Four Channel Digital Voltmeter with Display and Keyboard

INTRODUCTION

The PIC16C71 is a member of a new family of 8-bit microcontrollers, namely the PIC16CXX. The salient features of the PIC16C71 are:

- Improved and enhanced instruction set
- 14-bit instruction word
- Interrupt capability
- On-chip four channel, 8-bit A/D converter

This application note demonstrates the capability of the PIC16C71. To make this application note easier for the end user, it has been broken down into four sub-sections:

Section 1: Implements the multiplexing of four 7 segment LEDs with the PIC16C71.
Section 2: Implements the multiplexing of four 7 segment LEDs as well as the sampling of a 4x4 Keypad.
Section 3: Implements the multiplexing of four 7 segment LEDs as well as the sampling of one A/D channel.
Section 4: Implements the multiplexing of four 7 segment LEDs, sampling a 4x4 keypad and four A/D channels.

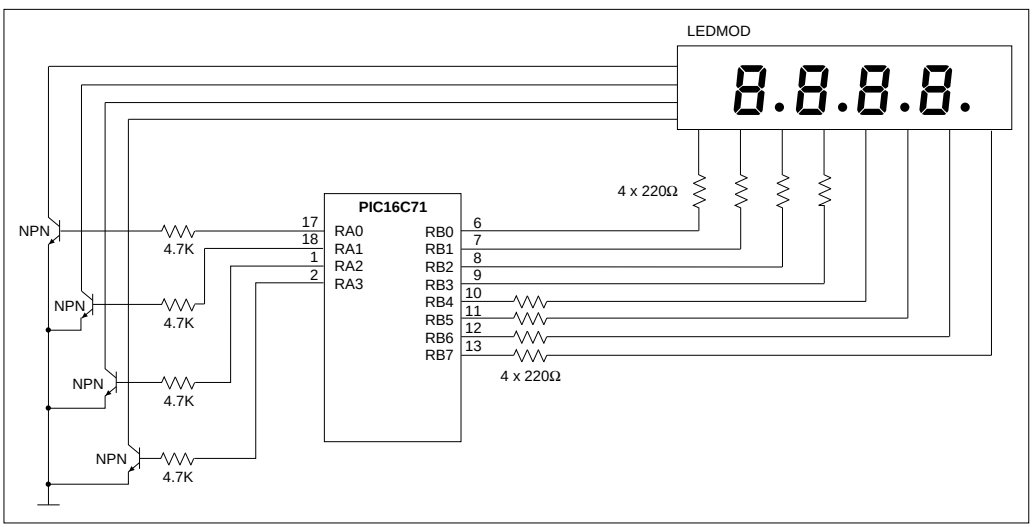
IMPLEMENTATION

SECTION 1: MULTIPLEXING FOUR 7 SEGMENT LED DISPLAYS

Hardware

The PIC16C71's I/O ports have an improved sink/source specification. Each I/O pin can sink up to 25 mA and source 20 mA, in addition total Port B source current is 100 mA and sink current is 150 mA. Port A is rated for 50 mA source current and 80 mA sink current. This makes the PIC16C71 ideal for driving 7 segment LEDs. Since the total number of I/O is limited to 13, the 8-bit Port B is used to drive the 8 LEDs, while external sink transistors or MOSFETs are used to sink the digit current (See Figure 1). Another alternative is to use ULN2003 open collector sink current drivers, which are available in 16 pin DIP or very small S0-16 packages. Each transistor on the ULN2003 can sink a maximum of 500 mA and the base drive can be directly driven from the Port A pins.

FIGURE 1 - MULTIPLEXING FOUR 7 SEGMENT LEDS



Software

The multiplexing is achieved by turning on each LED for a 5 msec duration every 20 µs. This gives an update rate of 50 Hz, which is quite exceptable to the human eye as a steady display. The 5 msec time base is generated by dividing the 4.096 MHz oscillator clock. The internal prescaler is configured to be a divide by 32 and assigned to the RTCC. The RTCC is pre-loaded with a value = 96. The RTCC will increment to FF and then roll over to 00 after a period = (256-96)*(32*4/4096000) = 5 µs. When the RTCC rolls over, the RTIF flag is set and since the RTIE and GIE bits are enabled, an interrupt is generated.

The software implements a simple timer which increments at a 1 second rate. Every second, the 4 nibble (two 8-bit registers MsdTime and LsdTime) are incremented in a BCD format. The lower 4 bits of LsdTime correspond to the least significant digit (LSD) on the display. The high 4 bits of LsdTime correspond to the second significant digit of the display and so on. Depending on which display is turned on, the corresponding 4 bit BCD value is extracted from either MsdTime or LsdTime, and decoded to a 7 segment display. The RTCC interrupt is generated at a steady rate of 5 µs and given an instruction time of 1 us. The entire display update program can reside in the interrupt service routine with no chance of getting an interrupt within an interrupt. The Code listing for Section 1 is in Appendix A.

SECTION 2: MULTIPLEXING FOUR 7 SEGMENT LED DISPLAYS AND SCANNING A 4X4 KEYPAD

Hardware

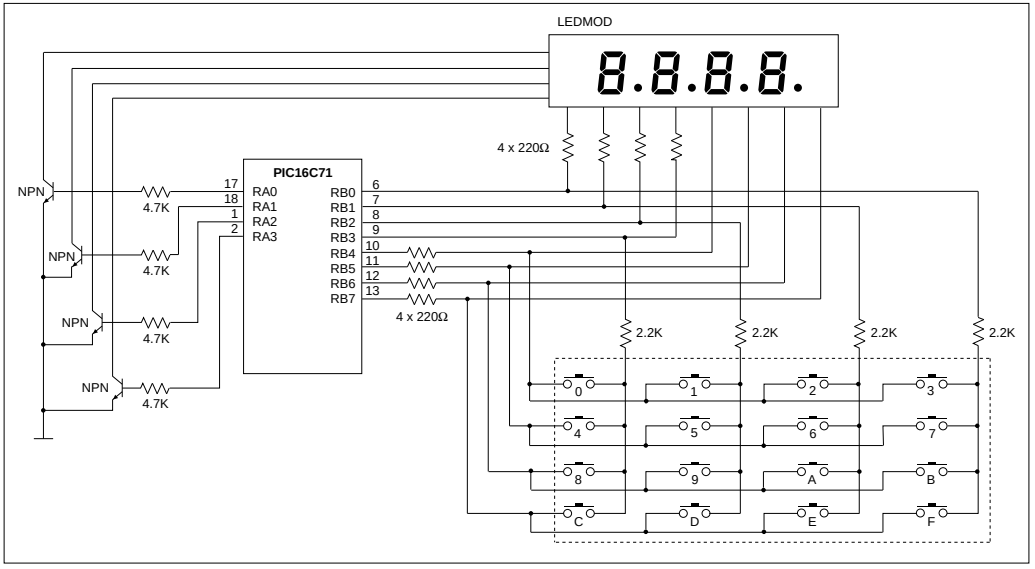
A 4x4 keypad can be very easily interfaced to the PIC16C71's Port B (see Figure 2). Internal pull-ups on pins RB4 to RB7 can be enabled and disabled by setting the RBPU bit in the OPTION register. The internal pull-ups have a value of 20K at 5V (typical). In order to sense a low level at the input, the switch is "connected" to ground through a 2.2K resistor. A key hit normally lasts from 50 msec to as long as a person holds the key down. In order not to miss any key hits, the keypad is sampled every 20 µs (just after the update of the MSD).

Software

To sample the keypad, the digit sinks are first disabled. Port B is then configured with RB4-RB7 as inputs and RB0 -RB3 as outputs driven high. The pull-ups on RB4-RB7 are enabled. Sequentially RB0 to RB3 are made low while RB4 to RB7 are checked for a key hit (a low level). One key hit per scan is demonstrated in this program. Multiple key hits per scan can very easily be implemented. Once the key hit is sensed, a 40 msec debounce period elapses before key sampling is resumed. No more key hits are sensed until the present key is released. This prevents erroneous key inputs.

The program basically inputs the key hit and displays its value as a hexadecimal character on the multiplexed 7-segment LEDs. The Code Listing for Section 2 is in Appendix B.

FIGURE 2 - MULTIPLEXING FOUR 7 SEGMENT LEDS WITH A 4X4 KEYPAD



SECTION 3: MULTIPLEXING FOUR 7 SEGMENT LED DISPLAYS AND THE A/D CHANNEL 0

Hardware

The four analog channels are connected to RA0-RA3. If any of these pins are used normally as digital I/O, they can momentarily be used as analog inputs. In order to avoid interference from the analog source, it is advisable to buffer the analog input through a voltage follower op amp, however, it is not always necessary. Figure 3A and 3B show some typical configurations. In this application, the analog input is a potentiometer whose wiper is connected through an RC network to channel 0. The RC is necessary in order to smooth out the analog voltage. The RC does contribute to a delay in the sampling time, however the stability of the analog reading is greatly improved.

Software

The analog input is sampled every 20 msec. The digit sinks and the drivers are turned off i.e. Port A is configured as an input and Port B outputs are made low. A 1msec settling time is allowed for the external RC network connected to the analog input to settle and then the A/D conversion is started. The result is read then converted from an 8-bit binary value to a 3-digit BCD value which is then displayed on the 7 Segment LEDs. The Code Listing for Section 3 is in Appendix C.

FIGURE 3A - TYPICAL CONNECTION FOR ANALOG/DIGITAL INPUT

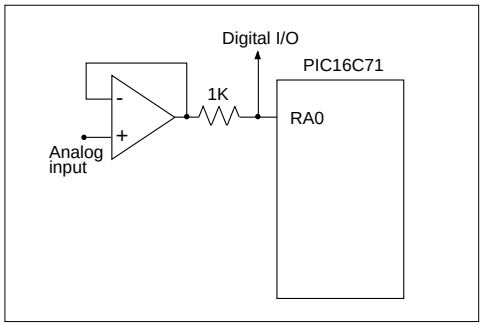
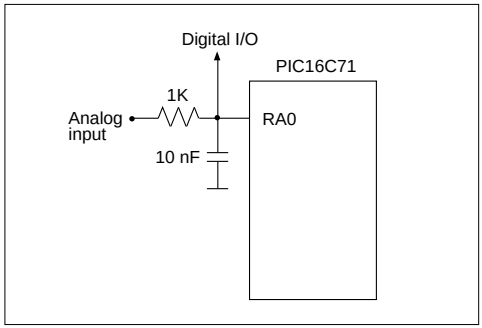


FIGURE 3B - TYPICAL CONNECTION FOR ANALOG/DIGITAL INPUT



SECTION 4: MULTIPLEXING FOUR 7 SEGMENT LED DISPLAYS WITH A 4X4 KEYPAD AND 4 A/D CHANNELS

Hardware

This section essentially incorporated Sections 1, 2 and 3 to give a complete four channel voltmeter. Figure 4 shows a typical configuration. The analog channels are connected through individual potentiometers to their respective analog inputs and are sampled every 20 msec in a round robin format. The sampling rate can be increased to as fast as once every 5 µs if required. The keypad sampling need not be any faster than once every 20 µs.

Software

The program samples the analog inputs and saves the result in four consecutive locations starting at "ADVALUE", with channel 0 saved at the first location and so on. By default, channel 0 is displayed. If Key 1 is pressed, channel 1 is displayed and so on. Key hits > 3 are ignored. The Code Listing for Section 4 is in Appendix D.

Code Size

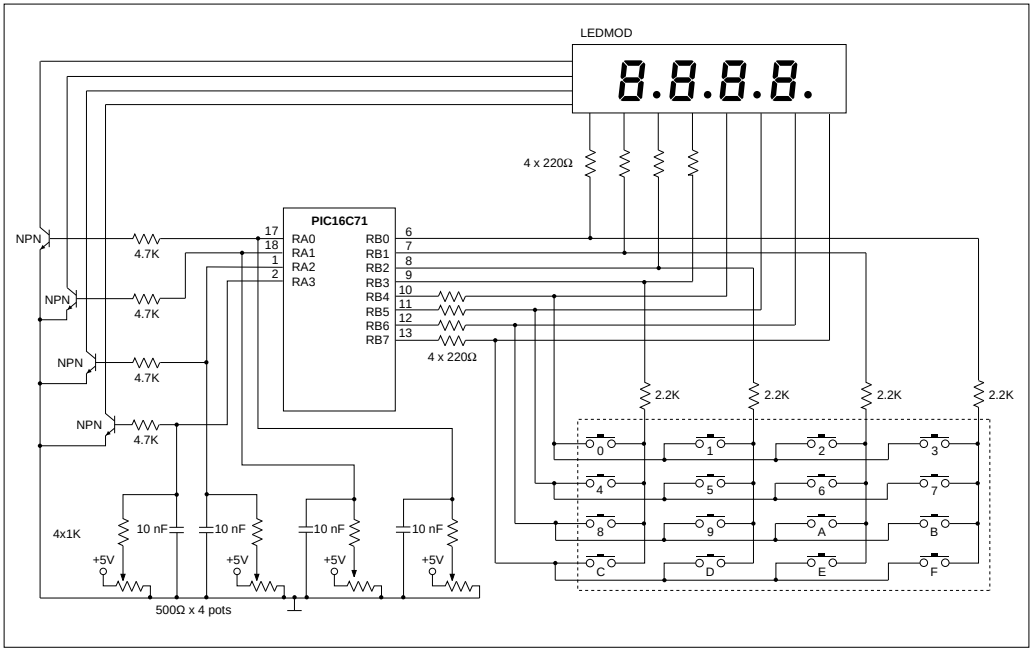
Section 1:	Program Memory:	139
	Data Memory:	6
Section 2:	Program Memory:	207
	Data Memory:	13
Section 3:	Program Memory:	207
	Data Memory:	17
Section 4:	Program Memory:	
	Data Memory:	

Conclusion

The four A/D channels on the PIC16C71 can be multiplexed with digital I/O, thus reducing overall pin counts and improving I/O pin usage in an analog application.

Author: Stan D'Souza, Logic Products Division

FIGURE 4 - FOUR CHANNEL VOLTMETER WITH DISPLAY AND KEYPAD



APPENDIX A

```
MPASM B0.50                                                    PAGE  1



                        ;*****************************************************************
                        ;This program is to demonstrate how to multiplex four 7 segment LED
                        ;digits using a PIC16C71. The four digits will start at 0000 and
                        ;increment at a 1 sec rate up to 9999.
                        ;The LEDs are updated every 5 msec, for a multiplexing rate of 20 msec.
                        ;The RTCC timer is used in internal interrupt mode to generate the 5 msec.
                        ;
                        ;                                       Stan D'Souza 5/8/93
                        ;*****************************************************************
                               LIST P=16C71, F=INHX8M
                        ;
                               include "picreg.equ"



                        ;
000C                    TempC          equ     0x0c           ;temp general purpose regs
000D                    TempD          equ     0x0d
000E                    TempE          equ     0x0e
000F                    Count          equ     0x0f           ;count
0010                    MsdTime        equ     0x10           ;most significant Timer
0011                    LsdTime        equ     0x11           ;Least significant Timer
0001                    OptionReg      equ     1
0002                    PCL            equ     2
0026                    BcdMsd         equ     26
0027                    Bcd            equ     27
                        ;
                                       org     0
0000 2805                              goto    Start          ;skip over interrupt vector
                        ;
                                       org     4
0004 281D                              goto    ServiceInterrupts
                        ;
                        Start
0005 2008                              call    InitPorts
0006 2012                              call    InitTimers
                        loop
0007 2807                              goto    loop
                        ;
                        InitPorts
0008 1683                              bsf     STATUS,RP0     ;select pg 1
0009 3003                              movlw   3              ;make RA0-3 digital I/O
000A 0108                              movwf   ADCON1         ;       /
000B 0205                              clrf    TRISA          ;make RA0-4 outputs
000C 0206                              clrf    TRISB          ;make RB0-7 outputs
000D 1283                              bcf     STATUS,RP0     ;select page 0
000E 0185                              clrf    PORT_A         ;make all outputs low
000F 0186                              clrf    PORT_B         ;       /
0010 1585                              bsf     PORT_A,3       ;enable MSB digit sink
0011 0008                              return
                        ;
                        ;
                        ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
                        ;the rtcc will be incremented every 31.25uS. If rtcc is preloaded
                        ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5msec. So the
                        ;end result is that we get a rtcc interrupt every 5msec.
                        InitTimers
0012 0190                              clrf    MsdTime        ;clr timers
0013 0191                              clrf    LsdTime        ;       /
0014 1683                              bsf     STATUS,RP0     ;select pg 1
0015 3084                              movlw   B'10000100'    ;assign ps to rtcc
```



```
0016 0081                       movwf   OptionReg              ;ps = 32
0017 1283                       bcf     STATUS,RP0             ;select pg 0
0018 3020                       movlw   B'00100000'            ;enable rtcc interrupt
0019 008B                       movwf   INTCON                 ;
001A 3060                       movlw   .96                    ;preload rtcc
001B 0081                       movwf   RTCC                   ;start counter
001C 0009                       retfie
                        ;
                        ServiceInterrupts
001D 190B                       btfsc   INTCON,RTIF            ;rtcc interrupt?
001E 2822                       goto    ServiceRTCC            ;yes then service
001F 3020                       movlw   B'00100000'            ;else clr rest
0020 008B                       movwf   INTCON
0021 0009                       retfie
                        ;
                        ServiceRTCC
0022 3060                       movlw   .96                    ;initialize rtcc
0023 0081                       movwf   RTCC
0024 110B                       bcf     INTCON,RTIF            ;clr int flag
0025 2028                       call    IncTimer               ;inc timer
0026 2050                       call    UpdateDisplay          ;update display
0027 0009                       retfie
                        ;
                        ;The display is incremented every 200*5msec = 1 Sec.
                        IncTimer
0028 0A0F                       incf    Count,w                ;inc count
0029 3AC8                       xorlw   .200                   ;= 200?
002A 1903                       btfsc   STATUS,Z               ;no then skip
002B 282E                       goto    DoIncTime              ;else inc time
002C 0A8F                       incf    Count
002D 0008                       return
                        DoIncTime
002E 018F                       clrf    Count                  ;clr count
002F 0A11                       incf    LsdTime,w              ;get lsd
0030 390F                       andlw   0x0F                   ;mask high nibble
0031 3A0A                       xorlw   0x0a                   ; = 10?
0032 1903                       btfsc   STATUS,Z               ;no then skip
0033 2836                       goto    IncSecondLsd           ;inc next lsd
0034 0A91                       incf    LsdTime                ;else inc timer
0035 0008                       return
                        IncSecondLsd
0036 0E11                       swapf   LsdTime,w              ;get hi in low nibble
0037 390F                       andlw   0x0F                   ;mask hi nibble
0038 3E01                       addlw   1                      ;inc it
0039 0091                       movwf   LsdTime                ;restore back
003A 0E91                       swapf   LsdTime                ;       /
003B 3A0A                       xorlw   0x0a                   ; = 10?
003C 1903                       btfsc   STATUS,Z               ;no then skip
003D 283F                       goto    IncThirdLsd            ;else inc next lsd
003E 0008                       return
                        IncThirdLsd
003F 0191                       clrf    LsdTime
0040 0A10                       incf    MsdTime,w              ;get 3rd lsd
0041 390F                       andlw   0x0F                   ;mask hi nibble
0042 3A0A                       xorlw   0x0a                   ;= 10?
0043 1903                       btfsc   STATUS,Z               ;no then skip
0044 2847                       goto    IncMsd                 ;else Msd
0045 0A90                       incf    MsdTime                ;else inc timer
0046 0008                       return
                        IncMsd
0047 0E10                       swapf   MsdTime,w              ;get hi in lo nibble
0048 390F                       andlw   0x0F                   ;mask hi nibble
0049 3E01                       addlw   1                      ;inc timer
004A 0090                       movwf   MsdTime                ;restore back
004B 0E90                       swapf   MsdTime                ;       /
004C 3A0A                       xorlw   0x0a                   ;= 10?
004D 1903                       btfsc   STATUS,Z               ;no then skip
004E 0190                       clrf    MsdTime                ;clr msd
004F 0008                       return
```

```
                        ;
                        ;
                        UpdateDisplay
0050 0805                       movf    PORT_A,w                ;present sink value in w
0051 0185                       clrf    PORT_A                  ;disable all digits sinks
0052 390F                       andlw   0x0f
0053 008C                       movwf   TempC                   ;save sink value in tempC
0054 160C                       bsf     TempC,4                 ;preset for lsd sink
0055 0C8C                       rrf     TempC                   ;determine next sink value
0056 1C03                       btfss   STATUS,CARRY            ;c=1?
0057 118C                       bcf     TempC,3                 ;no then reset LSD sink
0058 180C                       btfsc   TempC,0                 ;else see if Msd
0059 286B                       goto    UpdateMsd               ;yes then do Msd
005A 188C                       btfsc   TempC,1                 ;see if 3rdLsd
005B 2866                       goto    Update3rdLsd            ;yes then do 3rd Lsd
005C 190C                       btfsc   TempC,2                 ;see if 2nd Lsd
005D 2861                       goto    Update2ndLsd            ;yes then do 2nd lsd
                        UpdateLsd
005E 0811                       movf    LsdTime,w               ;get Lsd in w
005F 390F                       andlw   0x0f                    ;       /
0060 286F                       goto    DisplayOut              ;enable display
                        Update2ndLsd
0061 2080                       call    Chk2LsdZero             ;msd = 0 & 2 lsd 0?
0062 1D03                       btfss   STATUS,Z                ;yes then skip
0063 0E11                       swapf   LsdTime,w               ;get 2nd Lsd in w
0064 390F                       andlw   0x0f                    ;mask rest
0065 286F                       goto    DisplayOut              ;enable display
                        Update3rdLsd
0066 2088                       call    ChkMsdZero              ;msd = 0?
0067 1D03                       btfss   STATUS,Z                ;yes then skip
0068 0810                       movf    MsdTime,w               ;get 3rd Lsd in w
0069 390F                       andlw   0x0f                    ;mask low nibble
006A 286F                       goto    DisplayOut              ;enable display
                        UpdateMsd
006B 0E10                       swapf   MsdTime,w               ;get Msd in w
006C 390F                       andlw   0x0f                    ;mask rest
006D 1903                       btfsc   STATUS,Z                ;msd != 0 then skip
006E 300A                       movlw   0x0a
                        DisplayOut
006F 2074                       call    LedTable                ;get digit output
0070 0086                       movwf   PORT_B                  ;drive leds
0071 080C                       movf    TempC,w                 ;get sink value in w
0072 0085                       movwf   PORT_A
0073 0008                       return
                        ;
                        ;
                        LedTable
0074 0782                       addwf   PCL                     ;add to PC low
0075 343F                       retlw   B'00111111'             ;led drive for 0
0076 3406                       retlw   B'00000110'             ;led drive for 1
0077 345B                       retlw   B'01011011'             ;led drive for 2
0078 344F                       retlw   B'01001111'             ;led drive for 3
0079 3466                       retlw   B'01100110'             ;led drive for 4
007A 346D                       retlw   B'01101101'             ;led drive for 5
007B 347D                       retlw   B'01111101'             ;led drive for 6
007C 3407                       retlw   B'00000111'             ;led drive for 7
007D 347F                       retlw   B'01111111'             ;led drive for 8
007E 3467                       retlw   B'01100111'             ;led drive for 9
007F 3400                       retlw   B'00000000'             ;blank led drive
                        ;
                        ;
                        Chk2LsdZero
0080 2088                       call    ChkMsdZero              ;msd = 0?
0081 1D03                       btfss   STATUS,Z                ;yes then skip
0082 0008                       return                          ;else return
0083 0E11                       swapf   LsdTime,w               ;get 2nd lsd
0084 390F                       andlw   0x0f                    ;mask of LSD
0085 1D03                       btfss   STATUS,Z                ;0? then skip
0086 0008                       return
```

```
0087 340A                       retlw   10                      ;else return with 10
                        ;
                        ChkMsdZero
0088 0810                       movf    MsdTime,w               ;get Msd in w
0089 1D03                       btfss   STATUS,Z                ;= 0? skip
008A 0008                       return                          ;else return
008B 340A                       retlw   .10                     ;ret with 10
                        ;

                                end
                        ;
```

Appendix B

```
MPASM B0.50                                                          PAGE  1

                    ;*****************************************************************
                    ;This program is to demonstrate how to multiplex four 7 segment LED
                    ;digits and a 4X4 keypad using a PIC16C71.
                    ;The four digits will start as '0000' and when a key is hit
                    ;it is displayed on the 7 segment leds as a hex value 0 to F. The last
                    ;digit hit is always displayed on the right most led with the rest of
                    ;the digits shifted to the left. The left most digit is deleted.
                    ;The LEDs are updated every 20msec, the keypad is scanned at a rate of 20
msec.
                    ;The RTCC timer is used in internal interrupt mode to generate the 5 msec.
                    ;
                    ;                                        Stan D'Souza 5/8/93
                    ;*****************************************************************
                           LIST P=16C71, F=INHX8M
                    ;
                           include "picreg.equ"


                    ;
000C                TempC           equ     0x0c            ;temp general purpose regs
000D                TempD           equ     0x0d
000E                TempE           equ     0x0e
0020                PABuf           equ     0x20
0021                PBBuf           equ     0x21
000F                Count           equ     0x0f            ;count
0010                MsdTime         equ     0x10            ;most significant Timer
0011                LsdTime         equ     0x11            ;Least significant Timer
0012                KeyFlag         equ     0x12            ;flags related to key pad
0000                keyhit          equ     0               ;bit 0 -> key-press on
0001                DebnceOn        equ     1               ;bit 1 -> debounce on
0002                noentry         equ     2               ;no key entry = 0
0003                ServKey         equ     3               ;bit 3 -> service key
0013                Debnce          equ     0x13            ;debounce counter
0014                NewKey          equ     0x14
002F                WBuffer         equ     0x2f
002E                StatBuffer      equ     0x2e
0001                OptionReg       equ     1
0002                PCL             equ     2
                    ;
                    ;
                    push            macro
                                    movwf   WBuffer         ;save w reg in Buffer
                                    swapf   WBuffer         ;swap it
                                    swapf   STATUS,w        ;get status
                                    movwf   StatBuffer      ;save it
                                    endm
                    ;
                    pop             macro
                                    swapf   StatBuffer,w    ;restore status
                                    movwf   STATUS          ;       /
                                    swapf   WBuffer,w       ;restore W reg
                                    endm
                    ;
                                    org     0
0000 280D                           goto    Start           ;skip over interrupt vector
                    ;
                                    org     4
                    ;It is always a good practice to save and restore the w reg,
                    ;and the status reg during a interrupt.
                                    push
0004 00AF                           movwf   WBuffer         ;save w reg in Buffer
0005 0EAF                           swapf   WBuffer         ;swap it
0006 0E03                           swapf   STATUS,w        ;get status
0007 00AE                           movwf   StatBuffer      ;save it
0008 2036                           call    ServiceInterrupts
```



```
                        pop
0009 0E2E               swapf   StatBuffer,w        ;restore status
000A 0083               movwf   STATUS              ;       /
000B 0E2F               swapf   WBuffer,w           ;restore W reg

000C 0009               retfie
                    ;
                    Start
000D 2020               call    InitPorts
000E 202A               call    InitTimers
                    loop
000F 1992               btfsc   KeyFlag,ServKey     ;key service pending
0010 2012               call    ServiceKey          ;yes then service
0011 280F               goto    loop
                    ;
                    ;ServiceKey, does the software service for a keyhit. After a key service,
                    ;the ServKey flag is reset, to denote a completed operation.
                    ServiceKey
0012 0814               movf    NewKey,w            ;get key value
0013 008E               movwf   TempE               ;save in TempE
0014 0E10               swapf   MsdTime,w           ;move MSD out
0015 39F0               andlw   B'11110000'         ;clr lo nibble
0016 0090               movwf   MsdTime             ;save back
0017 0E11               swapf   LsdTime,w           ;get Lsd
0018 390F               andlw   B'00001111'         ;mask off lsd
0019 0490               iorwf   MsdTime             ;and left shift 3rd
001A 0E11               swapf   LsdTime,w           ;get Lsd again
001B 39F0               andlw   B'11110000'         ;mask off 2nd
001C 040E               iorwf   TempE,w             ;or with new lsd
001D 0091               movwf   LsdTime             ;make Lsd
001E 1192               bcf     KeyFlag,ServKey     ;reset service flag
001F 0008               return
                    ;
                    InitPorts
0020 1683               bsf     STATUS,RP0          ;select pg 1
0021 3003               movlw   3                   ;make RA0-3 digital I/O
0022 0108               movwf   ADCON1              ;       /
0023 0205               clrf    TRISA               ;make RA0-4 outputs
0024 0206               clrf    TRISB               ;make RB0-7 outputs
0025 1283               bcf     STATUS,RP0          ;select page 0
0026 0185               clrf    PORT_A              ;make all outputs low
0027 0186               clrf    PORT_B              ;       /
0028 1585               bsf     PORT_A,3            ;enable MSB digit sink
0029 0008               return
                    ;
                    ;
                    ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
                    ;the rtcc will be incremented every 31.25uS. If rtcc is preloaded
                    ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5msec. So the
                    ;end result is that we get a rtcc interrupt every 5msec.
                    InitTimers
002A 0190               clrf    MsdTime             ;clr timers
002B 0191               clrf    LsdTime             ;       /
002C 0192               clrf    KeyFlag             ;clr all flags
002D 1683               bsf     STATUS,RP0          ;select pg 1
002E 3084               movlw   B'10000100'         ;assign ps to rtcc
002F 0081               movwf   OptionReg           ;ps = 32
0030 1283               bcf     STATUS,RP0          ;select pg 0
0031 3020               movlw   B'00100000'         ;enable rtcc interrupt
0032 008B               movwf   INTCON              ;
0033 3060               movlw   .96                 ;preload rtcc
0034 0081               movwf   RTCC                ;start counter
0035 0009               retfie
                    ;
                    ServiceInterrupts
0036 190B               btfsc   INTCON,RTIF         ;rtcc interrupt?
0037 283B               goto    ServiceRTCC         ;yes then service
0038 018B               clrf    INTCON              ;else clr all int
0039 168B               bsf     INTCON,RTIE
```

```
003A 0008                       return
                        ;
                        ServiceRTCC
003B 3060                       movlw   .96                     ;initialize rtcc
003C 0081                       movwf   RTCC
003D 110B                       bcf     INTCON,RTIF             ;clr int flag
003E 1805                       btfsc   PORT_A,0                ;if msb on then do
003F 2042                       call    ScanKeys                ;do a quick key scan
0040 20A1                       call    UpdateDisplay           ;update display
0041 0008                       return
                        ;
                        ;
                        ;ScanKeys, scans the 4X4 keypad matrix and returns a key value in
                        ;NewKey (0 - F) if a key is pressed, if not it clears the keyhit flag.
                        ;Debounce for a given keyhit is also taken care of.
                        ;The rate of key scan is 20msec with a 4.096Mhz clock.
                        ScanKeys
0042 1C92                       btfss   KeyFlag,DebnceOn        ;debounce on?
0043 2848                       goto    Scan1                   ;no then scan keypad
0044 0B93                       decfsz  Debnce                  ;else dec debounce time
0045 0008                       return                          ;not over then return
0046 1092                       bcf     KeyFlag,DebnceOn        ;over, clr debounce flag
0047 0008                       return                          ;and return
                        Scan1
0048 208A                       call    SavePorts               ;save port values
0049 30EF                       movlw   B'11101111'             ;init TempD
004A 008D                       movwf   TempD
                        ScanNext
004B 0806                       movf    PORT_B,w                ;read to init port
004C 100B                       bcf     INTCON,RBIF             ;clr flag
004D 0C8D                       rrf     TempD                   ;get correct column
004E 1C03                       btfss   STATUS,C                ;if carry set?
004F 2862                       goto    NoKey                   ;no then end
0050 080D                       movf    TempD,w                 ;else output
0051 0086                       movwf   PORT_B                  ;low column scan line
0052 0000                       nop
0053 1C0B                       btfss   INTCON,RBIF             ;flag set?
0054 284B                       goto    ScanNext                ;no then next
0055 1812                       btfsc   KeyFlag,keyhit          ;last key released?
0056 2860                       goto    SKreturn                ;no then exit
0057 1412                       bsf     KeyFlag,keyhit          ;set new key hit
0058 0E06                       swapf   PORT_B,w                ;read port
0059 008E                       movwf   TempE                   ;save in TempE
005A 2064                       call    GetKeyValue             ;get key value 0 - F
005B 0094                       movwf   NewKey                  ;save as New key
005C 1592                       bsf     KeyFlag,ServKey         ;set service flag
005D 1492                       bsf     KeyFlag,DebnceOn        ;set flag
005E 3004                       movlw   4
005F 0093                       movwf   Debnce                  ;load debounce time
                        SKreturn
0060 2097                       call    RestorePorts            ;restore ports
0061 0008                       return
                        ;
                        NoKey
0062 1012                       bcf     KeyFlag,keyhit          ;clr flag
0063 2860                       goto    SKreturn
                        ;
                        ;GetKeyValue gets the key as per the following layout
                        ;
                        ;                   Col1    Col2    Col3    Col3
                        ;                   (RB3)   (RB2)   (RB1)   (RB0)
                        ;
                        ;Row1(RB4)            0       1       2       3
                        ;
                        ;Row2(RB5)            4       5       6       7
                        ;
                        ;Row3(RB6)            8       9       A       B
                        ;
                        ;Row4(RB7)            C       D       E       F
```

```
                        ;
                        GetKeyValue
0064 018C                       clrf    TempC
0065 1D8D                       btfss   TempD,3         ;first column
0066 286E                       goto    RowValEnd
0067 0A8C                       incf    TempC
0068 1D0D                       btfss   TempD,2         ;second col.
0069 286E                       goto    RowValEnd
006A 0A8C                       incf    TempC
006B 1C8D                       btfss   TempD,1         ;3rd col.
006C 286E                       goto    RowValEnd
006D 0A8C                       incf    TempC           ;last col.
                        RowValEnd
006E 1C0E                       btfss   TempE,0         ;top row?
006F 2878                       goto    GetValCom       ;yes then get 0,1,2&3
0070 1C8E                       btfss   TempE,1         ;2nd row?
0071 2877                       goto    Get4567         ;yes the get 4,5,6&7
0072 1D0E                       btfss   TempE,2         ;3rd row?
0073 2875                       goto    Get89ab         ;yes then get 8,9,a&b
                        Getcdef
0074 150C                       bsf     TempC,2         ;set msb bits
                        Get89ab
0075 158C                       bsf     TempC,3         ;       /
0076 2878                       goto    GetValCom       ;do common part
                        Get4567
0077 150C                       bsf     TempC,2
                        GetValCom
0078 080C                       movf    TempC,w
0079 0782                       addwf   PCL
007A 3400                       retlw   0
007B 3401                       retlw   1
007C 3402                       retlw   2
007D 3403                       retlw   3
007E 3404                       retlw   4
007F 3405                       retlw   5
0080 3406                       retlw   6
0081 3407                       retlw   7
0082 3408                       retlw   8
0083 3409                       retlw   9
0084 340A                       retlw   0a
0085 340B                       retlw   0b
0086 340C                       retlw   0c
0087 340D                       retlw   0d
0088 340E                       retlw   0e
0089 340F                       retlw   0f
                        ;
                        ;SavePorts, saves the porta and portb condition during a key scan
                        ;operation.
                        SavePorts
008A 0805                       movf    PORT_A,w        ;Get sink value
008B 00A0                       movwf   PABuf           ;save in buffer
008C 0185                       clrf    PORT_A          ;disable all sinks
008D 0806                       movf    PORT_B,w        ;get port b
008E 00A1                       movwf   PBBuf           ;save in buffer
008F 30FF                       movlw   0xff            ;make all high
0090 0086                       movwf   PORT_B          ;on port b
0091 1683                       bsf     STATUS,RP0      ;select page 1
0092 1381                       bcf     OptionReg,7     ;enable pull ups
0093 30F0                       movlw   B'11110000'     ;port b hi nibble inputs
0094 0106                       movwf   TRISB           ;lo nibble outputs
0095 1283                       bcf     STATUS,RP0      ;page 0
0096 0008                       return
                        ;
                        ;RestorePorts, restores the condition of porta and portb after a
                        ;key scan operation.
                        RestorePorts
0097 0821                       movf    PBBuf,w         ;get port n
0098 0086                       movwf   PORT_B
0099 0820                       movf    PABuf,w         ;get port a value
```

```
009A 0085                       movwf   PORT_A
009B 1683                       bsf     STATUS,RP0      ;select page 1
009C 1781                       bsf     OptionReg,7     ;disable pull ups
009D 0205                       clrf    TRISA           ;make port a outputs
009E 0206                       clrf    TRISB           ;as well as PORTB
009F 1283                       bcf     STATUS,RP0      ;page 0
00A0 0008                       return
                        ;
                        ;
                        UpdateDisplay
00A1 0805                       movf    PORT_A,w        ;present sink value in w
00A2 0185                       clrf    PORT_A          ;disable all digits sinks
00A3 390F                       andlw   0x0f
00A4 008C                       movwf   TempC           ;save sink value in tempC
00A5 160C                       bsf     TempC,4         ;preset for lsd sink
00A6 0C8C                       rrf     TempC           ;determine next sink value
00A7 1C03                       btfss   STATUS,CARRY    ;c=1?
00A8 118C                       bcf     TempC,3         ;no then reset LSD sink
00A9 180C                       btfsc   TempC,0         ;else see if Msd
00AA 28B8                       goto    UpdateMsd       ;yes then do Msd
00AB 188C                       btfsc   TempC,1         ;see if 3rdLsd
00AC 28B5                       goto    Update3rdLsd    ;yes then do 3rd Lsd
00AD 190C                       btfsc   TempC,2         ;see if 2nd Lsd
00AE 28B2                       goto    Update2ndLsd    ;yes then do 2nd lsd
                        UpdateLsd
00AF 0811                       movf    LsdTime,w       ;get Lsd in w
00B0 390F                       andlw   0x0f            ;       /
00B1 28BA                       goto    DisplayOut
                        Update2ndLsd
00B2 0E11                       swapf   LsdTime,w       ;get 2nd Lsd in w
00B3 390F                       andlw   0x0f            ;mask rest
00B4 28BA                       goto    DisplayOut      ;enable display
                        Update3rdLsd
00B5 0810                       movf    MsdTime,w       ;get 3rd Lsd in w
00B6 390F                       andlw   0x0f            ;mask low nibble
00B7 28BA                       goto    DisplayOut      ;enable display
                        UpdateMsd
00B8 0E10                       swapf   MsdTime,w       ;get Msd in w
00B9 390F                       andlw   0x0f            ;mask rest
                        DisplayOut
00BA 20BF                       call    LedTable        ;get digit output
00BB 0086                       movwf   PORT_B          ;drive leds
00BC 080C                       movf    TempC,w         ;get sink value in w
00BD 0085                       movwf   PORT_A
00BE 0008                       return
                        ;
                        ;
                        LedTable
00BF 0782                       addwf   PCL             ;add to PC low
00C0 343F                       retlw   B'00111111'     ;led drive for 0
00C1 3406                       retlw   B'00000110'     ;led drive for 1
00C2 345B                       retlw   B'01011011'     ;led drive for 2
00C3 344F                       retlw   B'01001111'     ;led drive for 3
00C4 3466                       retlw   B'01100110'     ;led drive for 4
00C5 346D                       retlw   B'01101101'     ;led drive for 5
00C6 347D                       retlw   B'01111101'     ;led drive for 6
00C7 3407                       retlw   B'00000111'     ;led drive for 7
00C8 347F                       retlw   B'01111111'     ;led drive for 8
00C9 3467                       retlw   B'01100111'     ;led drive for 9
00CA 3477                       retlw   B'01110111'     ;led drive for A
00CB 347C                       retlw   B'01111100'     ;led drive for b
00CC 3439                       retlw   B'00111001'     ;led drive for C
00CD 345E                       retlw   B'01011110'     ;led drive for d
00CE 3479                       retlw   B'01111001'     ;led drive for E
00CF 3471                       retlw   B'01110001'     ;led drive for F
                        ;
                        ;
                                end
                        ;
```

Appendix C

```
MPASM B0.50                                                     PAGE  1


                        ;*****************************************************************
                        ;This program is to demonstrate how to multiplex four 7 segment LED
                        ;and sample ch0 of the a/d in a PIC16C71. The a/d value is displayed
                        ;as a 3 digit decimal value of the a/d input (0 - 255).
                        ;The LEDs are updated every 20msec, the a/d is sampled every 20 msec.
                        ;The RTCC timer is used in internal interrupt mode to generate the 5 msec.
                        ;
                        ;                                       Stan D'Souza 5/8/93
                        ;*****************************************************************
                               LIST P=16C71, F=INHX8M
                        ;
                               include "picreg.equ"



                        ;
0026                    BcdMsd          equ     26
0027                    Bcd             equ     27
000C                    TempC           equ     0x0c            ;temp general purpose regs
000D                    TempD           equ     0x0d
000E                    TempE           equ     0x0e
0020                    PABuf           equ     0x20
0021                    PBBuf           equ     0x21
000F                    Count           equ     0x0f            ;count
0010                    MsdTime         equ     0x10            ;most significant Timer
0011                    LsdTime         equ     0x11            ;Least significant Timer
0012                    ADFlag          equ     0x12            ;flags related to key pad
0005                    ADOver          equ     5               ;bit 5 —> a/d over
002F                    WBuffer         equ     0x2f
002E                    StatBuffer      equ     0x2e
0001                    OptionReg       equ     1
0002                    PCL             equ     2
                        ;
                        push            macro
                        movwf           WBuffer                 ;save w reg in Buffer
                        swapf           WBuffer                 ;swap it
                        swapf           STATUS,w                ;get status
                        movwf           StatBuffer              ;save it
                        endm
                        ;
                        pop             macro
                        swapf           StatBuffer,w            ;restore status
                        movwf           STATUS                  ;       /
                        swapf           WBuffer,w               ;restore W reg
                        endm
                        ;
                        org             0
0000 280D               goto            Start                   ;skip over interrupt vector
                        ;
                        org             4
                        ;It is always a good practice to save and restore the w reg,
                        ;and the status reg during a interrupt.
                        push
0004 00AF               movwf           WBuffer                 ;save w reg in Buffer
0005 0EAF               swapf           WBuffer                 ;swap it
0006 0E03               swapf           STATUS,w                ;get status
0007 00AE       `       movwf           StatBuffer              ;save it

0008 2039               call            ServiceInterrupts
                        pop
0009 0E2E               swapf           StatBuffer,w            ;restore status
000A 0083                     movwf   STATUS          ;       /
```

```
000B 0E2F                       swapf   WBuffer,w               ;restore W reg

000C 0009                       retfie
                        ;
                        Start
000D 2021                       call    InitPorts
000E 202B                       call    InitTimers
000F 2036                       call    InitAd
                        loop
0010 1A92                       btfsc   ADFlag,ADOver           ;a/d over?
0011 2013                       call    UpdateAd                ;yes then update
0012 2810                       goto    loop
                        ;
                        UpdateAd
0013 1C88                       btfss   ADCON0,ADIF             ;a/d done?
0014 0008                       return                          ;no then leave
0015 0809                       movf    ADRES,W                 ;get a/d value
0016 00A1                       movwf   L_byte
0017 01A0                       clrf    H_byte
0018 20AD                       call    B2_BCD
0019 0824                       movf    R2,W                    ;get LSd
001A 0091                       movwf   LsdTime                 ;save in LSD
001B 0823                       movf    R1,W                    ;get Msd
001C 0090                       movwf   MsdTime                 ;save in Msd
001D 1088                       bcf     ADCON0,ADIF             ;clr interrupt flag
001E 1008                       bcf     ADCON0,ADON             ;turn off a/d
001F 1292                       bcf     ADFlag,ADOver           ;clr flag
0020 0008                       return
                        ;
                        ;
                        ;
                        InitPorts
0021 1683                       bsf     STATUS,RP0              ;select pg 1
0022 3003                       movlw   3                       ;make RA0-3 digital I/O
0023 0108                       movwf   ADCON1                  ;       /
0024 0205                       clrf    TRISA                   ;make RA0-4 outputs
0025 0206                       clrf    TRISB                   ;make RB0-7 outputs
0026 1283                       bcf     STATUS,RP0              ;select page 0
0027 0185                       clrf    PORT_A                  ;make all outputs low
0028 0186                       clrf    PORT_B                  ;       /
0029 1585                       bsf     PORT_A,3                ;enable MSB digit sink
002A 0008                       return
                        ;
                        ;
                        ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
                        ;the rtcc will be incremented every 31.25uS. If rtcc is preloaded
                        ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5msec. So the
                        ;end result is that we get a rtcc interrupt every 5msec.
                        InitTimers
002B 0190                       clrf    MsdTime                 ;clr timers
002C 0191                       clrf    LsdTime                 ;       /
002D 1683                       bsf     STATUS,RP0              ;select pg 1
002E 3084                       movlw   B'10000100'             ;assign ps to rtcc
002F 0081                       movwf   OptionReg               ;ps = 32
0030 1283                       bcf     STATUS,RP0              ;select pg 0
0031 3020                       movlw   B'00100000'             ;enable rtcc interrupt
0032 008B                       movwf   INTCON                  ;
0033 3060                       movlw   .96                     ;preload rtcc
0034 0081                       movwf   RTCC                    ;start counter
0035 0009                       retfie
                        ;
                        ;
                        InitAd
0036 30C8                       movlw   B'11001000'             ;init a/d
0037 0088                       movwf   ADCON0
0038 0008                       return
                        ;
```



```
                        ;
                        ServiceInterrupts
0039 190B                       btfsc   INTCON,RTIF            ;rtcc interrupt?
003A 283E                       goto    ServiceRTCC            ;yes then service
003B 018B                       clrf    INTCON
003C 168B                       bsf     INTCON,RTIE
003D 0008                       return
                        ;
                        ServiceRTCC
003E 3060                       movlw   .96                    ;initialize rtcc
003F 0081                       movwf   RTCC
0040 110B                       bcf     INTCON,RTIF            ;clr int flag
0041 1C05                       btfss   PORT_A,0               ;last digit?
0042 2045                       call    SampleAd               ;then sample a/d
0043 2071                       call    UpdateDisplay          ;else update display
0044 0008                       return
                        ;
                        ;
                        SampleAd
0045 205A                       call    SavePorts
0046 204C                       call    DoAd                   ;do a ad conversion
                        AdDone
0047 1908                       btfsc   ADCON0,GO              ;ad done?
0048 2847                       goto    AdDone                 ;no then loop
0049 1692                       bsf     ADFlag,ADOver          ;set a/d over flag
004A 2067                       call    RestorePorts           ;restore ports
004B 0008                       return
                        ;
                        ;
                        DoAd
004C 0186                       clrf    PORT_B                 ;turn off leds
004D 1683                       bsf     STATUS,RP0             ;select pg 1
004E 300F                       movlw   0x0f                   ;make port a hi-Z
004F 0105                       movwf   TRISA                  ;       /
0050 1283                       bcf     STATUS,RP0             ;select pg 0
0051 1408                       bsf     ADCON0,ADON            ;start a/d
0052 307D                       movlw   .125
0053 2056                       call    Wait
0054 1508                       bsf     ADCON0,GO              ;start conversion
0055 0008                       return
                        ;
                        ;
                        Wait
0056 008C                       movwf   TempC                  ;store in temp
                        Next
0057 0B8C                       decfsz  TempC
0058 2857                       goto    Next
0059 0008                       return


                        ;
                        ;SavePorts, saves the porta and portb condition during a key scan
                        ;operation.
                        SavePorts
005A 0805                       movf    PORT_A,w               ;Get sink value
005B 00A0                       movwf   PABuf                  ;save in buffer
005C 0185                       clrf    PORT_A                 ;disable all sinks
005D 0806                       movf    PORT_B,w               ;get port b
005E 00A1                       movwf   PBBuf                  ;save in buffer
005F 30FF                       movlw   0xff                   ;make all high
0060 0086                       movwf   PORT_B                 ;on port b
0061 1683                       bsf     STATUS,RP0             ;select page 1
0062 1381                       bcf     OptionReg,7            ;enable pull ups
0063 30F0                       movlw   B'11110000'            ;port b hi nibble inputs
0064 0106                       movwf   TRISB                  ;lo nibble outputs
0065 1283                       bcf     STATUS,RP0             ;page 0
0066 0008                       return
                        ;
                        ;RestorePorts, restores the condition of porta and portb after a
                        ;key scan operation.
```

```
                        RestorePorts
0067 0821                       movf    PBBuf,w         ;get port n
0068 0086                       movwf   PORT_B
0069 0820                       movf    PABuf,w         ;get port a value
006A 0085                       movwf   PORT_A
006B 1683                       bsf     STATUS,RP0      ;select page 1
006C 1781                       bsf     OptionReg,7     ;disable pull ups
006D 0205                       clrf    TRISA           ;make port a outputs
006E 0206                       clrf    TRISB           ;as well as PORTB
006F 1283                       bcf     STATUS,RP0      ;page 0
0070 0008                       return
                        ;
                        ;
                        UpdateDisplay
0071 0805                       movf    PORT_A,w        ;present sink value in w
0072 0185                       clrf    PORT_A          ;disable all digits sinks
0073 390F                       andlw   0x0f
0074 008C                       movwf   TempC           ;save sink value in tempC
0075 160C                       bsf     TempC,4         ;preset for lsd sink
0076 0C8C                       rrf     TempC           ;determine next sink value
0077 1C03                       btfss   STATUS,CARRY    ;c=1?
0078 118C                       bcf     TempC,3         ;no then reset LSD sink
0079 180C                       btfsc   TempC,0         ;else see if Msd
007A 288C                       goto    UpdateMsd       ;yes then do Msd
007B 188C                       btfsc   TempC,1         ;see if 3rdLsd
007C 2887                       goto    Update3rdLsd    ;yes then do 3rd Lsd
007D 190C                       btfsc   TempC,2         ;see if 2nd Lsd
007E 2882                       goto    Update2ndLsd    ;yes then do 2nd lsd
                        UpdateLsd
007F 0811                       movf    LsdTime,w       ;get Lsd in w
0080 390F                       andlw   0x0f            ;       /
0081 2890                       goto    DisplayOut      ;enable display
                        Update2ndLsd
0082 20A1                       call    Chk2LsdZero     ;msd = 0 & 2 lsd 0?
0083 1D03                       btfss   STATUS,Z        ;yes then skip
0084 0E11                       swapf   LsdTime,w       ;get 2nd Lsd in w
0085 390F                       andlw   0x0f            ;mask rest
0086 2890                       goto    DisplayOut      ;enable display
                        Update3rdLsd
0087 20A9                       call    ChkMsdZero      ;msd = 0?
0088 1D03                       btfss   STATUS,Z        ;yes then skip
0089 0810                       movf    MsdTime,w       ;get 3rd Lsd in w
008A 390F                       andlw   0x0f            ;mask low nibble
008B 2890                       goto    DisplayOut      ;enable display
                        UpdateMsd
008C 0E10                       swapf   MsdTime,w       ;get Msd in w
008D 390F                       andlw   0x0f            ;mask rest
008E 1903                       btfsc   STATUS,Z        ;msd != 0 then skip
008F 300A                       movlw   0x0a
                        DisplayOut
0090 2095                       call    LedTable        ;get digit output
0091 0086                       movwf   PORT_B          ;drive leds
0092 080C                       movf    TempC,w         ;get sink value in w
0093 0085                       movwf   PORT_A
0094 0008                       return
                        ;
                        ;
                        LedTable
0095 0782                       addwf   PCL             ;add to PC low
0096 343F                       retlw   B'00111111'     ;led drive for 0
0097 3406                       retlw   B'00000110'     ;led drive for 1
0098 345B                       retlw   B'01011011'     ;led drive for 2
0099 344F                       retlw   B'01001111'     ;led drive for 3
009A 3466                       retlw   B'01100110'     ;led drive for 4
009B 346D                       retlw   B'01101101'     ;led drive for 5
009C 347D                       retlw   B'01111101'     ;led drive for 6
009D 3407                       retlw   B'00000111'     ;led drive for 7
009E 347F                       retlw   B'01111111'     ;led drive for 8
009F 3467                       retlw   B'01100111'     ;led drive for 9
```

```
00A0 3400                       retlw           B'00000000'     ;blank led drive
                        ;
                        ;
                        Chk2LsdZero
00A1 20A9                       call            ChkMsdZero      ;msd = 0?
00A2 1D03                       btfss           STATUS,Z        ;yes then skip
00A3 0008                       return                          ;else return
00A4 0E11                       swapf           LsdTime,w       ;get 2nd lsd
00A5 390F                       andlw           0x0f            ;mask of LSD
00A6 1D03                       btfss           STATUS,Z        ;0? then skip
00A7 0008                       return
00A8 340A                       retlw           .10             ;else return with 10
                        ;
                        ChkMsdZero
00A9 0810                       movf            MsdTime,w       ;get Msd in w
00AA 1D03                       btfss           STATUS,Z        ;= 0? skip
00AB 0008                       return                          ;else return
00AC 340A                       retlw           .10             ;ret with 10
                        ;
                        ;
                        ;
0026                    count   equ     26
0027                    temp    equ     27
                        ;
0020                    H_byte  equ     20
0021                    L_byte  equ     21
0022                    R0      equ     22                      ; RAM Assignments
0023                    R1      equ     23
0024                    R2      equ     24
                        ;
                        ;
00AD 1003               B2_BCD  bcf     STATUS,0                ; clear the carry bit
00AE 3010                       movlw   .16
00AF 00A6                       movwf   count
00B0 01A2                       clrf    R0
00B1 01A3                       clrf    R1
00B2 01A4                       clrf    R2
00B3 0DA1               loop16  rlf     L_byte
00B4 0DA0                       rlf     H_byte
00B5 0DA4                       rlf     R2
00B6 0DA3                       rlf     R1
00B7 0DA2                       rlf     R0
                        ;
00B8 0BA6                       decfsz  count
00B9 28BB                       goto    adjDEC
00BA 3400                       RETLW   0
                        ;
00BB 3024               adjDEC  movlw   R2
00BC 0084                       movwf   FSR
00BD 20C5                       call    adjBCD
                        ;
00BE 3023                       movlw   R1
00BF 0084                       movwf   FSR
00C0 20C5                       call    adjBCD
                        ;
00C1 3022                       movlw   R0
00C2 0084                       movwf   FSR
00C3 20C5                       call    adjBCD
                        ;
00C4 28B3                       goto    loop16
                        ;
00C5 3003               adjBCD  movlw   3
00C6 0700                       addwf   0,W
00C7 00A7                       movwf   temp
00C8 19A7                       btfsc   temp,3                  ; test if result > 7
00C9 0080                       movwf   0
00CA 3030                       movlw   30
00CB 0700                       addwf   0,W
00CC 00A7                       movwf   temp
```

```
00CD 1BA7                       btfsc   temp,7          ; test if result > 7
00CE 0080                       movwf   0               ; save as MSD
00CF 3400                       RETLW   0
                        ;
                        ;

                                end
                        ;
```

APPENDIX D

```
MPASM 1.00 Released    MPLXAD.ASM   7-15-1994  13:43:14                PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001 ;*******************************************************************
                     0002 ;This program is to demonstrate how to multiplex four 7 segment LED
                     0003 ;digits and a 4X4 keypad along with 4 A/D inputs using a PIC16C71.
                     0004 ;The four digits will first display the decimal a/d value of ch0.
                     0005 ;When keys from 0 - 3 are hit the corresponding channel's a/d value
                     0006 ;is displayed in decimal.
                     0007 ;The LEDs are updated every 20mS, the keypad is scanned at a rate of 20 mS.
                     0008 ;All 4 channels are scanned at 20mS rate, so each channel gets scanned
                     0009 ;every 80mS. A faster rate of scanning is possible as required by
                     0010 ;the users application.
                     0011 ;The RTCC timer is used in internal interrupt mode to generate the
                     0012 ;5 mS.
                     0013 ;
                     0014 ;                                        Stan D'Souza 5/8/93
                     0015 ;
                     0016 ;Corrected error in display routine.
                     0017 ;                                        Stan D'Souza 2/27/94
                     0018 ;*******************************************************************
                     0019         LIST P=16C71, F=INHX8M
                     0020 ;
                     0021         include "picreg.equ"
                     0083
                     0084
                     0021
                     0022 ;
000C                 0023 TempC   equ     0x0c            ;temp general purpose regs
000D                 0024 TempD   equ     0x0d
000E                 0025 TempE   equ     0x0e
0020                 0026 PABuf   equ     0x20
0021                 0027 PBBuf   equ     0x21
000F                 0028 Count   equ     0x0f           ;count
0010                 0029 MsdTime equ     0x10            ;most significant Timer
0011                 0030 LsdTime equ     0x11            ;Least significant Timer
                     0031 ;
0012                 0032 Flag    equ     0x12            ;general purpose flag reg
0001                 0033  #define keyhit  Flag,0         ;bit 0 -> key-press on
0002                 0034  #define DebnceOn Flag,1        ;bit 1 -> debounce on
0003                 0035  #define noentry Flag,2         ;no key entry = 0
0004                 0036  #define ServKey Flag,3         ;bit 3 -> service key
0005                 0037  #define ADOver  Flag,4         ;bit 4 -> a/d conv. over
                     0038 ;
0013                 0039 Debnce  equ     0x13            ;debounce counter
0014                 0040 NewKey  equ     0x14
0015                 0041 DisplayCh equ   0x15            ;channel to be displayed
                     0042 ;
0016                 0043 ADTABLE equ     0x16            ;4 locations are reserved here
                     0044                                 ;from 0x16 to 0x19
                     0045 ;
002F                 0046 WBuffer equ     0x2f
002E                 0047 StatBuffer equ  0x2e
0001                 0048 OptionReg equ   1
0002                 0049 PCL     equ     2
                     0050 ;
                     0051 ;
                     0052 push    macro
                     0053         movwf   WBuffer         ;save w reg in Buffer
                     0054         swapf   WBuffer         ;swap it
                     0055         swapf   STATUS,w        ;get status
                     0056         movwf   StatBuffer      ;save it
                     0057         endm
                     0058 ;
```

```
                    0059 pop     macro
                    0060         swapf   StatBuffer,w    ;restore status
                    0061         movwf   STATUS          ;       /
                    0062         swapf   WBuffer,w       ;restore W reg
                    0063         endm
                    0064 ;
                    0065         org     0
0000 280D           0066         goto    Start           ;skip over interrupt vector
                    0067 ;
                    0068         org     4
                    0069 ;It is always a good practice to save and restore the w reg,
                    0070 ;and the status reg during a interrupt.
                    0071         push
0004 00AF              M         movwf   WBuffer         ;save w reg in Buffer
0005 0EAF              M         swapf   WBuffer         ;swap it
0006 0E03              M         swapf   STATUS,w        ;get status
0007 00AE              M         movwf   StatBuffer      ;save it
0008 2052           0072         call    ServiceInterrupts
                    0073         pop
0009 0E2E              M         swapf   StatBuffer,w    ;restore status
000A 0083              M         movwf   STATUS          ;       /
000B 0E2F              M         swapf   WBuffer,w       ;restore W reg
000C 0009           0074         retfie
                    0075 ;
                    0076 Start
000D 203B           0077         call    InitPorts
000E 20EE           0078         call    InitAd
000F 2045           0079         call    InitTimers
                    0080 loop
0010 1992           0081         btfsc   ServKey         ;key service pending
0011 2015           0082         call    ServiceKey      ;yes then service
0012 1A12           0083         btfsc   ADOver          ;a/d pending?
0013 2028           0084         call    ServiceAD       ;yes the service a/d
0014 2810           0085         goto    loop
                    0086 ;
                    0087 ;ServiceKey, does the software service for a keyhit. After a key service,
                    0088 ;the ServKey flag is reset, to denote a completed operation.
                    0089 ServiceKey
0015 1192           0090         bcf     ServKey         ;reset service flag
0016 0814           0091         movf    NewKey,w        ;get key value
0017 3C03           0092         sublw   3               ;key > 3?
0018 1C03           0093         btfss   STATUS,C        ;no then skip
0019 0008           0094         return                  ;else ignore key
001A 0814           0095         movf    NewKey,w
001B 0095           0096         movwf   DisplayCh       ;load new channel
                    0097 ;
                    0098 LoadAD
001C 3016           0099         movlw   ADTABLE         ;get top of table
001D 0715           0100         addwf   DisplayCh,w     ;add offset
001E 0084           0101         movwf   FSR             ;init FSR
001F 0800           0102         movf    0,w             ;get a/d value
0020 00A1           0103         movwf   L_byte
0021 01A0           0104         clrf    H_byte
0022 2106           0105         call    B2_BCD
0023 0824           0106         movf    R2,W            ;get LSd
0024 0091           0107         movwf   LsdTime         ;save in LSD
0025 0823           0108         movf    R1,W            ;get Msd
0026 0090           0109         movwf   MsdTime         ;save in Msd
0027 0008           0110         return
                    0111 ;
                    0112 ;This rountine essentially loads the ADRES value in the table location
                    0113 ;determined by the channel offset. If channel 0 then ADRES is saved
                    0114 ;in location ADTABLE. If channel 1 then ADRES is saved at ADTABLE + 1.
                    0115 ;and so on.
                    0116 ServiceAD
0028 0808           0117         movf    ADCON0,w        ;get adcon0
0029 008C           0118         movwf   TempC           ;save in temp
002A 3008           0119         movlw   B'00001000'     ;select next channel
002B 0708           0120         addwf   ADCON0,w        ;       /
```



```
002C 1A88           0121         btfsc   ADCON0,5        ;if <= ch3
002D 30C1           0122         movlw   B'11000001'     ;select ch0
002E 0088           0123         movwf   ADCON0
                    0124         ;now load adres in the table
002F 3016           0125         movlw   ADTABLE
0030 0084           0126         movwf   FSR             ;load FSR with top
0031 0C8C           0127         rrf     TempC
0032 0C8C           0128         rrf     TempC
0033 0C0C           0129         rrf     TempC,w         ;get in w reg
0034 3903           0130         andlw   3               ;mask off all but last 2
0035 0784           0131         addwf   FSR             ;add offset to table
0036 0809           0132         movf    ADRES,w         ;get a/d value
0037 0080           0133         movwf   0               ;load indirectly
0038 1212           0134         bcf     ADOver          ;clear flag
0039 201C           0135         call    LoadAD          ;load a/d value in display reg.
003A 0008           0136         return
                    0137
                    0138
                    0139
                    0140 ;
                    0141 InitPorts
003B 1683           0142         bsf     STATUS,RP0      ;select pg 1
003C 3003           0143         movlw   3               ;make RA0-3 digital I/O
003D 0088           0144         movwf   ADCON1          ;       /
003E 0185           0145         clrf    TRISA           ;make RA0-4 outputs
003F 0186           0146         clrf    TRISB           ;make RB0-7 outputs
0040 1283           0147         bcf     STATUS,RP0      ;select page 0
0041 0185           0148         clrf    PORT_A          ;make all outputs low
0042 0186           0149         clrf    PORT_B          ;       /
0043 1585           0150         bsf     PORT_A,3        ;enable MSB digit sink
0044 0008           0151         return
                    0152 ;
                    0153 ;
                    0154 ;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
                    0155 ;the rtcc will be incremented every 31.25uS. If rtcc is preloaded
                    0156 ;with 96, it will take (256-96)*31.25uS to overflow i.e. 5mS. So the
                    0157 ;end result is that we get a rtcc interrupt every 5mS.
                    0158 InitTimers
0045 0190           0159         clrf    MsdTime         ;clr timers
0046 0191           0160         clrf    LsdTime         ;       /
0047 0195           0161         clrf    DisplayCh       ;show channel 0
0048 0192           0162         clrf    Flag            ;clr all flags
0049 1683           0163         bsf     STATUS,RP0      ;select pg 1
004A 3084           0164         movlw   B'10000100'     ;assign ps to rtcc
004B 0081           0165         movwf   OptionReg       ;ps = 32
004C 1283           0166         bcf     STATUS,RP0      ;select pg 0
004D 3020           0167         movlw   B'00100000'     ;enable rtcc interrupt
004E 008B           0168         movwf   INTCON          ;
004F 3060           0169         movlw   .96             ;preload rtcc
0050 0081           0170         movwf   RTCC            ;start counter
0051 0009           0171         retfie
                    0172 ;
                    0173 ServiceInterrupts
0052 190B           0174         btfsc   INTCON,RTIF     ;rtcc interrupt?
0053 2857           0175         goto    ServiceRTCC     ;yes then service
0054 018B           0176         clrf    INTCON          ;else clr all int
0055 168B           0177         bsf     INTCON,RTIE
0056 0008           0178         return
                    0179 ;
                    0180 ServiceRTCC
0057 3060           0181         movlw   .96             ;initialize rtcc
0058 0081           0182         movwf   RTCC
0059 110B           0183         bcf     INTCON,RTIF     ;clr int flag
005A 1805           0184         btfsc   PORT_A,0        ;scan keys every 20 mS
005B 2060           0185         call    ScanKeys        ;when digit 1 is on
005C 1985           0186         btfsc   PORT_A,3        ;scan a/d every 20mS
005D 20F1           0187         call    SampleAd        ;when digit 4 is on
005E 20BF           0188         call    UpdateDisplay   ;update display
005F 0008           0189         return
```

```
                    0190 ;
                    0191 ;
                    0192 ;ScanKeys, scans the 4X4 keypad matrix and returns a key value in
                    0193 ;NewKey (0 - F) if a key is pressed, if not it clears the keyhit flag.
                    0194 ;Debounce for a given keyhit is also taken care of.
                    0195 ;The rate of key scan is 20mS with a 4.096Mhz clock.
                    0196 ScanKeys
0060 1C92           0197         btfss   DebnceOn        ;debounce on?
0061 2866           0198         goto    Scan1           ;no then scan keypad
0062 0B93           0199         decfsz  Debnce          ;else dec debounce time
0063 0008           0200         return                  ;not over then return
0064 1092           0201         bcf     DebnceOn        ;over, clr debounce flag
0065 0008           0202         return                  ;and return
                    0203 Scan1
0066 20A8           0204         call    SavePorts       ;save port values
0067 30EF           0205         movlw   B'11101111'     ;init TempD
0068 008D           0206         movwf   TempD
                    0207 ScanNext
0069 0806           0208         movf    PORT_B,w        ;read to init port
006A 100B           0209         bcf     INTCON,RBIF     ;clr flag
006B 0C8D           0210         rrf     TempD           ;get correct column
006C 1C03           0211         btfss   STATUS,C        ;if carry set?
006D 2880           0212         goto    NoKey           ;no then end
006E 080D           0213         movf    TempD,w         ;else output
006F 0086           0214         movwf   PORT_B          ;low column scan line
0070 0000           0215         nop
0071 1C0B           0216         btfss   INTCON,RBIF     ;flag set?
0072 2869           0217         goto    ScanNext        ;no then next
0073 1812           0218         btfsc   keyhit          ;last key released?
0074 287E           0219         goto    SKreturn        ;no then exit
0075 1412           0220         bsf     keyhit          ;set new key hit
0076 0E06           0221         swapf   PORT_B,w        ;read port
0077 008E           0222         movwf   TempE           ;save in TempE
0078 2082           0223         call    GetKeyValue     ;get key value 0 - F
0079 0094           0224         movwf   NewKey          ;save as New key
007A 1592           0225         bsf     ServKey         ;set service flag
007B 1492           0226         bsf     DebnceOn        ;set flag
007C 3004           0227         movlw   4
007D 0093           0228         movwf   Debnce          ;load debounce time
                    0229 SKreturn
007E 20B5           0230         call    RestorePorts    ;restore ports
007F 0008           0231         return
                    0232 ;
                    0233 NoKey
0080 1012           0234         bcf     keyhit          ;clr flag
0081 287E           0235         goto    SKreturn
                    0236 ;
                    0237 ;GetKeyValue gets the key as per the following layout
                    0238 ;
                    0239 ;                  Col1    Col2    Col3    Col3
                    0240 ;                  (RB3)   (RB2)   (RB1)   (RB0)
                    0241 ;
                    0242 ;Row1(RB4)           0       1       2       3
                    0243 ;
                    0244 ;Row2(RB5)           4       5       6       7
                    0245 ;
                    0246 ;Row3(RB6)           8       9       A       B
                    0247 ;
                    0248 ;Row4(RB7)           C       D       E       F
                    0249 ;
                    0250 GetKeyValue
0082 018C           0251         clrf    TempC
0083 1D8D           0252         btfss   TempD,3         ;first column
0084 288C           0253         goto    RowValEnd
0085 0A8C           0254         incf    TempC
0086 1D0D           0255         btfss   TempD,2         ;second col.
0087 288C           0256         goto    RowValEnd
0088 0A8C           0257         incf    TempC
```

```
0089 1C8D           0258         btfss   TempD,1         ;3rd col.
008A 288C           0259         goto    RowValEnd
008B 0A8C           0260         incf    TempC           ;last col.
                    0261 RowValEnd
008C 1C0E           0262         btfss   TempE,0         ;top row?
008D 2896           0263         goto    GetValCom       ;yes then get 0,1,2&3
008E 1C8E           0264         btfss   TempE,1         ;2nd row?
008F 2895           0265         goto    Get4567         ;yes the get 4,5,6&7
0090 1D0E           0266         btfss   TempE,2         ;3rd row?
0091 2893           0267         goto    Get89ab         ;yes then get 8,9,a&b
                    0268 Getcdef
0092 150C           0269         bsf     TempC,2         ;set msb bits
                    0270 Get89ab
0093 158C           0271         bsf     TempC,3         ;       /
0094 2896           0272         goto    GetValCom       ;do common part
                    0273 Get4567
0095 150C           0274         bsf     TempC,2
                    0275 GetValCom
0096 080C           0276         movf    TempC,w
0097 0782           0277         addwf   PCL
0098 3400           0278         retlw   0
0099 3401           0279         retlw   1
009A 3402           0280         retlw   2
009B 3403           0281         retlw   3
009C 3404           0282         retlw   4
009D 3405           0283         retlw   5
009E 3406           0284         retlw   6
009F 3407           0285         retlw   7
00A0 3408           0286         retlw   8
00A1 3409           0287         retlw   9
00A2 340A           0288         retlw   0a
00A3 340B           0289         retlw   0b
00A4 340C           0290         retlw   0c
00A5 340D           0291         retlw   0d
00A6 340E           0292         retlw   0e
00A7 340F           0293         retlw   0f
                    0294 ;
                    0295 ;SavePorts, saves the porta and portb condition during a key scan
                    0296 ;operation.
                    0297 SavePorts
00A8 0805           0298         movf    PORT_A,w        ;Get sink value
00A9 00A0           0299         movwf   PABuf           ;save in buffer
00AA 0185           0300         clrf    PORT_A          ;disable all sinks
00AB 0806           0301         movf    PORT_B,w        ;get port b
00AC 00A1           0302         movwf   PBBuf           ;save in buffer
00AD 30FF           0303         movlw   0xff            ;make all high
00AE 0086           0304         movwf   PORT_B          ;on port b
00AF 1683           0305         bsf     STATUS,RP0      ;select page 1
00B0 1381           0306         bcf     OptionReg,7     ;enable pull ups
00B1 30F0           0307         movlw   B'11110000'     ;port b hi nibble inputs
00B2 0086           0308         movwf   TRISB           ;lo nibble outputs
00B3 1283           0309         bcf     STATUS,RP0      ;page 0
00B4 0008           0310         return
                    0311 ;
                    0312 ;RestorePorts, restores the condition of porta and portb after a
                    0313 ;key scan operation.
                    0314 RestorePorts
00B5 0821           0315         movf    PBBuf,w         ;get port b
00B6 0086           0316         movwf   PORT_B
00B7 0820           0317         movf    PABuf,w         ;get port a value
00B8 0085           0318         movwf   PORT_A
00B9 1683           0319         bsf     STATUS,RP0      ;select page 1
00BA 1781           0320         bsf     OptionReg,7     ;disable pull ups
00BB 0185           0321         clrf    TRISA           ;make port a outputs
00BC 0186           0322         clrf    TRISB           ;as well as PORTB
00BD 1283           0323         bcf     STATUS,RP0      ;page 0
00BE 0008           0324         return
                    0325 ;
                    0326 ;
```

```
                  0327 UpdateDisplay
00BF 0805         0328         movf    PORT_A,w        ;present sink value in w
00C0 0185         0329         clrf    PORT_A          ;disable all digits sinks
00C1 390F         0330         andlw   0x0f
00C2 008C         0331         movwf   TempC           ;save sink value in tempC
00C3 160C         0332         bsf     TempC,4         ;preset for lsd sink
00C4 0C8C         0333         rrf     TempC           ;determine next sink value
00C5 1C03         0334         btfss   STATUS,CARRY    ;c=1?
00C6 118C         0335         bcf     TempC,3         ;no then reset LSD sink
00C7 180C         0336         btfsc   TempC,0         ;else see if Msd
00C8 28D6         0337         goto    UpdateMsd       ;yes then do Msd
00C9 188C         0338         btfsc   TempC,1         ;see if 3rdLsd
00CA 28D3         0339         goto    Update3rdLsd    ;yes then do 3rd Lsd
00CB 190C         0340         btfsc   TempC,2         ;see if 2nd Lsd
00CC 28D0         0341         goto    Update2ndLsd    ;yes then do 2nd lsd
                  0342 UpdateLsd
00CD 0811         0343         movf    LsdTime,w       ;get Lsd in w
00CE 390F         0344         andlw   0x0f            ;       /
00CF 28D8         0345         goto    DisplayOut
                  0346 Update2ndLsd
00D0 0E11         0347         swapf   LsdTime,w       ;get 2nd Lsd in w
00D1 390F         0348         andlw   0x0f            ;mask rest
00D2 28D8         0349         goto    DisplayOut      ;enable display
                  0350 Update3rdLsd
00D3 0810         0351         movf    MsdTime,w       ;get 3rd Lsd in w
00D4 390F         0352         andlw   0x0f            ;mask low nibble
00D5 28D8         0353         goto    DisplayOut      ;enable display
                  0354 UpdateMsd
00D6 0E10         0355         swapf   MsdTime,w       ;get Msd in w
00D7 390F         0356         andlw   0x0f            ;mask rest
                  0357 DisplayOut
00D8 20DD         0358         call    LedTable        ;get digit output
00D9 0086         0359         movwf   PORT_B          ;drive leds
00DA 080C         0360         movf    TempC,w         ;get sink value in w
00DB 0085         0361         movwf   PORT_A
00DC 0008         0362         return
                  0363 ;
                  0364 ;
                  0365 LedTable
00DD 0782         0366         addwf   PCL             ;add to PC low
00DE 343F         0367         retlw   B'00111111'     ;led drive for 0
00DF 3406         0368         retlw   B'00000110'     ;led drive for 1
00E0 345B         0369         retlw   B'01011011'     ;led drive for 2
00E1 344F         0370         retlw   B'01001111'     ;led drive for 3
00E2 3466         0371         retlw   B'01100110'     ;led drive for 4
00E3 346D         0372         retlw   B'01101101'     ;led drive for 5
00E4 347D         0373         retlw   B'01111101'     ;led drive for 6
00E5 3407         0374         retlw   B'00000111'     ;led drive for 7
00E6 347F         0375         retlw   B'01111111'     ;led drive for 8
00E7 3467         0376         retlw   B'01100111'     ;led drive for 9
00E8 3477         0377         retlw   B'01110111'     ;led drive for A
00E9 347C         0378         retlw   B'01111100'     ;led drive for b
00EA 3439         0379         retlw   B'00111001'     ;led drive for C
00EB 345E         0380         retlw   B'01011110'     ;led drive for d
00EC 3479         0381         retlw   B'01111001'     ;led drive for E
00ED 3471         0382         retlw   B'01110001'     ;led drive for F
                  0383
                  0384 ;
                  0385 ;
                  0386 InitAd
00EE 30C0         0387         movlw   B'11000000'     ;internal rc for tad
00EF 0088         0388         movwf   ADCON0          ;       /
                  0389         ;note that adcon1 is set in InitPorts
00F0 0008         0390         return
                  0391 ;
                  0392 SampleAd
00F1 20A8         0393         call    SavePorts
00F2 20F8         0394         call    DoAd                    ;do a ad conversion
                  0395 AdDone
```

```
00F3 1908           0396         btfsc   ADCON0,GO              ;ad done?
00F4 28F3           0397         goto    AdDone                 ;no then loop
00F5 1612           0398         bsf     ADOver                 ;set a/d over flag
00F6 20B5           0399         call    RestorePorts           ;restore ports
00F7 0008           0400         return
                    0401 ;
                    0402 ;
                    0403 DoAd
00F8 0186           0404         clrf    PORT_B                 ;turn off leds
00F9 1683           0405         bsf     STATUS,RP0             ;select pg 1
00FA 300F           0406         movlw   0x0f                   ;make port a hi-Z
00FB 0085           0407         movwf   TRISA                  ;       /
00FC 1283           0408         bcf     STATUS,RP0             ;select pg 0
00FD 1408           0409         bsf     ADCON0,ADON            ;start a/d
00FE 307D           0410         movlw   .125
00FF 2102           0411         call    Wait
0100 1508           0412         bsf     ADCON0,GO              ;start conversion
0101 0008           0413         return
                    0414 ;
                    0415 ;
                    0416 Wait
0102 008C           0417         movwf   TempC                  ;store in temp
                    0418 Next
0103 0B8C           0419         decfsz  TempC
0104 2903           0420         goto    Next
0105 0008           0421         return
                    0422
                    0423 ;
                    0424 ;
0026                0425 count   equ     26
0027                0426 temp    equ     27
                    0427 ;
0020                0428 H_byte  equ     20
0021                0429 L_byte  equ     21
0022                0430 R0      equ     22             ; RAM Assignments
0023                0431 R1      equ     23
0024                0432 R2      equ     24
                    0433 ;
                    0434 ;
0106 1003           0435 B2_BCD  bcf     STATUS,0               ; clear the carry bit
0107 3010           0436         movlw   .16
0108 00A6           0437         movwf   count
0109 01A2           0438         clrf    R0
010A 01A3           0439         clrf    R1
010B 01A4           0440         clrf    R2
010C 0DA1           0441 loop16  rlf     L_byte
010D 0DA0           0442         rlf     H_byte
010E 0DA4           0443         rlf     R2
010F 0DA3           0444         rlf     R1
0110 0DA2           0445         rlf     R0
                    0446 ;
0111 0BA6           0447         decfsz  count
0112 2914           0448         goto    adjDEC
0113 3400           0449         RETLW   0
                    0450 ;
0114 3024           0451 adjDEC  movlw   R2
0115 0084           0452         movwf   FSR
0116 211E           0453         call    adjBCD
                    0454 ;
0117 3023           0455         movlw   R1
0118 0084           0456         movwf   FSR
0119 211E           0457         call    adjBCD
                    0458 ;
011A 3022           0459         movlw   R0
011B 0084           0460         movwf   FSR
011C 211E           0461         call    adjBCD
                    0462 ;
011D 290C           0463         goto    loop16
```

```
                     0464 ;
011E 3003            0465 adjBCD  movlw   3
011F 0700            0466         addwf   0,W
0120 00A7            0467         movwf   temp
0121 19A7            0468         btfsc   temp,3          ; test if result > 7
0122 0080            0469         movwf   0
0123 3030            0470         movlw   30
0124 0700            0471         addwf   0,W
0125 00A7            0472         movwf   temp
0126 1BA7            0473         btfsc   temp,7          ; test if result > 7
0127 0080            0474         movwf   0               ; save as MSD
0128 3400            0475         RETLW   0
                     0476 ;
                     0477 ;
                     0478 ;
                     0479 ;
                     0480
                     0481         end
                     0482 ;
                     0483
                     0484
                     0485


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

0000 : X—XXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX

0080 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX

0100 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXX——— ————————
0140 : ———————— ———————— ———————— ————————

All other memory blocks unused.


Errors   :    0
Warnings :    0
```

NOTES:

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