Intelligent Remote Positioner (Motor Control)

Author: Steven Frank - Vesta Technology Inc.

INTRODUCTION

The excellent cost/performance ratio of the PIC16C5X are well suited for a low-cost proportional D.C. actuator controller. This application note depicts a design of a remote intelligent positioning system using a D.C. motor (up to 1/3 hp) run from 12 to 24 V. 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 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.

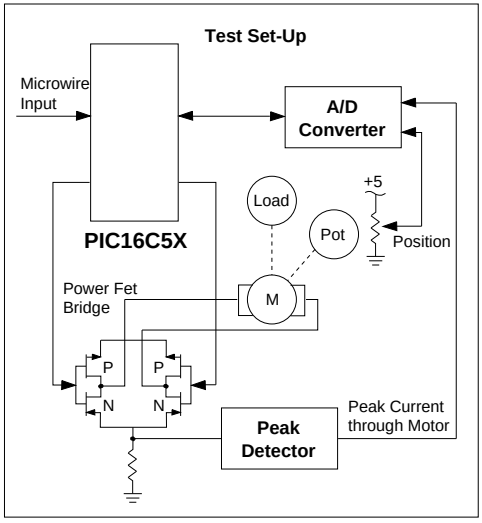
This system is ideally suited to remotely position valves and machinery. It can be used with D.C. motors to easily automate manual equipment. Because of the 5-wire serial interface, the positioner can be installed near its power supply and load. The remote intelligent positioner can then be linked to 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 need only send position commands and is therefore free to service the user interface, main application software and command many remote positioners.

The limit switch inputs provide a safety net this keeps the system from destroying itself in the event that the feedback device is lost. 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 50kb/s for the 4 MHz part. As currently implemented in this application note, the position request is the only communication. There are several variable locations available and they could be utilized to allow down-loading of the loop gain parameters, reading positioner information, or for setting 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 data by way of a microwire A/D converter. This part was chosen for low cost yet it provides adequate performance. The second channel of the A/D is shown hooked up to a peak current detector. If the user desired, the PIC16C5X could monitor and protect the motor from overcurrent by monitoring this information.

FIGURE 1 - BLOCK DIAGRAM



MICROWIRE™ is a trademark of National Semiconductor Corporation.

The H-bridge power amplifier will deliver 10 or more amps at up to 24 volts 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 for that is what the other terms are based on.

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 (Kp) 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. This term is traditionally formed by integrating the error over time. In this application it is done by integrating the Ki 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 Ki 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 (Kd) 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 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 H-bridge direction. 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 (Kp, Ki, Kd) 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 is simply 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. The systems that are generally controlled 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. Utilizing this technique the inexpensive PIC16C5X can implement PID control, PWM generation and 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 FLOW CHART

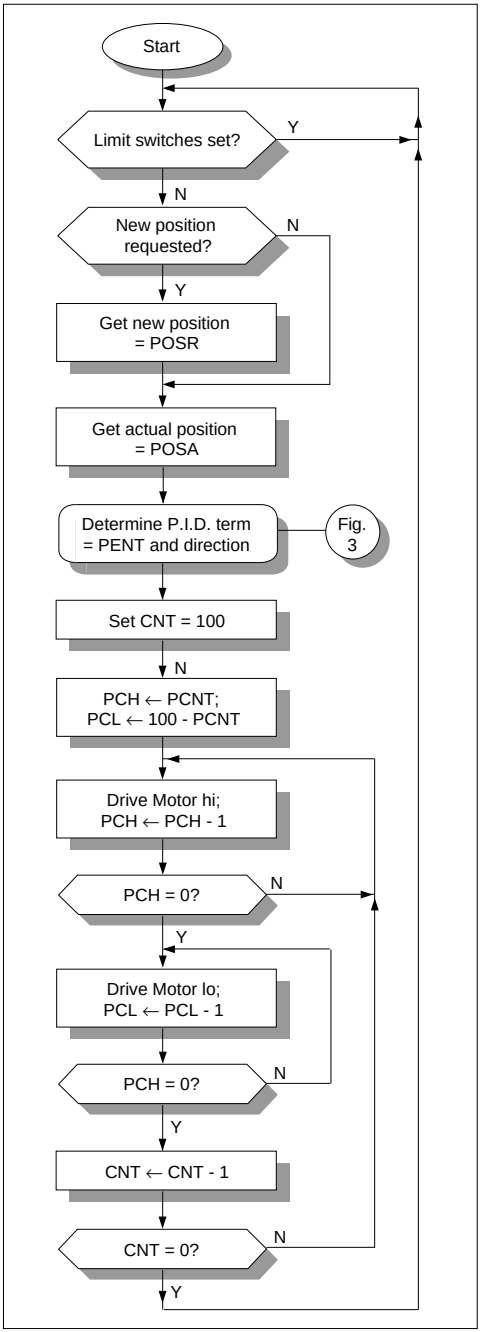


FIGURE 3 - P.I.D. ALGORITHM FLOW CHART

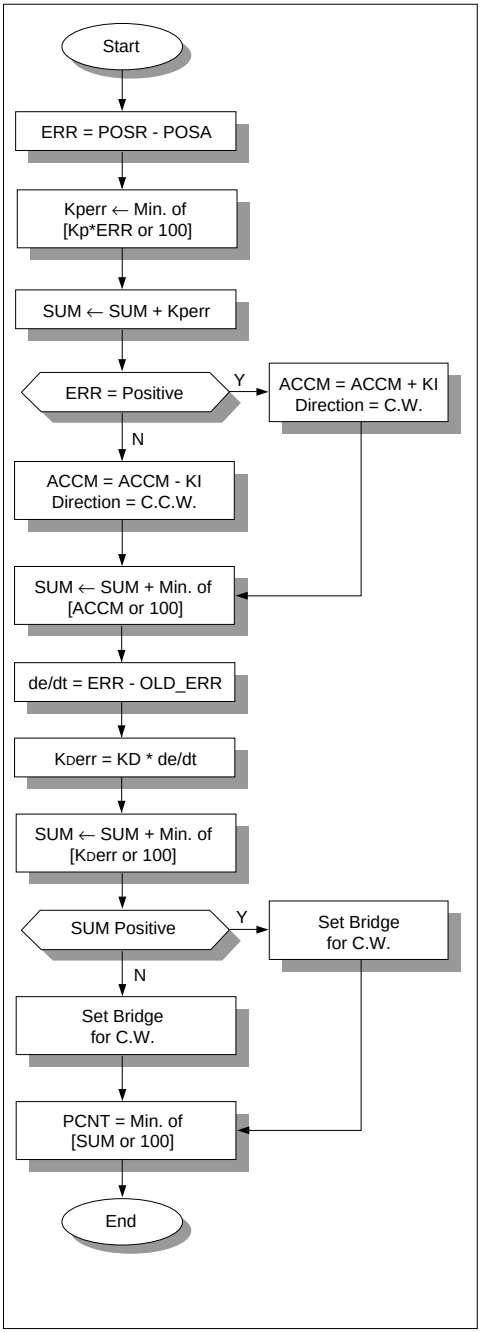
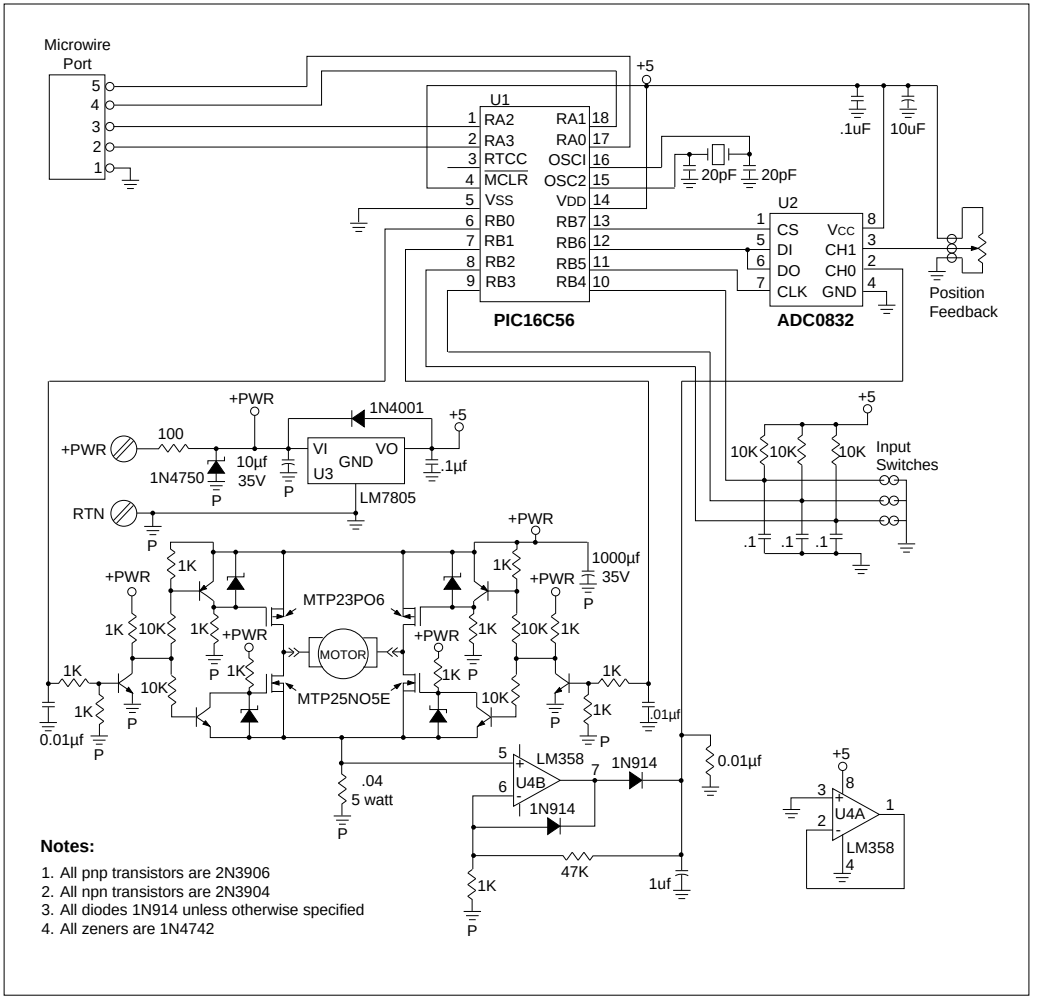


FIGURE 4 - SCHEMATIC



```
MPASM B0.54                                                    PAGE  1


                ;
                ;       mw8pos.asm
                ;
                        LIST P=16C56
                ;*****************************************************************
                ; REV. A        Original release  1/10/92 srf
                ;
                ; ***************************************************************
                ;
                ;REGISTER EQUATES
                ;
0000             W       EQU     0
0000             PNTR    EQU     00H             ; CONTENTS OF POINTER
0019             FLAGS   EQU     19H             ; USE THIS VARIABLE LOCATION AS FLAGS
                                                 ; 0 BIT IS SIGN OF ERROR 1 IS NEGATIVE
                                                 ; 1 BIT IS SIGN OF ERROR ACCUMULATOR
                                                 ; 2 BIT IS SIGN OF THE DE/DE TERM
                                                 ; 3 BIT IS DIRECTION 0 IS CW
                                                 ; 4 BIT IS SIGN OF THE OLD ERROR
0003             STATUS  EQU     03H
0003             SWR     EQU     03H             ; STATUS WORD REGISTER
                                                 ; 0 = CARRY
                                                 ; 1 = DC
                                                 ; 2 = Z, SET IF RESULT IS ZERO
0004             FSR     EQU     04H             ; FILE SELECT REGISTER
0005             PORTA   EQU     05H             ; I/O REG (A0-A3), (A4-A7 DEF=0)
0006             PORTB   EQU     06H             ; I/O REGISTER(B0-B7)
0007             HI      EQU     07H             ; NUMBER OF HIGH MICROSECONDS
0008             LO      EQU     08H             ; NUMBER OF LOW MICROSECONDS
0009             PCNT    EQU     09H             ; PERCENT DUTYCYCLE REQUEST
000A             HI_T    EQU     0AH             ; COUNTER FOR USECONDS LEFT/PULSE HI
000B             LO_T    EQU     0BH             ; COUNTER FOR USECONDS LEFT/PULSE LO
000C             ERROR   EQU     0CH             ; HOLDER FOR THE POSITIONAL ERROR
                                                 ; THIS IS AN 8 BIT MAGNITUDE WITH THE SIGN
                                                 ; KEPT IN THE FLAG REGISTER (9BIT SIGNED)
000D             SUMLO   EQU     0DH             ; PROGRESSIVE SUM OF THE PID TERMS
000E             ACCUM   EQU     0EH             ; ERROR ACCUMULATOR
000F             ERR_O   EQU     0FH             ; ERROR HISTORY USED FOR de/dt
                                                 ; THIS IS AN 8 BIT MAGNITUDE WITH THE SIGN
                                                 ; KEPT IN THE FLAG REGISTER (9BIT SIGNED)
0010             POSR    EQU     10H             ; POSITIONAL REQUEST
0011             POSA    EQU     11H             ; ACTUAL POSITION
0012             CYCLES  EQU     12H             ; COUNTER FOR CYCLES OUT

0013             mulcnd  equ     13H             ; 8 bit multiplicand
0013             ACCaLO  EQU     13H             ; same location used for the add routine
0014             mulplr  equ     14H             ; 8 bit multiplier
0014             ACCbLO  EQU     14H             ; same location used for the add routine
0015             H_byte  equ     15H             ; High byte of the 16 bit result
0015             ACCaHI  EQU     15H             ; same location used for the add routine
0016             L_byte  equ     16H             ; Low byte of the 16 bit result
0016             ACCbHI  EQU     16H             ; same location used for the add routine
0017             count   equ     17H             ; loop counter
0018             SUMHI   EQU     18H             ; HIGH BYTE OF THE LOOP SUM


                ; PORT ASSIGNMENTS AND CONSTANTS

0000             PWMCW   EQU     0               ; CLOCKWISE PWM OUTPUT BIT
0001             PWMCCW  EQU     1               ; COUNTERCLOCKWISE PWM OUTPUT BIT
0000             CARRY   EQU     0               ; CARRY BIT IN THE STATUS REGISTER
0002             Z       EQU     2               ; THE ZERO BIT OF THE STATUS REGISTER
0001             Same    equ     1               ;
```

```
0000                 ER_SGN  EQU     0               ; SIGN BIT FOR THE ERROR IN FLAG REGISTER
0001                 AC_SGN  EQU     1               ; SIGN BIT FOR THE ERROR ACCUMULATOR
0002                 DE_SGN  EQU     2               ; SIGN BIT FOR DE/DT
0004                 OER_SGN EQU     4               ; SIGN BIT FOR THE OLD ERROR
0030                 KP      EQU     30              ; PROPORTIONAL GAIN
0002                 KI      EQU     2               ; INTEGRAL GAIN
0020                 KD      EQU     20              ; DIFFERENTIAL GAIN
0003                 DIR     EQU     3               ; THE DIRECTION FLAG
0007                 CSN     EQU     7               ; CHIP SELECT NOT ON A/D
0006                 BV      EQU     6               ; DATA LINE FOR THE A/D
0005                 CK      EQU     5               ; CLOCK LINE FOR THE A/D
0002                 MWDO    EQU     2               ; MICROWIRE DATA OUT FROM POSITIONER
0001                 MWDI    EQU     1               ; MICROWIRE DATA IN TO POSITIONER
0000                 MWCS    EQU     0               ; MICROWIRE CHIP SELECT TO POSITIONER
0003                 MWCK    EQU     3               ; MICROWIRE CLOCK IN TO POSITIONER

                    ;***** MACROS *************************************************
                    ;
                    CLKUP   MACRO                   ; clock up macro for the microwire
                      BSF     PORTB,CK              ; data acquisition from the a/d
                      NOP
                      ENDM

                    CLKDN   MACRO                   ; clock down macro for the microwire
                      BCF     PORTB,CK              ; data acquisition from the a/d
                      NOP
                      ENDM

                    GET_BIT MACRO                   ; ** FOR RECEIVING A/D DATA **
                      BCF     SWR,CARRY
                      BSF     PORTB,CK              ; SET CLOCK BIT HIGH
                      BTFSC   PORTB,BV              ; LOOK AT DATA COMMING IN
                      BSF     SWR,CARRY             ; SET THE CARRY FOR A 1
                      RLF     POSA                  ; ROTATE THE W REG LEFT
                      BCF     PORTB,CK              ; SET THE CLOCK LOW
                      NOP                           ; DELAY
                      ENDM

0000 0B88                     GOTO    CLRREG

                    ;***** MATH ROUTINES **************************************
                    ;
                    ; **** 8 BIT MULTIPLY ********
                    ; ****************************         Begin Multiplier Routine
0001 0075           mpy_S     clrf    H_byte
0002 0076                     clrf    L_byte
0003 0C08                     movlw   8
0004 0037                     movwf   count
0005 0213                     movf    mulcnd,w
0006 0403                     bcf     STATUS,CARRY    ; Clear the carry bit in the status Reg.
0007 0334           loop      rrf     mulplr
0008 0603                     btfsc   STATUS,CARRY
0009 01F5                     addwf   H_byte,Same
000A 0335                     rrf     H_byte,Same
000B 0336                     rrf     L_byte,Same
000C 02F7                     decfsz  count
000D 0A07                     goto    loop
000E 0800                     retlw   0

                    ; ****************************
                    ; DOUBLE PRECISION ADD AND SUBTRACT ( ACCb-ACCa->ACCb )

000F 0917           D_sub     call    neg_A           ; At first negate ACCa, then add

                    ;***************
                    ; Double Precision Addition ( ACCb+ACCa->ACCb )

0010 0213           D_add     movf    ACCaLO,W
0011 01F4                     addwf   ACCbLO          ; add lsb
```

```
0012 0603                   btfsc   STATUS,CARRY    ; add in carry
0013 02B6                   incf    ACCbHI
0014 0215                   movf    ACCaHI,W
0015 01F6                   addwf   ACCbHI          ; add msb
0016 0800                   retlw   00
                    ;
                    ;
0017 0273           neg_A   comf    ACCaLO          ; negate ACCa
0018 02B3                   incf    ACCaLO
0019 0643                   btfsc   STATUS,Z
001A 00F5                   decf    ACCaHI
001B 0275                   comf    ACCaHI
001C 0800                   retlw   00

                    ; ****************************************
                    ; divide by 16 and limit to 100 Decimal

                    SHIFT   MACRO
                       BCF     SWR,CARRY
                       RRF     L_byte
                       BCF     SWR,CARRY
                       RRF     H_byte
                       BTFSC   SWR,CARRY
                       BSF     L_byte,7
                       ENDM
                    DIV_LMT
                       SHIFT
001D 0403                   BCF     SWR,CARRY
001E 0336                   RRF     L_byte
001F 0403                   BCF     SWR,CARRY
0020 0335                   RRF     H_byte
0021 0603                   BTFSC   SWR,CARRY
0022 05F6                   BSF     L_byte,7

                       SHIFT
0023 0403                   BCF     SWR,CARRY
0024 0336                   RRF     L_byte
0025 0403                   BCF     SWR,CARRY
0026 0335                   RRF     H_byte
0027 0603                   BTFSC   SWR,CARRY
0028 05F6                   BSF     L_byte,7

                       SHIFT
0029 0403                   BCF     SWR,CARRY
002A 0336                   RRF     L_byte
002B 0403                   BCF     SWR,CARRY
002C 0335                   RRF     H_byte
002D 0603                   BTFSC   SWR,CARRY
002E 05F6                   BSF     L_byte,7

                       SHIFT
002F 0403                   BCF     SWR,CARRY
0030 0336                   RRF     L_byte
0031 0403                   BCF     SWR,CARRY
0032 0335                   RRF     H_byte
0033 0603                   BTFSC   SWR,CARRY
0034 05F6                   BSF     L_byte,7


                    LMT100
0035 0C01                   MOVLW   1H              ; SUBTRACT 1 FROM THE HIGH BYTE TO SEE
0036 0095                   SUBWF   H_byte,0        ; IF THERE IS ANYTHING THERE, IF NOT,
0037 0703                   BTFSS   SWR,CARRY       ; THEN LEAVE THE LOW BYTE ALONE
0038 0A3C                   GOTO    L8_E            ; OTHERWISE GIVE THE LOW BYTE A FULL
0039 0C64                   MOVLW   64H             ; COUNT AND IT WILL HAVE BEEN LIMITED
003A 0036                   MOVWF   L_byte          ; TO 100
003B 0A42                   GOTO    LMT_EXIT
                    L8_E
003C 0C64                   MOVLW   64H             ; LIMIT THE MAGNITUDE OF THE VALUE TO
```

```
003D 0096                SUBWF   L_byte,0        ; 100 DECIMAL
003E 0703                BTFSS   SWR,CARRY
003F 0A42                GOTO    LMT_EXIT
0040 0C64                MOVLW   64H
0041 0036                MOVWF   L_byte
                  LMT_EXIT
0042 0800                RETLW   00
                  ;
                  ;THE ROUTINE CALCTIMES DOES THE FOLLOWING: PCNT = DUTY CYCLE IN %
                  ; 100 - PCNT —> LO  AND PCNT —> HI. ZERO VALUES IN EITHER LO OR HI
                  ;ARE FORCED TO 1.
                  CALCTIMES
0043 0209                MOVF    PCNT,W          ; PUT REQUESTED % INTO W REGISTER
0044 0027                MOVWF   HI              ; COPY ON MICROSECONDS IN TO HI TIME
0045 0C64                MOVLW   64H
0046 0028                MOVWF   LO
0047 0209                MOVF    PCNT,0
0048 00A8                SUBWF   LO,1            ; LEAVE 100-HI TIME IN LO TIME
0049 0207                MOVF    HI,0            ; INSPECT THE HIGH TIME
004A 0643                BTFSC   SWR,2           ; IF ITS IS ZERO
004B 02A7                INCF    HI,1            ; INCREMENT IT
004C 0208                MOVF    LO,0            ; INSPECT THE LO TIME
004D 0643                BTFSC   SWR,2           ; IF ITS ZERO
004E 02A8                INCF    LO,1            ; INCREMENT IT
004F 0800                RETLW   00


                  ;*****************************************************************
                  BEGIN
0050 0000                NOP                     ; STUBBED BEGINNING


                  ;****CHECKING THE LIMIT SWITCHES AND CHECKING FOR MW***************
                  ; This will check the switch inputs for closure and will terminate
                  ; pulsing is one is closed.  It doesn’t distinguish between the switches
                  ; so they are not dedicated to cw end and ccw end.

                  SW_TRAP
0051 0004                CLRWDT
0052 0746                BTFSS   PORTB,2         ; THIS WILL TEST ALL THREE OF THE
0053 0A51                GOTO    SW_TRAP         ; SWITCH INPUTS.  IF ANY ONE IS
0054 0766                BTFSS   PORTB,3         ; SET THEN EXECUTION OF THE CODE
0055 0A51                GOTO    SW_TRAP         ; WILL BE LIMITED TO LOOKING FOR
0056 0786                BTFSS   PORTB,4         ; IT TO BE CLEARED
0057 0A51                GOTO    SW_TRAP


                  ;****RECEIVING THE POSITIONAL REQUEST*******************************
                  ; The host system that wishes to send positional requests to the positioner
                  ; servo makes its desire known by setting the chip select to the positioner
                  ; low.  It then monitors the busy (Data Out) line from the positioner.  When
                  ; the positioner sets the busy line high, the host may begin sending its 8
                  ; request.  The data bits should be valid on the rising edge of the clock.
                  ; After 8 bits have been received by the positioner it will begin operation
                  ; to send the system to the received position.  It can be interrupted at any
                  ; point during the positioning process by the host sending a new command.
                  ; opportunity to update the command is issued every 100 pwm pulses (every 50
                  ; milliseconds).
                  ; If the host sends a zero positional command the positioner will stop the
                  ; system and remain inactive.
                  ; If the host does not successfully complete a microwire transmission of 8
                  ; data bits the watchdog timer will trip and reset the system to an inactive
                  ; “stopped” state.


                  REC_MW
0058 0C0B                MOVLW   0BH             ; RESET THE PORT FOR THREE INPUTS
```

```
0059 0005                       TRIS    PORTA           ; AND ONE OUTPUT
005A 0445                       BCF     PORTA,MWDO      ; SET THE DATA OUT LOW FOR BUSY
005B 0C20                       MOVLW   20H
005C 0037                       MOVWF   count
                        WATCH_CS
005D 0705                       BTFSS   PORTA,MWCS      ; CHECK FOR INCOMMING REQUESTS
005E 0A62                       GOTO    REC_CMD         ; RECEIVE A NEW POSITION REQUEST
005F 02F7                       DECFSZ  count,1
0060 0A5D                       GOTO    WATCH_CS
0061 0A71                       GOTO    REC_EXIT        ; NO REQUEST WAS MADE IN THE TIME ALLOTED
                        REC_CMD
0062 0545                       BSF     PORTA,MWDO      ; SET THE DATA OUT HIGH FOR "OK TO SEND"
0063 0C08                       MOVLW   8H              ; SET TO RECEIVE 8 BITS
0064 0037                       MOVWF   count
                        WAIT_UP
0065 0765                       BTFSS   PORTA,MWCK      ; WAIT FOR A RISING EDGE
0066 0A65                       GOTO    WAIT_UP
0067 0403                       BCF     SWR,CARRY       ; RESET THE CARRY TO A DEFAULT ZERO
0068 0625                       BTFSC   PORTA,MWDI      ; READ THE DATA IN
0069 0503                       BSF     SWR,CARRY       ; SET THE CARRY FOR A ONE
006A 0370                       RLF     POSR,1          ; ROTATE THE BIT INTO THE POSITION REQ.
006B 02F7                       DECFSZ  count,1         ; DECREMENT THE BIT COUNTER
006C 0A6E                       GOTO    WAIT_DN         ; WAIT FOR THE FALLING EDGE
006D 0A71                       GOTO    REC_EXIT        ; LAST BIT RECEIVED
                        WAIT_DN
006E 0665                       BTFSC   PORTA,MWCK      ; CHECK THE INCOMMING CLOCK
006F 0A6E                       GOTO    WAIT_DN         ; IF IT IS STILL HIGH WAIT FOR IT TO GO LOW
0070 0A65                       GOTO    WAIT_UP         ; IF IT GOES LOW GO BACK TO RECEIVE NEXT BIT

                        REC_EXIT
0071 0445                       BCF     PORTA,MWDO      ; SET THE BUSY FLAG


                        ;********** CHECK FOR THE DISABLE REQUEST ************************
                        ; Position 0 is considered a request to not drive the system.  In this way
                        ; the positioner will come up from a reset in a safe state and will not
                        ; try to move the system to some arbitrary location.

                        MOVE?
0072 0210                       MOVF    POSR,W          ; CHECK THE REQUESTED POSTION
0073 0643                       BTFSC   SWR,Z           ; IF IT IS ZERO THEN WAIT FOR A NON-ZERO
0074 0A50                       GOTO    BEGIN           ; REQUEST BY BRANCHING BACK TO THE BEGINNING

                        ;****READING THE A/D VALUES*****************************************
                        ;
                        ; Read the positional a/d channel (1) and store the value in the actual
                        ; position variable (POSA).
                        ; This is written in line to minimize the use of variables

                        READ_POS
0075 0071                       CLRF    POSA            ; CLEAN THE POSITION ACTUAL HOLDER
0076 04E6                       BCF     PORTB,CSN       ; SET THE CHIP SELECT LOW TO A/D
0077 0C1C                       MOVLW   1CH             ; SET THE DATA LINE TO OUTPUT
0078 0006                       TRIS    PORTB           ; FOR SENDING SET-UP BITS
0079 05C6                       BSF     PORTB,BV        ; SET FOR "START" BIT
007A 0000                       NOP
                          CLKUP                         ; CLOCK IN THE START BIT
007B 05A6                       BSF     PORTB,CK        ; data acquisition from the a/d
007C 0000                       NOP

                          CLKDN                         ; "
007D 04A6                       BCF     PORTB,CK        ; data acquisition from the a/d
007E 0000                       NOP

                          CLKUP                         ; CLOCK IN SINGLE-ENDED
007F 05A6                       BSF     PORTB,CK        ; data acquisition from the a/d
0080 0000                       NOP

                          CLKDN                         ; "
```

```
0081 04A6                     BCF     PORTB,CK        ; data acquisition from the a/d
0082 0000                     NOP

                        CLKUP                         ; CLOCK IN CHANNEL 1
0083 05A6                     BSF     PORTB,CK        ; data acquisition from the a/d
0084 0000                     NOP

                        CLKDN                         ; TO THE MUX
0085 04A6                     BCF     PORTB,CK        ; data acquisition from the a/d
0086 0000                     NOP

0087 0C5C                     MOVLW   5CH             ; SET THE DATA LINE TO INPUT
0088 0006                     TRIS    PORTB           ; TO RECEIVE DATA BITS FROM A/D
                        CLKUP                         ; CLOCK UP TO LET MUX SETTLE
0089 05A6                     BSF     PORTB,CK        ; data acquisition from the a/d
008A 0000                     NOP

                        CLKDN                         ; CLOCK DN TO LET MUX SETTLE
008B 04A6                     BCF     PORTB,CK        ; data acquisition from the a/d
008C 0000                     NOP

                        GET_BIT                       ; GET BIT 7
008D 0403                     BCF     SWR,CARRY
008E 05A6                     BSF     PORTB,CK        ; SET CLOCK BIT HIGH
008F 06C6                     BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
0090 0503                     BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
0091 0371                     RLF     POSA            ; ROTATE THE W REG LEFT
0092 04A6                     BCF     PORTB,CK        ; SET THE CLOCK LOW
0093 0000                     NOP                     ; DELAY

                        GET_BIT                       ; BIT 6
0094 0403                     BCF     SWR,CARRY
0095 05A6                     BSF     PORTB,CK        ; SET CLOCK BIT HIGH
0096 06C6                     BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
0097 0503                     BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
0098 0371                     RLF     POSA            ; ROTATE THE W REG LEFT
0099 04A6                     BCF     PORTB,CK        ; SET THE CLOCK LOW
009A 0000                     NOP                     ; DELAY
                        GET_BIT                       ; BIT 5
009B 0403                     BCF     SWR,CARRY
009C 05A6                     BSF     PORTB,CK        ; SET CLOCK BIT HIGH
009D 06C6                     BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
009E 0503                     BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
009F 0371                     RLF     POSA            ; ROTATE THE W REG LEFT
00A0 04A6                     BCF     PORTB,CK        ; SET THE CLOCK LOW
00A1 0000                     NOP                     ; DELAY

                        GET_BIT                       ; BIT 4
00A2 0403                     BCF     SWR,CARRY
00A3 05A6                     BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00A4 06C6                     BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00A5 0503                     BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00A6 0371                     RLF     POSA            ; ROTATE THE W REG LEFT
00A7 04A6                     BCF     PORTB,CK        ; SET THE CLOCK LOW
00A8 0000                     NOP                     ; DELAY

                        GET_BIT                       ; BIT 3
00A9 0403                     BCF     SWR,CARRY
00AA 05A6                     BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00AB 06C6                     BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00AC 0503                     BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00AD 0371                     RLF     POSA            ; ROTATE THE W REG LEFT
00AE 04A6                     BCF     PORTB,CK        ; SET THE CLOCK LOW
00AF 0000                     NOP                     ; DELAY

                         GET_BIT                      ; BIT 2
00B0 0403                     BCF     SWR,CARRY
00B1 05A6                     BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00B2 06C6                     BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
```

```
00B3 0503                   BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00B4 0371                   RLF     POSA            ; ROTATE THE W REG LEFT
00B5 04A6                   BCF     PORTB,CK        ; SET THE CLOCK LOW
00B6 0000                   NOP                     ; DELAY

                      GET_BIT                       ; BIT 1
00B7 0403                   BCF     SWR,CARRY
00B8 05A6                   BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00B9 06C6                   BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00BA 0503                   BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00BB 0371                   RLF     POSA            ; ROTATE THE W REG LEFT
00BC 04A6                   BCF     PORTB,CK        ; SET THE CLOCK LOW
00BD 0000                   NOP                     ; DELAY

                      GET_BIT                        ; BIT 0
00BE 0403                   BCF     SWR,CARRY
00BF 05A6                   BSF     PORTB,CK        ; SET CLOCK BIT HIGH
00C0 06C6                   BTFSC   PORTB,BV        ; LOOK AT DATA COMMING IN
00C1 0503                   BSF     SWR,CARRY       ; SET THE CARRY FOR A 1
00C2 0371                   RLF     POSA            ; ROTATE THE W REG LEFT
00C3 04A6                   BCF     PORTB,CK        ; SET THE CLOCK LOW
00C4 0000                   NOP                     ; DELAY

00C5 05E6                   BSF     PORTB,CSN       ; DESELECT THE CHIP


                    ;****************** CALCULATING THE PID TERMS **********************

                    ;****CALCULATE THE ERROR*******
                    ; The error is very simply the signed difference between where the
                    ; system is and where it is supposed to be at a particular instant
                    ; in time.  It is formed by subtracting the actual position from the
                    ; requested position (Position requested - Position actual).  This
                    ; difference is then used to determine the proportional,integral and
                    ; differential term contributions to the output.

                    C_ERR
00C6 0211                   MOVF    POSA,0          ; LOAD THE ACTUAL POSITION INTO W
00C7 0090                   SUBWF   POSR,0          ; SUBTRACT IT FROM THE REQUESTED POSITION
00C8 0603                   BTFSC   SWR,CARRY       ; CHECK THE CARRY BIT TO DETERMINE THE SIGN
00C9 0ACB                   GOTO    PLS_ER          ; ITS POSATIVE  (POSR>POSA)
00CA 0ACE                   GOTO    MNS_ER          ; ITS NEGATIVE  (POSA>POSR)

                    PLS_ER
00CB 002C                   MOVWF   ERROR           ; SAVE THE DIFFERENCE IN “ERROR”
00CC 0419                   BCF     FLAGS,ER_SGN    ; SET THE SIGN FLAG TO INDICATE POSATIVE
00CD 0AD2                   GOTO    CE_EXIT

                    MNS_ER
00CE 0210                   MOVF    POSR,0          ; RE-DO THE SUBTRACTION
00CF 0091                   SUBWF   POSA,0          ; ACTUAL - REQUESTED
00D0 002C                   MOVWF   ERROR           ; STORE THE DIFFERENCE IN “ERROR”
00D1 0519                   BSF     FLAGS,ER_SGN    ; SET THE SIGN FLAG FOR NEGATIVE

                    CE_EXIT
00D2 006D                   CLRF    SUMLO           ; CLEAN OLD VALUES OUT TO PREPARE
00D3 0078                   CLRF    SUMHI           ; FOR THIS CYCLES SUMMATION

                    ;****CALCULATE THE PROPORTIONAL TERM******
                    ; The proportional term is the error times the proportional gain term.
                    ; This term simply gives you more output drive the farther away you are
                    ; from where you want to be (error)*Kp.
                    ; The proportional gain term is a signed term between -100 and 100  The
                    ; more proportional gain you have the lower your system following error
                    ; will be.  The higher your proportional gain, the more integral and
                    ; differential term gains you will have to add to make the system stable.
                    ; The sum is being carried as a 16 bit signed value.

                    C_PROP
```

```
00D4 020C                   MOVF    ERROR,0         ; LOAD THE ERROR TERM INTO W
00D5 0033                   MOVWF   mulcnd          ; MULTIPLY IT BY THE PROPORTIONAL GAIN
00D6 0C30                   MOVLW   KP              ; KP AND THEN SCALE IT DOWN BY DIVIVING
00D7 0034                   MOVWF   mulplr          ; IT DOWN BY 16. IF IT IS STILL OVER
00D8 0901                   CALL    mpy_S           ; 255 THEN LIMIT IT TO 255
00D9 091D                   CALL    DIV_LMT

                    RESTORE_SGN
00DA 0719                   BTFSS   FLAGS,ER_SGN    ; IF THE ERROR SIGN IS NEGATIVE THEN
00DB 0ADE                   GOTO    ADDPROP         ; PUT THE SIGN INTO THE LOW BYTE
00DC 0276                   COMF    L_byte,1
00DD 02B6                   INCF    L_byte,1

                    ADDPROP
00DE 0216                   MOVF    L_byte,W        ; SAVE THE PROPORTIONAL PART
00DF 01ED                   ADDWF   SUMLO,1         ; IN THE SUM
00E0 0603                   BTFSC   SWR,CARRY       ; IF THE ADDITION CARRIED OUT THEN
00E1 02B8                   INCF    SUMHI,1         ; INCREMENT THE HIGH BYTE
00E2 0C00                   MOVLW   0               ; THEN
00E3 06ED                   BTFSC   SUMLO,7         ; SIGN EXTEND TO THE UPPER
00E4 0CFF                   MOVLW   0FF             ; BYTE
00E5 01F8                   ADDWF   SUMHI,1



                    ;****CALCULATE THE INTEGRAL TERM******
                    ; The integral term is an accumulation of the error thus far.  Its purpose
                    ; is to allow even a small error to effect a large change.  It does this
                    ; by adding a small number into an accumulator each cycle through the pro
                    ; Thusly even a small error that exist for a while will build up to a large
                    ; enough number to effect an output sufficient to move the system.  The
                    ; that this integral accumulator has is modulated by the integral gain term
                    ; The integral of the error over time is multiplied be KI and the result is
                    ; contribution to the final summation for determining the output value.  This
                    ; term helps to insure the long-term accuracy of the system is good.  A
                    ; amount is necessary for this purpose but too much will cause oscillations.
                    ; The integral is bounded in magnitude for two purposes.  The first is so
                    ; it never rolls over and changes sign.  The second is that it may saturate
                    ; long moves forcing an excessively large overshoot to “de-integrate” the
                    ; accumulated during the first of the move

                    C_INT
00E6 020C                   MOVF    ERROR,W         ; MOVE THE ERROR INTO THE W REG
00E7 0643                   BTFSC   SWR,Z           ; AND CHECK TO SEE IF IT IS ZERO
00E8 0AFF                   GOTO    ADDINT          ; IF SO THEN DONT CHANGE THE ACCUMULATOR
00E9 0619                   BTFSC   FLAGS,ER_SGN    ; TEST THE FLAGS TO FIND THE POLARITY
00EA 0AEE                   GOTO    MNS_1           ; OF THE ERROR .. 0 POSATIVE 1 NEGATIVE
                    PLS_1
00EB 0C02                   MOVLW   KI              ; IF POSATIVE ADD ONE TO
00EC 01EE                   ADDWF   ACCUM,1         ; THE ERROR ACCUMULATOR
00ED 0AF0                   GOTO    LMTACM          ; THEN LIMIT IT TO +/-100
                    MNS_1
00EE 0C02                   MOVLW   KI              ; IF NEGATIVE THEN SUBTRACT ONE
00EF 00AE                   SUBWF   ACCUM,1         ; FROM THE ERROR ACCUMULATOR
                    LMTACM
00F0 06EE                   BTFSC   ACCUM,7         ; CHECK THE SIGN BIT OF THE ERROR ACCUMULA-
TOR
00F1 0AF9                   GOTO    M_LMT           ; AND DO A POSATIVE OR NEGATIVE LIMIT
                    P_LMT
00F2 0C9C                   MOVLW   9CH             ; FOR THE POSATIVE LIMIT ADD 156 TO THE
00F3 01CE                   ADDWF   ACCUM,0         ; NUMBER AND SEE IF YOU GENERATE A CARRY
00F4 0703                   BTFSS   SWR,CARRY       ; BY CHECKING THE CARRY FLAG
00F5 0AFF                   GOTO    ADDINT          ; IF NOT THEN ITS O.K.
00F6 0C64                   MOVLW   64H             ; IF SO THEN FORCE THE ACCUMULATOR TO
00F7 002E                   MOVWF   ACCUM           ; 100 DECIMAL
00F8 0AFF                   GOTO    ADDINT
                    M_LMT
00F9 0C9C                   MOVLW   9CH             ; FOR THE NEGATIVE LIMIT SUBTRACT 156 FROM
00FA 008E                   SUBWF   ACCUM,0         ; THE NUMBER AND SEE IF YOU GENERATE A
```

```
00FB 0603                   BTFSC   SWR,CARRY       ; NON-CARRY CONDITION INDICATING A ROLL-OVER
00FC 0AFF                   GOTO    ADDINT          ; IF NOT THEN LEAVE THE ACCUMULATOR ALONE
00FD 0C9C                   MOVLW   9CH             ; IF SO THEN LIMIT IT TO -100 BY
00FE 002E                   MOVWF   ACCUM           ; FORCING THAT VALUE IN THE ACCUMULATOR

                    ADDINT
00FF 020E                   MOVF    ACCUM,W         ; ADD THE INTEGRAL ACCUMULATOR TO
0100 01ED                   ADDWF   SUMLO,1         ; THE LOW BYTE OF THE SUM
0101 0603                   BTFSC   SWR,CARRY       ; TEST FOR OVERFLOW, IF SO THEN
0102 02B8                   INCF    SUMHI,1         ; INCREMENT THE HI BYTE
0103 0C00                   MOVLW   0               ; LOAD 0 INTO THE W REGISTER
0104 06EE                   BTFSC   ACCUM,7         ; IF THE INTEGRAL ACCUMULATOR WAS NEGATIVE
0105 0240                   COMF    W,W             ; COMPLEMENT THE 0 TO GET SIGN FOR HIGH BYTE
0106 01F8                   ADDWF   SUMHI,1         ; ADD INTO THE HIGH BYTE OF THE SUM

                    U_DEXIT                        ; EXIT POINT FOR THE UP/DOWN CONTROL OF ACCUM


                    ;****CALCULATING THE DIFFERENTIAL TERM*************************
                    ; The differential term examines the error and determines how much
                    ; it has changed since the last cycle.  It does this by subtracting the
                    ; old error from the new error.  Since the cycle time is relatively fixed
                    ; we can use it as the “dt” of the desired “de/dt”.  This derivative of the
                    ; error is then multiplied by the differential gain term KD and becomes the
                    ; differential term contribution for the final summation.

                    ; First, create the “de” term by doing a signed subtaction of new error
                    ; minus the old error. (new_error - old_error)

                    C_DIFF
0107 020C                   MOVF    ERROR,W         ; LOAD THE NEW ERROR INTO REGISTER
0108 0719                   BTFSS   FLAGS,ER_SGN
0109 0B0D                   GOTO    LO_BYTE
010A 026C                   COMF    ERROR,1         ; CORRECT THE VALUE TO BE 16 BIT
010B 028C                   INCF    ERROR,W
010C 026C                   COMF    ERROR,1         ; RESTORE IT FOR FUTURE USE TO 8 BIT MAGNI-
TUDE
                    LO_BYTE
010D 0034                   MOVWF   ACCbLO          ; FOR SUBTRACTION
010E 0C00                   MOVLW   00
010F 0619                   BTFSC   FLAGS,ER_SGN    ; SIGN EXTEND THE UPPER BYTE
0110 0CFF                   MOVLW   0FF
0111 0036                   MOVWF   ACCbHI
0112 020F                   MOVF    ERR_O,W         ; LOAD THE OLD ERROR INTO OTHER REGISTER
0113 0799                   BTFSS   FLAGS,OER_SGN
0114 0B17                   GOTO    LO_BYTEO
0115 026F                   COMF    ERR_O,1         ; CORRECT THE VALUE TO BE 16 BIT
0116 028F                   INCF    ERR_O,W
                    LO_BYTEO
0117 0033                   MOVWF   ACCaLO          ; FOR SUBTRACTION
0118 0C00                   MOVLW   00
0119 0699                   BTFSC   FLAGS,OER_SGN   ; SIGN EXTEND THE UPPER BYTE
011A 0CFF                   MOVLW   0FF
011B 0035                   MOVWF   ACCaHI
011C 090F                   CALL    D_sub           ; PERFORM THE SUBTRACTION

                    STRIP_SGN
011D 06F6                   BTFSC   ACCbHI,7        ; TEST THE SIGN OF THE RESULT
011E 0B20                   GOTO    NEG_ABS
011F 0B25                   GOTO    POS_ABS
                    NEG_ABS
0120 0559                   BSF     FLAGS,DE_SGN    ; ITS NEGATIVE SO SET THE FLAG AND
0121 0274                   COMF    ACCbLO,1        ; COMPLEMENT THE VALUE
0122 0294                   INCF    ACCbLO,W
0123 002F                   MOVWF   ERR_O
0124 0B28                   GOTO    MULT_KD
                    POS_ABS
0125 0459                   BCF     FLAGS,DE_SGN    ; ITS POSATIVE SO SET RESET THE FLAG
```

```
0126 0214                   MOVF    ACCbLO,W        ; AND SAVE THE VALUE
0127 002F                   MOVWF   ERR_O

                    ; Then multiply by Kd

                    MULT_KD
0128 020F                   MOVF    ERR_O,W
0129 0033                   MOVWF   mulcnd          ; MOVE THE DE/DT TERM INTO THE MULCND REG.
012A 0C20                   MOVLW   KD              ; MOVE THE DIFFERENTIAL GAIN TERM INTO
012B 0034                   MOVWF   mulplr          ; MULPLR TO MULTIPLY THE DE/DT
012C 0901                   CALL    mpy_S           ; DO THE MULTIPLICATION
012D 091D                   CALL    DIV_LMT         ; SCALE AND LIMIT TO 100

                    RE_SGN
012E 0759                   BTFSS   FLAGS,DE_SGN    ; IF THE DE SIGN IS NEGATIVE THEN
012F 0B32                   GOTO    SAVE_DIFF       ; PUT THE SIGN INTO THE LOW BYTE
0130 0276                   COMF    L_byte,1
0131 02B6                   INCF    L_byte,1
                    SAVE_DIFF
0132 0216                   MOVF    L_byte,W
0133 0643                   BTFSC   SWR,Z
0134 0B45                   GOTO    ROLL_ER
0135 002F                   MOVWF   ERR_O

                    ; ADD THE DIFF TERM INTO THE SUMM ***************
                    ADDDIF
0136 0C00                   MOVLW   00
0137 0659                   BTFSC   FLAGS,DE_SGN    ; PUT THE KD*(DE/DT) TERM INTO THE
0138 0CFF                   MOVLW   0FF             ; REGISTERS TO ADD. AND
0139 0036                   MOVWF   ACCbHI          ; SIGN EXTEND THE UPPER BYTE
013A 020F                   MOVF    ERR_O,W
013B 0034                   MOVWF   ACCbLO
013C 020D                   MOVF    SUMLO,W         ; LOAD THE CURRENT SUM INTO THE
013D 0033                   MOVWF   ACCaLO          ; REGISTERS TO ADD
013E 0218                   MOVF    SUMHI,W
013F 0035                   MOVWF   ACCaHI
0140 0910                   CALL    D_add           ; ADD IN THE DIFFERENTIAL TERM
0141 0214                   MOVF    ACCbLO,W        ; SAVE THE RESULTS BACK
0142 002D                   MOVWF   SUMLO           ; INTO SUMLO AND HI
0143 0216                   MOVF    ACCbHI,W
0144 0038                   MOVWF   SUMHI

                    ROLL_ER
0145 020C                   MOVF    ERROR,W         ; TAKE THE CURRENT ERROR
0146 002F                   MOVWF   ERR_O           ; AND PUT IT IN THE ERROR HISTORY
0147 0499                   BCF     FLAGS,OER_SGN   ; SAVE THE CURRENT ERROR SIGN
0148 0619                   BTFSC   FLAGS,ER_SGN    ; IN THE OLD ERROR SIGN FOR
0149 0599                   BSF     FLAGS,OER_SGN   ; NEXT TIME THROUGH


                    ;****SET UP THE DIRECTION FOR THE BRIDGE*********************
                    ;
                    ; After the sum of all the components has been made, the sign of the
                    ; sum will determine which way the bridge should be powered.
                    ; If the sum is negative the bridge needs to be set to drive ccw; if the
                    ; sum is posative then the bridge needs to be set to drive cw.  This
                    ; is purely a convention and depends upon the polarity the motor and feedback
                    ; element are hooked up in.

                    SET_DIR
014A 0479                   BCF     FLAGS,DIR       ; SET FOR DEFAULT CLOCKWISE
014B 06F8                   BTFSC   SUMHI,7         ; LOOK AT THE SIGN BIT, IF IT IS SET
014C 0579                   BSF     FLAGS,DIR       ; THEN SET FOR CCW BRIDGE DRIVE


                    ;**** SCALE THE NUMBER TO BETWEEN 0 AND 100% **********************
                    ; After the direction is set the request for duty cycle is limited to between
                    ; 0 and 100 percent inclusive.  This value is passed to the dutycycle setting
                    ; routine by loading it in the variable “PCNT”.
```

```
                        L_SUMM
014D 07F8                       BTFSS   SUMHI,7         ; CHECK TO SEE IF IT IS NEGATIVE
014E 0B52                       GOTO    POS_LM
014F 0278                       COMF    SUMHI,1
0150 026D                       COMF    SUMLO,1
0151 02AD                       INCF    SUMLO,1

                        POS_LM
0152 0C01                       MOVLW   1H              ; SUBTRACT 1 FROM THE HIGH BYTE TO SEE
0153 0098                       SUBWF   SUMHI,0         ; IF THERE IS ANYTHING THERE, IF NOT,
0154 0703                       BTFSS   SWR,CARRY       ; THEN LEAVE THE LOW BYTE ALONE
0155 0B59                       GOTO    LB_L            ; OTHERWISE GIVE THE LOW BYTE A FULL
0156 0C64                       MOVLW   64H             ; COUNT AND IT WILL HAVE BEEN LIMITED
0157 002D                       MOVWF   SUMLO           ; TO 100
0158 0B5F                       GOTO    LP_EXIT         ; GOTO LIMIT PERCENT EXIT
                        LB_L
0159 0C64                       MOVLW   64H             ; LIMIT THE MAGNITUDE OF THE VALUE TO
015A 008D                       SUBWF   SUMLO,0         ; 100 DECIMAL
015B 0703                       BTFSS   SWR,CARRY
015C 0B5F                       GOTO    LP_EXIT
015D 0C64                       MOVLW   64H
015E 002D                       MOVWF   SUMLO

                        LP_EXIT
015F 020D                       MOVF    SUMLO,W         ; STORE THE LIMITED VALUE IN
0160 0029                       MOVWF   PCNT            ; THE PERCENT DUTYCYCLE REQUEST


                        ;*********************************************************
                        ; PWM GENERATING ROUTINE
                        ;
                        ; The important thing here is not to have to do too many decisions or
                        ; calculations while you are generating the 100 or so pulses.  These will
                        ; take time and limit the minimum or maximum duty cycle.

                        WHICH_DIR
0161 0679                       BTFSC   FLAGS,DIR       ; CHECK THE DIRECTION FLAG
0162 0B76                       GOTO    GOCCW           ; DO CCW PULSES FOR 1
0163 0B64                       GOTO    GOCW            ; DO CW PULSES FOR 0


                        GOCW
0164 0426                       BCF     PORTB,PWMCCW    ; SET THE BRIDGE FOR CW MOVE
0165 0C64                       MOVLW   64H             ;
0166 0032                       MOVWF   CYCLES          ; SET UP CYCLES COUNTER FOR 100 PULSES
0167 0943                       CALL    CALCTIMES       ; CALCULATE THE HI AND LO TIMES

                        RLDCW
0168 0207                       MOVF    HI,0            ; RE LOAD THE HI TIMER
0169 002A                       MOVWF   HI_T            ; WITH THE CALCULATED TIME
016A 0208                       MOVF    LO,0            ; RE LOAD THE LO TIMER
016B 002B                       MOVWF   LO_T            ; WITH THE CALCULATED TIME
016C 0004                       CLRWDT                  ; TAG THE WATCHDOG TIMER

                        CWHI
016D 0506                       BSF     PORTB,PWMCW     ; SET THE CLOCKWISE PWMBIT HIGH
016E 02EA                       DECFSZ  HI_T,1          ; DECREMENT THE HI USEC. COUNTER
016F 0B6D                       GOTO    CWHI            ; DO ANOTHER LOOP
                        CWLO
0170 0406                       BCF     PORTB,PWMCW     ; SET THE CLOCKWISE PWM BIT LOW
0171 02EB                       DECFSZ  LO_T,1          ; DECREMENT THE LO USEC. COUNTER
0172 0B70                       GOTO    CWLO            ; DO ANOTHER LOOP
0173 02F2                       DECFSZ  CYCLES,1        ; DECREMENT THE NUMBER OF CYCLES LEFT
0174 0B68                       GOTO    RLDCW           ; DO ANOTHER PULSE
0175 0A50                       GOTO    BEGIN           ; DO ANOTHER MAIN SYSTEM CYCLE
```

```
                        GOCCW
0176 0406                       BCF     PORTB,PWMCW     ; SET THE BRIDGE FOR CCW MOVE
0177 0C64                       MOVLW   64H             ;
0178 0032                       MOVWF   CYCLES          ; SET UP CYCLE COUNTER FOR 100 PULSES
0179 0943                       CALL    CALCTIMES       ; CALCULATE THE HI AND LO TIMES
                        RLDCCW
017A 0207                       MOVF    HI,0            ; RE LOAD THE HI TIMER
017B 002A                       MOVWF   HI_T            ; WITH THE CALCULATED TIME
017C 0208                       MOVF    LO,0            ; RE LOAD THE LO TIMER
017D 002B                       MOVWF   LO_T            ; WITH THE CALCULATED TIME
017E 0004                       CLRWDT                  ; TAG THE WATCHDOG

                        CCWHI
017F 0526                       BSF     PORTB,PWMCCW    ; SET THE COUNTERCLOCKWISE PWM BIT HIGH
0180 02EA                       DECFSZ  HI_T,1          ; DECREMENT THE HI USEC. COUNTER
0181 0B7F                       GOTO    CCWHI           ; DO ANOTHER LOOP
                        CCWLO
0182 0426                       BCF     PORTB,PWMCCW    ; SET THE COUNTERCLOCKWISE PWM BIT LOW
0183 02EB                       DECFSZ  LO_T,1          ; DECREMENT THE LO USEC. COUNTER
0184 0B82                       GOTO    CCWLO           ; DO ANOTHER LOOP
0185 02F2                       DECFSZ  CYCLES,1        ; DECREMENT THE NUMBER OF CYCLES LEFT
0186 0B7A                       GOTO    RLDCCW          ; DO ANOTHER PULSE
0187 0A50                       GOTO    BEGIN           ; DO ANOTHER MAIN SYSTEM CYCLE



                          ;************* START VECTOR ********************

                        CLRREG                          ;INITIALIZE REGISTERS

0188 0C0B                       MOVLW   0BH             ; SET PORT A FOR 3 INPUTS AND
0189 0005                       TRIS    PORTA           ; AN OUTPUT
018A 0C1C                       MOVLW   1CH             ; SET PORT B FOR INPUTS AND OUTPUTS
018B 0006                       TRIS    PORTB           ; THIS SETTING FOR SENDING TO A/D
018C 0040                       CLRW                    ; CLEAR THE W REGISTER
018D 0002                       OPTION                  ; STORE THE W REG IN THE OPTION REG
018E 0C08                       MOVLW   08H             ; STARTING REGISTER TO ZERO
018F 0024                       MOVWF   FSR             ;
                        GCLR
0190 0060                       CLRF    00              ;
0191 03E4                       INCFSZ  FSR             ; SKIP AFTER ALL REGISTERS
0192 0B90                       GOTO    GCLR            ; HAVE BEEN INITIALIZED
0193 0A50                       GOTO    BEGIN           ; START AT THE BEGINING OF THE PROGRAM

                                ORG     01FF            ;
01FF 0B88                       GOTO    CLRREG          ; START VECTOR


                              END


Errors   :     0
Warnings :     0
```

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.mchip.com/microhip

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: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
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

AMERICAS (continued)

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

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

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

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
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 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
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 Muenchen, Germany
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

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

9/22/95