A Clock Design Using the PIC16C54 for LED Displays and Switch Inputs

INTRODUCTION

The purpose of this application note is to design a clock while multiplexing the features as much as possible, allowing the circuit to use the 18-pin PIC16C54. Other devices in the Microchip Technology Inc. line expand on this part, making it a good starting point for learning the basics. This design is useful because it utilizes every pin for output and switches some of them to inputs briefly to read the keys. For a more extensive clock design, consult application note AN529.

THE DESIGN

This design is a simple time of day clock incorporating four seven-segment LED displays and three input switches. There is also an additional reset switch that would not normally be incorporated into the final design. The schematic is illustrated in Figure 1.

CONNECTIONS

The individual segments of each display are connected together, A-A-A-A, B-B-B-B, etc. The displays are numbered from the right, or least significant digit. The second display from the right is flipped upside down to align its decimal with the third display, creating the center clock colon. Therefore the segments are not tied together evenly straight across on the board, but must compensate for the change in one display's orientation. The common cathode for each display is turned on with transistors connected to the four I/O lines of Port A. The connections are RA0-CC4/Digit4, RA1-CC3/Digit3, RA2-CC2/Digit2, RA3-CC1/Digit1. A low output turns on the PNP transistor for the selected display. The Port B pins activate the LED segments. For this design only the center colon decimal points were connected. The connections are RB0-dp, RB1-A, RB2-B, RB3-C...RB7-G.

The switches are also connected to Port B I/O pins. Port B pins RB1, RB2, and RB3 are pulled low with 10K ohm resistors. This value is high enough to not draw current away from the LEDs when they are being driven on. Inputs are detected by pulling the pins high with a switch to V_{DD} through 820 ohm resistors. This value is low enough to pull the pin high quickly when the outputs have been turned off, and to create a 90% of V_{DD} high input.

OPERATION

Switches

When no buttons are pressed, the circuit will display the current time, starting at 12:00 on reset. Pressing SW1 will cause seconds to be displayed. The time is set by pressing SW2 to advance minutes, and SW3 to advance hours. Since each of the segments are tied together across all displays, only one display should be turned on at a time, or all displays turned on would display duplicate data. The displays are turned on right to left, with each display's value being output its turn. This is done fast enough so that there is no perceived flicker. The switches are read between display cycles.

Timing

The PIC16CXX prescaler is assigned to the RTCC as a 1:16 divide. The RTCC pin is tied low since it is not used. The OPTION Register is loaded with 03h to initialize this prescaler set up. The software is written with timing based on a 4.000mhz crystal. The instruction clock is 1.000 MHz after the internal divide by four. The 8-bit RTCC register rolls over every 256 cycles, for a final frequency of 244.1406 Hz. (exactly a 4.096 ms period) A variable named sec_nth is used to count 244 roll-overs of the RTCC for one second. The benefit of keeping time with a nth variable is that it can be written to as needed to adjust time in "nths" of a second, allowing almost any odd crystal frequency to be used. Simply determine the best prescale and "nth" divider, and compute the "nth" adjustment needed for each minute, hour, twelve hour roll-over. Time can be kept accurately to two "nths" a day (an "nth" is 1/244 of a second in this case). In this circuit, 9 "nths" are subtracted each minute, 34 "nths" are added each hour, and 6 "nths" subtracted every twelve hour roll-over. This leaves a computed error of 1.5 seconds/year except for crystal frequency drift. Another possible solution is to initialize the RTCC to some value that causes a roll-over at a predetermined time interval. Writing to the RTCC causes two clock cycles to be missed while clock edges realign, which would have to be accounted for. This is described in the *Microchip Data Book*.

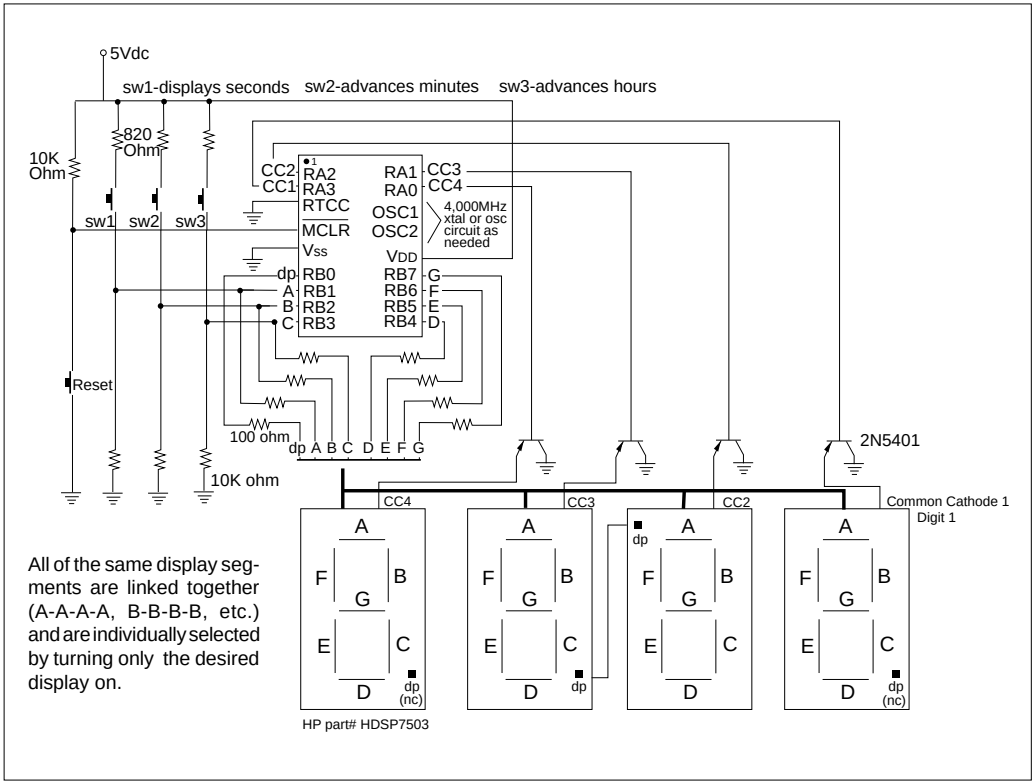
Displays

The program contains portions of code that act as a display driver. A variable exists for each of the four displays. A hex value from 0 to 9 can be written to these variables and they will be converted to display code and output to the displays. Only one display is actually on at a time, and its code is output into it in its turn. Another section of code takes the seconds, minutes, or hours value and separates it into the two digits needed for each display. In other words, 48 seconds would be separated into a "4" and an "8" and written to the appropriate display variable. The displays used were common cathode and turned on with transistors to avoid trying to sink too much current into the PIC16CXX. A display is enabled with a zero at the appropriate pin. 100 ohm resistors were used in series with the segments to obtain the desired brightness. Different values may be required if different displays are used. Since the displays are each on less than one fourth of the time, the resistor value must be low enough to compensate for the needed forward current.

CONCLUSION

The instruction execution speed of the PIC16C54 (and the rest of the PIC16/17 series) allows many functions to be implemented on a few pins by multiplexing them in software. User inputs, Real Time Clock Counter, and multiple LED displays are all accommodated with little or no sacrifice in functionality.

FIGURE 1: TIME OF DAY CLOCK USING PIC16C54



Author: Dan Matthews
Corporate Applications Manager

```
-LOC   LINE SOURCE TEXT

       0001  list P = 16C54
       0002 ;
       0003 ;*****************************************************************
       0004 ;                      Clock
       0005 ;*****************************************************************
       0006 ;
       0007 ;                 PROGRAM DESCRIPTION
       0008 ;
       0009 ; This program runs on a PIC16C54.
       0010 ;
       0011 ;                       Hardware Description
       0012 ;
       0013 ;  DISPLAYS
       0014 ; Four 7 segment displays are multiplexed.  The segments are tied together,
       0015 ; with the common cathode pins broken out separately.  The display appears
       0016 ; as a clock with a center semicolon ( 88:88 ).  The segments are assigned
       0017 ; to Port B, with the semicolon being RB0, and segments A through F
       0018 ; assigned as RB1 to RB7 respectively.
       0019 ;  The four common cathodes are activated by the four Port A pins through
       0020 ; transistors. RA0 for Digit4, RA1/Digit3, RA2/Digit2... through Digit4,
       0021 ; with Digit1 being in the rightmost position. The center semicolon is
       0022 ; made from the decimals of LED 2 and 3.
       0023 ;  Digit2 is turned upside down to put its decimal into position,
       0024 ; but it is wired with a corrected A-F assignment to compensate.  Both
       0025 ; decimals are tied together at RB0, but the display cathodes are still
       0026 ; separate.  Activating the decimal for digit2 AND 3 will turn on the
       0027 ; center colon.
       0028 ;
       0029 ;  SWITCHES
       0030 ; Because all twelve I/O pins are already used for the muxed displays,
       0031 ; eight for segments and four for digit selection, the three switches must
       0032 ; be read alternatingly through software.  The switches lie
       0033 ; across Port B pins, which are changed to inputs momentarily during read
       0034 ; and changed back to outputs during display.  Enough series resistance
       0035 ; must be used to prevent turning on or shorting segments during display
       0036 ; cycles if a switch is pressed.
       0037 ;
       0038 ; SW1-displays seconds, SW2-advances minutes,
       0039 ; SW3-advances hours, (none)-displays time
       0040 ;
       0041 ;*************************   Header ************************
       0042 ;
       0043 ;
01FF   0044 PIC54     equ     H'01FF' ; start address if used in a PIC16C54
03FF   0045 PIC56     equ     H'03FF' ;   "     "     "    "   " " PIC16C56
       0046                          ;
0000   0047 POINTER   equ     H'00'   ; address location f0 is an indirect address pointer
0001   0048 RTCC      equ     H'01'   ; address of RTCC clock value
0002   0049 PC        equ     H'02'   ; program counter
0003   0050 STATUS    equ     H'03'   ; F3 Reg is STATUS Reg.
0004   0051 FSR       equ     H'04'   ; F4 is File Select Register, address POINTER will direct to.
       0052                          ;
0005   0053 PORT_A    equ     H'05'   ; 7 segment Display Common Cathodes
0006   0054 PORT_B    equ     H'06'   ; Muxed Display Segments (Switches when inputs)
       0055                          ;
       0056                          ; STATUS REG. Bits
0000   0057 CARRY     equ     0       ; Carry Bit is Bit.0 of F3
0000   0058 C         equ     0
0001   0059 DCARRY    equ     1
0001   0060 DC        equ     1
0002   0061 Z_bit     equ     2       ; Bit 2 of F3 is Zero Bit
0002   0062 Z         equ     2
0003   0063 P_DOWN    equ     3
0003   0064 PD        equ     3
0004   0065 T_OUT     equ     4
0004   0066 TO        equ     4
0005   0067 PA0       equ     5       ;16C5X Status bits
```



```
0006    0068 PA1       equ     6            ;16C5X Status bits
0007    0069 PA2       equ     7            ;16C5X Status bits
        0070                                ;
007E    0071 ZERO      equ     H'7E'
000C    0072 ONE       equ     H'0C'
00B6    0073 TWO       equ     H'B6'
009E    0074 THREE     equ     H'9E'
00CC    0075 FOUR      equ     H'CC'
00DA    0076 FIVE      equ     H'DA'
00FA    0077 SIX       equ     H'FA'        ; coding of segments for display (PORT_B)
000E    0078 SEVEN     equ     H'0E'
00FE    0079 EIGHT     equ     H'FE'
00CE    0080 NINE      equ     H'CE'
0000    0081 BLANK     equ     H'00'
        0082                                ;
        0083                                ; timer variables start at a number that allows
        0084                                ; rollover in sync with time rollover, i.e. seconds
        0085                                ; starts at decimal 196 so that sixty 1-second
                                            ;   increments causes 0.
000C    0086 MAXNTHS   equ     D'12'        ;  initialization constants for timer count up
00C4    0087 MAXSECS   equ     D'196'       ;     see variable explanations for more info
00C4    0088 MAXMINS   equ     D'196'       ;
00F4    0089 MAXHRS    equ     D'244'
00F3    0090 MINHRS    equ     D'243'
0009    0091 ADJMIN    equ     D'9'         ; number of nths to be subtracted each minute for
                                            ;   accuracy
0022    0092 ADJHR     equ     D'34'        ; nths added each hour for accurate time
0006    0093 ADJDAY    equ     D'6'         ; nths subtracted each 1/2 day rollover
        0094 ;
00FE    0095 DISP4     equ     B'11111110'
00FD    0096 DISP3     equ     B'11111101'  ; Mapping of Active Display Selection (PORT_A)
00FB    0097 DISP2     equ     B'11111011'  ;  displays are active low
00F7    0098 DISP1     equ     B'11110111'
00FF    0099 DISPOFF   equ     H'FF'        ; turns all displays off when written to PORT_A
000E    0100 SWITCH    equ     B'00001110'  ; Used in tris B to set RB1-3 for switch inputs
        0101                                ;
        0102                                ;  Flag bit assignments
0000    0103 SEC       equ     H'0'         ; update time display values for sec, min, or hours
0001    0104 MIN       equ     H'1'
0002    0105 HRS       equ     H'2'
0003    0106 CHG       equ     H'3'         ; a change has occurred on a switch or a display
                                            ;   value
0004    0107 SW1       equ     H'4'         ; Flag bit assignments - switches that are on = 1
0005    0108 SW2       equ     H'5'         ;  SW1 is Seconds-minutes, SW2-hours, SW3-mode
0006    0109 SW3       equ     H'6'
0007    0110 SW_ON     equ     H'7'         ; indicates a switch has been pressed
        0111                                ;
        0112                                ; RAM VARIABLES
        0113 ;         equ     H'08'        ; not used
0009    0114 flags     equ     H'09'        ; bits:0-SEC,1-MIN,2-HRS,3-CHG,4-SW1,5-SW2,6-SW3,7-SW_ON
000B    0115 display   equ     H'0B'        ; SW_ON variable location - which display to update
000C    0116 digit1    equ     H'0C'        ; Rightmost display value
000D    0117 digit2    equ     H'0D'        ; Second display from right
000E    0118 digit3    equ     H'0E'        ; Third    "       "    "
000F    0119 digit4    equ     H'0F'        ; Fourth (and Leftmost)
0010    0120 sec_nth   equ     H'10'        ; seconds, fractional place
0011    0121 seconds   equ     H'11'        ; seconds
0012    0122 minutes   equ     H'12'        ; minutes
0013    0123 hours     equ     H'13'        ; hours
0014    0124 var       equ     H'14'        ; variable for misc math computations
0015    025  count     equ     H'15'        ; loop counter variable
        0126                                ;
        0127                                ;*******************************************
        0128                                ;
        0129                                ;  Initialize Ports all outputs, blank display
        030                                 ;
```

```
     OBJECT
-LOC  CODE  LINE SOURCE TEXT
0000  0C03  0131 START   movlw   H'03'        ; set option register, transition on clock,
0001  0002  0132         option               ; Prescale RTCC, 1:16
                         0133                      ;
0002  0C00  0134         movlw   0
0003  0005  0135         tris    PORT_A       ; Set all port pins as outputs
0004  0006  0136         tris    PORT_B
0005  0C00  0137         movlw   BLANK
0006  0026  0138         movwf   PORT_B       ; Blank the display
0007  04C3  0139         bcf     STATUS,PA1   ; page zero in case this is a higher PIC version
0008  04A3  0140         bcf     STATUS,PA0
            0141                              ;
            0142                              ;  initialize variables
0009  0C01  0143         movlw   H'01'
000A  0021  0144         movwf   RTCC         ; set RTCC above zero so initial wait period occurs
000B  0CF7  0145         movlw   DISP1
000C  002B  0146         movwf   display      ; initializes display selected to first display.
000D  0C00  0147         movlw   BLANK        ; put all displays to blank, no visible segments
000E  002C  0148         movwf   digit1
000F  002D  0149         movwf   digit2
0010  002E  0150         movwf   digit3
0011  002F  0151         movwf   digit4
0012  0C0C  0152         movlw   MAXNTHS      ; set timer variables to initial values
0013  0030  0153         movwf   sec_nth
0014  0CC4  0154         movlw   MAXSECS
0015  0031  0155         movwf   seconds
0016  0CC4  0156         movlw   MAXMINS
0017  0032  0157         movwf   minutes
0018  0CFF  0158         movlw   H'FF'        ; hours start at 12 which is max at FF
0019  0033  0159         movwf   hours
001A  0C00  0160         movlw   H'00'
001B  0029  0161         movwf   flags        ; clear the flags variable
001C  0004  0162         clrwdt               ; clear WatchDog Timer, must be within every 18ms
            0163                              ;
            0164                              ;
            0165 MAIN
            0166                              ;
            0167                              ;  wait for RTCC to roll-over
            0168 RTCC_FILL
001D  0201  0169         movf    RTCC,0
001E  0743  0170         btfss   STATUS,Z     ; note, RTCC is left free running
001F  0A1D  0171         goto    RTCC_FILL
            0172                              ;
0020  03F0  0173         incfsz  sec_nth,1    ;add 1 to nths, n X nths = 1 sec, n is based on prescaler
0021  0A54  0174         goto    TIME_DONE
0022  0004  0175         clrwdt
0023  0C0C  0176         movlw   MAXNTHS
0024  0030  0177         movwf   sec_nth      ; restore sec_nths variable for next round
            0178                              ;
            0179 CHECK_SW
0025  07E9  0180         btfss   flags,SW_ON  ; if no switches pressed, bypass this
0026  0A3C  0181         goto    SET_TIME
0027  0689  0182         btfsc  flags,SW1
0028  0A3C  0183         goto   SET_TIME      ; if seconds display is pressed, do not change time
0029  0CC4  0184         movlw  MAXSECS
002A  0031  0185         movwf  seconds       ; reset seconds to zero when setting clock
002B  0C7F  0186         movlw  H'7F'
002C  0030  0187         movwf  sec_nth       ; advance second timer 1/2 second to speed setting
002D  07A9  0188         btfss  flags,SW2
002E  0A35  0189         goto   HOURSET       ; if minutes do not need changing, check hours
002F  0CAF  0190         movlw  H'AF'
0030  0030  0191         movwf  sec_nth       ; advances timer faster when setting minutes
0031  03F2  0192         incfsz minutes,1     ; advances minutes 1
0032  0A35  0193         goto   HOURSET
0033  0CC4  0194         movlw  MAXMINS
0034   0032 0195         movwf  minutes       ; if minutes roll over to zero, reinitialize
                                               ;    minutes
            0196                              ;
```



```
0035  06A9  0197 HOURSET btfsc   flags,SW2
0036  0A60  0198         goto    CHECK_TIME      ; if not changing hours (changing minutes)
                                                 ; skip this
0037  03F3  0199         incfsz  hours,1         ; advance hours 1
0038  0A60  0200         goto    CHECK_TIME
0039  0CF4  0201         movlw   MAXHRS          ; if hours rolls over to zero, reinitialize
003A  0033  0202         movwf   hours
003B  0A60  0203         goto    CHECK_TIME      ;  skip time keeping, go to display changes
            0204                                 ;
            0205 SET_TIME
003C  0509  0206         bsf     flags,SEC       ; indicates seconds, if displayed, should be
                                                 ;   updated
003D  0569  0207         bsf     flags,CHG       ; inicates a flag change was made.
003E  03F1  0208         incfsz  seconds,1       ;  add 1 to seconds
003F  0A54  0209         goto    TIME_DONE
0040  0CC4  0210         movlw   MAXSECS
0041  0031  0211         movwf   seconds         ; restore seconds variable for next round
            0212                                 ;
0042  0529  0213         bsf     flags,MIN       ; minutes, if displayed, should be updated
0043  0569  0214         bsf     flags,CHG       ; indicates a flag change was made
0044  0C09  0215         movlw   ADJMIN
0045  00B0  0216         subwf   sec_nth,1       ; accuracy adjustment, do not go below 0
0046  03F2  0217         incfsz  minutes,1       ; add 1 to minutes
0047  0A54  0218         goto    TIME_DONE
0048  0CC4  0219         movlw   MAXMINS
0049  0032  0220         movwf   minutes         ; restore minutes variable for next hour
                                                 ;   countdown
            0221                                 ;
004A  0549  0222         bsf     flags,HRS       ; hours, if displayed, should be updated
004B  0569  0223         bsf     flags,CHG       ; indicates a flag change was made
004C  0C22  0224         movlw   ADJHR
004D  01F0  0225         addwf   sec_nth,1       ; add needed adjustment to nths for each hour
004E  03F3  0226         incfsz  hours,1         ; add 1 to hours
004F  0A54  0227         goto    TIME_DONE
0050  0CF4  0228         movlw   MAXHRS
0051  0033  0229         movwf   hours           ; restore hours variable for next round
0052  0C06  0230         movlw   ADJDAY
0053  00B0  0231         subwf   sec_nth,1       ; subtraction adjustment for each 1/2 day rollover
            0232                                 ;
            0233 TIME_DONE
0054  0769  0234         btfss   flags,CHG       ; if no switches or potentially dislayed
                                                 ;   numbers
0055  0A91  0235         goto    CYCLE           ;  were changed, then leave the display same
            0236                                 ;
            0237                                 ;
            0238 CHECK_SECONDS
            0239                                 ;  if seconds button was pushed display
                                                 ;   seconds
0056  0789  0240         btfss   flags,SW1
0057  0A60  0241         goto    CHECK_TIME      ; if seconds button not pressed, skip this
0058  0C00  0242         movlw   H'00'           ; zero time display variables except seconds
                                                 ;   (digit1)
0059  002D  0243         movwf   digit2          ; digit1 used to temporarily hold hex seconds
                                                 ;   or minutes
005A  002E  0244         movwf   digit3
005B  002F  0245         movwf   digit4
005C  0CC4  0246         movlw   MAXSECS
005D  0091  0247         subwf   seconds,0       ; subtract initialized preset to get actual
                                                 ;   seconds
005E  002C  0248         movwf   digit1          ; 1st digit variable temporarily holds hex
                                                 ;   value seconds
005F  0A69  0249         goto    SPLIT_HEX       ; done updating display variables in hex
            0250                                 ;
            0251 CHECK_TIME
0060  0C00  0252         movlw   H'00'
0061  002F  0253         movwf   digit4          ; zero out tens places in case there is no tens
                                                 ;   increment
0062  002D  0254         movwf   digit2
```

```
0063  0CF3   0255          movlw   MINHRS
0064  0093   0256          subwf   hours,0          ; subtract initialized preset to get actual
                                                    ;   hours
0065  002E   0257          movwf   digit3           ; 3rd digit variable temporarily holds hex
                                                    ;   value for hours
0066  0CC4   0258          movlw   MAXMINS
0067  0092   0259          subwf   minutes,0        ; subtract initialized preset to get actual
                                                    ;   minutes
0068  002C   0260          movwf   digit1           ; 1st digit temporarily holds hex value for
                                                    ;   minutes
              0261                                  ;
              0262                                  ; note digit variables are used for temp
                                                    ;   variables and final display variables
              0263                                  ;
              0264 SPLIT_HEX                        ;  split into two hex display variables and
                                                    ;   write
              0265                                  ;
0069  0C02   0266          movlw   H'02'
006A  0035   0267          movwf   count            ; convert each number - seconds - or minutes
                                                    ;   and hours
              0268
              0269                                  ;1st time through, FSR = digit1, 2nd time FSR =
                                                    ;   digit3
006B  0C0C   0270          movlw   digit1
006C  0024   0271          movwf   FSR              ; address of digit1 into File Select Register
                                                    ;   prepares POINTER
006D  0A70   0272          goto    LOOP             ; this loop is used to modify the minutes/
                                                    ;   seconds place
              0273                                  ;
006E  0C0E   0274 LOOP2    movlw   digit3
006F  0024   0275          movwf   FSR              ; this loop is used to modify the hours place
              0276                                  ;
              0277 LOOP
0070  0C0A   0278          movlw   D'10'
0071  00A0   0279          subwf   POINTER,1        ; find out how many tens in number,
0072  0603   0280          btfsc   STATUS,C         ; was a borrow needed?
0073  0A76   0281          goto    INCREMENT_10S    ;  if not, add 1 to tens position
0074  01E0   0282          addwf   POINTER,1        ;  if so, do not increment tens place, add ten
                                                    ;   back on
0075  0A7A   0283          goto    NEXT_DIGIT
              0284                                  ;
              0285 INCREMENT_10S
0076  02A4   0286          incf    FSR,1            ; bump address pointed to from 1s positoion to
                                                    ;   10s
0077  02A0   0287          incf    POINTER,1        ; add 1 to 10s position as determined by
                                                    ;   previous subtract
0078  00E4   0288          decf    FSR,1            ; put POINTER value back to 1s place for next
                                                    ;   subtraction
0079  0A70   0289          goto    LOOP             ; go back and keep subtracting until finished
              0290                                  ;
              0291 NEXT_DIGIT
007A  02F5   0292          decfsz  count,1
007B  0A6E   0293          goto    LOOP2            ; after splitting minutes into two places, go
                                                    ;   split hours
              0294                                  ;
              0295 CONVERT_HEX_TO_DISPLAY           ; converts digit variables to decimal display
                                                    ;   code
007C  0C0C   0296          movlw   digit1
007D  0024   0297          movwf   FSR              ; put the address of the digit1 into the FSR to
                                                    ;   enable POINTER
007E  0C04   0298          movlw   H'04'
007F  0035   0299          movwf   count            ; prepare count variable to loop for all four
                                                    ;   displays
              0300 NEXT_HEX
0080  0200   0301          movf    POINTER,0        ; get the hex value of the current digit vari
                                                    ;   able
0081  09C3   0302          call    RETURN_CODE      ; call for the hex to segment display code
                                                    ;   conversion
0082  0020   0303          movwf   POINTER          ; put the returned display code into the digit
```



```
                                                        ;   variable
0083  02A4   0304          incf    FSR,1               ; increment the pointer to the next digit
                                                        ;   address
0084  02F5   0305          decfsz  count,1             ; allow only count(4) times through loop
0085  0A80   0306          goto    NEXT_HEX
             0307                                       ;
             0308 FIX_DISPLAY
0086  0C7E   0309          movlw   ZERO
0087  008F   0310          subwf   digit4,0            ; check to see if left digit is a zero, if so
                                                        ;   blank it out
0088  0743   0311          btfss   STATUS,Z
0089  0A8C   0312          goto    FIX_SEC
008A  0C00   0313          movlw   BLANK
008B  002F   0314          movwf   digit4
             0315
008C  0789   0316 FIX_SEC  btfss   flags,SW1           ; if seconds are displayed, blank the third
                                                        ;   display too
008D  0A8F   0317          goto    CLEAR_FLAGS
008E  002E   0318          movwf   digit3
             0319                                       ;
             0320 CLEAR_FLAGS
008F  0CF0   0321          movlw   H'F0'
0090  0169   0322          andwf   flags,1             ; clear the lower 4 flag bits to show updated
                                                        ;   time status
             0323                                       ;
             0324 CYCLE
0091  0CFF   0325          movlw   DISPOFF
0092  0025   0326          movwf   PORT_A              ; Turn off LED Displays
0093  0C0E   0327          movlw   SWITCH
0094  0006   0328          tris    PORT_B              ; Set some port B pins as switch inputs
0095  0C0F   0329          movlw   H'0F'
0096  0169   0330          andwf   flags,1             ; reset switch flags to zero
0097  0000   0331          nop                         ; nop may not be needed, allows old outputs to
0098  0000   0332          nop                         ;   bleedoff through 10k R before reading port
                                                        ;   pins
0099  0000   0333          nop
009A  0206   0334          movf    PORT_B,0            ; read PORT_B for switch status
009B  0034   0335          movwf   var                 ;  write switch status to temporary variable
                                                        ;   "var"
009C  0734   0336          btfss   var,1
009D  0AA1   0337          goto    SWITCH2             ; indicate which switches are pressed in the
                                                        ;   flags variable
009E  0569   0338          bsf     flags,CHG
009F  0589   0339          bsf     flags,SW1
00A0  05E9   0340          bsf     flags,SW_ON
00A1  0754   0341 SWITCH2  btfss   var,2
00A2  0AA6   0342          goto    SWITCH3
00A3  0569   0343          bsf     flags,CHG
00A4  05A9   0344          bsf     flags,SW2
00A5  05E9   0345          bsf     flags,SW_ON
00A6  0774   0346 SWITCH3  btfss   var,3
00A7  0AAB   0347          goto    SETPORT
00A8  0569   0348          bsf     flags,CHG
00A9  05C9   0349          bsf     flags,SW3
00AA  05E9   0350          bsf     flags,SW_ON
             0351                                       ;
00AB  0C00   0352 SETPORT  movlw   H'00'               ; restore PORT_B as all outputs to displays
00AC  0006   0353          tris    PORT_B
00AD  0C00   0354             movlw   BLANK            ; blank display in preparation for next digit
                                                        ;   cycle
00AE  0026   0355             movwf   PORT_B
             0356                                       ;
             0357                                       ; determine which display needs updating and
                                                        ;   cycle it on
00AF  070B   0358             btfss   display,0        ; if 1st display, get 1st digit value into w
00B0  020F   0359             movf    digit4,0
00B1  072B   0360             btfss   display,1        ; if 2nd display, get 2nd digit
00B2  020E   0361             movf    digit3,0
00B3  074B   0362             btfss   display,2        ; if 3rd display, get 3rd digit
```

```
00B4  020D   0363            movf    digit2,0
00B5  076B   0364            btfss   display,3       ; if 4th display, get 4th digit
00B6  020C   0365            movf    digit1,0
00B7  0026   0366            movwf   PORT_B          ; put the number in w out to display
00B8  06F0   0367            btfsc   sec_nth,7
00B9  0506   0368            bsf     PORT_B,0        ; sets colon decimal on %50 duty using highest
                                                     ;   bit
00BA  020B   0369            movf    display,0       ; get display needing cycle on
00BB  0025   0370            movwf   PORT_A          ; enables proper display
00BC  002B   0371            movwf   display         ; enables display selected in last pass of
                                                     ;   CYCLE
00BD  036B   0372            rlf     display,1       ; rotate display “on” bit to next position
00BE  050B   0373            bsf     display,0       ; assures a 1 on lowest position since rotated
                                                     ;   in carry is zero
00BF  078B   0374            btfss   display,4       ; check if last display was already updated
00C0  040B   0375            bcf     display,0       ; if it was, set display back to 1st (bit 0
                                                     ;   cleared)
00C1  0004   0376            clrwdt                  ; this program pass completed normally, reset
                                                     ;   watch dog
             0377                                    ;
             0378                                    ;
             0379                                    ;
00C2  0A1D   0380            goto    MAIN
             0381                                    ;
             0382 RETURN_CODE
             0383                                    ;
00C3  01E2   0384            addwf   PC,1            ; the hex value in the display variable is
00C4  087E   0385            retlw   ZERO            ;   added to PC which causes a jump to return
                                                     ;   its display code
00C5  080C   0386            retlw   ONE
00C6  08B6   0387            retlw   TWO
00C7  089E   0388            retlw   THREE
00C8  08CC   0389            retlw   FOUR
00C9  08DA   0390            retlw   FIVE
00CA  08FA   0391            retlw   SIX
00CB  080E   0392            retlw   SEVEN
00CC  08FE   0393            retlw   EIGHT
00CD  08CE   0394            retlw   NINE
             0395                                    ;
             0396                                    ;
             0397            org     PIC54           ; reset location for this processor
Warning: Crossing page boundary — ensure page bits are set
01FF  0A00   0398            goto    START           ; begin program execution at START label
             0399                                    ;
             0400     END
             0401

MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX

0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXX-- -------- -------- --------

0180 : -------- -------- -------- --------
01C0 : -------- -------- -------- -------X

All other memory blocks unused.

Errors   :    0
Warnings :    1
```


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