
Measure Tilt Using PIC16F84A & ADXL202

*Author: Rodger Richey
Microchip Technology Inc.*

INTRODUCTION

Recent advances in accelerometer sensor technology, especially with silicon micromachined types, have driven the cost of these devices down significantly. As of today, you could obtain an accelerometer for less than \$5 per axis. Measurement of acceleration or one of the derivative properties such as vibration, shock, or tilt has become very commonplace in a wide range of products. At first you might think of seismic activity or machinery performance monitoring, but would automotive airbags, sports training products, or computer peripherals ever cross your mind? The technology behind acceleration sensors has advanced to provide a very cost effective and user friendly solution for almost any application.

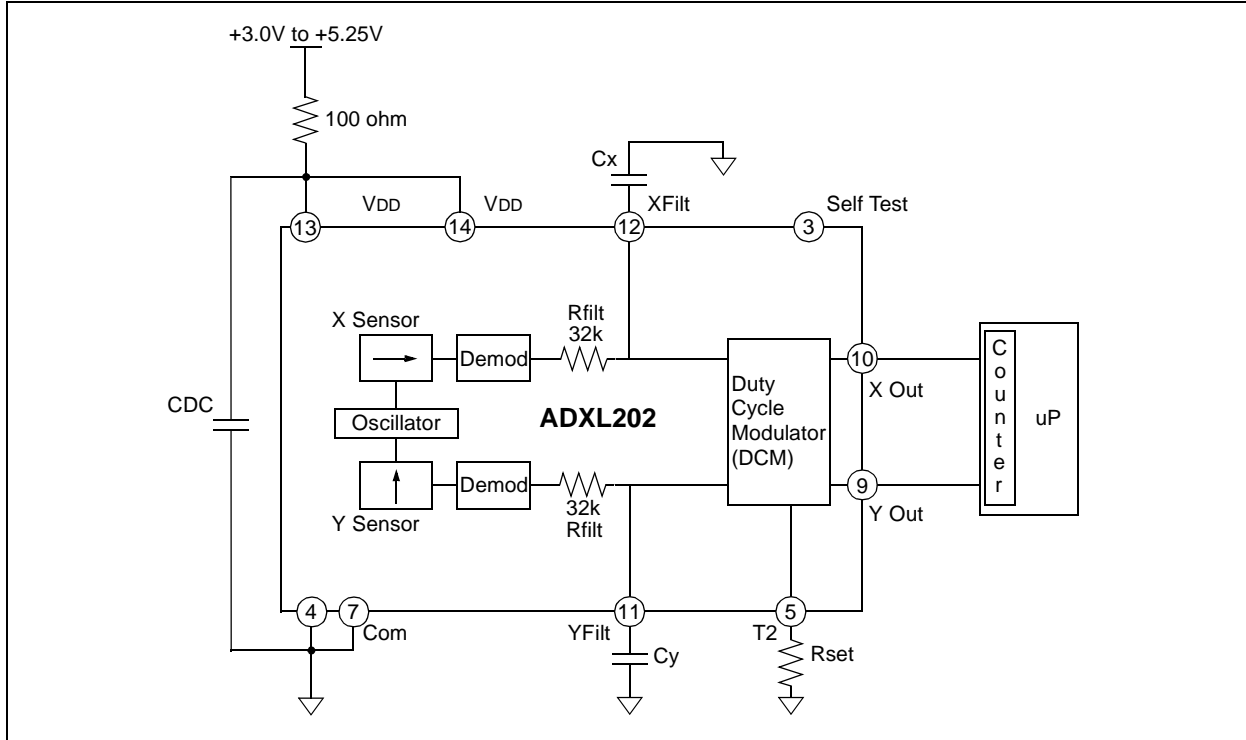
There are many types of sensors that measure acceleration, vibration, shock, or tilt. These sensors include piezo-film, electromechanical servo, piezoelectric, liquid tilt, bulk micromachined piezo resistive and capacitive sensors, as well as surface micromachined capacitive. Each of these sensors has distinct characteristics in the output signal of the sensor, cost to develop, and type of operating environment. Measurement of acceleration can also provide velocity by single integration and position by double integration. Vibration and shock can be used for machine health determination as well as motion and shock detection for car alarms. Static acceleration due to gravity can be used to determine tilt and inclination provided that the sensor is responsive to static acceleration.

This application note will focus on the surface micro-machined capacitive ADXL accelerometers from Analog Devices, in particular the ADXL202. The example application will use the ADXL202 accelerometer with the PIC16F84A in a tilt meter. The PIC16F84A is a good match with the ADXL202 because all acceleration measurements are digital only. Secondly, the Data EEPROM can be used to store the calibration constants and restore on reset. The external interface can also be changed easily to accommodate a LCD display (as shown in this application note) or a serial interface to the outside world.

MEMs SENSOR: THEORY OF OPERATION

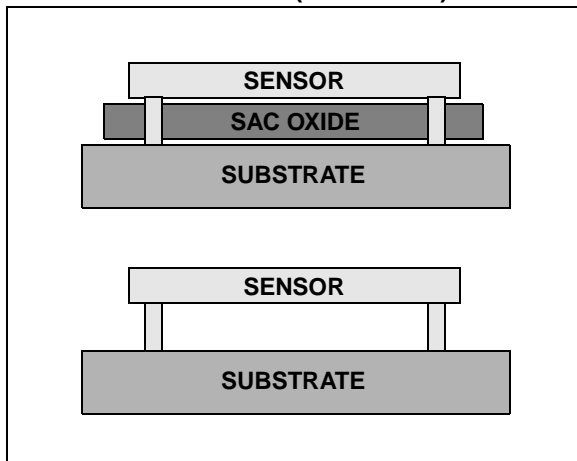
In recent years the silicon micromachined sensor has made tremendous advancements in terms of cost and level of on-chip integration for acceleration and/or vibration measurements. By implementing additional BiMOS circuitry on-chip, these products not only provide sensor but also signal conditioning in a single package that requires a few external components to complete the circuit. Some manufacturers have taken this approach one step further by converting the analog output of the sensor to a digital format such as duty cycle. This method not only lifts the burden of designing fairly complex analog circuitry for the sensor but also reduces cost and board area. Because of these advances, the micromachined accelerometer is finding its way into such products as joysticks and airbags that were previously impossible due to price or size limitations of the sensor. Figure 1 shows the block diagram of the ADXL202.

FIGURE 1: ADXL202 BLOCK DIAGRAM



A surface micromachined device is composed of springs, masses and motion sensing components. These sensors use standard integrated circuit processing techniques in standard wafer fabs, i.e., no additional cost to the user for special processes or fabs. As shown in Figure 2, normal IC processes take place by applying layers of oxide and polysilicon. Then using IC photolithography and selective etching the sensor is created as a 3-dimensional structure suspended above the substrate free to move in all directions. The surrounding area becomes the signal conditioning and output circuitry.

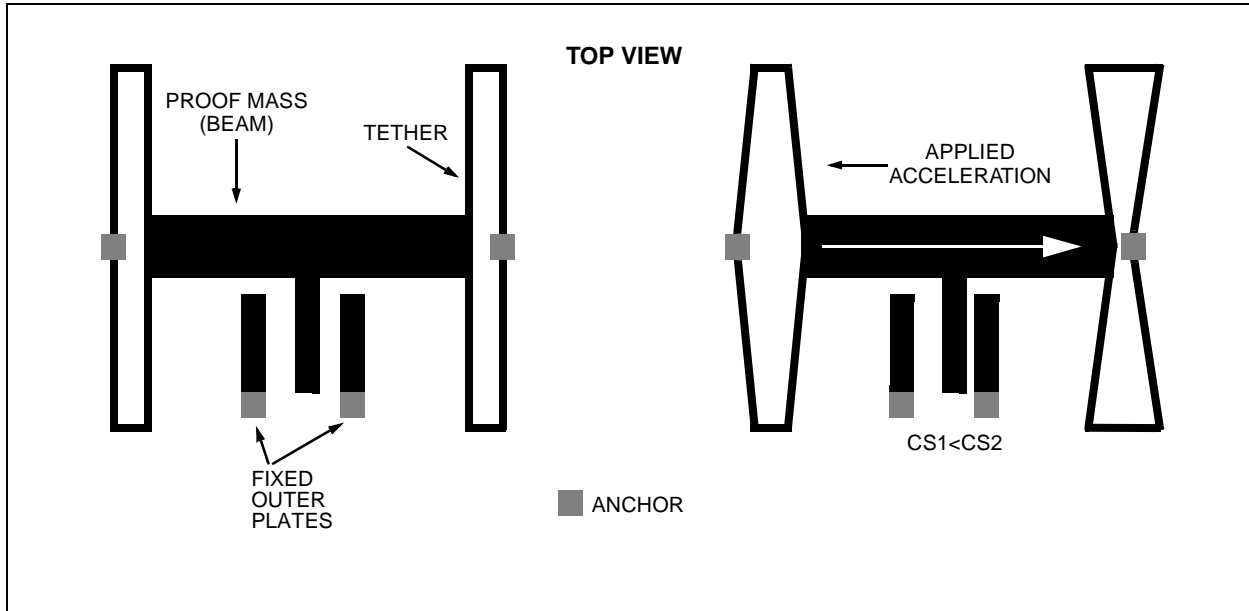
FIGURE 2: SILICON STRUCTURE OF ADXL202 (SIDE VIEW)



The core of the sensor is a surface micromachined polysilicon structure or mass that is suspended on top of the silicon wafer for each axis. The polysilicon "springs" hold the mass and provides resistance to movement due to acceleration forces. Both the mass and the substrate have plates that form a differential capacitor where the fixed plates on the substrate are driven 180° out of phase. Figure 3 shows an exaggerated diagram of the sensor. Any movement of the mass unbalances the differential capacitor resulting in a square wave output with the amplitude proportional to the acceleration. Each axis has a demodulator that rectifies the signal and determines the direction of the acceleration. This output is fed to a duty cycle modulator (DCM) that incorporates external capacitors to set the bandwidth of each axis. The analog signal is filtered and converted to a duty cycle output by the DCM. An external resistor sets the period of the duty cycle output. A 0g acceleration produces a 50% duty cycle output. A low-cost, all digital, microcontroller can be used to measure acceleration by timing both the duty cycle and the period of each axis. Refer to Figure 1 for interaction and connections between the various circuits inside the device as described above.

Some of the advantages with micromachined sensors are that they are low cost and most have on-chip signal conditioning.

FIGURE 3: SENSOR MECHANICAL OPERATION



CONFIGURING THE ADXL202

The end application for our ADXL202 is a simple tilt meter that shows X-axis and Y-axis tilt (or pitch and roll for you aeronautical buffs). The design procedure is somewhat iterative since the bandwidth, period, and microcontroller counter resolution play important roles in the minimum resolution of the measurement. Analog Devices has simplified the design procedure by providing an Excel spreadsheet entitled "The XL202 Interactive Designer" that can be downloaded off their website at www.analog.com and is shown in Appendix A. The specifications for our system are +5VDC operation, +/-1.0 degree tilt resolution, 25 samples per channel per second, and the microcontroller should operate at 4MHz or less.

Through the use of an iterative process, the designer can determine the external component values and the noise and resolution of the acceleration measurement without having to prototype a single circuit.

In Step 1 of the spreadsheet shown in Appendix A, the designer will enter the supply voltage which should be between 3.0V and 5.25V. We will enter 5.0V. Analog bandwidth is entered in Step 2, which calculates the values for the external capacitors. The bandwidth directly determines the noise floor and resolution of the accelerometer and therefore may have to be adjusted to provide the desired results based on calculations later in the spreadsheet. Enter 10Hz and the resulting capacitance is 0.50 μ F. Since 0.50 μ F is not a standard, we can modify the bandwidth to get a standard value. Using 10.5Hz yields a capacitor of 0.47 μ F.

In Step 3, the spreadsheet calculates the RMS and peak-to-peak (P-P) noise of the acceleration measurements. The designer must estimate the amount of time that the actual signal will be above the P-P noise using a multiplier. At this step enter 4, which in turn reveals

that the peak-to-peak noise will be 0.46 degrees of tilt. Now the designer must evaluate the P-P noise estimation because this noise determines the smallest acceleration resolution that the accelerometer can have. If this noise estimation is not acceptable, then the bandwidth must be lowered to reduce the P-P noise. This example is well within the 1.0 degree specification and so we will continue.

The next few steps set the period of the duty cycle output and the measurement resolution due to the counter on the microcontroller. Both the sample rate per channel and the percentage of time the ADXL202 will be powered are entered in Step 4. The designer also enters the time required to calculate the acceleration for two channels and the spreadsheet then calculates the period of the duty cycle output and the corresponding external resistor. We will use 25 samples per second per channel and the part will be powered up 100% of the time. Analog Devices has already calculated the time to acquire two channels and perform the calculations as 20ms. Of this time, it takes 3ms for calculations based on a previous application note, leaving 17ms for signal acquisition. This relates to 17,000 instruction cycles on the PICmicro[®] running at 4MHz.

In Step 5, the counter rate of the microcontroller is used to calculate the measurement resolution due to the counter in g's and degrees of tilt. The spreadsheet also determines the size of the counter on the microcontroller to prevent an overflow. Per our specifications, the microcontroller is clocked at 4MHz resulting in a 1MHz timer frequency (Timer0). With this timer rate, the resolution of the digital section of the ADXL202 is 0.06 degrees of tilt. The counter required to acquire the digital output must be 15-bits. We can easily implement a 15-bit counter using the Timer0 as the low byte of the count and for each Timer0 overflow increment an upper byte counter. The designer must again determine if this

resolution is acceptable. To increase the resolution, either increase the counter rate (Step 5) or decrease the number of samples per second (Step 4).

Step 6 checks for aliasing errors due to the sample rate. Nyquist requirements specify that the sample rate needs to be faster than the bandwidth by a factor of 2. Analog Devices recommends that at least a factor of 10 is used to minimize dynamic errors from the PWM sampling technique. For our case the, the ratio is 11.2 which according to Nyquist and Analog Devices is more than sufficient. If the spreadsheet calculates that the ratio is low, the designer must increase the sample rate in Step 4 or decrease the bandwidth in Step 2.

The results are in! The spreadsheet calculates the minimum resolution of the acceleration measurement due to RMS P-P noise and resolution of the counter in Step 7. It also provides a minimum resolution of a tilt measurement. Our calculated minimum resolution is 0.5 degrees of tilt which is acceptable according to the specification. If this resolution was not acceptable, then the bandwidth (Step 2), acquisition rate (Step 4), or the counter rate (Step 5) would have to be adjusted to reduce noise.

The spreadsheet also offers the designer the ability to explore how oversampling the PWM signal affects noise at the expense of sacrificing bandwidth in Step 8. Finally, Step 9 provides the estimated drift of the 0 g point due to temperature effects.

TILT METER APPLICATION

In the tilt meter application, the value of tilt in the X-axis and Y-axis is displayed on a 2-line by 8-character dot matrix LCD display. The only other function is a push button switch to perform a simple calibration cycle. The PIC16F84A makes an ideal companion to the ADXL202 because calibration parameters for the sensor can be stored in on-chip Data EEPROM memory for retrieval and usage in later calculations. Using the ADXL202 in conjunction with a PICmicro[®] not only reduces the time to market for the product but also overall system cost and power consumption.

Figure 4 shows the schematic for the simple tilt meter. For convenience, a 9V battery is used with a LM78L05 +5V regulator to provide power to the circuit. The ADXL202 is configured as shown in the ADXL202 Interactive Designer spreadsheet with 0.47 μ F capacitors on the XFILT and YFILT pins. A resistor of 1.0625M Ω is called out by the spreadsheet to connect to the RSET pin. Since the duty cycle generator's current source that determines the PWM frequency is only accurate to approximately 10%, a 1.0 M Ω , 5% resistor can be used. The 6% error between the two resistors will get corrected by the measurement of T2. The duty cycle output pins from the ADXL202 XOUT and YOUT are connected to RA0 and RA1 respectively.

The microcontroller circuit is also very simple. The 4MHz crystal uses two 33pF capacitors to complete the oscillator circuit. The push button switch is connected to RB4. This pin has internal pull-up resistors reducing the need for any external circuitry. The LCD display is driven using the 4-bit MPU mode which only requires 3 I/O pins for control and 4 I/O pins for data. Refer to the specifications for the Hitachi HD44780 LCD controller or the application note, AN587, "Interfacing PICmicro[®] to an LCD Module", for more interface information. The controls lines RS, R/W, and E connect to RB5, RA3, and RA2. The data lines are connected to RB<0:3>. There is also a 10KΩ potentiometer connected to pin 3 of the LCD display to control the contrast.

Acceleration is a vector quantity with both a direction and magnitude. The acceleration vector can be broken up into two vectors on the ADXL202, the X-axis and Y-axis. The ADXL202 is responsive to both static acceleration due to gravity as well as acceleration due to motion. The main problem with using this type of accelerometer to measure degrees tilt is not that it is sensitive to motion but that it can't distinguish between gravity and motion. The user must implement some type of time weighted filter to remove the effects of motion from the measurement (not implemented in this design). When the tilt angle is varied along the sensitive X- and Y-axis, the acceleration vector changes and the ADXL202 responds by changing the duty cycle outputs. The angle of tilt is defined by the following equation:

$$\theta = \arcsin[(V(out)-V(zero\ g)) / (1g \times Scale\ factor(V/g))]$$

This is a difficult calculation on an 8-bit microcontroller, therefore the calculation will be simplified. In spite of this, we still yield very good results (shown later in the firmware section).

The firmware is centered around the duty cycle measurement. The technical note from Analog Devices titled "Using the ADXL202 Duty Cycle Output" shows a very efficient method of measuring the period and duty cycle of the PWM waveforms. Figure 5 shows the waveforms from XOUT and YOUT. The most obvious method of measuring these waveforms is to measure the time from rising edge to falling edge to next rising edge for each of the waveforms. While very simple, this method takes 3 complete cycles to complete the process. If you take a closer look at Figure 5, you will see that the high time of XOUT and YOUT are centered about each other. We can use this to our advantage.

Figure 6 shows the waveform and measurement points for the improved measurement scheme. At Ta the counter is started. The program then records the times at Tb, Tc, and Td. By looking at the points of Figure 6, we can say that:

$$T1x = Tb - Ta = Tb \text{ (counter was 0 at } Ta)$$

$$T1y = Td - Tc$$

$$T2x = T2y = T2 = Te - Ta = Tg - Tf$$

Since we have already established that the center of T1 is aligned with the center of T2, the equation for T2 reduces to:

$$T2 = Td - T1y/2 - T1x/2$$

$$T2 = Td - (Td - Tc)/2 - Tb/2$$

This technique not only reduces the measurement time to two cycles, it also only calculates T2 once.

FIGURE 5: ADXL202 DUTY CYCLE OUTPUT

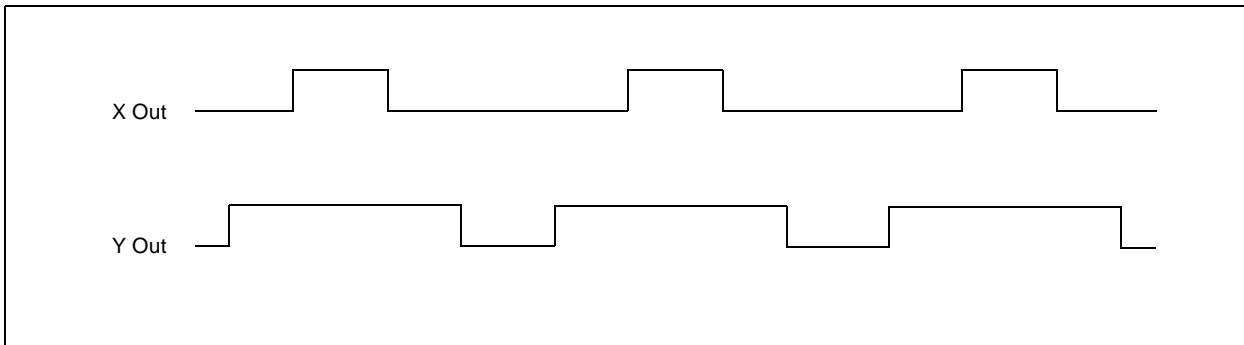
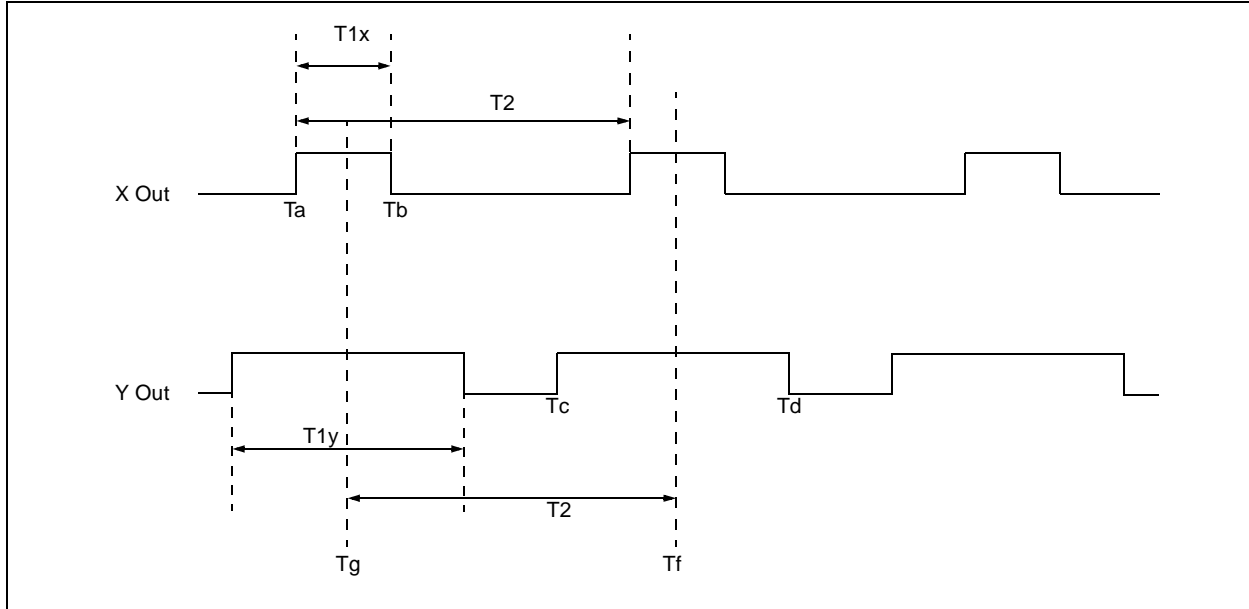


FIGURE 6: ADXL202 DUTY CYCLE MEASUREMENT



Now that our system is reading the duty cycle outputs of the ADXL202 and displaying the results on the LCD display, we need to consider how the system is calibrated. The first calibration step is the initial calibration of the tilt sensor with respect to gravity. The simplest method is to position the system such that the X-axis and Y-axis are both level. When instructed to calibrate, the PIC16F84A will calculate the duty cycle output T1 for both axis and the period T2. Several readings may be taken and averaged to improve the accuracy of the measurements. These values are now stored in both RAM as well as EEPROM as calibration constants. A scale factor is also used in the calibration process to create an n-bit result. These constants are defined as:

- T2cal, the value of T2 during the calibration phase. T2 must be stored because it can vary over temperature and has jitter from one measurement to another.
- ZXcal, the value of T1x during the calibration phase.
- ZYcal, the value of T1y during the calibration phase.
- K, the scale factor and is equal to $[4 * (T2cal * bit_scale_factor) / T2cal]$

K needs to be calculated only once. Since each axis will use this factor it is hard coded in the firmware. The bit_scale_factor is used to determine the size of the result. Since we are looking for a result of +/- 90 (1 degree of tilt per count), the bit scale factor would be 180. Therefore, K is assigned a value of 720. This is the simplification that was mentioned earlier.

Once we have calculated the calibration constants we can apply them to the duty cycle measurements to get degree of tilt. The following formulas give the degree of tilt for each axis:

$$ZX_{actual} = (ZX_{cal} * T2_{actual}) / T2_{cal} \quad (1)$$

$$ZY_{actual} = (ZY_{cal} * T2_{actual}) / T2_{cal} \quad (2)$$

T2actual is the current measurement of T2. This formula adjusts the 0g value for changes in T2 due to temperature or jitter.

$$X_{Acceleration} = [K * (T1x - ZX_{actual})] / T2_{actual} \quad (3)$$

$$Y_{Acceleration} = [K * (T1y - ZY_{actual})] / T2_{actual} \quad (4)$$

The values of T1x and T1y are the current measurements of T1 for each axis. The results in XAcceleration and YAcceleration are the degrees to tilt in the X-axis and Y-axis directions properly scaled for 1 degree per count. This method of calibration is very simple yet will suffer from small errors due to variance in duty cycle % per g (which was assumed to be 12.5%) from one part to the next.

AN715

The order of the math operations is deliberate to preserve the accuracy of the result. All math operations are done in fixed point math. Several variables are used in the math operations. The following table shows the two inputs to each routine and the location of the result of the routine.

TABLE 1: MATH OPERATIONS VARIABLE USAGE

Operation	Operand #1	Operand #2	Result
16 x 16 Addition	ACCHI, ACCLO	ARGH, ARGL	ACCHI, ACCLO
16 x 16 Subtraction	ACCHI, ACCLO	ARGH, ARGL	ACCHI, ACCLO
16 x 16 Multiply	ACCHI, ACCLO	ARGH, ARGL	PRODW3, PRODW2, PRODW1, PRODW0
32 x 16 Divide	PRODW3, PRODW2, PRODW1, PRODW0	DIV1, DIV0	ANS1, ANS0

For calculating Zactual in formula (1) and (2), the 16 x 16 multiply of Zcal * T2actual takes place first followed by the division of the result by T2cal. When calculating tilt (really is scaled acceleration) in the formulas (3) and (4), the subtraction of Zactual from T1 takes place first, followed by multiplication of the result by K, and finally the division of the result by T2actual.

Finally, the last two pieces of the code are for the LCD display and the Data EEPROM access. The LCD code is a derivative of that found in the application note, AN587, "Interfacing PICmicro[®] Microcontrollers to an LCD Module". Most of the changes were related to the different I/O pins used for data and control. The Data EEPROM routines use code directly from the PIC16F84A data sheet DS35007 for reads and writes. The WriteCal routine takes the calibration constants and writes them to the Data EEPROM. This routine is only called when a calibration cycle is performed. The RestoreCal routine is called when the PIC16F84A is reset. The calibration constants are grouped sequentially in memory so that these routines can use indirect addressing and shorten the length of code.

CONCLUSION

As in all applications, the type of acceleration sensor depends on the system requirements as well as the property being measured. Some accelerometers are better suited towards measuring vibration and shock such as the piezo-film and piezoelectric. Others are used for tilt measurements such as liquid tilt and micro-machined types. The type of sensor selected then dictates the signal conditioning circuitry requirements. Some accelerometers have AC response, some DC. Some sensors have analog outputs, others digital. In other words, one accelerometer does not fit into all applications. This application note has shown that a designer can quickly and easily complete an accelerometer based design using the ADXL sensors from Analog Devices. The use of a microcontroller further simplifies the design by giving the designer a more integrated, lower cost solution for the data measurement application.

REFERENCES

Data Sheets

- *ADXL202 Data Sheet*, Analog Devices Inc., Rev. 0
- *ADXL210 Data Sheet*, Analog Devices Inc., Rev. Pr.A
- *PIC16F84A Data Sheet*, Microchip Technology Inc., DS35007A

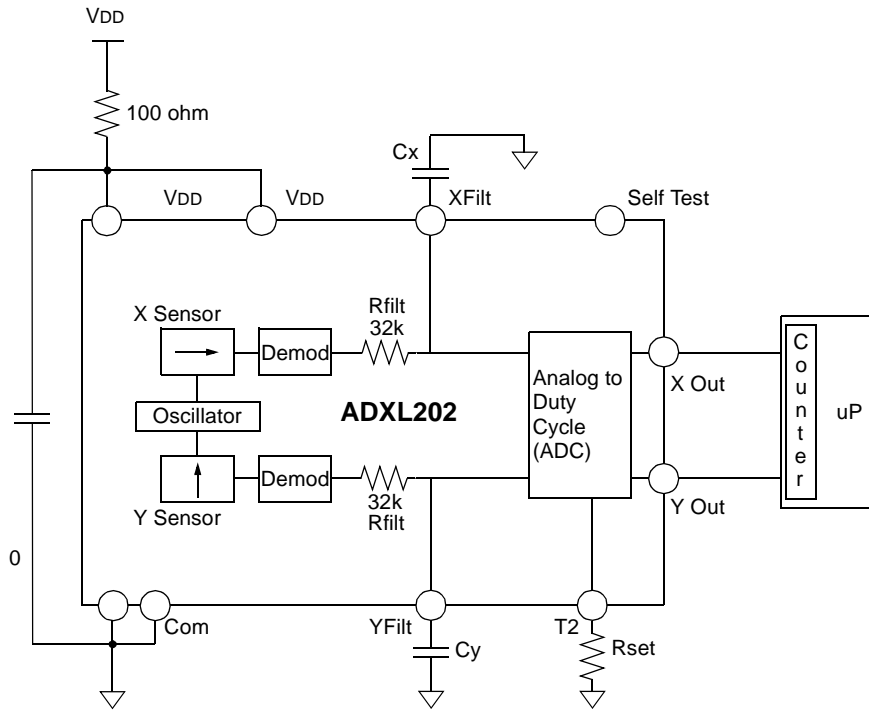
Technical Notes From Analog Devices

- *Using the ADXL202 Duty Cycle Output*
- *Accelerometer Design and Applications*
- *A Compact Algorithm Using The ADXL202 Duty Cycle Output*
- *The Interactive Designer ADXL202*

APPENDIX A: ADXL202 INTERACTIVE DESIGNER

The XL202 Interactive Designer

Enter values below. When your design is complete the values for your design will print out on this page.



Parameters

Supply voltage	5.0 V
Analog Bandwidth	10.5 Hz
Acquisition Rate	25 readings per second
Resolution (g's)	0.008 g
Resolution (deg of tilt)	0.47 deg of tilt
Microcontroller counter rate	1 MHz
T2	8.5 mS
Power cycling %	100% on time
Tmax	35 deg C
Tmin	15 deg C
Zero g drift Tmax	0.02 g
Zero g drift Tmin	0.02 g

Component Values

Supply Decoupling	0.1 uF
Xcap, Ycap	0.47 uF
Rset	1062.5 kohm

You will be asked to enter variety of design parameters important to your applications. This will include such issues as how fast is the signal you need to measure, what is the required update or acquisition rate, what is the counter speed on your microcontroller. After entering target values (inputs) the spreadsheet will calculate outputs such as the resolution of the accelerometer. You can then iterate the input values and trade off parameters as necessary to meet your design goals. Only enter values that are in **bold**.

1. Enter your nominal supply voltage

The XL202 will operate from 3.0V to 5.25V. Enter your nominal supply voltage here.

Vdd V

2. What is the fastest signal you want to be able to observe?

In this step you will determine the bandwidth for the analog stage of the accelerometer. The bandwidth generally determines the noise floor and thus the resolution of the accelerometer. In a later section you will also calculate digital noise sources from the PWM stage; the combination of these two noise sources determines the total noise floor. You will be measuring a real world acceleration, such as human or vehicle motion. What part of the signal content is important? If the signals are transient, such as shock or impulse, you may want to set a higher bandwidth. Human motion can often be measured at 10Hz or less. Don't forget to consider filter delays that could result in a lag between a stimulus and a response by the accelerometer, (dominated by the filter). Component values for the Xfilt and Yfilt capacitor are calculated below. You will probably want to iterate to a standard capacitor value.

Enter desired Bandwidth Hz Value for Cx, Cy 0.47 uF Component Value!

3. Estimate P-P noise

The peak to peak noise of the accelerometer is the best indicator of resolution of the accelerometer. Noise is a statistical process, and is best described by an RMS measurement, (available on the datasheet). P-P noise is then estimated using a statistical estimation. You need to select a RMS to P-P estimation. The table below tells you how various RMS to P-P noise multipliers, predict the amount of time the actual signal will EXCEED the estimated P-P noise. The lower the multiplier, the more likely it is that a noise event will exceed the P-P limit.

Enter RMS to P-P multiplier X of RMS

RMS Multiplier % of time a signal will exceed the P-P estimate

2X	32.00%
4X	4.60%
6X	0.27%
8X	0.01%

Calculated noise at the analog output Xfilt and Yfilt

Noise(rms) at Xfilt, Yfilt 0.002 g (max RMS)
 Noise(P-P) at Xfilt, Yfilt 0.008 g (max P-P) @4X RMS
 Noise(P-P) at Xfilt, Yfilt 0.47 Deg of tilt (max P-P) At 17mg/deg of tilt

Note: Noise level is inversely proportional to supply voltage

Note Decrease Noise (increase resolution) by decreasing BW.

3A. Iterate

Look at the P-P noise estimate; this is the noise limited resolution, (the smallest signal you can resolve). Is this acceptable for your application? If not you should consider adjusting the bandwidth down to reduce P-P noise and improve resolution.

4. How fast would you like to acquire the signals?

In this section we will begin the design of the digital output, and the microcontroller interface. You will input an acquisition rate, i.e. how many times per second you want a new reading from the accelerometer. You are also asked how long the part should be powered each second. Note that if you only want a few samples per second, but intend to keep the part powered all of the time, then you will need to set a faster acquisition rate in order to get reasonable values for the PWM output. The program requests that you input the time required to do the multiplies and divides to calculate the acceleration. 3.0ms is the time required for a Microchip 16C63 running at 4 Mhz. This section generates a component value for the Rset resistor.

Enter desired acquisition rate Each Channel per second

% of time part will be powered per second

Calculate Acquisition Time

Maximum time available to acquire 2 channels	20.0 ms
Time required to calculate two channels	<input type="text" value="3.0"/> ms
Time left for signal acquisition	17.0 ms (two channels)

This implies a requirement for the value of the PWM period T2

Thus, T2 =	8.5 mS or	118 Hz	This is the Sample Rate
Value for Rset	1062.5 kohm		Component Value!

5. Enter the counter rate of your Microcontroller and calculate the resolution of the digital output.

In Section 2, we calculated the resolution of the analog section. In this section we will calculate the resolution of the digital output; a function of the PWM rate T2 (calculated in section 4), and the counting rate of your microcontroller. Please note that the counting rate is different, and usually slower than the microcontroller clock rate. The output of this calculation is a measure of the quantization error of the counter. In some cases it may limit the ultimate resolution; we will explore this in

Counter Rate Mhz

Note: you will need a counter of size 17000 counts or 15 bits
To avoid overflowing the counter

Resolution	1062.5 Counts per g	Quantization bit size
Resolution	0.001 g	Based on 17mg/deg of tilt
Resolution	0.06 Deg of Tilt	

Note: Increase resolution by increasing counter rate or decreasing samples per second

6. Check for aliasing and other errors in sampling:

In all cases the sample rate (1/T2) needs to be faster than the bandwidth of the analog section by a factor of at least 2 in order to meet the requirements of Nyquist. Nyquist notwithstanding, a ratio of at least 10 is recommend to minimize dynamic errors that are endemic to PWM sampling techniques. If your ratio is low, you can improve it by either increasing the sample rate (by increasing the acquisition rate in section 4) or decreasing the analog bandwidth (in section 2).

Ratio of sample rate (1/T2) to analog BW: 11.2 Good!

7. Estimate of total resolution (iterate to meet design objective)

We are now in a position to bring together the various calculation above to determine the resolution of the complete analog and digital design. The ultimate resolution is determined by both the noise at the analog output (Xcap and Ycap) and the quantization bit size of the PWM + counter system. At this point check the total system resolution to see if it meets your requirements. If it does not, then revisit bandwidth at Xfilt and Yfilt, acquisition rate or counting rate to reduce noise. You may also want to consider digital filtering, (oversampling) to reduce noise at the expense of sampling rate as discussed in the next section.

Noise due to analog section	0.008 g (max P-P) @4X RMS	This is the noise contribution at the analog output Xcap, Ycap
Resolution of digital output- counter	0.001 g	This is the quantization noise of the digital output
Estimated Total Noise (resolution):	0.008 g P-P	This is the total P-P noise, which is the root sum square of the analog and digital noise.
Estimated Total Noise (resolution):	0.5 deg of tilt @ 17mg/deg	
Noise (Resolution) is limited by:	Bandwidth at Xfilt, Yfilt; reduce bandwidth if lower noise desired (section 2)	

8. Option: Reduce noise by oversampling (at expense of bandwidth)

Another design option is to use digital filtering (averaging) in order to reduce noise, at the expense of bandwidth. By averaging several samples you are in effect filtering the signal. Implementing averages of 2,4 8, 16 samples are simple right shifts in microcontroller code (very efficient). For oversampling to work, samples need to be taken at a rate no faster than 10 times the analog bandwidth. Note: make sure oversampling is set to 1 sample if you don't want to use oversampling!

Estimated Noise (resolution) with average of:	<input type="text" value="1"/> Samples	Noise before oversampling	0.008 g P-P	
Note: Samples should be taken about:	95.2381 mS apart	Noise after oversampling	0.008 g P-P	0% Reduction
		Bandwidth before oversampling	10.5 Hz	
		Bandwidth after oversampling	10.5 Hz	

9. Estimated Drift of Zero g point

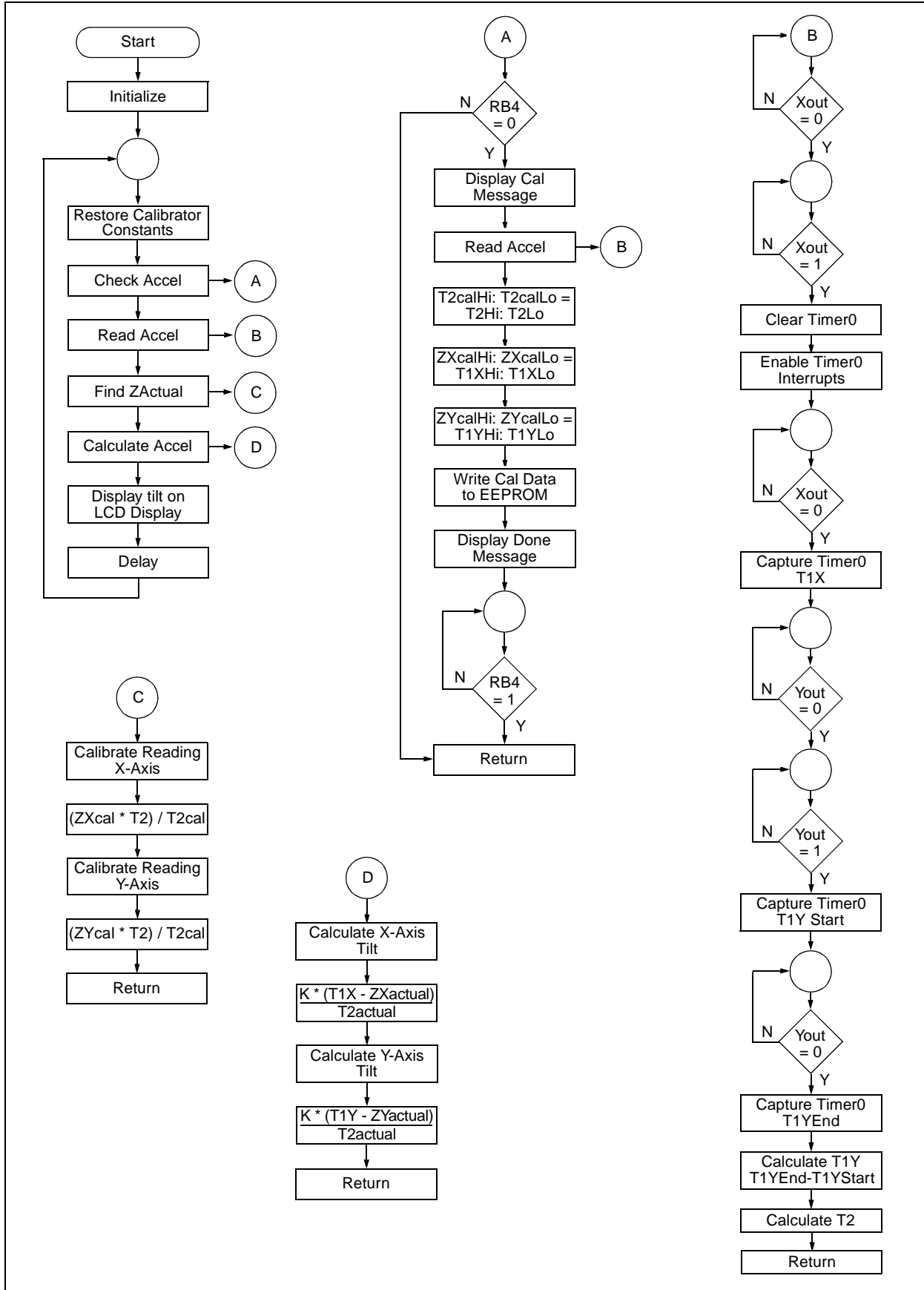
You can estimate the zero g temperature shift by entering your expected temperature range below and an estimate of the drift in mg/C (from the data sheet). Note that zero g drift can be positive or negative, but in general is very linear. X axis and Yaxis drift are uncorrelated.

Drift in mG per degree C
 Max Temp
 Min Temp

0.002	g/C
35	C
15	C

Drift from 25C Value		
Drift in mG	Drift in deg of tilt	
at Tmax 0.02 mg	1.2 deg of tilt	at 17mg/deg C
at Tmin 0.02 mg	1.2 deg of tilt	at 17mg/deg C

APPENDIX B: TILT SENSOR FIRMWARE FLOWCHART



APPENDIX C: TILT MOTOR SOURCE CODE LISTING

```

00001 list p=16f84a
00002 include <p16f84a.inc>
00001 LIST
00002 ; P16F84A.INC Standard Header File,Version 2.00 Microchip Technology
00134 LIST
00003
2007 3FF1 00004 __config __CP_OFF&_WDT_OFF&_XT_OSC&_PWRTE_ON
00005 ;Assembled using MPASM V2.30
00006 ;PORTA defines
00007 #define XOUT 0
00008 #define YOUT 1
00009 #define E 2
00010 #define RW 3
00011
00012 ;PORTB defines
00013 #define CAL 4
00014 #define RS 5
00015
00016 ;=====
00017 ; RAM EQUATES
00018 ;=====
00019 cblock 0x0c
000000C 00020 T1XHi
0000000D 00021 T1XLo
0000000E 00022 ArgL
0000000F 00023 ArgH
00000010 00024 AccHi
00000011 00025 AccLo
00000012 00026 DivCnt
00000013 00027 PRODW3
00000014 00028 PRODW2
00000015 00029 PRODW1
00000016 00030 PRODW0
00000017 00031 DIV0
00000018 00032 DIV1
00000019 00033 ANS0
0000001A 00034 ANS1
0000001B 00035 T2Hi
0000001C 00036 T2Lo
0000001D 00037 T1YStartLo
0000001E 00038 T1YStartHi
0000001F 00039 T1YEndLo
00000020 00040 T1YEndHi
00000021 00041 T1YHi
00000022 00042 T1YLo
00000023 00043 ZXcalHi
00000024 00044 ZXcalLo
00000025 00045 ZYcalHi
00000026 00046 ZYcalLo
00000027 00047 T2calHi
00000028 00048 T2calLo
00000029 00049 ZXActualHi
0000002A 00050 ZXActualLo
0000002B 00051 ZYActualHi
0000002C 00052 ZYActualLo
0000002D 00053 XAccel
0000002E 00054 YAccel

```

AN715

```
0000002F 00055      Temp0
00000030 00056      Temp1
00000031 00057      Temp2
00000032 00058      Temp3
00000033 00059      Timer0H
00000034 00060      EADR
00000035 00061      EDATA
      00062      endc
      00063
0000000E 00064 Count1 equ  ArgL
0000000F 00065 Count2 equ  ArgH
0000002F 00066 Temp   equ  Temp0
00000019 00067 CMD    equ  ANS0
00000019 00068 LDATA  equ  CMD
00000017 00069 Digit0 equ  DIV0
00000018 00070 Digit1 equ  DIV1
      00071
00000002 00072 KHi   equ  0x02
000000D0 00073 KLo   equ  0xd0
      00074
0000 00075      org  0x0000
0000 2808 00076      goto  Start      ;Go to start of program
      00077
0004 00078      org  0x0004
0004 0AB3 00079      incf  Timer0H,F
0005 110B 00080      bcf  INTCON,T0IF
0006 118B 00081      bcf  INTCON,RBIE
0007 0009 00082      retfie
      00083
0008 00084 Start
0008 1283 00085      bcf  STATUS,RP0
0009 0185 00086      clrf  PORTA
000A 0186 00087      clrf  PORTB
000B 1683 00088      bsf  STATUS,RP0      ;Bank1
000C 3003 00089      movlw B'00000011'    ;Set up the I/O ports
000D 0085 00090      movwf TRISA
000E 3010 00091      movlw B'00010000'
000F 0086 00092      movwf TRISB
0010 300F 00093      movlw B'00001111'
0011 0081 00094      movwf OPTION_REG
0012 1283 00095      bcf  STATUS,RP0      ;Bank0
0013 21E1 00096      call OpenXLCD      ;Initialize LCD
0014 22C6 00097      call RestoreCal    ;Restore calibration constants
0015 178B 00098      bsf  INTCON,GIE
      00099
0016 00100 Main
0016 20F2 00101      call  CheckCal      ;Check if need to calibrate
0017 2020 00102      call  ReadAccel    ;Read the acceleration
0018 2060 00103      call  FindZActual  ;Calibrate readings
0019 2085 00104      call  CalculateAccel ;Calculate tilt (acceleration)
001A 2194 00105      call  DisplayAccel ;Display results
001B 30FF 00106      movlw 0xff          ;Delay for a while
001C 22A2 00107      call  Delay_Ms_4MHz
001D 30FF 00108      movlw 0xff
001E 22A2 00109      call  Delay_Ms_4MHz
001F 2816 00110      goto  Main          ;Do it again
00111 ;*****
00112
00113 ;=====
00114 ;===== Acceleration Measurement/Calculation Routines =====
```



```

00115 ;=====
00116 ;*****
00117 ;ReadAccel
00118 ;   This subroutine acquires and calculates T1X, T1Y, and T2.
00119 ;       T1X is in registers T1XHi,T1XLo
00120 ;       T1Y is in registers T1YHi,T1YLo
00121 ;       T2 is in registers T2Hi,T2Lo
00122 ;*****
0020   00123 ReadAccel
0020   00124 EDGE1
0020 1805 00125   btfsc PORTA,XOUT   ;Wait for low on XOUT
0021 2820 00126   goto   EDGE1
0022   00127 EDGE2
0022 1C05 00128   btfss PORTA,XOUT   ;Wait for high on XOUT
0023 2822 00129   goto   EDGE2
0024 0181 00130   clrf  TMR0       ;Clear Timer
0025 01B3 00131   clrf  Timer0H
0026 110B 00132   bcf  INTCON,T0IF ;Enable Timer0 overflow interrupt
0027 168B 00133   bsf  INTCON,T0IE
0028   00134 EDGE3
0028 1805 00135   btfsc PORTA,XOUT   ;Look for falling edge on XOUT
0029 2828 00136   goto   EDGE3
002A 0801 00137   movf  TMR0,W       ;Save Timer0H:TMR0 as T1X
002B 008D 00138   movwf T1XLo
002C 0833 00139   movf  Timer0H,W
002D 008C 00140   movwf T1XHi
002E   00141 EDGE4
002E 1885 00142   btfsc PORTA,YOUT   ;Look a low level on YOUT
002F 282E 00143   goto   EDGE4
0030   00144 EDGE5
0030 1C85 00145   btfss PORTA,YOUT   ;Look for rising edge on YOUT
0031 2830 00146   goto   EDGE5
0032 0801 00147   movf  TMR0,W       ;Save Timer0H:TMR0 for start
0033 009D 00148   movwf T1YStartLo ;of T1Y
0034 0833 00149   movf  Timer0H,W
0035 009E 00150   movwf T1YStartHi
0036   00151 EDGE6
0036 1885 00152   btfsc PORTA,YOUT   ;Look for falling edge on YOUT
0037 2836 00153   goto   EDGE6
0038 0801 00154   movf  TMR0,W       ;Save Timer0H:TMR0 as the end
0039 009F 00155   movwf T1YEndLo   ;of T1Y
003A 0833 00156   movf  Timer0H,W
003B 00A0 00157   movwf T1YEndHi
003C 128B 00158   bcf  INTCON,T0IE
00159
003D 0820 00160   movf  T1YEndHi,W   ;Calculate T1Y
003E 0090 00161   movwf AccHi
003F 081F 00162   movf  T1YEndLo,W
0040 0091 00163   movwf AccLo
0041 081E 00164   movf  T1YStartHi,W
0042 008F 00165   movwf ArgH
0043 081D 00166   movf  T1YStartLo,W
0044 008E 00167   movwf ArgL
0045 210E 00168   call Sub16x16
0046 0810 00169   movf  AccHi,W
0047 00A1 00170   movwf T1YHi
0048 0811 00171   movf  AccLo,W
0049 00A2 00172   movwf T1YLo
004A 0820 00173   movf  T1YEndHi,W   ;CALCULATE T2
004B 0090 00174   movwf AccHi       ;2*(T2Hi,T2Lo) = (T1YEndHi:T1YEndLo)+

```

AN715

```
004C 081F 00175    movf  T1YEndLo,W    ;(T1YStartHi:T1YStartLo)-(T1XHi:T1XLo)
004D 0091 00176    movwf AccLo
004E 081E 00177    movf  T1YStartHi,W
004F 008F 00178    movwf ArgH
0050 081D 00179    movf  T1YStartLo,W
0051 008E 00180    movwf ArgL
0052 2107 00181    call Add16x16      ;ACCHI,ACCLO=(T1YEndHi:T1YEndLo)+
0053 080C 00182    movf  T1XHi,W      ; (T1YStartHi:T1YStartLo)
0054 008F 00183    movwf ArgH
0055 080D 00184    movf  T1XLo,W
0056 008E 00185    movwf ArgL
0057 210E 00186    call Sub16x16     ;ACCHI,ACCLO = 2*T2
0058 1003 00187    bcf   STATUS,C
0059 0C90 00188    rrf   AccHi,F
005A 0C91 00189    rrf   AccLo,F
005B 0810 00190    movf  AccHi,W
005C 009B 00191    movwf T2Hi
005D 0811 00192    movf  AccLo,W
005E 009C 00193    movwf T2Lo
005F 0008 00194    return
00195
00196
00197 ;*****
0060 00198 FindZActual
00199 ; This subroutine finds the value of ZActual for the X and Y
00200 ; axis.
00201 ; Output is ZXActualHi & ZXActualLo for the X-axis and
00202 ; ZYActualHi & ZXActualLo for the Y-axis.
00203 ;*****
0060 00204 FindZActual
0060 0823 00205    movf  ZXcalHi,W    ;First the X-axis
0061 0090 00206    movwf AccHi
0062 0824 00207    movf  ZXcalLo,W
0063 0091 00208    movwf AccLo
0064 081B 00209    movf  T2Hi,W
0065 008F 00210    movwf ArgH
0066 081C 00211    movf  T2Lo,W
0067 008E 00212    movwf ArgL        ;PRODW3,PRODW2,PRODW1,PRODW0 =
0068 211A 00213    call Mul16x16     ;(ZXCAL_HI,ZXCAL_LO)*(T2HI,T2LO)
0069 0827 00214    movf  T2calHi,W
006A 0098 00215    movwf DIV1
006B 0828 00216    movf  T2calLo,W
006C 0097 00217    movwf DIV0
006D 2165 00218    call Div32x16     ;ANS1,ANS0 = (ZXcal * T2) / T2cal
006E 081A 00219    movf  ANS1,W
006F 00A9 00220    movwf ZXActualHi
0070 0819 00221    movf  ANS0,W
0071 00AA 00222    movwf ZXActualLo
0072 0825 00223    movf  ZYcalHi,W   ;Same thing for the Y-axis
0073 0090 00224    movwf AccHi
0074 0826 00225    movf  ZYcalLo,W
0075 0091 00226    movwf AccLo
0076 081B 00227    movf  T2Hi,W
0077 008F 00228    movwf ArgH
0078 081C 00229    movf  T2Lo,W
0079 008E 00230    movwf ArgL        ;PRODW3,PRODW2,PRODW1,PRODW0 =
007A 211A 00231    call Mul16x16     ;(ZYCAL_HI,ZYCAL_LO)*(T2HI,T2LO)
007B 0827 00232    movf  T2calHi,W
007C 0098 00233    movwf DIV1
007D 0828 00234    movf  T2calLo,W
```

```

007E 0097 00235    movwf  DIV0
007F 2165 00236    call  Div32x16    ;ANS1,ANS0 = (ZYcal * T2) / T2cal
0080 081A 00237    movf   ANS1,W
0081 00AB 00238    movwf  ZYActualHi
0082 0819 00239    movf   ANS0,W
0083 00AC 00240    movwf  ZYActualLo
0084 0008 00241    return
00242
00243
00244 ;*****
00245 ;CalculateAccel
00246 ;   This subroutine performs the acceleration calculation for
00247 ;   each axis. The formula is:
00248 ;       ACCEL = [K * (T1-Zactual) / T2actual]
00249 ;   Output is XAccel and YAccel
00250 ;*****
0085 00251 CalculateAccel
0085 0829 00252    movf  ZXActualHi,W    ;Check if acceleration is positive
0086 020C 00253    subwf T1XHi,W        ;or negative by comparing
0087 1C03 00254    btfss STATUS,C       ;T1X and ZXactual
0088 28A5 00255    goto  CA1            ;Jump if T1X < ZXactual
0089 1D03 00256    btfss STATUS,Z       ;Test if T1XHI=ZX_ACTUAL_HI
008A 288F 00257    goto  CA2            ;Jump if T1X > ZXactual
008B 082A 00258    movf  ZXActualLo,W
008C 020D 00259    subwf T1XLo,W
008D 1C03 00260    btfss STATUS,C
008E 28A5 00261    goto  CA1            ;Jump if TX1 < ZXactual
008F 00262 CA2
008F 080C 00263    movf  T1XHi,W        ;T1X - ZXactual
0090 0090 00264    movwf AccHi
0091 080D 00265    movf  T1XLo,W
0092 0091 00266    movwf AccLo
0093 0829 00267    movf  ZXActualHi,W
0094 008F 00268    movwf ArgH
0095 082A 00269    movf  ZXActualLo,W
0096 008E 00270    movwf ArgL
0097 210E 00271    call  Sub16x16
0098 3002 00272    movlw KHi
0099 008F 00273    movwf ArgH
009A 30D0 00274    movlw KLo
009B 008E 00275    movwf ArgL
009C 211A 00276    call  Mul16x16    ;PRODW3,PRODW2,PRODW1,PRODW0 =
00277    ;K * (T1X - ZXactual)
009D 081B 00278    movf  T2Hi,W
009E 0098 00279    movwf DIV1
009F 081C 00280    movf  T2Lo,W
00A0 0097 00281    movwf DIV0
00A1 2165 00282    call  Div32x16    ;ANS1:ANS0=
00A2 0819 00283    movf  ANS0,W        ; [K*(T1X-ZXactual)]/T2actual
00A3 00AD 00284    movwf XAccel        ;The result will be a signed 8-bit #
00A4 28BB 00285    goto  DoYCalc
00A5 00286 CA1
00A5 0829 00287    movf  ZXActualHi,W    ;ZXactual - T1X
00A6 0090 00288    movwf AccHi
00A7 082A 00289    movf  ZXActualLo,W
00A8 0091 00290    movwf AccLo
00A9 080C 00291    movf  T1XHi,W
00AA 008F 00292    movwf ArgH
00AB 080D 00293    movf  T1XLo,W
00AC 008E 00294    movwf ArgL

```

AN715

```
00AD 210E 00295    call  Sub16x16
00AE 3002 00296    movlw  KHi
00AF 008F 00297    movwf  ArgH
00B0 30D0 00298    movlw  KLo
00B1 008E 00299    movwf  ArgL
00B2 211A 00300    call  Mul16x16    ;PRODW3,PRODW2,PRODW1,PRODW0 =
00301              ;K * (ZXactual - T1X)
00B3 081B 00302    movf   T2Hi,W
00B4 0098 00303    movwf  DIV1
00B5 081C 00304    movf   T2Lo,W
00B6 0097 00305    movwf  DIV0
00B7 2165 00306    call  Div32x16    ;ANS1,ANS0 =
00B8 0919 00307    comf   ANS0,W     ; [K*(ZXactual-T1X)]/T2actual
00B9 3E01 00308    addlw  0x01       ;The result will be a signed 8-bit #
00BA 00AD 00309    movwf  XAccel
00BB      00310 DoYCalc
00BB 082B 00311    movf   ZYActualHi,W ;Check if acceleration is positive
00BC 0221 00312    subwf  T1YHi,W    ;or negative by comparing
00BD 1C03 00313    btfss  STATUS,C   ;T1Y and ZYactual
00BE 28DB 00314    goto  CA3         ;Jump if T1Y < ZYactual
00BF 1D03 00315    btfss  STATUS,Z   ;Test if T1YHI=ZY_ACTUAL_HI
00C0 28C5 00316    goto  CA4         ;Jump if T1Y > ZYactual
00C1 082C 00317    movf   ZYActualLo,W
00C2 0222 00318    subwf  T1YLo,W
00C3 1C03 00319    btfss  STATUS,C   ;Jump if TY1 < ZYactual
00C4 28DB 00320    goto  CA3
00C5      00321 CA4
00C5 0821 00322    movf   T1YHi,W    ;T1Y - ZYactual
00C6 0090 00323    movwf  AccHi
00C7 0822 00324    movf   T1YLo,W
00C8 0091 00325    movwf  AccLo
00C9 082B 00326    movf   ZYActualHi,W
00CA 008F 00327    movwf  ArgH
00CB 082C 00328    movf   ZYActualLo,W
00CC 008E 00329    movwf  ArgL
00CD 210E 00330    call  Sub16x16
00CE 3002 00331    movlw  KHi
00CF 008F 00332    movwf  ArgH
00D0 30D0 00333    movlw  KLo
00D1 008E 00334    movwf  ArgL
00D2 211A 00335    call  Mul16x16    ;PRODW3,PRODW2,PRODW1,PRODW0 =
00336              ;K * (T1Y - ZYactual)
00D3 081B 00337    movf   T2Hi,W
00D4 0098 00338    movwf  DIV1
00D5 081C 00339    movf   T2Lo,W
00D6 0097 00340    movwf  DIV0
00D7 2165 00341    call  Div32x16    ;ANS1,ANS0 =
00D8 0819 00342    movf   ANS0,W     ; [K*(T1Y-ZYactual)]/T2actual
00D9 00AE 00343    movwf  YAccel    ;The result will be a signed 8-bit #
00DA 0008 00344    return
00DB      00345 CA3
00DB 082B 00346    movf   ZYActualHi,W ;ZYactual - T1Y
00DC 0090 00347    movwf  AccHi
00DD 082C 00348    movf   ZYActualLo,W
00DE 0091 00349    movwf  AccLo
00DF 0821 00350    movf   T1YHi,W
00E0 008F 00351    movwf  ArgH
00E1 0822 00352    movf   T1YLo,W
00E2 008E 00353    movwf  ArgL
00E3 210E 00354    call  Sub16x16
```

```

00E4 3002 00355    movlw  KHi
00E5 008F 00356    movwf  ArgH
00E6 30D0 00357    movlw  KLo
00E7 008E 00358    movwf  ArgL
00E8 211A 00359    call  Mul16x16    ;PRODW3,PRODW2,PRODW1,PRODW0 =
00360                ;K * (ZYactual - T1Y)
00E9 081B 00361    movf   T2Hi,W
00EA 0098 00362    movwf  DIV1
00EB 081C 00363    movf   T2Lo,W
00EC 0097 00364    movwf  DIV0
00ED 2165 00365    call  Div32x16    ;ANS1,ANS0 =
00EE 0919 00366    comf   ANS0,W     ;[K*(ZYactual-T1Y)]/T2actual
00EF 3E01 00367    addlw  0x01       ;The result will be a signed 8-bit #
00F0 00AE 00368    movwf  YAccel
00F1 0008 00369    return
00370
00371
00372 ;*****
00373 ;CheckCal
00374 ; This subroutine reads the CAL pushbutton switch (RB4) and if
00375 ; it is low, performs a simple calibration routine.
00376 ;*****
00F2 00377 CheckCal
00F2 1A06 00378    btfsc  PORTB,CAL    ;Is RB4 low?
00F3 0008 00379    return            ;If not then exit routine
00F4 2276 00380    call  DisplayCal
00F5 2020 00381    call  ReadAccel    ;If yes then perform a read cycle
00F6 081B 00382    movf   T2Hi,W     ;Save the measured values in the
00F7 00A7 00383    movwf  T2calHi    ;calibration registers
00F8 081C 00384    movf   T2Lo,W
00F9 00A8 00385    movwf  T2calLo
00FA 080C 00386    movf   T1XHi,W
00FB 00A3 00387    movwf  ZXcalHi
00FC 080D 00388    movf   T1XLo,W
00FD 00A4 00389    movwf  ZXcalLo
00FE 0821 00390    movf   T1YHi,W
00FF 00A5 00391    movwf  ZYcalHi
0100 0822 00392    movf   T1YLo,W
0101 00A6 00393    movwf  ZYcalLo
0102 22B9 00394    call  WriteCal     ;Write the calibration data to EEPROM
0103 2292 00395    call  DisplayDone  ;Write message to LCD display
0104 00396 CCLoop
0104 1E06 00397    btfss  PORTB,CAL    ;Wait for pushbutton switch to be
0105 2904 00398    goto  CCLoop      ;released
0106 0008 00399    return
00400 ;*****
00401
00402
00403 ;=====
00404 ;===== Mathematical Operations =====
00405 ;=====
00406 ;*****
00407 ;Add16x16
00408 ; This subroutine performs a 16-bit by 16-bit addition.
00409 ; Note that this routine does not check for possible overflow
00410 ; results i.e., 17-bit sum.
00411 ; Inputs are AccHi:AccLo and ArgH:ArgL
00412 ; Result is in AccHi:AccLo
00413 ; (AccHi:AccLo) = (AccHi:AccLo)+(ArgH:ArgL)
00414 ;*****

```

AN715

```
0107 00415 Add16x16
0107 080E 00416      movf  ArgL,W      ;Add low bytes together
0108 0791 00417      addwf AccLo,F
0109 1803 00418      btfsc STATUS,C      ;Check for carry out of addition
010A 0A90 00419      incf  AccHi,F      ;If yes, increment AccHi
010B 080F 00420      movf  ArgH,W      ;Add high bytes together
010C 0790 00421      addwf AccHi,F
010D 0008 00422      return
00423
00424 ;*****
00425 ;Sub16x16
00426 ; This subroutine performs a 16-bit by 16-bit subtraction.
00427 ; Inputs are AccHi:AccLo and ArgH:ArgL
00428 ; Result is in AccHi:AccLo
00429 ; (AccHi:AccLo) = (AccHi:AccLo)-(ArgH:ArgL)
00430 ;*****
010E 00431 Sub16x16
010E 098E 00432      comf  ArgL,F      ;2's complement ArgH:ArgL
010F 0A8E 00433      incf  ArgL,F
0110 1903 00434      btfsc STATUS,2
0111 038F 00435      decf  ArgH,F
0112 098F 00436      comf  ArgH,F
0113 080E 00437      movf  ArgL,W      ;Now perform a 16-bit addition
0114 0791 00438      addwf AccLo,F
0115 1803 00439      btfsc STATUS,W
0116 0A90 00440      incf  AccHi,F
0117 080F 00441      movf  ArgH,W
0118 0790 00442      addwf AccHi,F
0119 0008 00443      return
00444
00445 ;*****
00446 ;Mul16x16
00447 ; This subroutine performs a 16-bit by 16-bit multiplication.
00448 ; It produces a 32-bit number. Multiplication by 0 is checked
00449 ; and performed correctly, ie, A * 0 = 0.
00450 ; Inputs are (AccHi:AccLo) and (ArgH:ArgL)
00451 ; Output is (PRODW3:PRODW2:PRODW1:PRODW0)
00452 ; (PRODW3:PRODW2:PRODW1:PRODW0) = (AccHi:AccLo) * (ArgH:ArgL)
00453 ;*****
011A 00454 Mul16x16
011A 01AF 00455      clrf  Temp0      ;Clear the temporary variables used
011B 01B0 00456      clrf  Temp1      ;in this routine
011C 01B1 00457      clrf  Temp2
011D 01B2 00458      clrf  Temp3
011E 0196 00459      clrf  PRODW0
011F 0195 00460      clrf  PRODW1
0120 0194 00461      clrf  PRODW2
0121 0193 00462      clrf  PRODW3
0122 0811 00463      movf  AccLo,W
0123 00AF 00464      movwf Temp0      ;Move contents of AccHi:AccLo
0124 0810 00465      movf  AccHi,W      ;into Temp1:Temp0
0125 00B0 00466      movwf Temp1
0126 0890 00467      movf  AccHi,F      ;Test if AccHi:AccLo = 0000
0127 1D03 00468      btfss STATUS,Z
0128 292C 00469      goto  CheckNext   ;AccHi:AccLo not zero
0129 0891 00470      movf  AccLo,F
012A 1903 00471      btfsc STATUS,Z
012B 2960 00472      goto  Equal0      ;AccHi:AccLo = 0000
012C 00473 CheckNext
012C 088F 00474      movf  ArgH,F      ;Test if ArgH:ArgL = 0000
```

```

012D 1D03 00475    btfss STATUS,Z
012E 2932 00476    goto  DoMultiply    ;ArgH:ArgL not zero
012F 088E 00477    movf  ArgL,F
0130 1903 00478    btfsc STATUS,Z
0131 2960 00479    goto  Equal0       ;ArgH:ArgL = 0000
0132    00480 DoMultiply
0132 088F 00481    movf  ArgH,F       ;Test if ArgH:ArgL has been reduced
0133 1D03 00482    btfss STATUS,Z    ;to 0
0134 2938 00483    goto  TestLSB     ;ArgH:ArgL has not been reduced to 0
0135 088E 00484    movf  ArgL,F
0136 1903 00485    btfsc STATUS,Z
0137 0008 00486    return            ;ArgH:ArgL has been reduced to zero
00487                ;so multiplication is done
0138    00488 TestLSB
0138 1003 00489    bcf   STATUS,C     ;Shift ArgH:ArgL right
0139 0C8F 00490    rrf   ArgH,F
013A 0C8E 00491    rrf   ArgL,F
013B 1C03 00492    btfss STATUS,C    ;Is LSb of ArgH:ArgL = 1
013C 295A 00493    goto  DoShift     ;Jump if LSb = 0
013D 082F 00494    movf  Temp0,W     ;If LSb = 1 then
013E 0796 00495    addwf PRODW0,F   ;PRODW3:PRODW2:PRODW1:PRODW0 =
013F 1C03 00496    btfss STATUS,C    ;PRODW3:PRODW2:PRODW1:PRODW0 +
0140 294A 00497    goto  ADD2        ;Temp3:Temp2:Temp1:Temp0
0141 3001 00498    movlw 0x01       ;Add carry bit if necessary
0142 0795 00499    addwf PRODW1,F
0143 1C03 00500    btfss STATUS,C
0144 294A 00501    goto  ADD2
0145 3001 00502    movlw 0x01       ;Add carry bit if PRODW1 overflows
0146 0794 00503    addwf PRODW2,F   ;as a result of the addition of the
0147 1C03 00504    btfss STATUS,C    ;previous carry
0148 294A 00505    goto  ADD2
0149 0A93 00506    incf  PRODW3,F
014A    00507 ADD2
014A 0830 00508    movf  Temp1,W
014B 0795 00509    addwf PRODW1,F
014C 1C03 00510    btfss STATUS,C
014D 2953 00511    goto  ADD3
014E 3001 00512    movlw 0x01
014F 0794 00513    addwf PRODW2,F
0150 1C03 00514    btfss STATUS,C
0151 2953 00515    goto  ADD3
0152 0A93 00516    incf  PRODW3,F
0153    00517 ADD3
0153 0831 00518    movf  Temp2,W
0154 0794 00519    addwf PRODW2,F
0155 1C03 00520    btfss STATUS,C
0156 2958 00521    goto  ADD4
0157 0A93 00522    incf  PRODW3,F
0158    00523 ADD4
0158 0832 00524    movf  Temp3,W
0159 0793 00525    addwf PRODW3,F
015A    00526 DoShift
015A 1003 00527    bcf   STATUS,C     ;Shift temp registers left
015B 0DAF 00528    rlf   Temp0,F
015C 0DB0 00529    rlf   Temp1,F
015D 0DB1 00530    rlf   Temp2,F
015E 0DB2 00531    rlf   Temp3,F
015F 2932 00532    goto  DoMultiply
0160    00533 Equal0
0160 0196 00534    clrf  PRODW0     ;Since one argument equals zero

```

AN715

```
0161 0195 00535   clrf  PRODW1       ;PRODW3,PRODW2,PRODW1,PRODW0 = 0
0162 0194 00536   clrf  PRODW2
0163 0193 00537   clrf  PRODW3
0164 0008 00538   return
00539
00540 ;*****
00541 ;Div32x16
00542 ; This subroutine performs a 32-bit x 16-bit division.
00543 ; Division is performed by binary long division.
00544 ; Inputs are (PRODW3:PRODW2:PRODW1:PRODW0) and (DIV1:DIV0).
00545 ; Output is (ANS1:ANS0)
00546 ; (ANS1:ANS0) = (PRODW3:PRODW2:PRODW1:PRODW0) / (DIV1:DIV0)
00547 ;*****
0165 00548 Div32x16
0165 019A 00549   clrf  ANS1         ;Clear the result registers
0166 0199 00550   clrf  ANS0
0167 3011 00551   movlw 0x11        ;DivCnt = 17d
0168 0092 00552   movwf DivCnt
0169 00553 DA1
0169 0818 00554   movf  DIV1,W
016A 0213 00555   subwf PRODW3,W    ;Is DIV1 > PRODW3
016B 1C03 00556   btfss STATUS,C
016C 2973 00557   goto  NoSub      ;Jump if DIV1 > PRODW3
016D 1D03 00558   btfss STATUS,2   ;Is DIV1 = PRODW3
016E 297C 00559   goto  DoSubs     ;Jump if DIV1 < PRODW3
016F 0817 00560   movf  DIV0,W     ;Is DIV0 > PRODW2
0170 0214 00561   subwf PRODW2,W
0171 1803 00562   btfsc STATUS,C
0172 297C 00563   goto  DoSubs     ;Jump if DIV0 < PRODW2
0173 00564 NoSub
0173 1003 00565   bcf  STATUS,C    ;Clear the carry bit
0174 0D99 00566   rlf  ANS0,F      ;Add 0 to LSb of ANS1,ANS0
0175 0D9A 00567   rlf  ANS1,F
0176 1003 00568   bcf  STATUS,C    ;Clear the carry bit
0177 0D96 00569   rlf  PRODW0,F    ;Shift PRODW3,2,1,0 left
0178 0D95 00570   rlf  PRODW1,F
0179 0D94 00571   rlf  PRODW2,F
017A 0D93 00572   rlf  PRODW3,F
017B 2991 00573   goto  ChkCnt
017C 00574 DoSubs
017C 0813 00575   movf  PRODW3,W
017D 0090 00576   movwf AccHi
017E 0814 00577   movf  PRODW2,W
017F 0091 00578   movwf AccLo
0180 0818 00579   movf  DIV1,W
0181 008F 00580   movwf ArgH
0182 0817 00581   movf  DIV0,W
0183 008E 00582   movwf ArgL
0184 210E 00583   call  Sub16x16   ;(PRODW3:2) = (PRODW3:2)-(DIV1:0)
0185 0810 00584   movf  AccHi,W
0186 0093 00585   movwf PRODW3
0187 0811 00586   movf  AccLo,W
0188 0094 00587   movwf PRODW2
0189 1403 00588   bsf  STATUS,C
018A 0D99 00589   rlf  ANS0,F
018B 0D9A 00590   rlf  ANS1,F      ;Add 1 to LSb of ANS1:ANS0
018C 1003 00591   bcf  STATUS,C
018D 0D96 00592   rlf  PRODW0,F    ;Shift PRODW3,2,1,0, left
018E 0D95 00593   rlf  PRODW1,F
018F 0D94 00594   rlf  PRODW2,F
```



```

0190 0D93 00595    rif    PRODW3,F
0191    00596 ChkCnt
0191 0B92 00597    decfsz DivCnt,F    ;Check for 17 operations
0192 2969 00598    goto   DA1        ;If not then loop
0193 0008 00599    return
      00600 ;*****
      00601
      00602
      00603 ;=====
00604 ;===== Display Routines =====
00605 ;=====
00606 ;*****
00607 ;DisplayAccel
00608 ; This subroutine takes the values int XAccel and YAccel and
00609 ; displays the ASCII equivalent on the LCD display.
00610 ;*****
0194    00611 DisplayAccel
0194 223E 00612    call   BusyXLCD    ;Wait for LCD to not be busy
0195 3001 00613    movlw  0x01        ;Reset cursor to home position
0196 221C 00614    call   WriteCmdXLCD ;of line 1
      00615
0197 1FAD 00616    btfss XAccel,7    ;Check if XAccel is negative
0198 29A0 00617    goto   XSpace
0199 223E 00618    call   BusyXLCD    ;Is negative
019A 302D 00619    movlw  '-'        ;Print a '-' to the display
019B 2254 00620    call   WriteDataXLCD
019C 092D 00621    comf   XAccel,W    ;2's complement XAccel
019D 3E01 00622    addlw  0x01
019E 00AD 00623    movwf  XAccel
019F 29A3 00624    goto   DispX
01A0    00625 XSpace    ;Not negative
01A0 223E 00626    call   BusyXLCD
01A1 3020 00627    movlw  ' '        ;Print a space to the display
01A2 2254 00628    call   WriteDataXLCD
01A3    00629 DispX
01A3 082D 00630    movf   XAccel,W    ;Convert XAccel to 2-digit ASCII
01A4 22AC 00631    call   Bin2Ascii
01A5 223E 00632    call   BusyXLCD
01A6 0818 00633    movf   Digit1,W    ;Write the upper digit to the LCD
01A7 2254 00634    call   WriteDataXLCD
01A8 223E 00635    call   BusyXLCD
01A9 0817 00636    movf   Digit0,W    ;Write the lower digit to the LCD
01AA 2254 00637    call   WriteDataXLCD
01AB 223E 00638    call   BusyXLCD
01AC 30DF 00639    movlw  0xdf        ;Write a degrees symbol to the LCD
01AD 2254 00640    call   WriteDataXLCD
01AE 223E 00641    call   BusyXLCD
01AF 3020 00642    movlw  ' '        ;Write " Pit" to the LCD
01B0 2254 00643    call   WriteDataXLCD ;for the word pitch which refers
01B1 223E 00644    call   BusyXLCD    ;to the X-axis
01B2 3050 00645    movlw  'P'
01B3 2254 00646    call   WriteDataXLCD
01B4 223E 00647    call   BusyXLCD
01B5 3069 00648    movlw  'i'
01B6 2254 00649    call   WriteDataXLCD
01B7 223E 00650    call   BusyXLCD
01B8 3074 00651    movlw  't'
01B9 2254 00652    call   WriteDataXLCD
01BA 223E 00653    call   BusyXLCD
01BB 30A8 00654    movlw  0xa8        ;Change the cursor position to home

```

AN715

```
01BC 221C 00655    call  WriteCmdXLCD    ;of line 2
                   00656
01BD 1FAE 00657    btfs  YAccel,7        ;Check if YAccel is negative
01BE 29C6 00658    goto  YSpace
01BF 223E 00659    call  BusyXLCD        ;Is negative
01C0 302D 00660    movlw '-'             ;Print a '-' to the display
01C1 2254 00661    call  WriteDataXLCD
01C2 092E 00662    comf  YAccel,W        ;2's complement YAccel
01C3 3E01 00663    addlw 0x01
01C4 00AE 00664    movwf YAccel
01C5 29C9 00665    goto  DispY
01C6    00666 YSpace    ;Not negative
01C6 223E 00667    call  BusyXLCD
01C7 3020 00668    movlw ''             ;Print a space to the display
01C8 2254 00669    call  WriteDataXLCD
01C9    00670 DispY
01C9 082E 00671    movf  YAccel,W        ;Convert YAccel to 2-digit ASCII
01CA 22AC 00672    call  Bin2Ascii
01CB 223E 00673    call  BusyXLCD
01CC 0818 00674    movf  Digit1,W        ;Write the upper digit to the LCD
01CD 2254 00675    call  WriteDataXLCD
01CE 223E 00676    call  BusyXLCD
01CF 0817 00677    movf  Digit0,W        ;Write the lower digit t the LCD
01D0 2254 00678    call  WriteDataXLCD
01D1 223E 00679    call  BusyXLCD
01D2 30DF 00680    movlw 0xdf           ;Write a degrees symbol to the LCD
01D3 2254 00681    call  WriteDataXLCD
01D4 223E 00682    call  BusyXLCD
01D5 3020 00683    movlw ''             ;Write " Rol" to the LCD
01D6 2254 00684    call  WriteDataXLCD ;for the word roll which refers
01D7 223E 00685    call  BusyXLCD      ;to the Y-axis
01D8 3052 00686    movlw 'R'
01D9 2254 00687    call  WriteDataXLCD
01DA 223E 00688    call  BusyXLCD
01DB 306F 00689    movlw 'o'
01DC 2254 00690    call  WriteDataXLCD
01DD 223E 00691    call  BusyXLCD
01DE 306C 00692    movlw 'l'
01DF 2254 00693    call  WriteDataXLCD
01E0 0008 00694    return
                   00695
01E0 0008 00694    return
                   00696 ;*****
01E0 0008 00694    return
                   00697 ;OpenXLCD
01E0 0008 00694    return
                   00698 ; This subroutine initializes the LCD display. It is
01E0 0008 00694    return
                   00699 ; cleared and blank upon exit of this routine
01E0 0008 00694    return
                   00700 ;*****
01E1    00701 OpenXLCD
01E1 301E 00702    movlw 0x1e           ;Delay for POR
01E2 22A2 00703    call  Delay_Ms_4MHz
                   00704
01E3 30F0 00705    movlw 0xf0           ;Write upper byte of configuration
01E4 1683 00706    bsf  STATUS,RP0      ;value to the LCD three times
01E5 0586 00707    andwf TRISB,F        ;After this the LCD can be read
01E6 1283 00708    bcf  STATUS,RP0
01E7 0586 00709    andwf PORTB,F
01E8 3003 00710    movlw 0x03
01E9 0486 00711    iorwf PORTB,F        ;Output data to the port, 8-bit mode
01EA 1505 00712    bsf  PORTA,E         ;Clock the data in
01EB 0000 00713    nop
01EC 1105 00714    bcf  PORTA,E
```

```
00715
01ED 300A 00716    movlw 0x0a      ;Wait for ~5ms
01EE 22A2 00717    call Delay_Ms_4MHz
00718
01EF 30F0 00719    movlw 0xf0
01F0 0586 00720    andwf PORTB,F
01F1 3003 00721    movlw 0x03
01F2 0486 00722    iorwf PORTB,F  ;Output data to the port, 8-bit mode
01F3 1505 00723    bsf  PORTA,E   ;Clock the data in
01F4 0000 00724    nop
01F5 1105 00725    bcf  PORTA,E
00726
01F6 300A 00727    movlw 0x0a      ;Wait for ~5ms
01F7 22A2 00728    call Delay_Ms_4MHz
00729
01F8 30F0 00730    movlw 0xf0
01F9 0586 00731    andwf PORTB,F
01FA 3003 00732    movlw 0x03
01FB 0486 00733    iorwf PORTB,F  ;Output data to the port, 8-bit mode
01FC 1505 00734    bsf  PORTA,E   ;Clock the data in
01FD 0000 00735    nop
01FE 1105 00736    bcf  PORTA,E
00737
01FF 30F0 00738    movlw 0xf0
0200 0586 00739    andwf PORTB,F
0201 1486 00740    bsf  PORTB,1   ;Output data to the port, 4-bit mode
0202 1505 00741    bsf  PORTA,E
0203 0000 00742    nop
0204 1105 00743    bcf  PORTA,E
00744
0205 300F 00745    movlw 0x0f
0206 1683 00746    bsf  STATUS,RP0
0207 0486 00747    iorwf TRISB,F
0208 1283 00748    bcf  STATUS,RP0
00749
0209 223E 00750    call BusyXLCD   ;Function Set: 4-bit mode, 2 lines,
020A 302F 00751    movlw 0x2f     ;5x8 dots
020B 221C 00752    call WriteCmdXLCD
00753
020C 223E 00754    call BusyXLCD   ;Display Cntrl: display, cursor off
020D 3008 00755    movlw 0x08
020E 221C 00756    call WriteCmdXLCD
00757
020F 223E 00758    call BusyXLCD   ;Display Cntrl: display & cursor on,
0210 300F 00759    movlw 0x0f     ;blinking on
0211 221C 00760    call WriteCmdXLCD
00761
0212 223E 00762    call BusyXLCD   ;Clear Display
0213 3001 00763    movlw 0x01
0214 221C 00764    call WriteCmdXLCD
00765
0215 223E 00766    call BusyXLCD   ;Shift Cntrl: cursor moves to left
0216 3013 00767    movlw 0x13
0217 221C 00768    call WriteCmdXLCD
00769
0218 223E 00770    call BusyXLCD   ;Set DDRAM address to 0
0219 3080 00771    movlw 0x80
021A 221C 00772    call WriteCmdXLCD
021B 0008 00773    return
00774
```

AN715

```
00775
00776 ;*****
00777 ;WriteCmdXLCD
00778 ; This subroutine writes a command to the LCD display using
00779 ; a 4-bit interface.
00780 ;*****
021C 00781 WriteCmdXLCD
021C 1283 00782 bcf STATUS,RP0
021D 0099 00783 movwf CMD ;Save command in WREG to CMD
021E 30F0 00784 movlw 0xf0 ;Setup up data port for write
021F 1683 00785 bsf STATUS,RP0
0220 0586 00786 andwf TRISB,F
0221 1283 00787 bcf STATUS,RP0
0222 0586 00788 andwf PORTB,F
0223 0819 00789 movf CMD,W ;Write upper 4-bits to data port
0224 00AF 00790 movwf Temp
0225 0EAF 00791 swapf Temp,F
0226 300F 00792 movlw 0x0f
0227 052F 00793 andwf Temp,W
0228 390F 00794 andlw 0x0f
0229 0486 00795 iorwf PORTB,F
022A 1185 00796 bcf PORTA,RW ;Set the control bits for write
022B 1286 00797 bcf PORTB,RS ;and command
022C 0000 00798 nop
022D 1505 00799 bsf PORTA,E ;Clock the upper nibble in
022E 0000 00800 nop
022F 1105 00801 bcf PORTA,E
0230 30F0 00802 movlw 0xf0
0231 0586 00803 andwf PORTB,F
0232 300F 00804 movlw 0x0f
0233 0519 00805 andwf CMD,W ;Output the lower 4-bits to data port
0234 0486 00806 iorwf PORTB,F
0235 0000 00807 nop
0236 1505 00808 bsf PORTA,E ;Clock the lower nibble in
0237 0000 00809 nop
0238 1105 00810 bcf PORTA,E
0239 300F 00811 movlw 0x0f
023A 1683 00812 bsf STATUS,RP0
023B 0486 00813 iorwf TRISB,F
023C 1283 00814 bcf STATUS,RP0
023D 0008 00815 return
00816
00817
00818 ;*****
00819 ;BusyXLCD
00820 ; This subroutine monitors the busy bit from the LCD display
00821 ; It returns when the LCD is no longer busy.
00822 ;*****
023E 00823 BusyXLCD
023E 1283 00824 bcf STATUS,RP0
023F 1585 00825 bsf PORTA,RW ;Set up for a read
0240 1286 00826 bcf PORTB,RS ;Read the busy bit/address
0241 0000 00827 nop
0242 1505 00828 bsf PORTA,E ;Clock the data out
0243 0000 00829 nop
0244 1D86 00830 btfs PORTB,3 ;Read the busy bit
0245 2A4D 00831 goto BNHI
0246 1105 00832 bcf PORTA,E ;Still busy
0247 0000 00833 nop
0248 1505 00834 bsf PORTA,E ;Clock out the lower nibble
```

```

0249 0000 00835    nop
024A 1105 00836    bcf  PORTA,E
024B 1185 00837    bcf  PORTA,RW
024C 2A3E 00838    goto BusyXLCD    ;Try again
024D    00839 BNHI
024D 1105 00840    bcf  PORTA,E    ;LCD not busy
024E 0000 00841    nop
024F 1505 00842    bsf  PORTA,E    ;Clock out the lower nibble
0250 0000 00843    nop
0251 1105 00844    bcf  PORTA,E
0252 1185 00845    bcf  PORTA,RW
0253 0008 00846    return
    00847
    00848
    00849 ;*****
00850 ;WriteDataXLCD
00851 ;   This subroutine writes a byte of data to the LCD display
00852 ;   using the 4-bit interface.
00853 ;*****
0254    00854 WriteDataXLCD
0254 1283 00855    bcf  STATUS,RP0
0255 0099 00856    movwf LDATA    ;Save the data in LDATA
0256 30F0 00857    movlw 0xf0    ;Setup the data port
0257 1683 00858    bsf  STATUS,RP0
0258 0586 00859    andwf TRISB,F
0259 1283 00860    bcf  STATUS,RP0
025A 0586 00861    andwf PORTB,F
025B 0819 00862    movf LDATA,W    ;Write the upper nibble of data
025C 00AF 00863    movwf Temp    ;to the data port
025D 0EAF 00864    swapf Temp,F
025E 300F 00865    movlw 0x0f
025F 052F 00866    andwf Temp,W
0260 390F 00867    andlw 0x0f
0261 0486 00868    iorwf PORTB,F
0262 1686 00869    bsf  PORTB,RS    ;Set control signals for write
0263 1185 00870    bcf  PORTA,RW    ;to data registers
0264 0000 00871    nop
0265 1505 00872    bsf  PORTA,E    ;Clock the upper nibble in
0266 0000 00873    nop
0267 1105 00874    bcf  PORTA,E
0268 30F0 00875    movlw 0xf0
0269 0586 00876    andwf PORTB,F
026A 300F 00877    movlw 0x0f
026B 0519 00878    andwf LDATA,W    ;Write the lower nibble to data port
026C 0486 00879    iorwf PORTB,F
026D 0000 00880    nop
026E 1505 00881    bsf  PORTA,E    ;Clock the lower nibble in
026F 0000 00882    nop
0270 1105 00883    bcf  PORTA,E
0271 300F 00884    movlw 0x0f
0272 1683 00885    bsf  STATUS,RP0
0273 0486 00886    iorwf TRISB,F
0274 1283 00887    bcf  STATUS,RP0
0275 0008 00888    return
    00889
    00890 ;*****
00891 ;DisplayCal
00892 ;   This subroutine displays a message to the LCD display
00893 ;   indicating that a calibration cycle is in progress.
00894 ;*****

```

AN715

```
0276 00895 DisplayCal
0276 223E 00896 call BusyXLCD
0277 3001 00897 movlw 0x01
0278 221C 00898 call WriteCmdXLCD
0279 223E 00899 call BusyXLCD
027A 3043 00900 movlw 'C'
027B 2254 00901 call WriteDataXLCD
027C 223E 00902 call BusyXLCD
027D 3061 00903 movlw 'a'
027E 2254 00904 call WriteDataXLCD
027F 223E 00905 call BusyXLCD
0280 306C 00906 movlw 'l'
0281 2254 00907 call WriteDataXLCD
0282 223E 00908 call BusyXLCD
0283 3069 00909 movlw 'i'
0284 2254 00910 call WriteDataXLCD
0285 223E 00911 call BusyXLCD
0286 3062 00912 movlw 'b'
0287 2254 00913 call WriteDataXLCD
0288 223E 00914 call BusyXLCD
0289 3072 00915 movlw 'r'
028A 2254 00916 call WriteDataXLCD
028B 223E 00917 call BusyXLCD
028C 3061 00918 movlw 'a'
028D 2254 00919 call WriteDataXLCD
028E 223E 00920 call BusyXLCD
028F 3074 00921 movlw 't'
0290 2254 00922 call WriteDataXLCD
0291 0008 00923 return
00924
00925
00926 ;*****
00927 ;DisplayDone
00928 ; This subroutine displays a message to the LCD display
00929 ; indicating that a calibration cycle has completed.
00930 ;*****
0292 00931 DisplayDone
0292 223E 00932 call BusyXLCD
0293 30A8 00933 movlw 0xa8
0294 221C 00934 call WriteCmdXLCD
0295 223E 00935 call BusyXLCD
0296 3044 00936 movlw 'D'
0297 2254 00937 call WriteDataXLCD
0298 223E 00938 call BusyXLCD
0299 306F 00939 movlw 'o'
029A 2254 00940 call WriteDataXLCD
029B 223E 00941 call BusyXLCD
029C 306E 00942 movlw 'n'
029D 2254 00943 call WriteDataXLCD
029E 223E 00944 call BusyXLCD
029F 3065 00945 movlw 'e'
02A0 2254 00946 call WriteDataXLCD
02A1 0008 00947 return
00948
00949
00950
00951 ;=====
00952 ;===== Misc. Routines =====
00953 ;=====
00954 ;*****
```

```

00955 ;Delay_Ms_4MHz
00956 ;   Generic delay routine. Delay length in ms is loaded
00957 ;   into WREG before calling.
00958 ;*****
02A2   00959 Delay_Ms_4MHz
02A2 1283 00960   bcf   STATUS,RP0
02A3 008E 00961   movwf Count1
02A4   00962 DLMS2M1
02A4 307C 00963   movlw 0x7c
02A5 008F 00964   movwf Count2
02A6   00965 DLMS2M2
02A6 0000 00966   nop
02A7 0B8F 00967   decfsz Count2,F
02A8 2AA6 00968   goto  DLMS2M2
02A9 0B8E 00969   decfsz Count1,F
02AA 2AA4 00970   goto  DLMS2M1
02AB 0008 00971   return
00972
00973
00974 ;*****
00975 ;Bin2Ascii
00976 ;   This routine converts a binary number to a 2-digit ASCII
00977 ;   number. The binary number is sent in WREG.
00978 ;*****
02AC   00979 Bin2Ascii
02AC 0198 00980   clrf  Digit1   ;Clear the upper digit
02AD 0097 00981   movwf Digit0   ;Save the binary number
02AE   00982 B2A1
02AE 300A 00983   movlw 0x0a     ;Repeadedly subtract 10 from the
02AF 0217 00984   subwf Digit0,W ;number until the result is less
02B0 1C03 00985   btfss STATUS,C ;then 10
02B1 2AB5 00986   goto  B2A2
02B2 0097 00987   movwf Digit0
02B3 0A98 00988   incf Digit1,F
02B4 2AAE 00989   goto  B2A1
02B5   00990 B2A2
02B5 3030 00991   movlw 0x30     ;Add 0x30 to make the result
02B6 0797 00992   addwf Digit0,F ;ASCII
02B7 0798 00993   addwf Digit1,F
02B8 3400 00994   retlw 0
00995 ;*****
00996
00997
00998 ;=====
00999 ;===== Data EEPROM Routines =====
01000 ;=====
01001 ;*****
01002 ;WriteCal
01003 ;   This subroutine takes 6 bytes starting with address
01004 ;   ZXcalHi and writes them to the internal Data EEPROM.
01005 ;   Calls to WriteEE perform the actual write sequence.
01006 ;*****
02B9   01007 WriteCal
02B9 3006 01008   movlw 0x06     ;Load byte counter with 6
02BA 008E 01009   movwf Count1
02BB 3023 01010   movlw ZXcalHi ;Load the starting address into FSR
02BC 0084 01011   movwf FSR
02BD 01B4 01012   clrf  EADR     ;Start writing data to EE address 0
02BE   01013 WCLoop
02BE 0800 01014   movf  INDF,W   ;Load data

```

AN715

```
02BF 00B5 01015    movwf  EDATA
02C0 22D2 01016    call  WriteEE      ;Call routine to write data
02C1 0AB4 01017    incf  EADR,F       ;Increment EE address
02C2 0A84 01018    incf  FSR,F        ;Increment FSR
02C3 0B8E 01019    decfsz Count1,F   ;Decrement count
02C4 2ABE 01020    goto  WCLoop
02C5 0008 01021    return
    01022
    01023
    01024 ;*****
    01025 ;RestoreCal
    01026 ; This subroutine reads 6 bytes from the Data EE starting
    01027 ; with address 0 and saves them starting with ZXcalHi.
    01028 ; Calls to ReadEE perform the actual read sequence.
    01029 ;*****
02C6 01030 RestoreCal
02C6 3006 01031    movlw  0x06        ;Load byte counter
02C7 008E 01032    movwf  Count1
02C8 3023 01033    movlw  ZXcalHi    ;Load starting address into FSR
02C9 0084 01034    movwf  FSR
02CA 01B4 01035    clrf  EADR        ;Load starting EE address with 0
02CB 01036 RCLoop
02CB 22E4 01037    call  ReadEE      ;Read data from EE
02CC 0080 01038    movwf  INDF       ;Save in register
02CD 0AB4 01039    incf  EADR,F      ;Increment EE address
02CE 0A84 01040    incf  FSR,F       ;Increment FSR
02CF 0B8E 01041    decfsz Count1,F  ;Decrement count
02D0 2ACB 01042    goto  RCLoop
02D1 0008 01043    return
    01044
    01045 ;*****
    01046 ;WriteEE
    01047 ; This is the subroutine to load the address and data into
    01048 ; the special EE access registers and perform the EE write
    01049 ; sequence.
    01050 ;*****
02D2 01051 WriteEE
02D2 1283 01052    bcf  STATUS,RP0
02D3 0834 01053    movf  EADR,W      ;Load EE address
02D4 0089 01054    movwf  EEADR
02D5 0835 01055    movf  EDATA,W    ;Load EE data
02D6 0088 01056    movwf  EEDATA
02D7 1683 01057    bsf  STATUS,RP0
02D8 1208 01058    bcf  EECON1,EEIF
02D9 1508 01059    bsf  EECON1,WREN ;EE write sequence
02DA 3055 01060    movlw  0x55      ;must be performed
02DB 0089 01061    movwf  EECON2    ;in this order
02DC 30AA 01062    movlw  0xaa     ;otherwise write
02DD 0089 01063    movwf  EECON2    ;does not take
02DE 1488 01064    bsf  EECON1,WR   ;place correctly
02DF 01065 eBusy
02DF 1E08 01066    btfss EECON1,EEIF ;Wait for write to complete
02E0 2ADF 01067    goto  eBusy
02E1 1108 01068    bcf  EECON1,WREN ;Disable writes
02E2 1283 01069    bcf  STATUS,RP0
02E3 0008 01070    return
    01071
    01072
    01073 ;*****
    01074 ;ReadEE
```



```

01075 ; This is the subroutine to read from the data EE using the
01076 ; special EE access registers.
01077 ;*****
02E4 01078 ReadEE
02E4 1283 01079 bcf STATUS,RP0
02E5 0834 01080 movf EADR,W ;Load EE address
02E6 0089 01081 movwf EEADR
02E7 1683 01082 bsf STATUS,RP0
02E8 1408 01083 bsf EECON1,RD ;Perform the EE write sequence
02E9 1283 01084 bcf STATUS,RP0
02EA 0808 01085 movf EEDATA,W ;Move data into WREG
02EB 0008 01086 return
01087
01088
01089 end

```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```

0000 : X--XXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0200 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0240 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0280 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
02C0 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX-----
2000 : -----X-----

```

All other memory blocks unused.

Program Memory Words Used: 745

Program Memory Words Free: 279

Errors : 0

Warnings : 0 reported, 0 suppressed



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-786-7200 Fax: 480-786-7277
Technical Support: 480-786-7627
Web Address: <http://www.microchip.com>

Atlanta

Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago

Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

Microchip Technology Inc.
4570 Westgrove Drive, Suite 160
Addison, TX 75248
Tel: 972-818-7423 Fax: 972-818-2924

Dayton

Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit

Microchip Technology Inc.
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles

Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

AMERICAS (continued)

Toronto

Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong

Microchip Asia Pacific
Unit 2101, Tower 2
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

Beijing

Microchip Technology, Beijing
Unit 915, 6 Chaoyangmen Bei Dajie
Dong Erhuan Road, Dongcheng District
New China Hong Kong Manhattan Building
Beijing 100027 PRC
Tel: 86-10-85282100 Fax: 86-10-85282104

India

Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Japan

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222-0033 Japan
Tel: 81-45-471-6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai

Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700 Fax: 86 21-6275-5060

ASIA/PACIFIC (continued)

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C

Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5858 Fax: 44-118 921-5835

Denmark

Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hof 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

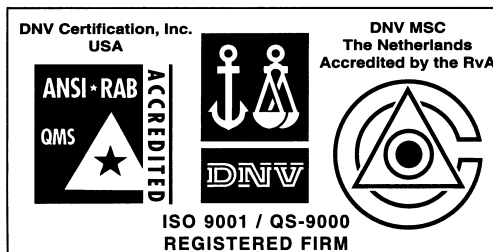
Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

11/15/99



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and water fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOC® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

All rights reserved. © 1999 Microchip Technology Incorporated. Printed in the USA. 11/99 Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.