One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

A Compact Algorithm Using the ADXL213 Duty Cycle Output

by Harvey Weinberg and Nitzan Gadish

INTRODUCTION

In many applications, high accuracy measurement of acceleration is less important than having a simple and compact software algorithm. This application note outlines a decode algorithm that measures only the pulse width (t1) output of the ADXL213 and translates it to degrees of tilt. In this algorithm, the period (t2) is not measured, and no binary division is used.

In PIC assembly code, a total of 199 bytes of program memory and 18 bytes of data memory are used. Even more efficient memory usage (particularly of data memory) can be obtained with further optimization. A flowchart of the algorithm is included so that the user can modify it or port it to any 4-bit or 8-bit microcontroller with little effort (see Figure 2).

A discussion of error sources inherent in this method of measurement is also included.

PRINCIPLE OF OPERATION

The ADXL213 outputs a pulse-width modulated (PWM) signal proportional to acceleration. Assuming that the scale factor is fixed at 30% per *g*, the following equation can be used:

$$acceleration = \frac{\left(\frac{t1}{t2}\right) - 0\,g \text{ duty cycle}}{30\%}$$

where:
t1 is the pulse width.
t2 is the period of the PWM output of the ADXL213.

In a temperature-stable environment, it can be assumed that the average value of t2 does not change. Therefore, the formula for acceleration can be rearranged as follows:

$$acceleration = \frac{\left(\frac{t1 - (t1 \text{ at } 0\,g)}{t2}\right)}{30\%}$$

Over a range of ±35° of tilt, each degree of tilt is very close to 16 m*g*. By choosing particular values of t2, we can take advantage of very easy modulo-2 division to minimize computational requirements when calculating tilt angle. For example,

$$t2 = 208\ \mu s$$
$$1g \Leftrightarrow (208\ \mu s) \times (30\%) = 62.4\ \mu s$$
$$1\ \mu s \Leftrightarrow (1g/62.4) = 16\ mg$$

Using this technique, tilt angle calculation is reduced to a simple 1 μs per degree relationship. Any modulo-2 factor of 208 μs (416 μs, 832 μs, and so on) can be used as required.

ERROR SOURCES

Scale error is the most significant error source encountered when using this algorithm. It is assumed that the overall scale factor is 16 m*g* per microsecond (or some modulo-2 multiple) in this algorithm, but the actual scale factor can be anything from 27% per *g* to 33% per *g*. This results in a ±4° error over ±40° of tilt. Another obvious error source is using the wrong value for t2. A 1% error in t2 results in a 1% error in tilt angle resolution. These errors can be eliminated by adding a trim to t2.

Scale factor error and t2 error can be trimmed out together by adjusting t2 such that the 16 m*g* per microsecond (or some modulo-2 multiple) relationship is maintained. This is expressed by the following equation:

$$t2 = \frac{1}{scale\_factor \times 0.016}$$

For example, for a scale factor of 10%,

$$t2 = \frac{1}{0.27 \times 0.016} = 231\ \mu s$$

Adjusting t2 to 231 μs in this case eliminates the errors due to scale factor and t2 accuracy.

Because scale factor variation can result in such large errors, trimming t2 by adding a potentiometer in series with R_{SET}, as shown in Figure 1, is recommended. In applications where only changes in tilt angle are of interest and errors due to scale factor and t2 inaccuracy can be tolerated, this trim can be omitted.

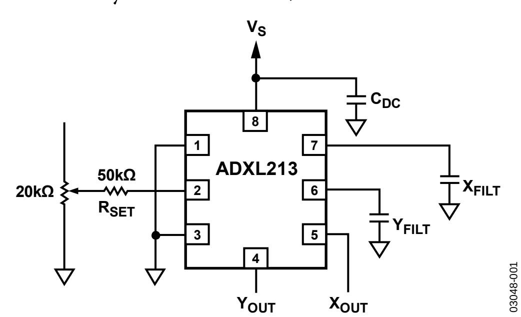

Figure 1. Circuit for Trimming t2

t2 may drift over temperature by as much as a few percentage points. This drift is very difficult to compensate for using this type of algorithm. It is recommended that another algorithm be used in situations where temperature drift is a problem.

The assumption that over ±35° of tilt, each degree of tilt is very close to 16 m*g* is, of course, an approximation. At a 1° tilt angle, one degree of tilt is 17.45 mg; at a 35° tilt angle, one degree of tilt is 14.38 m*g*. Although these values may appear to introduce a large source of error, in fact, they represent only ±1° of error over a ±40° range of tilt, as shown in Table 1.

Table 1. Tilt Angle vs. Error

Tilt Angle (°)	*g* Generated	t1 (µs)	Error (°)
0	0.000	0	0
2	0.035	2	0
4	0.070	4	0
6	0.105	7	1
8	0.139	9	1
10	0.174	11	1
12	0.208	13	1
14	0.242	15	1
16	0.276	17	1
18	0.309	19	1
20	0.342	21	1
22	0.375	23	1
24	0.407	25	1
26	0.438	27	1
28	0.469	29	1
30	0.500	31	1
32	0.530	33	1
34	0.559	35	1
36	0.588	37	1
38	0.616	38	0
40	0.643	40	0

There is normally a certain amount of jitter in t2. Because the duty cycle does not change as a result of this jitter, t1 changes proportionally with t2. This error source is minimized in the 0 *g* calibration routine by taking the average value of t1 over 16 readings. This averaging is not done in normal sampling to allow wider bandwidth operation. If wide bandwidth is not a concern, the user may wish to modify the algorithm to include a similar averaging scheme in normal sampling to minimize the error due to t2 jitter.

The final source of error is from aliasing in the duty cycle modulator itself. The analog bandwidth should be limited to 1/10 the duty cycle modulator frequency. For a t2 period of 1000 µs, the analog bandwidth should be 100 Hz or less.

FLOWCHART

Figure 2 is a flowchart of the algorithm.

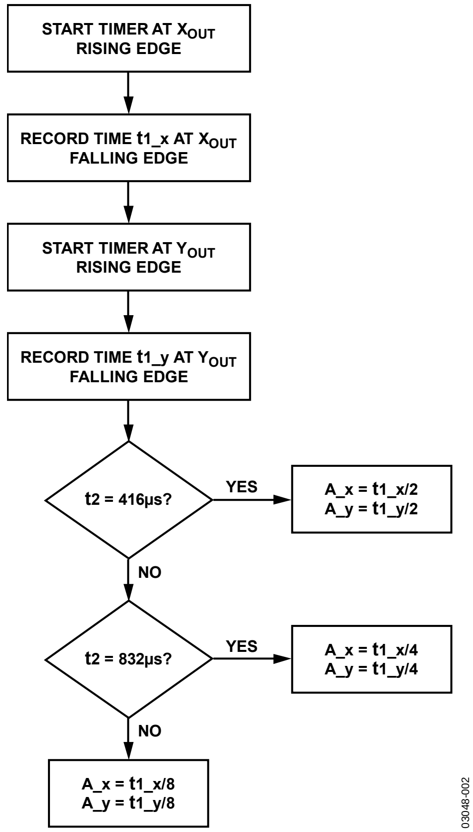

Figure 2. Flowchart

PROGRAM LISTING

```
;***********************************************************************
;
;********** 213-T1.ASM ***************************************
;********** REVISION: 1 **************************************
;********** MODIFIED FROM 202-T1.ASM *************************
;
;      RELEASED: SEPT. 16, 1998
;      REVISED: JAN. 16, 2009
;
; THIS SOFTWARE USES t1 MEASUREMENTS ONLY TO DETERMINE ACCELERATION
; EXPERIENCED BY THE ADXL213. THE OUTPUT IS A 1-BYTE HEXADECIMAL
; NUMBER PER AXIS OF RANGE 00 TO FF. THE MOST SIGNIFICANT BIT IS A SIGN
; BIT. A 1 IN THE MSB INDICATES POSITIVE ACCELERATION. A 0 IN THE MSB
; INDICATES NEGATIVE ACCELERATION. TO MAKE THE SOFTWARE AS COMPACT AS
; POSSIBLE, t2 IS ASSUMED TO HAVE A FIXED VALUE. VARIATION FROM THIS
; VALUE WILL RESULT IN ERROR. IT IS ALSO ASSUMED THAT THE FACTOR OF g/t1
; IS FIXED AS SHOWN IN THE TABLE BELOW. SO FOR TILT MEASUREMENT OVER
; ±40 DEGREES, THIS ROUTINE IS ACCURATE TO APPROXIMATELY ONE DEGREE.
; SINCE THE OUTPUT IS A 1-BYTE NUMBER, RESPONSE IS LIMITED TO ±1 g.
;
;      t2 (IN µSEC)     g/t1 (HOW MANY g FOR 1 µSEC)         µSEC/DEGREE
;      416                 0.008                                  2
;      832                 0.004                                  4
;      1664                0.002                                  8
;      3328                0.001                                  16
;
;======================================================================
LIST P=16C62A              ;SPECIFY PROCESSOR
;======================================================================
;======================================================================
;
; REGISTER DEFINITIONS
;
```

```
;======================================================================
W EQU H'0000'
F EQU H'0001'
;----- REGISTER FILES--------------------------------------------------
INDF EQU H'0000'
TMR0 EQU H'0001'
PCL EQU H'0002'
STATUS EQU H'0003'
FSR EQU H'0004'
PORTA EQU H'0005'
PORTB EQU H'0006'
PORTC EQU H'0007'
PCLATH EQU H'000A'
INTCON EQU H'000B'
PIR1 EQU H'000C'
TMR1L EQU H'000E'
TMR1H EQU H'000F'
T1CON EQU H'0010'
TMR2 EQU H'0011'
T2CON EQU H'0012'
SSPBUF EQU H'0013'
SSPCON EQU H'0014'
CCPR1L EQU H'0015'
CCPR1H EQU H'0016'
CCP1CON EQU H'0017'
OPTION_REG EQU H'0081'
TRISA EQU H'0085'
TRISB EQU H'0086'
TRISC EQU H'0087'
PIE1 EQU H'008C'
PCON EQU H'008E'
PR2 EQU H'0092'
SSPADD EQU H'0093'
SSPSTAT EQU H'0094'
```

```
;----- STATUS BITS ----------------------------------------------------
IRP EQU H'0007'
RP1 EQU H'0006'
RP0 EQU H'0005'
NOT_TO EQU H'0004'
NOT_PD EQU H'0003'
Z EQU H'0002'
DC EQU H'0001'
C EQU H'0000'
;----- INTCON BITS ----------------------------------------------------
GIE EQU H'0007'
PEIE EQU H'0006'
T0IE EQU H'0005'
INTE EQU H'0004'
RBIE EQU H'0003'
T0IF EQU H'0002'
INTF EQU H'0001'
RBIF EQU H'0000'
;----- PIR1 BITS ------------------------------------------------------
SSPIF EQU H'0003'
CCP1IF EQU H'0002'
TMR2IF EQU H'0001'
TMR1IF EQU H'0000'
;----- T1CON BITS -----------------------------------------------------
T1CKPS1 EQU H'0005'
T1CKPS0 EQU H'0004'
T1OSCEN EQU H'0003'
NOT_T1SYNC EQU H'0002'
T1INSYNC EQU H'0002'       ;BACKWARD COMPATIBILITY
TMR1CS EQU H'0001'
TMR1ON EQU H'0000'
;----- T2CON BITS -----------------------------------------------------
TOUTPS3 EQU H'0006'
TOUTPS2 EQU H'0005'
TOUTPS1 EQU H'0004'
```

```
TOUTPS0 EQU H'0003'
TMR2ON EQU H'0002'
T2CKPS1 EQU H'0001'
T2CKPS0 EQU H'0000'
;----- SSPCON BITS ---------------------------------------------------
WCOL EQU H'0007'
SSPOV EQU H'0006'
SSPEN EQU H'0005'
CKP EQU H'0004'
SSPM3 EQU H'0003'
SSPM2 EQU H'0002'
SSPM1 EQU H'0001'
SSPM0 EQU H'0000'
;----- CCP1CON BITS --------------------------------------------------
CCP1X EQU H'0005'
CCP1Y EQU H'0004'
CCP1M3 EQU H'0003'
CCP1M2 EQU H'0002'
CCP1M1 EQU H'0001'
CCP1M0 EQU H'0000'
;----- OPTION BITS ---------------------------------------------------
NOT_RBPU EQU H'0007'
INTEDG EQU H'0006'
T0CS EQU H'0005'
T0SE EQU H'0004'
PSA EQU H'0003'
PS2 EQU H'0002'
PS1 EQU H'0001'
PS0 EQU H'0000'
;----- PIE1 BITS -----------------------------------------------------
SSPIE EQU H'0003'
CCP1IE EQU H'0002'
TMR2IE EQU H'0001'
TMR1IE EQU H'0000'
```

```
;----- PCON BITS ------------------------------------------------------
NOT_POR EQU H'0001'
;----- SSPSTAT BITS ---------------------------------------------------
D EQU H'0005'
I2C_DATA EQU H'0005'
NOT_A EQU H'0005'
NOT_ADDRESS EQU H'0005'
D_A EQU H'0005'
DATA_ADDRESS EQU H'0005'
P EQU H'0004'
I2C_STOP EQU H'0004'
S EQU H'0003'
I2C_START EQU H'0003'
R EQU H'0002'
I2C_READ EQU H'0002'
NOT_W EQU H'0002'
NOT_WRITE EQU H'0002'
R_W EQU H'0002'
READ_WRITE EQU H'0002'
UA EQU H'0001'
BF EQU H'0000'
;======================================================================
;
; RAM DEFINITION
;
;======================================================================
__MAXRAM H'BF'
__BADRAM H'08'-H'09', H'0D', H'18'-H'1F'
__BADRAM H'88'-H'89', H'8D', H'8F'-H'91',H'95'-H'9F'
;======================================================================
;
; RAM EQUATES
;
;======================================================================
T1X_1 EQU 20
```

```
T1X_0 EQU 21
ARGL EQU 22
ARGH EQU 23
ACCHI EQU 24
ACCLO EQU 25
T1Y_1 EQU 26
T1Y_0 EQU 27
T1XCAL_2 EQU 28
T1XCAL_1 EQU 29
T1XCAL_0 EQU 2A
T1YCAL_2 EQU 2B
T1YCAL_1 EQU 2C
T1YCAL_0 EQU 2D
X_ACCEL EQU 2E
Y_ACCEL EQU 2F
T1CAL_COUNT EQU 30
ROTCNT EQU 31
;======================================================================
;
; CONFIGURATION BITS
;
;======================================================================
_CP_ALL EQU H'3F8F'
_CP_75 EQU H'3F9F'
_CP_50 EQU H'3FAF'
_CP_OFF EQU H'3FBF'
_PWRTE_ON EQU H'3FBF'
_PWRTE_OFF EQU H'3FB7'
_WDT_ON EQU H'3FBF'
_WDT_OFF EQU H'3FBB'
_LP_OSC EQU H'3FBC'
_XT_OSC EQU H'3FBD'
_HS_OSC EQU H'3FBE'
_RC_OSC EQU H'3FBF'
;======================================================================
```

```
;***** PROGRAM ******
;***** MAIN PROGRAM *****
;***** RESET ROUTINE *****
ORG 0000
GOTO PROG_START          ;GO TO START OF PROGRAM
GOTO PROG_START
GOTO PROG_START          ;THESE COMMANDS ARE HERE TO
GOTO PROG_START          ;KICK THE PROGRAM COUNTER PAST
RETURN                   ;THE INTERRUPT VECTORS IN CASE
RETURN                   ;OF A GLITCH
PROG_START
CLRF PORTA
CLRF PORTB
CLRF PORTC
BSF STATUS,5             ;RAM PAGE 1
MOVLW B'11111111'        ;SET UP THE I/O PORTS
MOVWF TRISA              ;PORT A, ALL INPUTS
MOVLW B'11111111'
MOVWF TRISB              ;PORT B, ALL INPUTS
MOVLW B'11111111'
MOVWF TRISC              ;PORT C, ALL INPUTS
BCF STATUS,5             ;SET RAM PAGE 0
MAIN_LOOP
CALL CHECK_CAL           ;CHECK IF CALIBRATION ROUTINE
                         ;SHOULD BE PERFORMED
CALL READ_T1             ;READ ACCELERATION
MOVF T1X_1,0             ;CHECK ACCELERATION POLARITY
SUBWF T1XCAL_1,0
BTFSS STATUS,C
GOTO ACCX_GT_ZX
BTFSS STATUS,Z
GOTO ACCX_LT_ZX
MOVF T1X_0,0
SUBWF T1XCAL_0,0
BTFSS STATUS,C
```

```
GOTO ACCX_GT_ZX
ACCX_LT_ZX                ;X ACCELERATION IS NEGATIVE
MOVF T1XCAL_0,0
MOVWF ACCLO
MOVF T1XCAL_1,0
MOVWF ACCHI
MOVF T1X_0,0
MOVWF ARGL
MOVF T1X_1,0
MOVWF ARGH
CALL SUB_16X16
BCF STATUS,C              ;DIVIDE BY 2 (1 SHIFT) IF t2=416µS
RRF ACCHI,1               ;DIVIDE BY 4 (2 SHIFTS) IF t2=832µS
RRF ACCLO,0               ;DIVIDE BY 8 (3 SHIFTS) IF t2=1664µS
MOVWF X_ACCEL
BCF X_ACCEL,7             ;CLEAR THE SIGN BIT AS ACCEL IS -
GOTO DO_Y_AXIS
ACCX_GT_ZX                ;X ACCELERATION IS POSITIVE
MOVF T1X_0,0
MOVWF ACCLO
MOVF T1X_1,0
MOVWF ACCHI
MOVF T1XCAL_0,0
MOVWF ARGL
MOVF T1XCAL_1,0
MOVWF ARGH
CALL SUB_16X16
BCF STATUS,C              ;DIVIDE BY 2 (1 SHIFT) IF t2=416µS
RRF ACCHI,1               ;DIVIDE BY 4 (2 SHIFTS) IF t2=832µS
RRF ACCLO,0               ;DIVIDE BY 8 (3 SHIFTS) IF t2=1664µS
MOVWF X_ACCEL
BSF X_ACCEL,7             ;SET THE SIGN BIT AS ACCEL IS +
DO_Y_AXIS
MOVF T1Y_1,0              ;CHECK FOR ACCELERATION POLARITY
SUBWF T1YCAL_1,0
```

```
BTFS S STATUS,C
GOTO ACCY_GT_ZY
BTFS S STATUS,Z
GOTO ACCY_LT_ZY
MOVF T1Y_0
SUBWF T1YCAL_0,0
BTFSS STATUS,C
GOTO ACCY_GT_ZY
ACCY_LT_ZY                ;Y ACCELERATION IS NEGATIVE
MOVF T1YCAL_0,0
MOVWF ACCLO
MOVF T1YCAL_1,0
MOVWF ACCHI
MOVF T1Y_0,0
MOVWF ARGL
MOVF T1Y_1,0
MOVWF ARGH
CALL SUB_16X16
BCF STATUS,C              ;DIVIDE BY 2 (1 SHIFT) IF t2=416µS
RRF ACCHI,1               ;DIVIDE BY 4 (2 SHIFTS) IF t2=832µS
RRF ACCLO,0               ;DIVIDE BY 8 (3 SHIFTS) IF t2=1664µS
MOVWF Y_ACCEL
BCF Y_ACCEL,7             ;CLEAR THE SIGN BIT AS ACCEL IS -
GOTO MAIN_LOOP
ACCY_GT_ZY                ;Y ACCELERATION IS POSITIVE
MOVF T1Y_0,0
MOVWF ACCLO
MOVF T1Y_1,0
MOVWF ACCHI
MOVF T1YCAL_0,0
MOVWF ARGL
MOVF T1YCAL_1,0
MOVWF ARGH
CALL SUB_16X16
```

```
BCF STATUS,C                ;DIVIDE BY 2 (1 SHIFT) IF t2=416µS
RRF ACCHI,1                 ;DIVIDE BY 4 (2 SHIFTS) IF t2=832µS
RRF ACCLO,0                 ;DIVIDE BY 8 (3 SHIFTS) IF t2=1664µS
MOVWF Y_ACCEL
BSF Y_ACCEL,7               ;SET THE SIGN BIT AS ACCEL IS +
GOTO MAIN_LOOP
;***** SUBROUTINES *****
;**********************************************************************
CHECK_CAL                   ;THIS SUBROUTINE READS THE "CAL" PIN (RA4). IF IT
                            ;IS HI, A SIMPLE CALIBRATION ROUTINE IS PERFORMED
                            ;TO MEASURE THE 0 g VALUE OF t1. SIXTEEN SAMPLES OF
                            ;t1 ARE AVERAGED (BY ADDING TOGETHER AND THEN
                            ;DIVIDING BY 16) TO INCREASE ACCURACY.
BTFSS PORTA,3               ;IS RA4 HI
RETURN                      ;IF NOT THEN NO CAL ROUTINE
CLRF T1XCAL_2               ;IF YES THEN ACQUIRE CAL DATA
CLRF T1XCAL_1               ;START BY CLEARING ALL
CLRF T1XCAL_0               ;OF THE CALIBRATION REGISTERS
CLRF T1YCAL_2
CLRF T1YCAL_1
CLRF T1YCAL_0
MOVLW 10                    ;SET AVERAGING COUNTER TO 16
MOVWF T1CAL_COUNT
ZCAL_A
MOVF T1CAL_COUNT,1          ;TEST IF 16 PASSES HAVE OCCURRED BY
BTFSC STATUS,Z              ;TESTING IF THE LOOP COUNTER = 0
GOTO ZCAL_B
CALL READ_T1                ;READ t1
MOVF T1X_0,0                ;DO AVERAGING CALCULATIONS OF t1X
ADDWF T1XCAL_0,1
BTFSS STATUS,C              ;CHECK IF A CARRY WAS GENERATED
GOTO ZCAL_C
MOVLW 01                    ;IF A CARRY WAS GENERATED INCREMENT
ADDWF T1XCAL_1
```

```
BTFSC STATUS,C              ;CHECK IF A CARRY WAS GENERATED
INCF T1XCAL_2,1
ZCAL_C
MOVF T1X_1,0
ADDWF T1XCAL_1
BTFSC STATUS,C              ;CHECK IF A CARRY WAS GENERATED
INCF T1XCAL_2
MOVF T1Y_0,0
ADDWF T1YCAL_0,1            ;DO AVERAGING CALCULATIONS OF t1Y
BTFSS STATUS,C
GOTO ZCAL_D
MOVLW 01
ADDWF T1YCAL_1
BTFSC STATUS,C
INCF T1YCAL_2,1
ZCAL_D
MOVF T1Y_1,0
ADDWF T1YCAL_1
BTFSC STATUS,C
INCF T1YCAL_2
DECF T1CAL_COUNT            ;DECREMENT LOOP COUNTER
GOTO ZCAL_A                 ;LOOP
ZCAL_B
MOVLW 04                    ;DIVIDE T1CAL BY 16
MOVWF ROTCNT
ZCAL_E
RRF T1XCAL_2,1
RRF T1XCAL_1,1
RRF T1XCAL_0,1
RRF T1YCAL_2,1
RRF T1YCAL_1,1
RRF T1YCAL_0,1
MOVLW 01
SUBWF ROTCNT,1
BTFSS STATUS,Z
```

```
GOTO ZCAL_E
RETURN
;*********************************************************************
READ_T1                    ;THIS SUBROUTINE ACQUIRES t1X AND t1Y
                           ;t1X IS IN REGISTERS T1X_1,T1X_0
                           ;t1Y IS IN REGISTERS T1Y_1,T1Y_0
CLRF T1CON                 ;SET TIMER 1 TO ZERO
CLRF TMR1L
CLRF TMR1H
EDGE1
BTFSC PORTB,2              ;WAIT FOR RISING EDGE
GOTO EDGE1
EDGE2
BTFSS PORTB,2
GOTO EDGE2
BSF T1CON,TMR1ON           ;TURN TIMER 1 ON
NOP                        ;WAIT 3 µSEC TO DEGLITCH
NOP
NOP
EDGE3
BTFSC PORTB,2              ;LOOK FOR FALLING EDGE
GOTO EDGE3
BCF T1CON,TMR1ON           ;STOP TIMER 1 TO READ RELIABLY
MOVFT MR1H,0
MOVWF T1X_1
MOVF TMR1L,0
MOVWF T1X_0
CLRF TMR1L                 ;CLEAR THE TIMER RESULT REGISTERS
CLRF TMR1H                 ;IN PREPARATION FOR t1Y CAPTURE
EDGE4
BTFSC PORTB,1              ;LOOK FOR THE RISING EDGE ON
GOTO EDGE4                 ;Y CHANNEL
EDGE5
BTFSS PORTB,1
GOTO EDGE5
```

```
BSF T1CON,TMR1ON            ;TURN TIMER 1 BACK ON AT RISING EDGE
NOP                         ;WAIT 3 µSEC TO DEGLITCH
NOP
NOP
EDGE6
BTFSC PORTB,1               ;LOOK FOR FALLING EDGE SIGNIFYING
GOTO EDGE6                  ;THE END OF t1Y
BCF T1CON,TMR1ON            ;STOP TIMER 1 TO READ END OF t1Y
MOVF TMR1H,0
MOVWF T1Y_1
MOVF TMR1L,0
MOVWF T1Y_0
RETURN
;**********************************************************************
SUB_16X16                   ;THIS SUBROUTINE PERFORMS A 16 BIT BY 16 BIT
                            ;SUBTRACTION.
                            ;(ACCHI,ACCLO)=(ACCHI,ACCLO)-(ARGH,ARGL)
COMF ARGL
INCF ARGL
BTFSC STATUS,2
DECF ARGH
COMF ARGH                   ;NEGATE ZERO
MOVF ARGL,W                 ;THEN ADD
ADDWF ACCLO,F
BTFSC STATUS,W
INCF ACCHI
MOVF ARGH,W
ADDWF ACCHI,F
RETURN
;*********************************************************************
END
```