AN531

Intelligent Remote Positioner (Motor Control)

Author: *Steven Frank*
Vesta Technology Inc.

INTRODUCTION

The excellent cost/performance ratio of the PIC16C5X makes it well suited as a low-cost proportional D.C. actuator controller. This application note depicts a design for a remote intelligent positioning system using a D.C. motor (up to 1/3 hp) run from 12V to 24V. The position accuracy is one in eight bits or 0.4%. The PIC16C5X receives its command and control information via a Microwire® serial bus. However, any serial communication method is applicable.

IMPLEMENTATION

The PIC16C5X based controller receives movement commands from a host, compares them to the actual position, calculates the desired motor drive level and then pulses a full H-bridge (Figure 2). In this way it serves as a remote intelligent positioner, driving the load until it has reached the commanded position. It can be used to control any proportional D.C. actuator (i.e., D.C. motor or proportional valve).

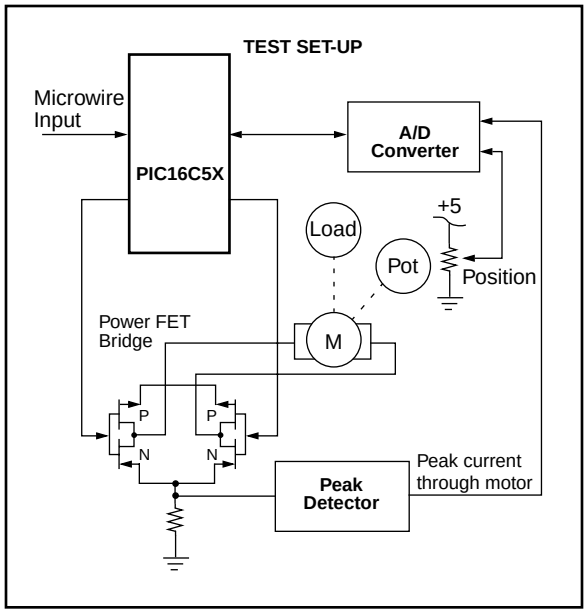
This system is ideally suited for remotely positioned valves and machinery. It can be used with D.C. motors to easily automate manual equipment. Because of the 5-wire serial interface, the positioner can be installed near its power supply and load. The remote intelligent positioner can then be linked to the central control processor by a small diameter easily routed cable. Since the positioner is running its own closed-loop PID algorithm (Figure 3), the host central processor needs only to send position commands and is therefore free to service the user interface, main application software and command multiple remote positioners.

The limit switch inputs provide a safety net which keeps the system from destroying itself in the event that the feedback device is damaged. The optional current sense input can be used to determine if the load has jammed and prevent overheating of the actuator and drive electronics.

The commanded positions are presented to the PIC16C5X via a microwire type protocol at bit-rates of up to 50 Kbs for a 4 MHz part. As currently implemented in this application note, the position request is the only communication. There are several variable locations available which could be used to down-load the loop gain parameters, read positioner information, or set a current limit. The host that is sending the position request must set the chip select low, and wait for the PIC16C5X to raise the "busy" (DO) line high. At this point, eight data bits can be clocked into the PIC16C5X. The requested position is sent most significant bit first and can be any 8-bit value. Values 1 through 255 represent valid positions with 0 being reserved for drive disable.

The PIC16C5X acquires its data by way of a Microwire® A/D converter. This part was chosen for low cost while providing adequate performance. In Figure 1 the second channel of the A/D converter is shown connected to a peak current detector. If the user desires, the PIC16C5X could monitor and protect the motor from overcurrent conditions by monitoring the second channel.

FIGURE 1: BLOCK DIAGRAM



Microwire is a registered trademark of National Semiconductor Corporation.

The H-bridge power amplifier will deliver 10 or more amps at upto 24V when properly heat-sinked. It is wired for a modified 4-quadrant mode of operation. One leg of the bridge is used to control direction and the other leg pulses the low FET and the high FET alternately to generate the desired duty-cycle. In this way the system will operate well to produce a desired "speed" without the use of a separate speed control loop. This allows use of the PIC16C5X to control the PID algorithm for position directly while having reasonable speed control. The capacitance at the gates of the FETs combined with the impedance of the drive circuits provides for turn-off of the upper FET before the lower FET turns on... an important criteria.

The PID algorithm itself is where most of the meat of this application note is located so let's look at it more closely. The algorithm is formed by summing the contribution of three basic components. The first calculation is the error upon which the other terms are based.

The **error** is the requested position minus the actual position. It is a signed number whose magnitude can be 255. In order not to lose resolution, the error is stored as an 8-bit magnitude with the sign stored separately in the FLAGS register under ER_SGN. This allows us to resolve a full signed 8-bit error with 8-bit math.

The **proportional term** is merely the algebraic difference of the requested position minus the actual position. It is scaled by a gain term (K_P) called the "proportional gain". The sign (+,-) of this term is important for it tells the system which direction it must drive to correct the error. The proportional term is limited to ±100. Increasing the proportional gain term will improve the dynamic and static accuracy of the system. Increasing it too much will cause oscillations.

The next term that gets calculated is the **integral term (K_I)**. This term is traditionally formed by integrating the error over time. In this application it is done by integrating the K_I term over time. When the error is zero, no integration is performed. This is a more practical way to handle a potentially large number in 8-bit math. By increasing the K_I term the D.C. or static gain of the system is improved. Increasing the integral gain too much can lead to low frequency oscillations.

The **differential term** (K_D) is a stabilizing term that helps keep the integral and proportional terms from overdriving the system through the desired position and thus creating oscillations. As you use more proportional and integral gain you will need more differential gain as well. The differential gain is calculated by looking at the rate of change of the positional error with respect to time. It is actually formed as "delta error/delta time" with the delta time being a program cycle.

The three terms (proportional, integral, differential) are summed algebraically and scaled to produce a percentage speed request between 0 and 100%. The sign of the sum is used to control the direction of the H-bridge. The loop calculations run approximately 20 times per second on a 4 MHz part. This yields sufficient gain-bandwidth for most positioning applications. If higher system performance is desired, the number of pulses can be reduced to 20 and a 16 MHz PIC16C5X can be used. Your loop gains (K_P, K_I, K_D) will have to be recalculated, but the system sample rate will be increased to 400 Hz. This should be sufficient to control a system that has a response time of 20 milliseconds or more.

The key to using the PIC16C5X series parts for PID control and PWM generation is to separate the two into separate tasks. There simply is not the hardware support or the processing speed to accurately do both concurrently. It is fortunate therefore that it is not necessary to do both concurrently. Most systems can be stabilized with a much lower information update rate than the PWM frequency. This supports the approach of calculating the desired percentage, outputting the PWM for a period of time and then recalculating the new desired percentage. Using this technique, the inexpensive PIC16C5X can implement PID control, PWM generation, and will still have processing time left over for monitor or communication functions.

About the Author:

Steven Frank has been designing analog and digital control systems for ten years. His background is in medical and consumer electronics. He has received numerous patents in control systems and instrumentation. At Vesta Technology Inc., Mr. Frank works with a number of engineers on custom embedded control systems designs. Vesta Technology Inc. is a provider of embedded control systems from an array of standard products and designs. Vesta offers custom design services and handles projects from concept to manufacturing.

FIGURE 2: PROGRAM FLOWCHART

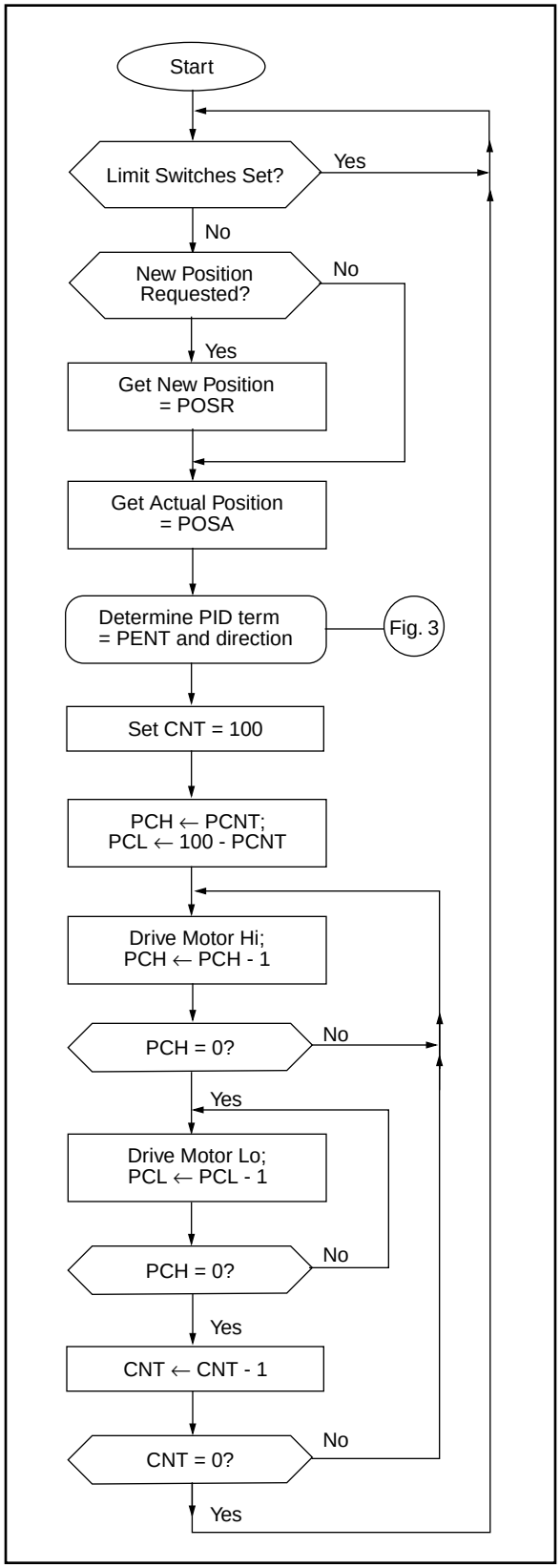


FIGURE 3: PID ALGORITHM FLOWCHART

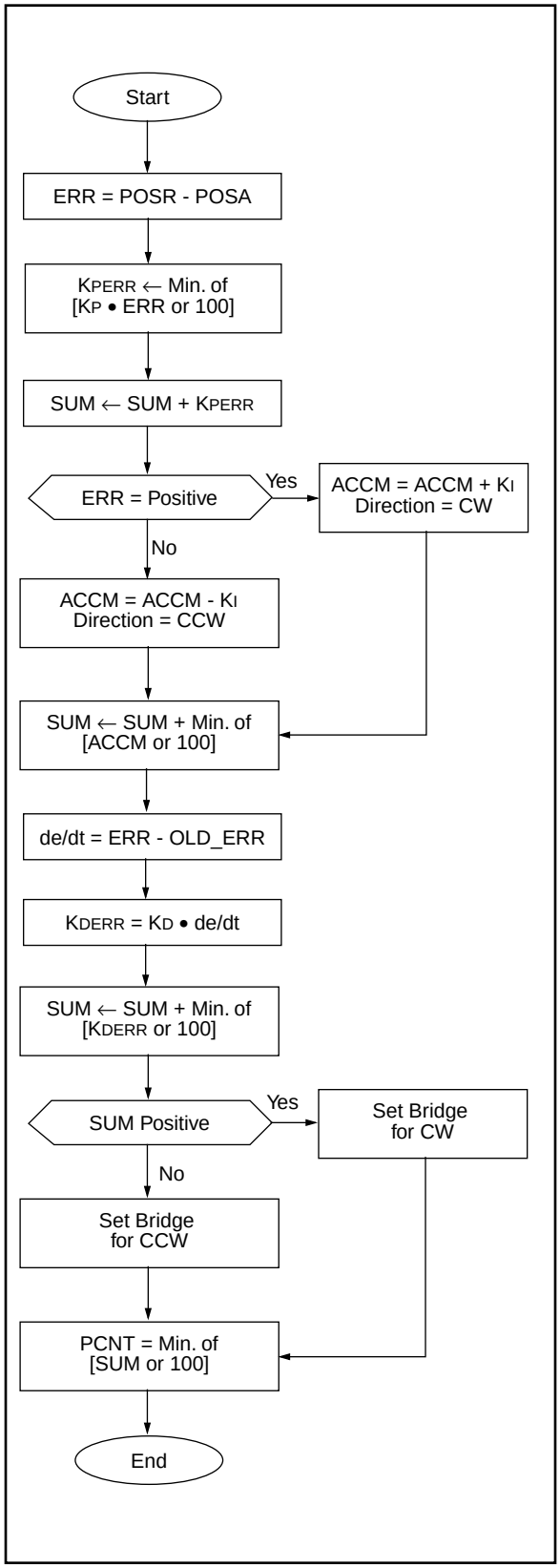
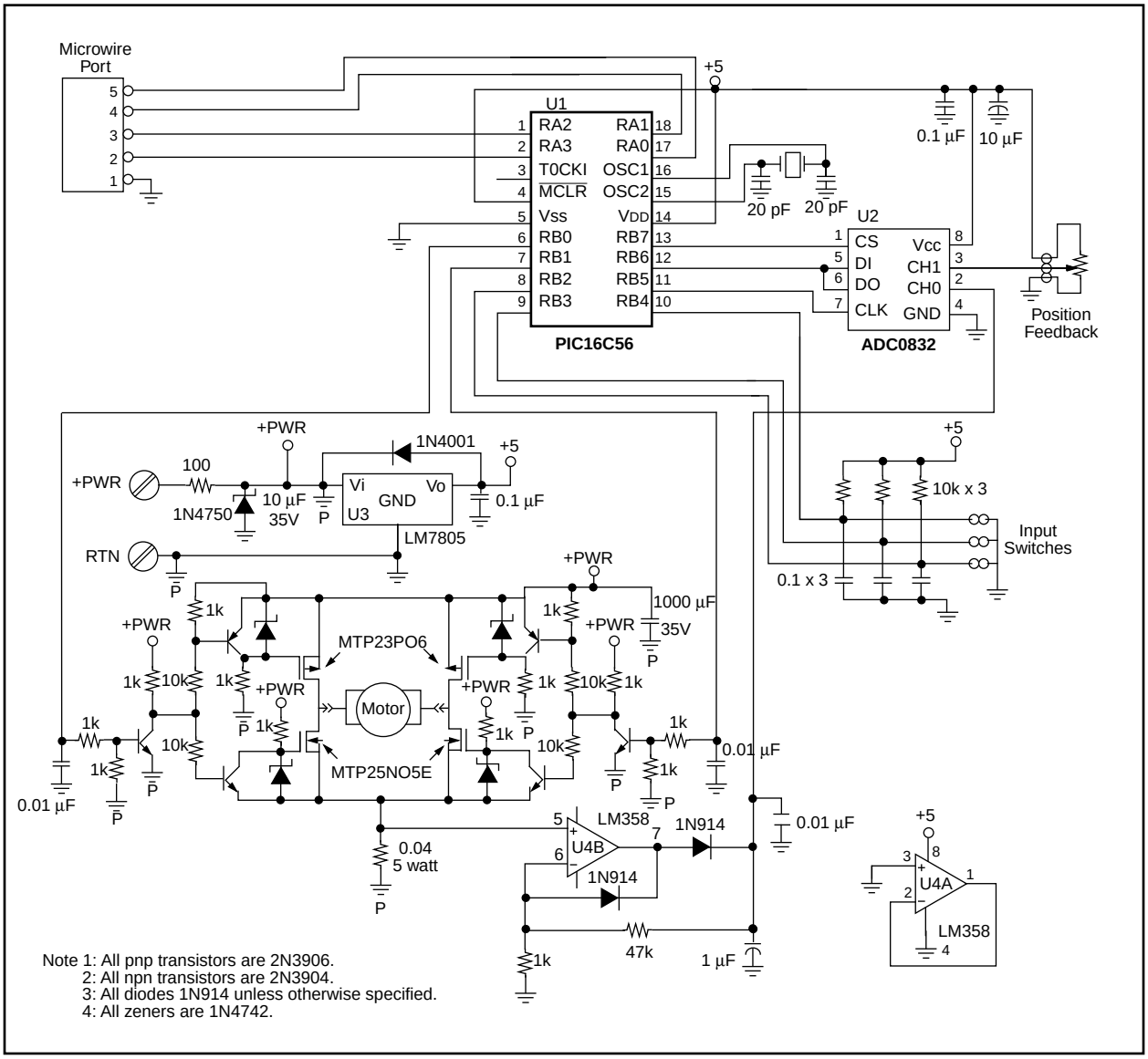


FIGURE 4: SCHEMATIC



Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX A: MWPOS.ASM

```
MPASM 01.40 Released           MWPOS.ASM   1-16-1997  13:16:02         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001    TITLE " MicroWire Positioner "
                     00002 ;
                     00003 ;        mw8pos.asm
                     00004 ;
                     00005         LIST P=16C56
                     00006 ;
                     00007 ;*****************************************************************
                     00008 ;
                     00009 ;       Program:          MWPOS.ASM
                     00010 ;       Revision Date:    1/10/92 srf   REV. A
                     00011 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                     00012 ;
                     00013 ;*******************************************************************************
                     00014 ;
                     00015 ;REGISTER EQUATES
                     00016 ;
  00000000           00017 PNTR    EQU     00H             ; CONTENTS OF POINTER
  00000019           00018 FLAGS   EQU     19H             ; USE THIS VARIABLE LOCATION AS FLAGS
                     00019                                 ; 0 BIT IS SIGN OF ERROR 1 IS NEGATIVE
                     00020                                 ; 1 BIT IS SIGN OF ERROR ACCUMULATOR
                     00021                                 ; 2 BIT IS SIGN OF THE DE/DT TERM
                     00022                                 ; 3 BIT IS DIRECTION 0 IS CW
                     00023                                 ; 4 BIT IS SIGN OF THE OLD ERROR
  00000003           00024 STATUS  EQU     03H
  00000001           00025 F       EQU     1
  00000000           00026 W       EQU     0
  00000003           00027 SWR     EQU     03H             ; STATUS WORD REGISTER
                     00028                                 ; 0 = CARRY
                     00029                                 ; 1 = DC
                     00030                                 ; 2 = Z, SET IF RESULT IS ZERO
  00000004           00031 FSR     EQU     04H             ; FILE SELECT REGISTER
  00000005           00032 PORTA   EQU     05H             ; I/O REG (A0-A3), (A4-A7 DEF=0)
  00000006           00033 PORTB   EQU     06H             ; I/O REGISTER(B0-B7)
  00000007           00034 HI      EQU     07H             ; NUMBER OF HIGH MICROSECONDS
  00000008           00035 LO      EQU     08H             ; NUMBER OF LOW MICROSECONDS
  00000009           00036 PCNT    EQU     09H             ; PERCENT DUTYCYCLE REQUEST
  0000000A           00037 HI_T    EQU     0AH             ; COUNTER FOR USECONDS LEFT/PULSE HI
  0000000B           00038 LO_T    EQU     0BH             ; COUNTER FOR USECONDS LEFT/PULSE LO
  0000000C           00039 ERR1    EQU     0CH             ; HOLDER FOR THE POSITIONAL ERROR
                     00040                                 ; THIS IS AN 8 BIT MAGNITUDE WITH THE SIGN
                     00041                                 ; KEPT IN THE FLAG REGISTER (9BIT SIGNED)
  0000000D           00042 SUMLO   EQU     0DH             ; PROGRESSIVE SUM OF THE PID TERMS
  0000000E           00043 ACCUM   EQU     0EH             ; ERROR ACCUMULATOR
  0000000F           00044 ERR_O   EQU     0FH             ; ERROR HISTORY USED FOR de/dt
                     00045                                 ; THIS IS AN 8 BIT MAGNITUDE WITH THE SIGN
                     00046                                 ; KEPT IN THE FLAG REGISTER (9BIT SIGNED)
  00000010           00047 POSR    EQU     10H             ; POSITIONAL REQUEST
  00000011           00048 POSA    EQU     11H             ; ACTUAL POSITION
  00000012           00049 CYCLES  EQU     12H             ; COUNTER FOR CYCLES OUT
                     00050
  00000013           00051 mulcnd  equ     13H             ; 8 bit multiplicand
  00000013           00052 ACCaLO  EQU     13H             ; same location used for the add routine
  00000014           00053 mulplr  equ     14H             ; 8 bit multiplier
  00000014           00054 ACCbLO  EQU     14H             ; same location used for the add routine
  00000015           00055 H_byte  equ     15H             ; High byte of the 16 bit result
  00000015           00056 ACCaHI  EQU     15H             ; same location used for the add routine
  00000016           00057 L_byte  equ     16H             ; Low byte of the 16 bit result
  00000016           00058 ACCbHI  EQU     16H             ; same location used for the add routine
  00000017           00059 count   equ     17H             ; loop counter
  00000018           00060 SUMHI   EQU     18H             ; HIGH BYTE OF THE LOOP SUM
                     00061
                     00062
                     00063 ; PORT ASSIGNMENTS AND CONSTANTS
                     00064
  00000000           00065 PWMCW   EQU     0               ; CLOCKWISE PWM OUTPUT BIT
  00000001           00066 PWMCCW  EQU     1               ; COUNTERCLOCKWISE PWM OUTPUT BIT
  00000000           00067 CARRY   EQU     0               ; CARRY BIT IN THE STATUS REGISTER
  00000002           00068 Z       EQU     2               ; THE ZERO BIT OF THE STATUS REGISTER
  00000001           00069 Same    equ     1               ;
  00000000           00070 ER_SGN  EQU     0               ; SIGN BIT FOR THE ERROR IN FLAG REGISTER
```

```
  00000001          00071 AC_SGN  EQU     1               ; SIGN BIT FOR THE ERROR ACCUMULATOR
  00000002          00072 DE_SGN  EQU     2               ; SIGN BIT FOR DE/DT
  00000004          00073 OER_SGN EQU     4               ; SIGN BIT FOR THE OLD ERROR
  00000030          00074 KP      EQU     30              ; PROPORTIONAL GAIN
  00000002          00075 KI      EQU     2               ; INTEGRAL GAIN
  00000020          00076 KD      EQU     20              ; DIFFERENTIAL GAIN
  00000003          00077 DIR     EQU     3               ; THE DIRECTION FLAG
  00000007          00078 CSN     EQU     7               ; CHIP SELECT NOT ON A/D
  00000006          00079 BV      EQU     6               ; DATA LINE FOR THE A/D
  00000005          00080 CK      EQU     5               ; CLOCK LINE FOR THE A/D
  00000002          00081 MWDO    EQU     2               ; MICROWIRE DATA OUT FROM POSITIONER
  00000001          00082 MWDI    EQU     1               ; MICROWIRE DATA IN TO POSITIONER
  00000000          00083 MWCS    EQU     0               ; MICROWIRE CHIP SELECT TO POSITIONER
  00000003          00084 MWCK    EQU     3               ; MICROWIRE CLOCK IN TO POSITIONER
                    00085
                    00086
                    00087 ;***** MACROS ********************************************
                    00088 ;
                    00089 CLKUP   MACRO                   ; clock up macro for the microwire
                    00090         BSF     PORTB,CK        ; data acquisition from the a/d
                    00091         NOP
                    00092         ENDM
                    00093
                    00094 CLKDN   MACRO                   ; clock down macro for the microwire
                    00095         BCF     PORTB,CK        ; data acquisition from the a/d
                    00096         NOP
                    00097         ENDM
                    00098
                    00099 GET_BIT MACRO                   ; ** FOR RECEIVING A/D DATA **
                    00100         BCF     SWR,CARRY
                    00101         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
                    00102         BTFSC   PORTB,BV        ; LOOK AT DATA COMING IN
                    00103         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
                    00104         RLF     POSA, F         ; ROTATE THE W REG LEFT
                    00105         BCF     PORTB,CK        ; SET THE CLOCK LOW
                    00106         NOP                     ; DELAY
                    00107         ENDM
                    00108
0000 0B88           00109         GOTO    CLRREG
                    00110
                    00111 ;***** MATH ROUTINES ************************************
                    00112 ;
                    00113
                    00114 ; **** 8 BIT MULTIPLY ********
                    00115 ; ****************************         Begin Multiplier Routine
0001 0075           00116 mpy_S   clrf    H_byte
0002 0076           00117         clrf    L_byte
0003 0C08           00118         movlw   8
0004 0037           00119         movwf   count
0005 0213           00120         movf    mulcnd,W
0006 0403           00121         bcf     STATUS,CARRY    ; Clear the carry bit in the status Reg.
0007 0334           00122 loop    rrf     mulplr, F
0008 0603           00123         btfsc   STATUS,CARRY
0009 01F5           00124         addwf   H_byte,Same
000A 0335           00125         rrf     H_byte,Same
000B 0336           00126         rrf     L_byte,Same
000C 02F7           00127         decfsz  count, F
000D 0A07           00128         goto    loop
000E 0800           00129         retlw   0
                    00130
                    00131 ; ****************************
                    00132 ; DOUBLE PRECISION ADD AND SUBTRACT ( ACCb-ACCa->ACCb )
                    00133
000F 0917           00134 D_sub   call    neg_A           ; At first negate ACCa, then add
                    00135
                    00136 ;***************
                    00137 ; Double Precision Addition ( ACCb+ACCa->ACCb )
                    00138
0010 0213           00139 D_add   movf    ACCaLO,W
0011 01F4           00140         addwf   ACCbLO, F       ; add lsb
0012 0603           00141         btfsc   STATUS,CARRY    ; add in carry
0013 02B6           00142         incf    ACCbHI, F
0014 0215           00143         movf    ACCaHI,W
0015 01F6           00144         addwf   ACCbHI, F       ; add msb
0016 0800           00145         retlw   00
                    00146 ;
                    00147 ;
0017 0273           00148 neg_A   comf    ACCaLO, F       ; negate ACCa
0018 02B3           00149         incf    ACCaLO, F
0019 0643           00150         btfsc   STATUS,Z
001A 00F5           00151         decf    ACCaHI, F
001B 0275           00152         comf    ACCaHI, F
001C 0800           00153         retlw   00
```

```
                  00154
                  00155 ; ******************************************
                  00156 ; divide by 16 and limit to 100 Decimal
                  00157
                  00158 SHIFT   MACRO
                  00159         BCF     SWR,CARRY
                  00160         RRF     L_byte, F
                  00161         BCF     SWR,CARRY
                  00162         RRF     H_byte, F
                  00163         BTFSC   SWR,CARRY
                  00164         BSF     L_byte,7
                  00165         ENDM
                  00166
001D              00167 DIV_LMT
                  00168         SHIFT
001D 0403             M         BCF     SWR,CARRY
001E 0336             M         RRF     L_byte, F
001F 0403             M         BCF     SWR,CARRY
0020 0335             M         RRF     H_byte, F
0021 0603             M         BTFSC   SWR,CARRY
0022 05F6             M         BSF     L_byte,7
                  00169         SHIFT
0023 0403             M         BCF     SWR,CARRY
0024 0336             M         RRF     L_byte, F
0025 0403             M         BCF     SWR,CARRY
0026 0335             M         RRF     H_byte, F
0027 0603             M         BTFSC   SWR,CARRY
0028 05F6             M         BSF     L_byte,7
                  00170         SHIFT
0029 0403             M         BCF     SWR,CARRY
002A 0336             M         RRF     L_byte, F
002B 0403             M         BCF     SWR,CARRY
002C 0335             M         RRF     H_byte, F
002D 0603             M         BTFSC   SWR,CARRY
002E 05F6             M         BSF     L_byte,7
                  00171         SHIFT
002F 0403             M         BCF     SWR,CARRY
0030 0336             M         RRF     L_byte, F
0031 0403             M         BCF     SWR,CARRY
0032 0335             M         RRF     H_byte, F
0033 0603             M         BTFSC   SWR,CARRY
0034 05F6             M         BSF     L_byte,7
                  00172
0035              00173 LMT100
0035 0C01         00174         MOVLW   1H              ; SUBTRACT 1 FROM THE HIGH BYTE TO SEE
0036 0095         00175         SUBWF   H_byte,0        ; IF THERE IS ANYTHING THERE, IF NOT,
0037 0703         00176         BTFSS   SWR,CARRY       ; THEN LEAVE THE LOW BYTE ALONE
0038 0A3C         00177         GOTO    L8_E            ; OTHERWISE GIVE THE LOW BYTE A FULL
0039 0C64         00178         MOVLW   64H             ; COUNT AND IT WILL HAVE BEEN LIMITED
003A 0036         00179         MOVWF   L_byte          ; TO 100
003B 0A42         00180         GOTO    LMT_EXIT
003C              00181 L8_E
003C 0C64         00182         MOVLW   64H             ; LIMIT THE MAGNITUDE OF THE VALUE TO
003D 0096         00183         SUBWF   L_byte,0        ; 100 DECIMAL
003E 0703         00184         BTFSS   SWR,CARRY
003F 0A42         00185         GOTO    LMT_EXIT
0040 0C64         00186         MOVLW   64H
0041 0036         00187         MOVWF   L_byte
0042              00188 LMT_EXIT
0042 0800         00189         RETLW   00
                  00190 ;
                  00191 ;THE ROUTINE CALCTIMES DOES THE FOLLOWING: PCNT = DUTY CYCLE IN %
                  00192 ; 100 - PCNT --> LO  AND PCNT --> HI. ZERO VALUES IN EITHER LO OR HI
                  00193 ;ARE FORCED TO 1.
0043              00194 CALCTIMES
0043 0209         00195         MOVF    PCNT,W          ; PUT REQUESTED % INTO W REGISTER
0044 0027         00196         MOVWF   HI              ; COPY ON MICROSECONDS IN TO HI TIME
0045 0C64         00197         MOVLW   64H
0046 0028         00198         MOVWF   LO
0047 0209         00199         MOVF    PCNT,0
0048 00A8         00200         SUBWF   LO,1            ; LEAVE 100-HI TIME IN LO TIME
0049 0207         00201         MOVF    HI,0            ; INSPECT THE HIGH TIME
004A 0643         00202         BTFSC   SWR,2           ; IF ITS IS ZERO
004B 02A7         00203         INCF    HI,1            ; INCREMENT IT
004C 0208         00204         MOVF    LO,0            ; INSPECT THE LO TIME
004D 0643         00205         BTFSC   SWR,2           ; IF ITS ZERO
004E 02A8         00206         INCF    LO,1            ; INCREMENT IT
004F 0800         00207         RETLW   00
                  00208
                  00209
                  00210 ;*****************************************************************
0050              00211 BEGIN
0050 0000         00212         NOP             ; STUBBED BEGINNING
```

```
                 00213
                 00214
                 00215 ;****CHECKING THE LIMIT SWITCHES AND CHECKING FOR MW**************
                 00216 ; This will check the switch inputs for closure and will terminate
                 00217 ; pulsing if one is closed.  It doesn't distinguish between the switches
                 00218 ; so they are not dedicated to cw end and ccw end.
                 00219
0051             00220 SW_TRAP
0051 0004        00221         CLRWDT
0052 0746        00222         BTFSS   PORTB,2         ; THIS WILL TEST ALL THREE OF THE
0053 0A51        00223         GOTO    SW_TRAP         ; SWITCH INPUTS.  IF ANY ONE IS
0054 0766        00224         BTFSS   PORTB,3         ; SET THEN EXECUTION OF THE CODE
0055 0A51        00225         GOTO    SW_TRAP         ; WILL BE LIMITED TO LOOKING FOR
0056 0786        00226         BTFSS   PORTB,4         ; IT TO BE CLEARED
0057 0A51        00227         GOTO    SW_TRAP
                 00228
                 00229
                 00230 ;****RECEIVING THE POSITIONAL REQUEST*****************************
                 00231 ; The host system that wishes to send positional requests to the positioner
                 00232 ; servo makes its desire known by setting the chip select to the positioner
                 00233 ; low.  It then monitors the busy (Data Out) line from the positioner.  When
                 00234 ; the positioner sets the busy line high, the host may begin sending its 8 bit
                 00235 ; request.  The data bits should be valid on the rising edge of the clock.
                 00236 ; After 8 bits have been received by the positioner it will begin operation
                 00237 ; to send the system to the received position.  It can be interrupted at any
                 00238 ; point during the positioning process by the host sending a new command.  The
                 00239 ; opportunity to update the command is issued every 100 pwm pulses (every 50
                 00240 ; milliseconds).
                 00241 ; If the host sends a zero positional command the positioner will stop the
                 00242 ; system and remain inactive.
                 00243 ; If the host does not successfully complete a microwire transmission of 8
                 00244 ; data bits the watchdog timer will trip and reset the system to an inactive
                 00245 ; "stopped" state.
                 00246
                 00247
0058             00248 REC_MW
0058 0C0B        00249         MOVLW   0BH             ; RESET THE PORT FOR THREE INPUTS
0059 0005        00250         TRIS    PORTA           ; AND ONE OUTPUT
005A 0445        00251         BCF     PORTA,MWDO      ; SET THE DATA OUT LOW FOR BUSY
005B 0C20        00252         MOVLW   20H
005C 0037        00253         MOVWF   count
005D             00254 WATCH_CS
005D 0705        00255         BTFSS   PORTA,MWCS      ; CHECK FOR INCOMING REQUESTS
005E 0A62        00256         GOTO    REC_CMD         ; RECEIVE A NEW POSITION REQUEST
005F 02F7        00257         DECFSZ  count,1
0060 0A5D        00258         GOTO    WATCH_CS
0061 0A71        00259         GOTO    REC_EXIT        ; NO REQUEST WAS MADE IN THE TIME ALLOTED
0062             00260 REC_CMD
0062 0545        00261         BSF     PORTA,MWDO      ; SET THE DATA OUT HIGH FOR "OK TO SEND"
0063 0C08        00262         MOVLW   8H              ; SET TO RECEIVE 8 BITS
0064 0037        00263         MOVWF   count
0065             00264 WAIT_UP
0065 0765        00265         BTFSS   PORTA,MWCK      ; WAIT FOR A RISING EDGE
0066 0A65        00266         GOTO    WAIT_UP
0067 0403        00267         BCF     SWR,CARRY       ; RESET THE CARRY TO A DEFAULT ZERO
0068 0625        00268         BTFSC   PORTA,MWDI      ; READ THE DATA IN
0069 0503        00269         BSF     SWR,CARRY       ; SET THE CARRY FOR A ONE
006A 0370        00270         RLF     POSR,1          ; ROTATE THE BIT INTO THE POSITION REQ.
006B 02F7        00271         DECFSZ  count,1         ; DECREMENT THE BIT COUNTER
006C 0A6E        00272         GOTO    WAIT_DN         ; WAIT FOR THE FALLING EDGE
006D 0A71        00273         GOTO    REC_EXIT        ; LAST BIT RECEIVED
006E             00274 WAIT_DN
006E 0665        00275         BTFSC   PORTA,MWCK      ; CHECK THE INCOMING CLOCK
006F 0A6E        00276         GOTO    WAIT_DN         ; IF IT IS STILL HIGH WAIT FOR IT TO GO LOW
0070 0A65        00277         GOTO    WAIT_UP         ; IF IT GOES LOW GO BACK TO RECEIVE NEXT BIT
                 00278
0071             00279 REC_EXIT
0071 0445        00280         BCF     PORTA,MWDO      ; SET THE BUSY FLAG
                 00281
                 00282
                 00283 ;********** CHECK FOR THE DISABLE REQUEST ************************
                 00284 ; Position 0 is considered a request to not drive the system.  In this way
                 00285 ; the positioner will come up from a reset in a safe state and will not
                 00286 ; try to move the system to some arbitrary location.
                 00287
0072             00288 MOVE
0072 0210        00289         MOVF    POSR,W          ; CHECK THE REQUESTED POSTION
0073 0643        00290         BTFSC   SWR,Z           ; IF IT IS ZERO THEN WAIT FOR A NON-ZERO
0074 0A50        00291         GOTO    BEGIN           ; REQUEST BY BRANCHING BACK TO THE BEGINNING
                 00292
                 00293 ;****READING THE A/D VALUES**************************************
                 00294 ;
                 00295 ; Read the positional a/d channel (1) and store the value in the actual
```

```
                00296 ; position variable (POSA).
                00297 ; This is written in line to minimize the use of variables
                00298
0075            00299 READ_POS
0075 0071       00300         CLRF    POSA            ; CLEAN THE POSITION ACTUAL HOLDER
0076 04E6       00301         BCF     PORTB,CSN       ; SET THE CHIP SELECT LOW TO A/D
0077 0C1C       00302         MOVLW   1CH             ; SET THE DATA LINE TO OUTPUT
0078 0006       00303         TRIS    PORTB           ; FOR SENDING SET-UP BITS
0079 05C6       00304         BSF     PORTB,BV        ; SET FOR “START” BIT
007A 0000       00305         NOP
                00306         CLKUP                   ; CLOCK IN THE START BIT
007B 05A6           M         BSF     PORTB,CK        ; data acquisition from the a/d
007C 0000           M         NOP
                00307         CLKDN                   ; “
007D 04A6           M         BCF     PORTB,CK        ; data acquisition from the a/d
007E 0000           M         NOP
                00308         CLKUP                   ; CLOCK IN SINGLE-ENDED
007F 05A6           M         BSF     PORTB,CK        ; data acquisition from the a/d
0080 0000           M         NOP
                00309         CLKDN                   ; “
0081 04A6           M         BCF     PORTB,CK        ; data acquisition from the a/d
0082 0000           M         NOP
                00310         CLKUP                   ; CLOCK IN CHANNEL 1
0083 05A6           M         BSF     PORTB,CK        ; data acquisition from the a/d
0084 0000           M         NOP
                00311         CLKDN                   ; TO THE MUX
0085 04A6           M         BCF     PORTB,CK        ; data acquisition from the a/d
0086 0000           M         NOP
0087 0C5C       00312         MOVLW   5CH             ; SET THE DATA LINE TO INPUT
0088 0006       00313         TRIS    PORTB           ; TO RECEIVE DATA BITS FROM A/D
                00314         CLKUP                   ; CLOCK UP TO LET MUX SETTLE
0089 05A6           M         BSF     PORTB,CK        ; data acquisition from the a/d
008A 0000           M         NOP
                00315         CLKDN                   ; CLOCK DN TO LET MUX SETTLE
008B 04A6           M         BCF     PORTB,CK        ; data acquisition from the a/d
008C 0000           M         NOP
                00316         GET_BIT                 ; GET BIT 7
008D 0403           M         BCF     SWR,CARRY
008E 05A6           M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
008F 06C6           M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
0090 0503           M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
0091 0371           M         RLF     POSA, F         ; ROTATE THE W REG LEFT
0092 04A6           M         BCF     PORTB,CK        ; SET THE CLOCK LOW
0093 0000           M         NOP                     ; DELAY
                00317         GET_BIT                 ; BIT 6
0094 0403           M         BCF     SWR,CARRY
0095 05A6           M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
0096 06C6           M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
0097 0503           M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
0098 0371           M         RLF     POSA, F         ; ROTATE THE W REG LEFT
0099 04A6           M         BCF     PORTB,CK        ; SET THE CLOCK LOW
009A 0000           M         NOP                     ; DELAY
                00318         GET_BIT                 ; BIT 5
009B 0403           M         BCF     SWR,CARRY
009C 05A6           M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
009D 06C6           M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
009E 0503           M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
009F 0371           M         RLF     POSA, F         ; ROTATE THE W REG LEFT
00A0 04A6           M         BCF     PORTB,CK        ; SET THE CLOCK LOW
00A1 0000           M         NOP                     ; DELAY
                00319         GET_BIT                 ; BIT 4
00A2 0403           M         BCF     SWR,CARRY
00A3 05A6           M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00A4 06C6           M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00A5 0503           M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00A6 0371           M         RLF     POSA, F         ; ROTATE THE W REG LEFT
00A7 04A6           M         BCF     PORTB,CK        ; SET THE CLOCK LOW
00A8 0000           M         NOP                     ; DELAY
                00320         GET_BIT                 ; BIT 3
00A9 0403           M         BCF     SWR,CARRY
00AA 05A6           M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00AB 06C6           M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00AC 0503           M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00AD 0371           M         RLF     POSA, F         ; ROTATE THE W REG LEFT
00AE 04A6           M         BCF     PORTB,CK        ; SET THE CLOCK LOW
00AF 0000           M         NOP                     ; DELAY
                00321         GET_BIT                 ; BIT 2
00B0 0403           M         BCF     SWR,CARRY
00B1 05A6           M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00B2 06C6           M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00B3 0503           M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00B4 0371           M         RLF     POSA, F         ; ROTATE THE W REG LEFT
00B5 04A6           M         BCF     PORTB,CK        ; SET THE CLOCK LOW
```

```
00B6 0000              M         NOP                     ; DELAY
                     00322       GET_BIT                 ; BIT 1
00B7 0403              M         BCF     SWR,CARRY
00B8 05A6              M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00B9 06C6              M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00BA 0503              M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00BB 0371              M         RLF     POSA, F         ; ROTATE THE W REG LEFT
00BC 04A6              M         BCF     PORTB,CK        ; SET THE CLOCK LOW
00BD 0000              M         NOP                     ; DELAY
                     00323       GET_BIT                 ; BIT 0
00BE 0403              M         BCF     SWR,CARRY
00BF 05A6              M         BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00C0 06C6              M         BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00C1 0503              M         BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00C2 0371              M         RLF     POSA, F         ; ROTATE THE W REG LEFT
00C3 04A6              M         BCF     PORTB,CK        ; SET THE CLOCK LOW
00C4 0000              M         NOP                     ; DELAY
00C5 05E6            00324       BSF     PORTB,CSN       ; DESELECT THE CHIP
                     00325
                     00326
                     00327 ;****************** CALCULATING THE PID TERMS **********************
                     00328
                     00329 ;****CALCULATE THE ERROR*******
                     00330 ; The error is very simply the signed difference between where the
                     00331 ; system is and where it is supposed to be at a particular instant
                     00332 ; in time.  It is formed by subtracting the actual position from the
                     00333 ; requested position (Position requested - Position actual).  This
                     00334 ; difference is then used to determine the proportional,integral and
                     00335 ; differential term contributions to the output.
                     00336
00C6                 00337 C_ERR
00C6 0211            00338         MOVF    POSA,0          ; LOAD THE ACTUAL POSITION INTO W
00C7 0090            00339         SUBWF   POSR,0          ; SUBTRACT IT FROM THE REQUESTED POSITION
00C8 0603            00340         BTFSC   SWR,CARRY       ; CHECK THE CARRY BIT TO DETERMINE THE SIGN
00C9 0ACB            00341         GOTO    PLS_ER          ; ITS POSITIVE(POSR>POSA)
00CA 0ACE            00342         GOTO    MNS_ER          ; ITS NEGATIVE  (POSA>POSR)
                     00343
00CB                 00344 PLS_ER
00CB 002C            00345         MOVWF   ERR1            ; SAVE THE DIFFERENCE IN “ERROR”
00CC 0419            00346         BCF     FLAGS,ER_SGN    ; SET THE SIGN FLAG TO INDICATE POSITIVE
00CD 0AD2            00347         GOTO    CE_EXIT
                     00348
00CE                 00349 MNS_ER
00CE 0210            00350         MOVF    POSR,0          ; RE-DO THE SUBTRACTION
00CF 0091            00351         SUBWF   POSA,0          ; ACTUAL - REQUESTED
00D0 002C            00352         MOVWF   ERR1            ; STORE THE DIFFERENCE IN “ERROR”
00D1 0519            00353         BSF     FLAGS,ER_SGN    ; SET THE SIGN FLAG FOR NEGATIVE
                     00354
00D2                 00355 CE_EXIT
00D2 006D            00356         CLRF    SUMLO           ; CLEAN OLD VALUES OUT TO PREPARE
00D3 0078            00357         CLRF    SUMHI           ; FOR THIS CYCLES SUMMATION
                     00358
                     00359 ;****CALCULATE THE PROPORTIONAL TERM******
                     00360 ; The proportional term is the error times the proportional gain term.
                     00361 ; This term simply gives you more output drive the farther away you are
                     00362 ; from where you want to be (error)*Kp.
                     00363 ; The proportional gain term is a signed term between -100 and 100  The
                     00364 ; more proportional gain you have the lower your system following error
                     00365 ; will be.  The higher your proportional gain, the more integral and
                     00366 ; differential term gains you will have to add to make the system stable.
                     00367 ; The sum is being carried as a 16 bit signed value.
                     00368
00D4                 00369 C_PROP
00D4 020C            00370         MOVF    ERR1,0          ; LOAD THE ERROR TERM INTO W
00D5 0033            00371         MOVWF   mulcnd          ; MULTIPLY IT BY THE PROPORTIONAL GAIN
00D6 0C30            00372         MOVLW   KP              ; KP AND THEN SCALE IT DOWN BY DIVIDING
00D7 0034            00373         MOVWF   mulplr          ; IT DOWN BY 16. IF IT IS STILL OVER
00D8 0901            00374         CALL    mpy_S           ; 255 THEN LIMIT IT TO 255
00D9 091D            00375         CALL    DIV_LMT
                     00376
00DA                 00377 RESTORE_SGN
00DA 0719            00378         BTFSS   FLAGS,ER_SGN    ; IF THE ERROR SIGN IS NEGATIVE THEN
00DB 0ADE            00379         GOTO    ADDPROP         ; PUT THE SIGN INTO THE LOW BYTE
00DC 0276            00380         COMF    L_byte,1
00DD 02B6            00381         INCF    L_byte,1
                     00382
00DE                 00383 ADDPROP
00DE 0216            00384         MOVF    L_byte,W        ; SAVE THE PROPORTIONAL PART
00DF 01ED            00385         ADDWF   SUMLO,1         ; IN THE SUM
00E0 0603            00386         BTFSC   SWR,CARRY       ; IF THE ADDITION CARRIED OUT THEN
00E1 02B8            00387         INCF    SUMHI,1         ; INCREMENT THE HIGH BYTE
00E2 0C00            00388         MOVLW   0               ; THEN
00E3 06ED            00389         BTFSC   SUMLO,7         ; SIGN EXTEND TO THE UPPER
```

```
00E4 0CFF          00390         MOVLW   0FF             ; BYTE
00E5 01F8          00391         ADDWF   SUMHI,1
                   00392
                   00393
                   00394
                   00395 ;****CALCULATE THE INTEGRAL TERM******
                   00396 ; The integral term is an accumulation of the error thus far.  Its purpose
                   00397 ; is to allow even a small error to effect a large change.  It does this
                   00398 ; by adding a small number into an accumulator each cycle through the program.
                   00399 ; Thusly even a small error that exists for a while will build up to a large
                   00400 ; enough number to effect an output sufficient to move the system.  The effect
                   00401 ; that this integral accumulator has is modulated by the integral gain term KI.
                   00402 ; The integral of the error over time is multiplied by KI and the result is its
                   00403 ; contribution to the final summation for determining the output value.  This
                   00404 ; term helps to insure the long-term accuracy of the system is good.  A certain
                   00405 ; amount is necessary for this purpose but too much will cause oscillations.
                   00406 ; The integral is bounded in magnitude for two purposes.  The first is so that
                   00407 ; it never rolls over and changes sign.  The second is that it may saturate on
                   00408 ; long moves forcing an excessively large overshoot to “de-integrate” the error
                   00409 ; accumulated during the first of the moves.
                   00410
00E6               00411 C_INT
00E6 020C          00412         MOVF    ERR1,W          ; MOVE THE ERROR INTO THE W REG
00E7 0643          00413         BTFSC   SWR,Z           ; AND CHECK TO SEE IF IT IS ZERO
00E8 0AFF          00414         GOTO    ADDINT          ; IF SO THEN DONT CHANGE THE ACCUMULATOR
00E9 0619          00415         BTFSC   FLAGS,ER_SGN    ; TEST THE FLAGS TO FIND THE POLARITY
00EA 0AEE          00416         GOTO    MNS_1           ; OF THE ERROR .. 0 POSITIVE 1 NEGATIVE
00EB               00417 PLS_1
00EB 0C02          00418         MOVLW   KI              ; IF POSITIVE ADD ONE TO
00EC 01EE          00419         ADDWF   ACCUM,1         ; THE ERROR ACCUMULATOR
00ED 0AF0          00420         GOTO    LMTACM          ; THEN LIMIT IT TO +/-100
00EE               00421 MNS_1
00EE 0C02          00422         MOVLW   KI              ; IF NEGATIVE THEN SUBTRACT ONE
00EF 00AE          00423         SUBWF   ACCUM,1         ; FROM THE ERROR ACCUMULATOR
00F0               00424 LMTACM
00F0 06EE          00425         BTFSC   ACCUM,7         ; CHECK THE SIGN BIT OF THE ERROR ACCUMULATOR
00F1 0AF9          00426         GOTO    M_LMT           ; AND DO A POSITIVE OR NEGATIVE LIMIT
00F2               00427 P_LMT
00F2 0C9C          00428         MOVLW   9CH             ; FOR THE POSITIVE LIMIT ADD 156 TO THE
00F3 01CE          00429         ADDWF   ACCUM,0         ; NUMBER AND SEE IF YOU GENERATE A CARRY
00F4 0703          00430         BTFSS   SWR,CARRY       ; BY CHECKING THE CARRY FLAG
00F5 0AFF          00431         GOTO    ADDINT          ; IF NOT THEN ITS O.K.
00F6 0C64          00432         MOVLW   64H             ; IF SO THEN FORCE THE ACCUMULATOR TO
00F7 002E          00433         MOVWF   ACCUM           ; 100 DECIMAL
00F8 0AFF          00434         GOTO    ADDINT
00F9               00435 M_LMT
00F9 0C9C          00436         MOVLW   9CH             ; FOR THE NEGATIVE LIMIT SUBTRACT 156 FROM
00FA 008E          00437         SUBWF   ACCUM,0         ; THE NUMBER AND SEE IF YOU GENERATE A
00FB 0603          00438         BTFSC   SWR,CARRY       ; NON-CARRY CONDITION INDICATING A ROLL-OVER
00FC 0AFF          00439         GOTO    ADDINT          ; IF NOT THEN LEAVE THE ACCUMULATOR ALONE
00FD 0C9C          00440         MOVLW   9CH             ; IF SO THEN LIMIT IT TO -100 BY
00FE 002E          00441         MOVWF   ACCUM           ; FORCING THAT VALUE IN THE ACCUMULATOR
                   00442
00FF               00443 ADDINT
00FF 020E          00444         MOVF    ACCUM,W         ; ADD THE INTEGRAL ACCUMULATOR TO
0100 01ED          00445         ADDWF   SUMLO,1         ; THE LOW BYTE OF THE SUM
0101 0603          00446         BTFSC   SWR,CARRY       ; TEST FOR OVERFLOW, IF SO THEN
0102 02B8          00447         INCF    SUMHI,1         ; INCREMENT THE HI BYTE
0103 0C00          00448         MOVLW   0               ; LOAD 0 INTO THE W REGISTER
0104 06EE          00449         BTFSC   ACCUM,7         ; IF THE INTEGRAL ACCUMULATOR WAS NEGATIVE
0105 0240          00450         COMF    W,W             ; COMPLEMENT THE 0 TO GET SIGN FOR HIGH BYTE
0106 01F8          00451         ADDWF   SUMHI,1         ; ADD INTO THE HIGH BYTE OF THE SUM
                   00452
                   00453
0107               00454 U_DEXIT                         ; EXIT POINT FOR THE UP/DOWN CONTROL OF ACCUM
                   00455
                   00456
                   00457
                   00458 ;****CALCULATING THE DIFFERENTIAL TERM*************************
                   00459 ; The differential term examines the error and determines how much
                   00460 ; it has changed since the last cycle.  It does this by subtracting the
                   00461 ; old error from the new error.  Since the cycle time is relatively fixed
                   00462 ; we can use it as the “dt” of the desired “de/dt”.  This derivative of the
                   00463 ; error is then multiplied by the differential gain term KD and becomes the
                   00464 ; differential term contribution for the final summation.
                   00465
                   00466 ; First, create the “de” term by doing a signed subtaction of new error
                   00467 ; minus the old error. (new_error - old_error)
                   00468
0107               00469 C_DIFF
0107 020C          00470         MOVF    ERR1,W          ; LOAD THE NEW ERROR INTO REGISTER
0108 0719          00471         BTFSS   FLAGS,ER_SGN
0109 0B0D          00472         GOTO    LO_BYTE
```

```
010A 026C           00473         COMF    ERR1,1          ; CORRECT THE VALUE TO BE 16 BIT
010B 028C           00474         INCF    ERR1,W
010C 026C           00475         COMF    ERR1,1          ; RESTORE IT FOR FUTURE USE TO 8 BIT MAGNITUDE
010D                00476 LO_BYTE
010D 0034           00477         MOVWF   ACCbLO          ; FOR SUBTRACTION
010E 0C00           00478         MOVLW   00
010F 0619           00479         BTFSC   FLAGS,ER_SGN    ; SIGN EXTEND THE UPPER BYTE
0110 0CFF           00480         MOVLW   0FF
0111 0036           00481         MOVWF   ACCbHI
0112 020F           00482         MOVF    ERR_O,W         ; LOAD THE OLD ERROR INTO OTHER REGISTER
0113 0799           00483         BTFSS   FLAGS,OER_SGN
0114 0B17           00484         GOTO    LO_BYTEO
0115 026F           00485         COMF    ERR_O,1         ; CORRECT THE VALUE TO BE 16 BIT
0116 028F           00486         INCF    ERR_O,W
0117                00487 LO_BYTEO
0117 0033           00488         MOVWF   ACCaLO          ; FOR SUBTRACTION
0118 0C00           00489         MOVLW   00
0119 0699           00490         BTFSC   FLAGS,OER_SGN   ; SIGN EXTEND THE UPPER BYTE
011A 0CFF           00491         MOVLW   0FF
011B 0035           00492         MOVWF   ACCaHI
011C 090F           00493         CALL    D_sub           ; PERFORM THE SUBTRACTION
                    00494
011D                00495 STRIP_SGN
011D 06F6           00496         BTFSC   ACCbHI,7        ; TEST THE SIGN OF THE RESULT
011E 0B20           00497         GOTO    NEG_ABS
011F 0B25           00498         GOTO    POS_ABS
0120                00499 NEG_ABS
0120 0559           00500         BSF     FLAGS,DE_SGN    ; ITS NEGATIVE SO SET THE FLAG AND
0121 0274           00501         COMF    ACCbLO,1        ; COMPLEMENT THE VALUE
0122 0294           00502         INCF    ACCbLO,W
0123 002F           00503         MOVWF   ERR_O
0124 0B28           00504         GOTO    MULT_KD
0125                00505 POS_ABS
0125 0459           00506         BCF     FLAGS,DE_SGN    ; ITS POSITIVE SO SET RESET THE FLAG
0126 0214           00507         MOVF    ACCbLO,W        ; AND SAVE THE VALUE
0127 002F           00508         MOVWF   ERR_O
                    00509
                    00510 ; Then multiply by Kd
                    00511
0128                00512 MULT_KD
0128 020F           00513         MOVF    ERR_O,W
0129 0033           00514         MOVWF   mulcnd          ; MOVE THE DE/DT TERM INTO THE MULCND REG.
012A 0C20           00515         MOVLW   KD              ; MOVE THE DIFFERENTIAL GAIN TERM INTO
012B 0034           00516         MOVWF   mulplr          ; MULPLR TO MULTIPLY THE DE/DT
012C 0901           00517         CALL    mpy_S           ; DO THE MULTIPLICATION
012D 091D           00518         CALL    DIV_LMT         ; SCALE AND LIMIT TO 100
                    00519
012E                00520 RE_SGN
012E 0759           00521         BTFSS   FLAGS,DE_SGN    ; IF THE DE SIGN IS NEGATIVE THEN
012F 0B32           00522         GOTO    SAVE_DIFF       ; PUT THE SIGN INTO THE LOW BYTE
0130 0276           00523         COMF    L_byte,1
0131 02B6           00524         INCF    L_byte,1
0132                00525 SAVE_DIFF
0132 0216           00526         MOVF    L_byte,W
0133 0643           00527         BTFSC   SWR,Z
0134 0B45           00528         GOTO    ROLL_ER
0135 002F           00529         MOVWF   ERR_O
                    00530
                    00531 ; ADD THE DIFF TERM INTO THE SUMM ***************
                    00532
0136                00533 ADDDIF
0136 0C00           00534         MOVLW   00
0137 0659           00535         BTFSC   FLAGS,DE_SGN    ; PUT THE KD*(DE/DT) TERM INTO THE
0138 0CFF           00536         MOVLW   0FF             ; REGISTERS TO ADD. AND
0139 0036           00537         MOVWF   ACCbHI          ; SIGN EXTEND THE UPPER BYTE
013A 020F           00538         MOVF    ERR_O,W
013B 0034           00539         MOVWF   ACCbLO
013C 020D           00540         MOVF    SUMLO,W         ; LOAD THE CURRENT SUM INTO THE
013D 0033           00541         MOVWF   ACCaLO          ; REGISTERS TO ADD
013E 0218           00542         MOVF    SUMHI,W
013F 0035           00543         MOVWF   ACCaHI
0140 0910           00544         CALL    D_add           ; ADD IN THE DIFFERENTIAL TERM
0141 0214           00545         MOVF    ACCbLO,W        ; SAVE THE RESULTS BACK
0142 002D           00546         MOVWF   SUMLO           ; INTO SUMLO AND HI
0143 0216           00547         MOVF    ACCbHI,W
0144 0038           00548         MOVWF   SUMHI
                    00549
0145                00550 ROLL_ER
0145 020C           00551         MOVF    ERR1,W          ; TAKE THE CURRENT ERROR
0146 002F           00552         MOVWF   ERR_O           ; AND PUT IT IN THE ERROR HISTORY
0147 0499           00553         BCF     FLAGS,OER_SGN   ; SAVE THE CURRENT ERROR SIGN
0148 0619           00554         BTFSC   FLAGS,ER_SGN    ; IN THE OLD ERROR SIGN FOR
0149 0599           00555         BSF     FLAGS,OER_SGN   ; NEXT TIME THROUGH
```

```
                    00556
                    00557
                    00558 ;****SET UP THE DIRECTION FOR THE BRIDGE********************
                    00559 ;
                    00560 ; After the sum of all the components has been made, the sign of the
                    00561 ; sum will determine which way the bridge should be powered.
                    00562 ; If the sum is negative the bridge needs to be set to drive ccw; if the
                    00563 ; sum is Positive then the bridge needs to be set to drive cw.  This
                    00564 ; is purely a convention and depends upon the polarity the motor and feedback
                    00565 ; element are hooked up in.
                    00566
014A                00567 SET_DIR
014A 0479           00568         BCF     FLAGS,DIR       ; SET FOR DEFAULT CLOCKWISE
014B 06F8           00569         BTFSC   SUMHI,7         ; LOOK AT THE SIGN BIT, IF IT IS SET
014C 0579           00570         BSF     FLAGS,DIR       ; THEN SET FOR CCW BRIDGE DRIVE
                    00571
                    00572
                    00573 ;**** SCALE THE NUMBER TO BETWEEN 0 AND 100% **********************
                    00574
                    00575 ; After the direction is set the request for duty cycle is limited to between
                    00576 ; 0 and 100 percent inclusive.  This value is passed to the dutycycle setting
                    00577 ; routine by loading it in the variable “PCNT”.
                    00578
                    00579
014D                00580 L_SUMM
014D 07F8           00581         BTFSS   SUMHI,7         ; CHECK TO SEE IF IT IS NEGATIVE
014E 0B52           00582         GOTO    POS_LM
014F 0278           00583         COMF    SUMHI,1
0150 026D           00584         COMF    SUMLO,1
0151 02AD           00585         INCF    SUMLO,1
                    00586
0152                00587 POS_LM
0152 0C01           00588         MOVLW   1H              ; SUBTRACT 1 FROM THE HIGH BYTE TO SEE
0153 0098           00589         SUBWF   SUMHI,0         ; IF THERE IS ANYTHING THERE, IF NOT,
0154 0703           00590         BTFSS   SWR,CARRY       ; THEN LEAVE THE LOW BYTE ALONE
0155 0B59           00591         GOTO    LB_L            ; OTHERWISE GIVE THE LOW BYTE A FULL
0156 0C64           00592         MOVLW   64H             ; COUNT AND IT WILL HAVE BEEN LIMITED
0157 002D           00593         MOVWF   SUMLO           ; TO 100
0158 0B5F           00594         GOTO    LP_EXIT         ; GOTO LIMIT PERCENT EXIT
0159                00595 LB_L
0159 0C64           00596         MOVLW   64H             ; LIMIT THE MAGNITUDE OF THE VALUE TO
015A 008D           00597         SUBWF   SUMLO,0         ; 100 DECIMAL
015B 0703           00598         BTFSS   SWR,CARRY
015C 0B5F           00599         GOTO    LP_EXIT
015D 0C64           00600         MOVLW   64H
015E 002D           00601         MOVWF   SUMLO
                    00602
015F                00603 LP_EXIT
015F 020D           00604         MOVF    SUMLO,W         ; STORE THE LIMITED VALUE IN
0160 0029           00605         MOVWF   PCNT            ; THE PERCENT DUTYCYCLE REQUEST
                    00606
                    00607
                    00608 ;*******************************************************
                    00609 ; PWM GENERATING ROUTINE
                    00610 ;
                    00611 ; The important thing here is not to have to do too many decisions or
                    00612 ; calculations while you are generating the 100 or so pulses.  These will
                    00613 ; take time and limit the minimum or maximum duty cycle.
                    00614
0161                00615 WHICH_DIR
0161 0679           00616         BTFSC   FLAGS,DIR       ; CHECK THE DIRECTION FLAG
0162 0B76           00617         GOTO    GOCCW           ; DO CCW PULSES FOR 1
0163 0B64           00618         GOTO    GOCW            ; DO CW PULSES FOR 0
                    00619
                    00620
0164                00621 GOCW
0164 0426           00622         BCF     PORTB,PWMCCW    ; SET THE BRIDGE FOR CW MOVE
0165 0C64           00623         MOVLW   64H             ;
0166 0032           00624         MOVWF   CYCLES          ; SET UP CYCLES COUNTER FOR 100 PULSES
0167 0943           00625         CALL    CALCTIMES       ; CALCULATE THE HI AND LO TIMES
                    00626
0168                00627 RLDCW
0168 0207           00628         MOVF    HI,0            ; RELOAD THE HI TIMER
0169 002A           00629         MOVWF   HI_T            ; WITH THE CALCULATED TIME
016A 0208           00630         MOVF    LO,0            ; RELOAD THE LO TIMER
016B 002B           00631         MOVWF   LO_T            ; WITH THE CALCULATED TIME
016C 0004           00632         CLRWDT                  ; TAG THE WATCHDOG TIMER
                    00633
016D                00634 CWHI
016D 0506           00635         BSF     PORTB,PWMCW     ; SET THE CLOCKWISE PWMBIT HIGH
016E 02EA           00636         DECFSZ  HI_T,1          ; DECREMENT THE HI USEC. COUNTER
016F 0B6D           00637         GOTO    CWHI            ; DO ANOTHER LOOP
0170                00638 CWLO
```

```
0170 0406           00639         BCF     PORTB,PWMCW     ; SET THE CLOCKWISE PWM BIT LOW
0171 02EB           00640         DECFSZ  LO_T,1          ; DECREMENT THE LO USEC. COUNTER
0172 0B70           00641         GOTO    CWLO            ; DO ANOTHER LOOP
0173 02F2           00642         DECFSZ  CYCLES,1        ; DECREMENT THE NUMBER OF CYCLES LEFT
0174 0B68           00643         GOTO    RLDCW           ; DO ANOTHER PULSE
0175 0A50           00644         GOTO    BEGIN           ; DO ANOTHER MAIN SYSTEM CYCLE
                    00645
                    00646
0176                00647 GOCCW
0176 0406           00648         BCF     PORTB,PWMCW     ; SET THE BRIDGE FOR CCW MOVE
0177 0C64           00649         MOVLW   64H             ;
0178 0032           00650         MOVWF   CYCLES          ; SET UP CYCLE COUNTER FOR 100 PULSES
0179 0943           00651         CALL    CALCTIMES       ; CALCULATE THE HI AND LO TIMES
017A                00652 RLDCCW
017A 0207           00653         MOVF    HI,0            ; RE LOAD THE HI TIMER
017B 002A           00654         MOVWF   HI_T            ; WITH THE CALCULATED TIME
017C 0208           00655         MOVF    LO,0            ; RE LOAD THE LO TIMER
017D 002B           00656         MOVWF   LO_T            ; WITH THE CALCULATED TIME
017E 0004           00657         CLRWDT                  ; TAG THE WATCHDOG
                    00658
017F                00659 CCWHI
017F 0526           00660         BSF     PORTB,PWMCCW    ; SET THE COUNTERCLOCKWISE PWM BIT HIGH
0180 02EA           00661         DECFSZ  HI_T,1          ; DECREMENT THE HI USEC. COUNTER
0181 0B7F           00662         GOTO    CCWHI           ; DO ANOTHER LOOP
0182                00663 CCWLO
0182 0426           00664         BCF     PORTB,PWMCCW    ; SET THE COUNTERCLOCKWISE PWM BIT LOW
0183 02EB           00665         DECFSZ  LO_T,1          ; DECREMENT THE LO USEC. COUNTER
0184 0B82           00666         GOTO    CCWLO           ; DO ANOTHER LOOP
0185 02F2           00667         DECFSZ  CYCLES,1        ; DECREMENT THE NUMBER OF CYCLES LEFT
0186 0B7A           00668         GOTO    RLDCCW          ; DO ANOTHER PULSE
0187 0A50           00669         GOTO    BEGIN           ; DO ANOTHER MAIN SYSTEM CYCLE
                    00670
                    00671
                    00672
                    00673
                    00674
                    00675         ;************* START VECTOR ********************
                    00676
0188                00677 CLRREG                          ;INITIALIZE REGISTERS
                    00678
0188 0C0B           00679         MOVLW   0BH             ; SET PORT A FOR 3 INPUTS AND
0189 0005           00680         TRIS    PORTA           ; AN OUTPUT
018A 0C1C           00681         MOVLW   1CH             ; SET PORT B FOR INPUTS AND OUTPUTS
018B 0006           00682         TRIS    PORTB           ; THIS SETTING FOR SENDING TO A/D
018C 0040           00683         CLRW                    ; CLEAR THE W REGISTER
018D 0002           00684         OPTION                  ; STORE THE W REG IN THE OPTION REG
018E 0C08           00685         MOVLW   08H             ; STARTING REGISTER TO ZERO
018F 0024           00686         MOVWF   FSR             ;
0190                00687 GCLR
0190 0060           00688         CLRF    00              ;
0191 03E4           00689         INCFSZ  FSR, F          ; SKIP AFTER ALL REGISTERS
0192 0B90           00690         GOTO    GCLR            ; HAVE BEEN INITIALIZED
0193 0A50           00691         GOTO    BEGIN           ; START AT THE BEGINING OF THE PROGRAM
                    00692
01FF                00693         ORG     01FF            ;
01FF 0B88           00694         GOTO    CLRREG          ; START VECTOR
                    00695
                    00696
                    00697         END

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

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXX------------ ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   405
Program Memory Words Free:   619


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: 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.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

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

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

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

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
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

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

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, Hongiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore
Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 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 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
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 Müchen, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleone
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81-4-5471- 6166 Fax: 81-4-5471-6122

5/8/97

MICROCHIP