     INTCON,T0IF
        bcf     INTCON,RBIE
        retfie

Start

        bcf     STATUS,RP0
        clrf    PORTA
        clrf    PORTB
```

```

        bsf          STATUS,RP0                ;Bank1
        movlw       B'00000011'              ;Set up the I/O ports
        movwf      TRISA
        movlw       B'00010000'
        movwf      TRISB
        movlw       B'00001111'
        movwf      OPTION_REG
        bcf          STATUS,RP0                ;Bank0
        bsf          INTCON,GIE

        Movlw       b'00100011'
        Movwf      T1CON
        Movlw       b'00000101'
        Movwf      CCP1
        bsf          INTCON,GIE

EdgeA      btfsc   PORTA,0
           Goto   EdgeA

EdgeB      btfss   PORTA,0                ;Look for the high transmission at Ta
           Goto   EdgeB                ;Keep looking for high transmission
           Clrf   TMR1L                ;Start timing
           Clrf   TMR1H
           Bcf    PIR1,TMR1IF          ;Enabling the timer1 overflow interrupt
           bsf    PIE1,TMR1IE

EdgeC      btfsc   PORTA,0                ;Look for the low transmission at Tb
           Goto   EdgeC                ;Keep looking for low transmission
           Movf   TMR1L,w              ;Record and save the time in register T1X
           Movwf T1XEndlo
           Movf   TMR1H
           Movwf T1XEndHi

EdgeD      btfsc   PORTB,2
           goto   EdgeD

EdgeE      btfsc   PORTB,2                ;Look for the high transmission at Tc
           Goto   EdgeE                ;Keep looking for high transmission
           Movf   TMR1L,w              ;Record and save the time in T1Ybeginlo
           Movwf T1Ybeginlo
           Movf   TMR1H,w
           Movwf T1YbeginHi

EdgeF      btfsc   PORTB,2                ;Look for the low transmission at Td
           Goto   EdgeF                ;Keep looking for low transmission

```

```

Movf  TMR1L,w           ;Record and save the time in
Movwf  T1YEndlo        ; T1YEndlo
Movf   TMR1H,w
Movwf  T1YEndHi

Movf   T1YEndHi,w
Movwf  Arg_hi
Movf   T1YEndlo,w
Movwf  Arg_lo
Movf   T1YbeginHi,w
Movwf  Sum_Hi
Movf   T1YbeginLo,w
Movwf  Sum_Lo
call   Subtract
Movf   Sum_Hi,w
Movwf  T1YHi
Movf   Sum_Lo,w
Movwf  T1YLo

```

;CALCULATE T2

;2*(T2Hi,T2Lo) = (T1YEndHi:T1YEndLo)+
;(T1YStartHi:T1YStartLo)-(T1XHi:T1XLo)

```

movf   T1YEndHi,w
movwf  Arg_Hi
movf   T1YEndLo,w
movwf  Arg_Lo
movf   T1YbeginHi,w
movwf  Sum_Hi
movf   T1YbeginLo,w
movwf  Sum_lo
call   add
                                ;Sum_hi,Sum_lo=(T1YEndHi:T1YEendLo)+
movf   T1XEndHi,w           ; (T1YBeginHi:T1YBeginLo)
movwf  Sum_Hi
movf   T1XEndLo,W
movwf  Sum_lo
call   Subtract             ;Sum_hi:Sum_lo = 2*T2
bcf    STATUS,C
rrf    Arg_hi,F             ;rotate one bit means multiply
rrf    Arg_lo,F             ; by two
movf   Arg_hi,W
movwf  T2Hi
movf   Arg_lo,W
movwf  T2Lo
return

```

; Calculation of the Z value based on the formula
 ; $Z_{actual} = (Z_{cal} * T2_{actual})/T2_{cal}$

```
ZActual_value    movf      ZXcalHi,w
                  Movwf     Arg_Hi
                  Movf      ZXcalLo,w
                  Movwf     Arg_Lo
                  Movf      T2Hi,w
                  Movwf     Sum_Hi
                  Movf      T2Lo,w
                  Movwf     Sum_Lo
                  Call       Mul

                  Movf      T2calHi,w
                  Movwf     Divisor1
                  Movf      T2calLo,w
                  Movwf     Divisor0
                  Call       Division
                  Movf      Quo_1,w
                  Movwf     ZXActualHi
                  Movf      Quo_0,w
                  Movwf     ZXActualLo
```

; The x-axis acceleration value is programmed based on the formula
 ; $Acceleration = K*(T1-Z_{actual})/T2_{actual}$

```
X_Accel_value    movf      ZXActualHi,w ; This is to check whether the
                  Subwf     T1XHi,w ; numerator is positive or negative
                  Btfss    STATUS,c
                  Goto     Num_negx
                  Btfss    STATUS,z
                  Goto     Num_posx
                  Movf     ZXActualLo,w
                  Subwf     T1XLo,w
                  Btfss    STATUS,c
                  Goto     Num_posx
```

;This subroutine is chosen if the x-axis acceleration value is negative

```
Num_negx         movf      ZXActualHi,w
                  Movwf     Arg_Hi
                  Movf      ZXActualLo,w
                  Movwf     Arg_Lo
                  Movf      T1XHi,w
                  Movwf     Sum_hi
                  Movf      T1XLo,w
```

```

Movwf    Sum_lo
Call     Subtract
Movlw   KHi
Movwf   Arg_Hi
Movlw   KLo
Movwf   Arg_Lo
Call    Mul

Movf    T2Hi,w
Movwf   Divisor1
Movf    T2Lo,w
Movwf   Divisor0
Call    Division

Movf    T2Hi,w
Movwf   Divisor1
Movf    T2Lo,w
Movwf   Divisor0
Call    Division
movf    Quo_0,w
movwf   u_term_hi_acce
movf    Quo_1,w
movwf   u_term_lo_acce

movf    u_term_hi_acce,w
sublw   b'00000010' ; Upper byte for analog value 3V
call    no_drive
movwf   temp_lo
btfsc  STATUS,c
call    ccw
call    cw

no_drive    movwf   temp_lo    ;No Signal is output to the h-bridge
            btfsc  STATUS,z
            call   next_byte3
            return

;This section is to determine which direction the motor should turn
next_byte3

bcf     STATUS,c
bcf     STATUS,z
bsf     STATUS,RP0
movf    u_term_lo_acce,w
bcf     STATUS,RP0
sublw   b'00000001' ;lower byte for analog value 3V
call    no_drive_test2 ;This is the balanced state value
btfsc  STATUS,c

```

```

call      ccw      ;If no carry then go to ccw subroutine
call      cw       ;If carry then go to cw subroutine

no_drive_test2  movwf  temp_lo
                btfsc  STATUS,z
                call   clr_drive ;No Signal is output to the h-bridge
                return

```

;This ccw subroutine contains a few individual error values, which in turn will call the appropriate duty cycle subroutines.

```

ccw
    movf      u_term_hi_acce,w
    sublw    b'00000000' ; upper byte error value (3 – 1.8)v
    btfsc    STATUS,z
    call     upper_byte

```

```

valtest1
    movf      u_term_hi_acce,w
    sublw    b'00000000' ; upper byte error value (3-1.9)v
    movwf    test_bytelo
    btfsc    STATUS,z
    call     upper_byte2
    movwf    test_bytelo
    btfss    STATUS,c
    call     set_cycle1

```

```

valtest2
    movf      u_term_hi_acce,w
    sublw    b'00000000' ; upper byte error value (3-2)v
    movwf    test_bytelo
    btfsc    STATUS,z
    call     upper_byte3
    movwf    test_bytelo
    btfss    STATUS,c
    call     set_cycle2

```

```

valtest3
    movf      u_term_hi_acce,w
    sublw    b'00000000' ; upper byte error value (3-2.1)v
    movwf    test_bytelo
    btfsc    STATUS,z
    call     upper_byte4
    movwf    test_bytelo
    btfss    STATUS,c
    call     set_cycle3

```

valtest4	<pre> movf u_term_hi_acce,w sublw b'00000000' ; upper byte error value(3-2.2)v movwf test_bytelo btfsc STATUS,z call upper_byte5 movwf test_bytelo btfss STATUS,c call set_cycle4 </pre>
valtest5	<pre> movf u_term_hi_acce,w sublw b'00000000' ; upper byte error value (3-2.3)v movwf test_bytelo btfsc STATUS,z call upper_byte6 movwf test_bytelo btfss STATUS,c call set_cycle5 </pre>
valtest6	<pre> movf u_term_hi_acce,w sublw b'00000000' ; upper byte error value (3-2.4)v movwf test_bytelo btfsc STATUS,z call upper_byte7 movwf test_bytelo btfss STATUS,c call set_cycle6 </pre>
valtest7	<pre> movf u_term_hi_acce,w sublw b'00000000' ; upper byte error value (3-2.5)v movwf test_bytelo btfsc STATUS,z call upper_byte8 movwf test_bytelo btfss STATUS,c call set_cycle7 </pre>
valtest8	<pre> movf u_term_hi_acce,w sublw b'00000000' ; upper byte error value (3-2.6)v movwf test_bytelo btfsc STATUS,z call upper_byte9 movwf test_bytelo btfss STATUS,c </pre>

	call	set_cycle8
valtest9	movf sublw movwf btfsc call movwf btfss call	u_term_hi_acce,w b'00000000' ; upper byte error value (3-2.7)v test_bytelo STATUS,z upper_byte8 test_bytelo STATUS,c set_cycle9
valtest10	movf sublw movwf btfsc call movwf btfss call	u_term_hi_acce,w b'00000000' ; upper byte error value (3-2.8)v test_bytelo STATUS,z upper_byte9 test_bytelo STATUS,c set_cycle10
valtest11	movf sublw movwf btfsc call movwf btfss call call	u_term_hi_acce,w b'00000000' ; upper byte error value (3-2.9)v test_bytelo STATUS,z upper_byte10 test_bytelo STATUS,c set_cycle11 clr_drive
upper_byte	movf sublw movwf btfss call	u_term_lo_acce,w b'11110110' ; lower byte error value (3 – 1.8)v test_bytelo STATUS,c set_cycle1
upper_byte2	movf sublw movwf btfss call call	u_term_lo_acce,w b'11100010' ; lower byte error value (3 – 1.9)v test_bytahi STATUS,c set_cycle1 valtest2
upper_byte3	movf sublw movwf btfss call	u_term_lo_acce,w b'11001101' ; lower byte error value (3 – 2.0)v test_bytahi STATUS,c set_cycle2

	call	valtest3
upper_byte4	movf sublw movwf btfss call call	u_term_lo_acce,w b'10111001' ; lower byte error value (3 – 2.1)v test_bytehi STATUS,c set_cycle3 valtest4
upper_byte5	movf sublw movwf btfss call call	u_term_lo_acce,w b'10100100' ; lower byte error value (3 – 2.2)v test_bytehi STATUS,c set_cycle4 valtest5
upper_byte6	movf sublw movwf btfss call call	u_term_lo_acce,w ;lower byte error value (3 – 2.3)v b'10010000' test_bytehi STATUS,c set_cycle5 valtest6
upper_byte7	movf sublw movwf btfss call call	u_term_lo_acce,w ;lower byte error value (3 – 2.4)v b'01111011' test_bytehi STATUS,c set_cycle6 valtest7
upper_byte8	movf sublw movwf btfss call call	u_term_lo_acce,w ;lower byte error value (3 – 2.5)v b'01100111' test_bytehi STATUS,c set_cycle7 valtest8
upper_byte9	movf sublw movwf btfss call call	u_term_lo_acce,w ;lower byte error value (3 – 2.6)v b'01010010' test_bytehi STATUS,c set_cycle8 valtest9
upper_byte10	movf sublw movwf	u_term_lo_acce,w ;lower byte error value (3 – 2.7)v b'00111110' test_bytehi

```

        btfs          STATUS,c
        call         set_cycle9
        call         valtest10

upper_byte11    movf          u_term_lo_acce,w ;lower byte error value (3 – 2.8)v
                sublw        b'00101001'
                movwf       test_bytehi
                btfs          STATUS,c
                call         set_cycle10
                call         valtest11

upper_byte12    movf          u_term_lo_acce,w ;lower byte error value (3 – 2.9)v
                sublw        b'00010101'
                movwf       test_bytehi
                btfs          STATUS,c
                call         set_cycle11
                goto         $-1

```

;The duty cycle values based on ON and OFF Pulse Width Modulation

```

set_cycle1      movlw        h'E5'                ; 100% duty cycle
                movwf       fr_cnt
                movlw       h'FE'
                movwf       fr_cnt2
                call         dir_chg
                call         ramp_rou

set_cycle2      movlw        h'E7'                ;95% duty cycle
                movwf       fr_cnt
                movlw       h'FD'
                movwf       fr_cnt2
                call         dir_chg
                call         ramp_rou

set_cycle3      movlw        h'E8'                ;90% duty cycle
                movwf       fr_cnt
                movlw       h'FC'
                movwf       fr_cnt2
                call         dir_chg
                call         ramp_rou

set_cycle4      movlw        h'E9'                ;85% duty cycle
                movwf       fr_cnt
                movlw       h'FB'
                movwf       fr_cnt2
                call         dir_chg
                call         ramp_rou

```

set_cycle5	movlw movwf movlw movwf call call	h'EA' fr_cnt h'FA' fr_cnt2 dir_chg ramp_rou	;80% duty cycle
set_cycle6	movlw movwf movlw movwf call call	h'EC' fr_cnt h'F9' fr_cnt2 dir_chg ramp_rou	;75% duty cycle
set_cycle7	movlw movwf movlw movwf call call	h'ED' fr_cnt h'F7' fr_cnt2 dir_chg ramp_rou	;70% duty cycle
set_cycle8	movlw movwf movlw movwf call call	h'EE' fr_cnt h'F6' fr_cnt2 dir_chg ramp_rou	;65% duty cycle
set_cycle9	movlw movwf movlw movwf call call	h'EF' fr_cnt h'F5' fr_cnt2 dir_chg ramp_rou	;60% duty cycle
set_cycle10	movlw movwf movlw movwf call call	h'F1' fr_cnt h'F4' fr_cnt2 dir_chg ramp_rou	;55% duty cycle
set_cycle11	movlw movwf movlw movwf	h'F2' fr_cnt h'F2' fr_cnt2	;50% duty cycle

```

                                call    dir_chg
                                call    ramp_rou

clr_drive    bcf    PORTB,5
             bcf    PORTB,7
             bcf    PORTB,1
             bcf    PORTB,2
             movlw  h'e5'
             movwf  fr_cnt
             call   delay1
             call   dir_chg2

```

; Time delay in ensuring the current is fully flowed to ground before
; Starting to turn the other direction of motor. This is to prevent short circuit from
happening. The delay is about 500ms

```

dir_chg2    movlw  h'FF'
            movwf  cnt4
Con2        movlw  h'FF'
            Movwf  cnt5
            decfsz cnt5,1
            goto  $-1
            decf  cnt4,1
            movf  cnt4,w
            sublw h'be'
            btfss status,z
            goto  Con2
            incf  num_times
            movf  num_times,w
            subwf fr_cnt1,w

            btfss status,z
            goto  dir_chg2
            call  EdgeA

```

;The other time delay for motor to turn the other direction. The function is same as above
;subroutine

```

dir_chg    movlw  h'FF'
            movwf  cnt4
Con3        movlw  h'FF'
            movwf  cnt5
            decfsz cnt5,1
            goto  $-1
            decf  cnt4,1
            movf  cnt4,w

```

```

        sublw    h'be'
        btfss   status,z
        goto    Con3
        incf    num_times
        movf    num_times,w
        sublw   d'20'

        btfss   status,z
        goto    dir_chg
        return

```

; Thie Ramp_rou subroutine is to start the ramping up of motor voltage

```

ramp_rou    movlw    h'fc'
            movwf    ramp_cnt
            movlw    h'ea'
            movwf    ramp_cnt2
            call     PWM_ramp
            decf    ramp_cnt
            movf    fr_cnt,w
            subwf   ramp_cnt,w
            btfss   STATUS,c
            call     PWMbegin
            incf    ramp_cnt2
            call     PWM_ramp
            goto    ramp_rou

```

```

PWM_ramp    bcf     PORTB,7
            bsf     PORTB,4
            call    delay1
            bcf     PORTB,2
            bsf     PORTB,1
            call    delay2
            bcf     PORTB,7
            bcf     PORTB,4
            call    delay1
            bcf     PORTB,2
            bcf     PORTB,1
            call    delay2
            return

```

; Output the PWM signal to respective PORTS

```

PWMbegin    bcf     PORTB,7
            bsf     PORTB,4
            call    delay1
            bcf     PORTB,2
            bsf     PORTB,1
            call    delay2
            bcf     PORTB,7

```

```

bcf      PORTB,4
call    delay1
bcf      PORTB,2
bcf      PORTB,1
call    delay2
incf    cnt3
movf    cnt3,w
sublw   d'50'
btfss   status,z
goto    PWMbegin
clrf    cnt3
clrf    num_times
call    EdgeA

```

;Below are the subroutines to select the appropriate duty cycle values based on the on and ;off time of pulses.

```

set2_cycle1    movlw    h'E5'
                movwf    fr_cnt
                movlw    h'FE'
                movwf    fr_cnt2
                call     dir_chg
                call     ramp_rou

```

```

set2_cycle2    movlw    h'E7'
                movwf    fr_cnt
                movlw    h'FD'
                movwf    fr_cnt2
                call     dir_chg
                call     ramp_rou

```

```

set2_cycle3    movlw    h'E8'
                movwf    fr_cnt
                movlw    h'FC'
                movwf    fr_cnt2
                call     dir_chg
                call     ramp_rou

```

```

set2_cycle4    movlw    h'E9'
                movwf    fr_cnt
                movlw    h'FB'
                movwf    fr_cnt2
                call     dir_chg
                call     ramp_rou

```

```

set2_cycle5    movlw    h'EA'
                movwf    fr_cnt
                movlw    h'FA'
                movwf    fr_cnt2

```

	call	dir_chg	
	call	ramp_rou	
set2_cycle6	movlw	h'EC'	
	movwf	fr_cnt	
	movlw	h'F9'	
	movwf	fr_cnt2	
	call	dir_chg	
	call	ramp_rou	
set2_cycle7	movlw	h'ED'	
	movwf	fr_cnt	
	movlw	h'F7'	
	movwf	fr_cnt2	
	call	dir_chg	
	call	ramp_rou	
set2_cycle8	movlw	h'EE'	
	movwf	fr_cnt	
	movlw	h'F6'	
	movwf	fr_cnt2	
	call	dir_chg	
	call	ramp_rou	
set2_cycle9	movlw	h'EF'	
	movwf	fr_cnt	
	movlw	h'F5'	
	movwf	fr_cnt2	
	call	dir_chg	
	call	ramp_rou	
set2_cycle10	movlw	h'F1'	
	movwf	fr_cnt	
	movlw	h'F4'	
	movwf	fr_cnt2	
	call	dir_chg	
	call	ramp_rou	
set2_cycle11	movlw	h'F2'	
	movwf	fr_cnt	
	movlw	h'F2'	
	movwf	fr_cnt2	
	call	dir_chg	
	call	ramp_rou	
delay1	movlw	h'FF'	; PWM modulation based on the
	movwf	cnt2	; values given by fr_cnt and fr_cnt2

Con	movlw movwf decfsz goto decf movf subwf btfss goto return	h'FF' cnt1 cnt1,1 \$-1 cnt2,1 cnt2,w fr_cnt,w status,z Con
delay2	movlw movwf	h'FF' cnt2
Con5	movlw movwf decfsz goto decf movf subwf btfss goto return	h'FF' cnt1 cnt1,1 \$-1 cnt2,1 cnt2,w fr_cnt2,w status,z Con5
cw	comf comf movf sublw btfsc call	u_term_hi_acce,f u_term_lo_acce,f u_term_hi_acce,w b'00000000' ;Upper byte error value (3.1-3)v STATUS,z upperbyte
val_test1	movf sublw movwf btfsc call movwf btfsc call	u_term_hi_acce,w b'00000000' ;Upper byte error value (3.2-3)v test_bytelo STATUS,z upperbyte9 test_bytelo STATUS,c set2_cycle11
val_test2	movf sublw movwf	u_term_hi_acce,w b'00000000' ;Upper byte error value (3.3-3)v test_bytelo

	<pre> btfsc STATUS,z call upperbyte11 movwf test_bytelo btfsc STATUS,c call set2_cycle10 </pre>	
val_test3	<pre> movf u_term_hi_acce,w sublw b'00000000' ;Upper byte error value (3.4-3)v movwf test_bytelo btfsc STATUS,z call upperbyte10 movwf test_bytelo btfsc STATUS,c call set2_cycle9 </pre>	
val_test4	<pre> movf u_term_hi_acce,w sublw b'00000000' ;Upper byte error value (3.5-3)v movwf test_bytelo btfsc STATUS,z call upperbyte9 movwf test_bytelo btfsc STATUS,c call set2_cycle8 </pre>	
val_test5	<pre> movf u_term_hi_acce,w sublw b'00000000' ;Upper byte error value (3.6-3)v movwf test_bytelo btfsc STATUS,z call upperbyte8 movwf test_bytelo btfsc STATUS,c call set2_cycle7 </pre>	
val_test6	<pre> movf u_term_hi_acce,w sublw b'00000000' ;Upper byte error value (3.7-3)v movwf test_bytelo btfsc STATUS,z call upperbyte7 movwf test_bytelo btfsc STATUS,c call set2_cycle6 </pre>	
val_test7	<pre> movf u_term_hi_acce,w </pre>	

	sublw	b'00000000'	;Upper byte error value (3.8-3)v
	movwf	test_bytelo	
	btfs	STATUS,z	
	call	upperbyte6	
	movwf	test_bytelo	
	btfs	STATUS,c	
	call	set2_cycle5	
val_test8	movf	u_term_hi_acce,w	
	sublw	b'00000000'	;Upper byte error value (3.9-3)v
	movwf	test_bytelo	
	btfs	STATUS,z	
	call	upperbyte5	
	movwf	test_bytelo	
	btfs	STATUS,c	
	call	set2_cycle4	
val_test9	movf	u_term_hi_acce,w	
	sublw	b'00000000'	;Upper byte error value (4.0-3)v
	movwf	test_bytelo	
	btfs	STATUS,z	
	call	upperbyte4	
	movwf	test_bytelo	
	btfs	STATUS,c	
	call	set2_cycle3	
val_test10	movf	u_term_hi_acce,w	
	sublw	b'00000000'	;Upper byte error value (4.1-3)v
	movwf	test_bytelo	
	btfs	STATUS,z	
	call	upperbyte3	
	movwf	test_bytelo	
	btfs	STATUS,c	
	call	set2_cycle2	
val_test11	movf	u_term_hi_acce,w	;Upper byte error value (4.2-3)v
	sublw	b'00000000'	
	movwf	test_bytelo	
	btfs	STATUS,z	
	call	upperbyte2	
	movwf	test_bytelo	
	btfs	STATUS,c	
	call	set2_cycle1	
	call	clr_drive	

upperbyte	movf sublw movwf btfsc call	u_term_lo_acce,w b'00010100' ;lower byte error value (3.1-3)v test_bytehi STATUS,c set2_cycle11
upperbyte2	movf sublw movwf btfsc call call	u_term_lo_acce,w b'00101001' ;lower byte error value (3.2-3)v test_bytehi STATUS,c set2_cycle11 val_test11
upperbyte3	movf sublw movwf btfsc call call	u_term_lo_acce,w b'00111101' ;lower byte error value (3.3-3)v test_bytehi STATUS,c set2_cycle10 val_test10
upperbyte4	movf sublw movwf btfsc call call	u_term_lo_acce,w b'01010010' ;lower byte error value (3.4-3)v test_bytehi STATUS,c set2_cycle9 val_test9
upperbyte5	movf sublw movwf btfsc call call	u_term_lo_acce,w b'01100110' ;lower byte error value (3.5-3)v test_bytehi STATUS,c set2_cycle8 val_test8
upperbyte6	movf sublw movwf btfsc call call	u_term_lo_acce,w b'01111011' ;lower byte error value (3.6-3)v test_bytehi STATUS,c set2_cycle7 val_test7
upperbyte7	movf sublw movwf btfsc call	u_term_lo_acce,w b'10001111' ;lower byte error value (3.7-3)v test_bytehi STATUS,c set2_cycle6

	call	val_test6
upperbyte8	movf sublw movwf btfsc call call	u_term_lo_acce,w ;lower byte error value (3.8-3)v b'10100100' test_bytehi STATUS,c set2_cycle5 val_test5
upperbyte9	movf sublw movwf btfsc call call	u_term_lo_acce,w ;lower byte error value (3.9-3)v b'10111000' test_bytehi STATUS,c set2_cycle4 val_test4
upperbyte10	movf sublw movwf btfsc call call	u_term_lo_acce,w ;lower byte error value (4-3)v b'11001101' test_bytehi STATUS,c set2_cycle3 valtest3
upperbyte11	movf sublw movwf btfsc call call	u_term_lo_acce,w;lower byte error value (4.1-3)v b'11100001' test_bytehi STATUS,c set2_cycle2 valtest2
upperbyte12	movf sublw movwf btfsc call call	u_term_lo_acce,w ;lower byte error value (4.2-3)v b'11110110' test_bytehi STATUS,c set2_cycle1 valtest1

Overall balancing program (Gyro only) (MPLAB)

```
List    p=16F877a
        include "p16f877a.inc"
        __config __cp_off & __wdt_off & __xt_osc & __pwrt_on
```

cnt1	equ	2AH
cnt2	equ	2CH
cnt3	equ	2DH
cnt4	equ	2EH
cnt5	equ	3BH
num_times	equ	3DH
fr_cnt	equ	3FH
fr_cnt2	equ	7AH
fr_cnt1	equ	7BH
test_bytehi	equ	6AH
test_bytelo	equ	6CH
gy_calc_angle_vel_new_hi	equ	20H
gy_calc_angle_vel_new_lo	equ	21H
gy_calc_angle_new_hi	equ	22H
gy_calc_angle_new_lo	equ	23H
tilt_temp_hi	equ	24H
tilt_temp_lo	equ	25H
Kt_tilt_hi	equ	26H
Kt_tilt_lo	equ	27H
Kv_tilt_hi	equ	28H
Kv_tilt_lo	equ	29H
Ka_tilt_hi	equ	30H
Ka_tilt_lo	equ	31H
tilt_rate_hi	equ	32H
tilt_rate_term_lo	equ	33H
gy_calc_angle_old_hi	equ	34H
gy_calc_angle_old_lo	equ	35H
gy_calc_angle_vel_old_hi	equ	36H
gy_calc_angle_vel_old_lo	equ	37H
u_term_hi	equ	38H
u_term_lo	equ	39H
Time_lo	equ	40H
Time_hi	equ	41H
set_point_lo	equ	43H
set_point_hi	equ	45H
prod_res0	equ	54H
prod_res1	equ	53H
prod_res2	equ	52H
prod_res3	equ	51H
Arg_hi	equ	57H

Arg_lo	equ	58H
Sum_hi	equ	59H
Sum_lo	equ	60H
var_hi	equ	61H
var_lo	equ	62H
pwm_bit_cnt	equ	63H
pwm_bit	equ	64H
pwm_bit_cnt2	equ	65H
pwm_bit2	equ	66H
count	equ	67H
TILT_RATE_LO	equ	68H
tilt_term_hi	equ	69H
tilt_term_lo	equ	70H
tilt_rate_term_hi	equ	71H
temp_lo	equ	72H
gy_temp_angle_old_hi	equ	73H
gy_temp_angle_old_lo	equ	74H
tilt_hi	equ	75H
tilt_lo	equ	76H
ramp_cnt	equ	77H
ramp_cnt2	equ	78H

```

; Start at the reset vector
org 0x000
goto start

```

Start

```

bsf STATUS,RP0 ;Bank 1
bcf STATUS,RP1
clrf TRISB ;PORTB [7-0] outputs
movlw b'10000000'
movwf ADCON1 ;Right justified, all A/D
movlw b'00000000'
movwf OPTION_REG
bcf STATUS,RP0 ;Bank 0
movlw B'01011001' ;Fosc/8 [7-6], A/D ch3 [5-3], A/D on [0]
movwf ADCON0
clrf PORTB
clrf cnt3
clrf num_times
movlw d'8'
movwf num_samples

```

Main

```

call ad_portb
goto Main

```

```

ad_portb
                                ;wait for acquisition time (20uS)
                                ;Start A/D conversion
        bsf    ADCON0,GO
Wait
        btfsc  ADCON0,GO        ;Wait for conversion to complete
        goto   Wait

;tilt error value = (ADC(x)-3.0)

comp_ute
        movwf  Arg_lo
        movf   ADRESH,w ; Store in ADC 10-bit result
        movwf  Arg_hi      ; register
        movf   temp_acchi,w
        movwf  Sum_hi
        movf   temp_acclo,w
        movwf  Sum_lo

        call   add
        movf   Sum_hi,w
        movwf  gy_calc_angle_new_hi
        movf   Sum_lo,w
        movwf  gy_calc_angle_new_lo
        decfsz num_samples
        call   wait
        movlw  d'8'          ;Averaging of 8 samples
        movwf  num_samples

        movf   gy_calc_angle_new_hi,w
        movwf  Arg_hi
        movf   gy_calc_angle_new_lo,w
        movwf  Arg_lo
        movlw  b'00000010'
        movwf  Sum_hi
        movlw  b'00000001'
        movwf  Sum_lo
        call   subtract
        movf   Sum_hi,w
        movwf  tilt_temp_hi
        movf   tilt_temp_hi,w
        movwf  Arg_hi
        movf   Sum_lo,w
        movwf  tilt_temp_lo
        movf   tilt_temp_lo,w
        movwf  Arg_lo

        movlw  b'00000000'

```



```

movwf    Kt_tilt_hi
movf     Kt_tilt_hi,w
movwf    var_hi
movlw    b'00000001'
movwf    Kt_tilt_lo
movf     Kt_tilt_lo,w
movwf    var_lo
call     mul
movf     prod_res1,w
movwf    tilt_hi
movf     prod_res0,w
movwf    tilt_lo

```

```

movlw    b'00000000'
movwf    Kv_tilt_hi
movlw    b'00000001'
movwf    Kv_tilt_lo
movf     Kv_tilt_hi,w
movwf    Arg_hi
movf     Kv_tilt_lo,w
movwf    Arg_lo
movf     tilt_hi,w
movwf    var_hi
movf     tilt_lo,w
movwf    var_lo
call     mul
movf     prod_res1,w
movwf    u_term_hi
movf     prod_res0,w
movwf    u_term_lo

```

```

movf     ADRESH,w
sublw    b'00000010' ; Upper byte for analog value 3V
call     no_drive
movwf    temp_lo
btfsc   STATUS,c
call     ccw
call     cw

```

The following program is similar to the list of program from pg 94, no-drive subroutine to pg 109 .

Multiplication program

Mul

```
;movlw      b'00010010'
;movwf      Arg_lo
;movlw      b'00011000'
;movwf      Arg_hi
;movlw      b'10010100'
;movwf      var_lo
;movlw      b'01110000'
;movwf      var_hi
clrf        prod_res0
clrf        prod_res1
clrf        prod_res2
clrf        prod_res3
clrf        count
movlw      d'17'
movwf      count

movf        var_lo,w          ; Place the value from other program
;loop
movwf      prod_res0          ;into the variable of this
;multiplication loop.

btss       status,z          ;Check whether the product_res0 is zero
call       Check_n
movf       var_hi,w          ;the product_res0 is zero
movwf      prod_res1          ;Check the next upper byte.
btfs       status,z
call       equal_zero

Check_n    btss       status,c
Call       add_var
movf       var_hi,w
Movwf      prod_res1

Check_Arg  movf       Arg_Lo,f
btss       status,z          ;Test if value of Arg_Lo is zero
call       test_lsb          ;Arg_lo is not zero
movf       Arg_Hi,w          ;Arg_lo is zero
btfs       status,z          ;Test if value of Arg_Hi is zero
call       equal_zero        ;Arg_Hi is equal to zero
call       test_lsb

add_var    incf       prod_res1
movf       var_hi,w
movwf      prod_res1
```

```

test_lsb    bcf      status,c
            rrf      prod_res3
            rrf      prod_res2
            rrf      prod_res1
            rrf      prod_res0
            btfss   status,c    ;Test if there is carry bit
            call    shift        ; There is no carry
            movf    Arg_lo,w     ;Upper two bytes is added with the bytes
            addwf   prod_res2,f  ;of the Arg_Hi:Arg_Lo.
            Btfsc   status,c
            call    Add_Hi
            movf    Arg_Hi,w
            addwf   prod_res3,f
            call    shift

equal_zero  clrf      prod_res0    ; the multiplication result is zero.
            clrf      prod_res1
            clrf      prod_res2
            clrf      prod_res3
            call    stop          ;Exit from the mul subroutine

Add_Hi      incf      prod_res3
            Movf     Arg_Hi,w
            Addwf   prod_res3

shift       decfsz   count
            call    test_lsb
            goto    stop

stop clrf   count

```

Division program

Division

```
;movlw    b'00010010' ;Inserting the values into divisor and the
;movwf    divisor0    ;number to be divided.
;movlw    b'00011000'
;movwf    divisor1
;movlw    b'00111000'
;movwf    divisor2
;movlw    b'01001000'
;movwf    divisor3
;movlw    b'10010100'
;movwf    prod_res0
;movlw    b'01110000'
;movwf    prod_res1
;movlw    b'00000100'
;movwf    prod_res2
;movlw    b'11100000'
;movwf    prod_res3
```

Clrf

```
rem_dr0    ;Initialise variables by clearing all the values
Clrf       rem_dr1
clrf       rem_dr2
clrf       rem_dr3
clrf       Quo_0
clrf       Quo_1
clrf       Quo_2
clrf       Quo_3
Movlw     32
Movwf     bitcnt
```

Loop

```
rlf       prod_res0    ;Clear the 32 bit result registers
rlf       prod_res1
rlf       prod_res2
rlf       prod_res3
rlf       rem_dr0
rlf       rem_dr1
rlf       rem_dr2
rlf       rem_dr3
```

```
movf      divisor3,w    ;Compare divisor and remainder
```

```

subwf    rem_dr3,w
btfss   STATUS,z
call    test_more3    ;test if rem_dr3 is more than divisor if not

equal   movf    divisor2,w    ;High byte is equal, so test the lower byte
subwf   rem_dr2,w
btfss   STATUS,z
call    test_more2
movf    divisor1,w    ;Test the lower bytes
subwf   rem_dr1,w
btfss   STATUS,z
call    test_more1
movf    divisor0
subwf   rem_dr0,w
call    test_more0

```

;These sections are to test the result byte whether the divisor value is
;larger or smaller than the rem_dr value.
;If the rem_dr value is larger then the divisor then goto subs subroutine

```

test_more3  btfss   STATUS,c
            call    last
            call    subs2

test_more2  btfss   STATUS,c
            call    last
            call    subs1

test_more1  btfss   STATUS,c
            call    last
            call    subs0

test_more0  btfss   STATUS,c
            call    last
            bsf    prod_res0,0
            call    last

```

;These sections subs2, subs1 and subs3 are to compare
;the adjacent bytes on whether there is any carry bits.
;If there is any carry, adjust affected bytes by decreasing one
;bit value

```

subs2      movf      divisor2,w
           subwf     rem_dr2,w
           btfss    STATUS,c
           decf     rem_dr3
           movf     divisor3,w
           subwf     rem_dr3,w
           movf     divisor1,w
           subwf     rem_dr1,w
           btfss    STATUS,c
           decf     rem_dr2
           movf     divisor2,w
           subwf     rem_dr2,w
           movf     divisor0,w
           subwf     rem_dr0,w
           btfss    STATUS,c
           decf     rem_dr1
           movf     divisor1,w
           subwf     rem_dr1,w
           bsf      prod_res0,0
           call     last

```

```

subs1      movf      divisor1,w
           subwf     rem_dr1,w
           btfss    STATUS,c
           decf     rem_dr2
           movf     divisor2,w
           subwf     rem_dr2,w
           movf     divisor0,w
           subwf     rem_dr0,w
           btfss    STATUS,c
           decf     rem_dr1
           movf     divisor1,w
           subwf     rem_dr1,w
           bsf      prod_res0,0
           call     last

```

```

subs0      movf      divisor0,w
           subwf     rem_dr0,w
           btfss    STATUS,c
           decf     rem_dr1
           movf     divisor1,w
           subwf     rem_dr1,w
           bsf      prod_res0,0
           call     last

```

```

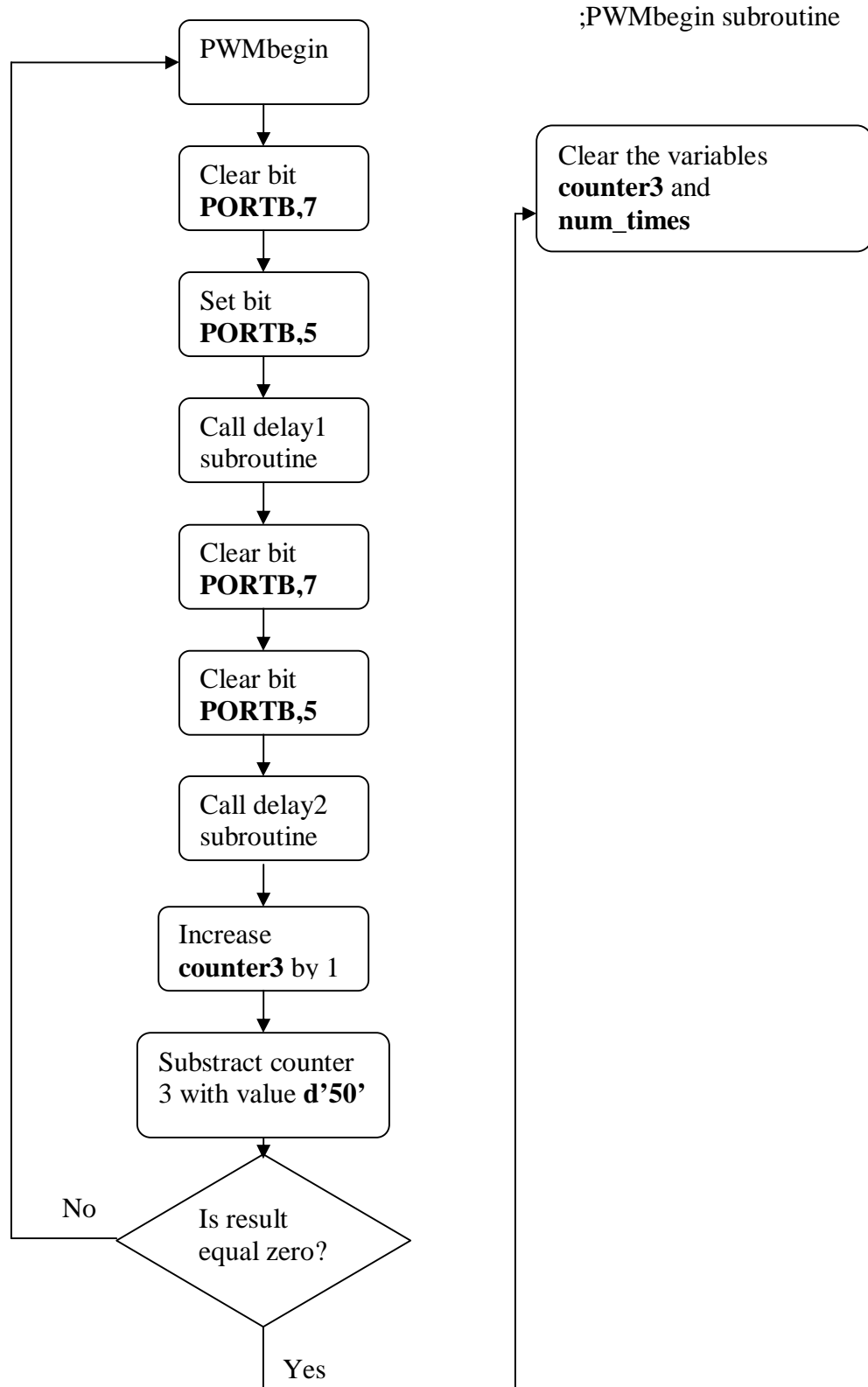
last      decfsz   bitcnt           ;Test whether 32 rotations has been
           Goto    loop           ;executed

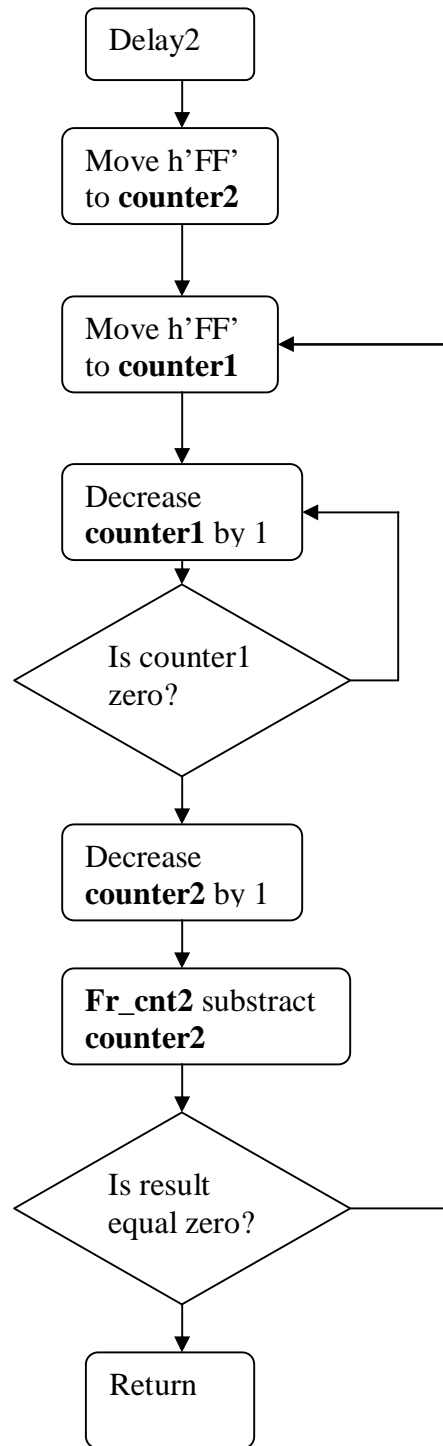
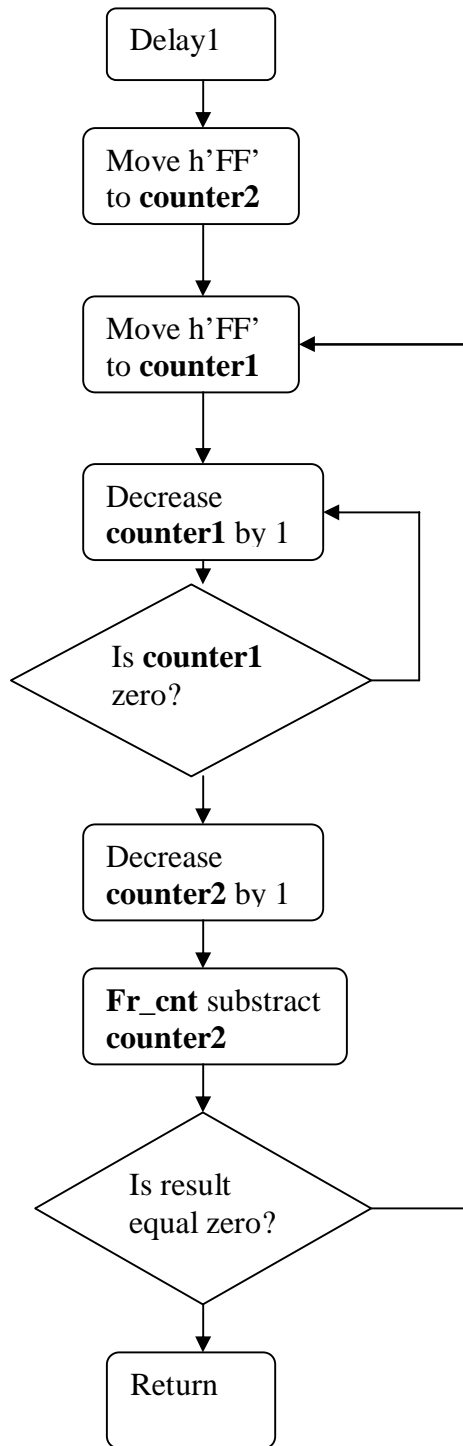
```

```
Movf      prod_res0,w
Movwf    Quo_0      ;Result is stored in Quo variables
Movf      prod_res1,w
Movwf    Quo_1      ;The result is 32 bits values
Movf      prod_res2,w
Movwf    Quo_2
Movf      prod_res3,w
Movwf    Quo_3

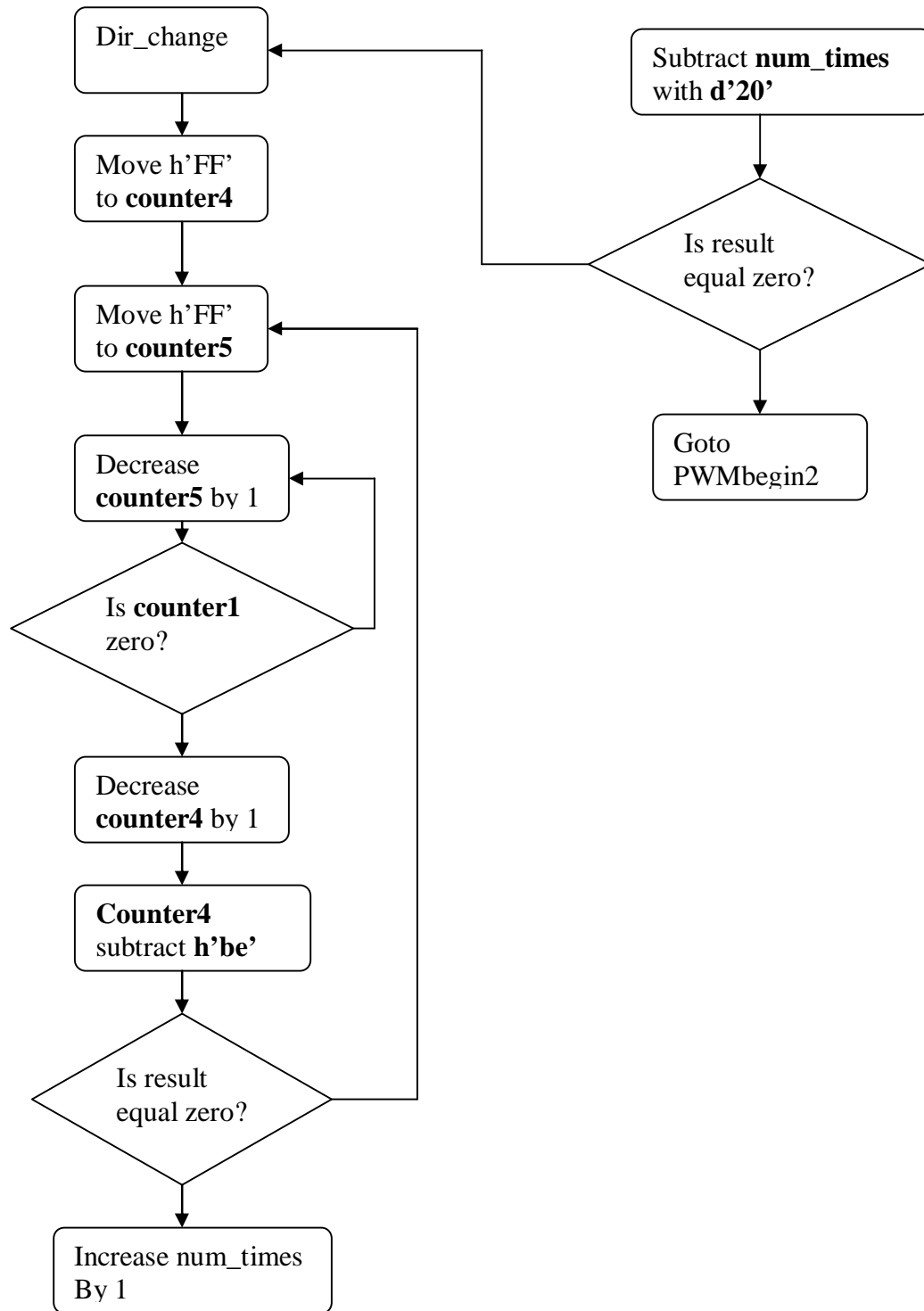
End
```

Programming flow-chart (On h-bridge (bi-direction motor turn))

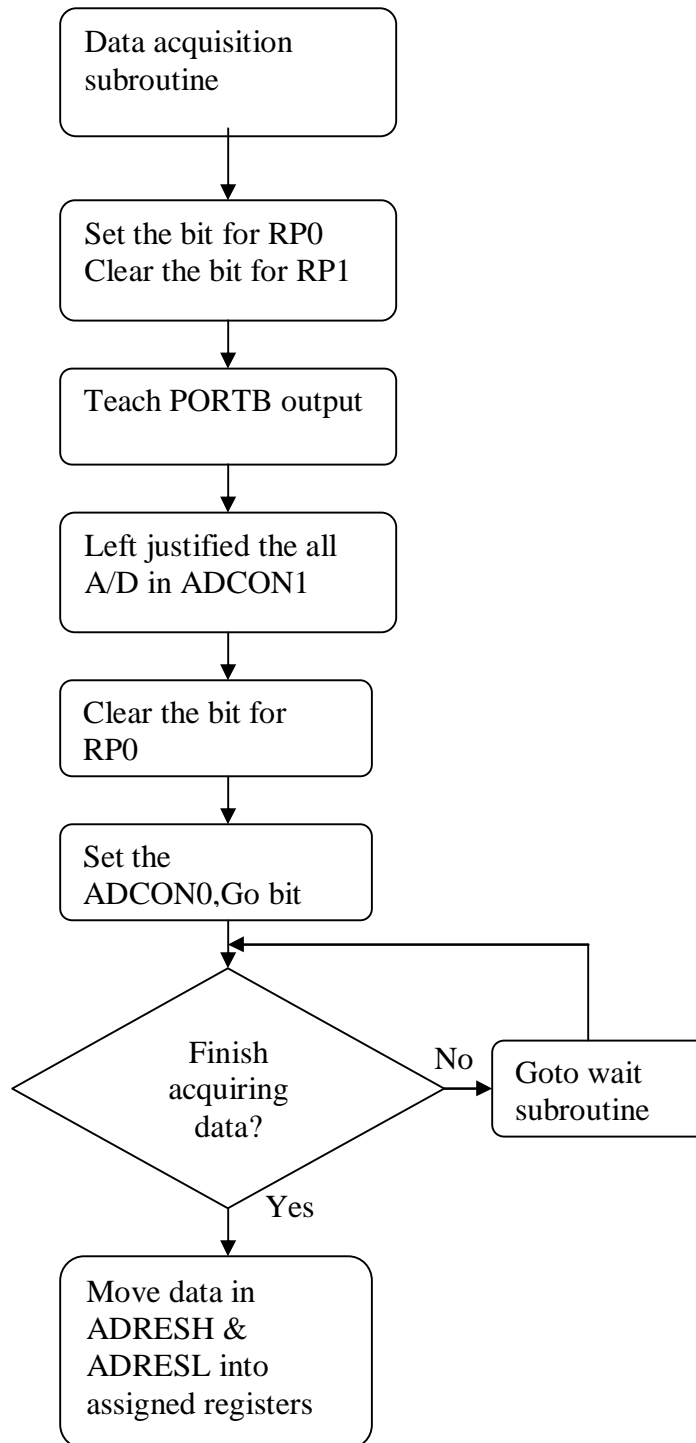




Direction change subroutine



Data Acquisition flow chart



Gyro-Accelerometer

;If one of the portc,2 exhibit a 1 signal then use the accelerometer data

;If portc,2 exhibit a 0 signal then use the gyro data

;Both the accelerometer and gyro run simulaneously

;Refer to the CD for more details

```
Data_chg      btfsc      PORTC,2
               call      acce_data
               call      gyro_data

acce_data     movf      u_term_lo_acce,w
               movwf     u_term_lo
               movf      u_term_hi_acce,w
               movwf     u_term_hi
               return

gyro_data     movf      u_term_lo_gyro,w
               movwf     u_term_lo
               movf      u_term_hi_gyro,w
               movwf     u_term_hi
               return

               end
```