

---

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

---

Author: Dan Matthews  
Microchip Technology Inc.

### 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 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 PORTA. 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 PORTB 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 PORTB I/O pins. PORTB pins RB1, RB2, and RB3 are pulled low with 10 k $\Omega$  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 VDD through 820 $\Omega$  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 VDD 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 in its turn. This is done fast enough so that there is no perceived flicker. The switches are read between display cycles.

#### Timing

The PIC16C5X prescaler is assigned to TMR0 as a 1:16 divide. The T0CKI pin is tied low since it is not used. The OPTION Register is loaded with 03h to initialize this prescaler setup. The software is written with timing based on a 4.000 MHz crystal. The instruction clock is 1.000 MHz after the internal divide by four. The 8-bit TMR0 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 TMR0 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 "nth's" 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 "nth's" a day (an "nth" is 1/244 of a second in this case). In this circuit, 9 "nth's" are subtracted each minute, 34 "nth's" are added each hour, and 6 "nth's" 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 TMR0 to some value that causes a roll-over at a predetermined time interval. Writing to TMR0 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*.

\_\_\_\_\_

be provided if different displays are used. Second,

## CONCLUSION

DATE	TIME	BY	DESCRIPTION	AMOUNT
10/1/19	10:00	JOHN	PAID FOR	100.00
10/2/19	10:00	JOHN	PAID FOR	100.00
10/3/19	10:00	JOHN	PAID FOR	100.00
10/4/19	10:00	JOHN	PAID FOR	100.00
10/5/19	10:00	JOHN	PAID FOR	100.00
10/6/19	10:00	JOHN	PAID FOR	100.00
10/7/19	10:00	JOHN	PAID FOR	100.00
10/8/19	10:00	JOHN	PAID FOR	100.00
10/9/19	10:00	JOHN	PAID FOR	100.00
10/10/19	10:00	JOHN	PAID FOR	100.00
10/11/19	10:00	JOHN	PAID FOR	100.00
10/12/19	10:00	JOHN	PAID FOR	100.00
10/13/19	10:00	JOHN	PAID FOR	100.00
10/14/19	10:00	JOHN	PAID FOR	100.00
10/15/19	10:00	JOHN	PAID FOR	100.00
10/16/19	10:00	JOHN	PAID FOR	100.00
10/17/19	10:00	JOHN	PAID FOR	100.00
10/18/19	10:00	JOHN	PAID FOR	100.00
10/19/19	10:00	JOHN	PAID FOR	100.00
10/20/19	10:00	JOHN	PAID FOR	100.00
10/21/19	10:00	JOHN	PAID FOR	100.00
10/22/19	10:00	JOHN	PAID FOR	100.00
10/23/19	10:00	JOHN	PAID FOR	100.00
10/24/19	10:00	JOHN	PAID FOR	100.00
10/25/19	10:00	JOHN	PAID FOR	100.00
10/26/19	10:00	JOHN	PAID FOR	100.00
10/27/19	10:00	JOHN	PAID FOR	100.00
10/28/19	10:00	JOHN	PAID FOR	100.00
10/29/19	10:00	JOHN	PAID FOR	100.00
10/30/19	10:00	JOHN	PAID FOR	100.00
10/31/19	10:00	JOHN	PAID FOR	100.00
11/1/19	10:00	JOHN	PAID FOR	100.00
11/2/19	10:00	JOHN	PAID FOR	100.00
11/3/19	10:00	JOHN	PAID FOR	100.00
11/4/19	10:00	JOHN	PAID FOR	100.00
11/5/19	10:00	JOHN	PAID FOR	100.00
11/6/19	10:00	JOHN	PAID FOR	100.00
11/7/19	10:00	JOHN	PAID FOR	100.00
11/8/19	10:00	JOHN	PAID FOR	100.00
11/9/19	10:00	JOHN	PAID FOR	100.00
11/10/19	10:00	JOHN	PAID FOR	100.00
11/11/19	10:00	JOHN	PAID FOR	100.00
11/12/19	10:00	JOHN	PAID FOR	100.00
11/13/19	10:00	JOHN	PAID FOR	100.00
11/14/19	10:00	JOHN	PAID FOR	100.00
11/15/19	10:00	JOHN	PAID FOR	100.00
11/16/19	10:00	JOHN	PAID FOR	100.00
11/17/19	10:00	JOHN	PAID FOR	100.00
11/18/19	10:00	JOHN	PAID FOR	100.00
11/19/19	10:00	JOHN	PAID FOR	100.00
11/20/19	10:00	JOHN	PAID FOR	100.00
11/21/19	10:00	JOHN	PAID FOR	100.00
11/22/19	10:00	JOHN	PAID FOR	100.00
11/23/19	10:00	JOHN	PAID FOR	100.00
11/24/19	10:00	JOHN	PAID FOR	100.00
11/25/19	10:00	JOHN	PAID FOR	100.00
11/26/19	10:00	JOHN	PAID FOR	100.00
11/27/19	10:00	JOHN	PAID FOR	100.00
11/28/19	10:00	JOHN	PAID FOR	100.00
11/29/19	10:00	JOHN	PAID FOR	100.00
11/30/19	10:00	JOHN	PAID FOR	100.00
12/1/19	10:00	JOHN	PAID FOR	100.00
12/2/19	10:00	JOHN	PAID FOR	100.00
12/3/19	10:00	JOHN	PAID FOR	100.00
12/4/19	10:00	JOHN	PAID FOR	100.00
12/5/19	10:00	JOHN	PAID FOR	100.00
12/6/19	10:00	JOHN		

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

## APPENDIX A: CLOCK54.ASM

MPASM 01.40 Released      CLOCK54.ASM    1-16-1997   17:23:58      PAGE   1

LOC	OBJECT CODE	LINE	SOURCE TEXT	VALUE
00001		;		
00002		;	*****	
00003		LIST	P = 16C54, n = 66	
00004		;		
00005			Clock	
00006		;	*****	
00007		;		
00008			PROGRAM DESCRIPTION	
00009		;		
00010		;	This program runs on a PIC16C54.	
00011		;		
00012			Hardware Description	
00013		;		
00014		DISPLAYS		
00015		;	Four 7 segment displays are multiplexed. The segments are tied together, with	
00016		;	the common cathode pins broken out separately. The display appears as a clock	
00017		;	with a center semicolon ( 88:88 ). The segments are assigned to Port B, with the	
00018		;	semicolon being RB0, and segments A through F assigned as RB1 to RB7 respectively.	
00019		;	The four common cathodes are attached to the four Port A pins through transistors.	
00020		;	RA0 for LED0, RA1/LED1... through LED3. The center semicolon is made from the decimals	
00021		;	of LED 2 and 3. LED display 2 is turned upside down to put its decimal into position,	
00022		;	but it is wired with a corrected A-F assignment to compensate. Both decimals	
00023		;	are tied together at RB0, but the display cathodes are still separate.	
00024		;		
00025		SWITCHES		
00026		;	Because all twelve I/O pins are already used for the muxed displays, the four	
00027		;	switches must be switched in alternately through software. The switches	
00028		;	lie across Port B pins, which will be changed to inputs momentarily during read	
00029		;	and changed back to outputs during display.	
00030		;		
00031		;		
00032		;		
00033		Program:	CLOCK54.ASM	
00034		Revision Date:		
00035			1-16-97      Compatibility with MPASMWIN 1.40	
00036		;		

```

00037 ;
00038 ;***** Header *****
00039 ;
00040 ;
000001FF 00041 PIC54 equ H'01FF'
000003FF 00042 PIC56 equ H'03FF'
00043 ;
00000000 00044 POINTER equ H'00'
00000001 00045 TMR0 equ H'01'
00000002 00046 PC equ H'02'
00000003 00047 STATUS equ H'03' ; F3 Reg is STATUS Reg.
00000004 00048 FSR equ H'04'
00049 ;
00000005 00050 PORT_A equ H'05' ; 7 segment Display Common Cathodes
00000006 00051 PORT_B equ H'06' ; Center Colon and Muxed Display Segments (Switches when inputs)
00052 ;
00053 ; STATUS REG. Bits
00000000 00054 CARRY equ 0 ; Carry Bit is Bit.0 of F3
00000000 00055 C equ 0
00000001 00056 DCARRY equ 1
00000001 00057 DC equ 1
00000002 00058 Z_bit equ 2 ; Bit 2 of F3 is Zero Bit
00000002 00059 Z equ 2
00000003 00060 P_DOWN equ 3
00000003 00061 PD equ 3
00000004 00062 T_OUT equ 4
00000004 00063 TO equ 4
00000005 00064 PA0 equ 5 ;16C5X Status bits
00000006 00065 PA1 equ 6 ;16C5X Status bits
00000007 00066 PA2 equ 7 ;16C5X Status bits
00067 ;
0000007E 00068 ZERO equ H'7E'
0000000C 00069 ONE equ H'0C'
000000B6 00070 TWO equ H'B6'
0000009E 00071 THREE equ H'9E'
000000CC 00072 FOUR equ H'CC'
000000DA 00073 FIVE equ H'DA'
000000FA 00074 SIX equ H'FA' ; Mapping of segments for display (PORT_B)
0000000E 00075 SEVEN equ H'0E'
000000FE 00076 EIGHT equ H'FE'
000000CE 00077 NINE equ H'CE'
00000001 00078 COLON equ H'01'
000000F0 00079 T equ H'F0'
00000000 00080 BLANK equ H'00'
00081 ;
0000000C 00082 MAXNTHS equ D'12' ; constants for timer variable count up
000000C4 00083 MAXSECS equ D'196' ; variables roll over in HEX at time roll over, see variable

```

```

000000C4      00084 MAXMINS equ    D'196'      ; explanation
000000F4      00085 MAXHRS  equ    D'244'
000000F3      00086 MINHRS  equ    D'243'
00000009      00087 ADJMIN  equ    D'9'        ; number of nth's that need to be subtracted each minute
00000022      00088 ADJHR   equ    D'34'       ; nth's added each hour for accurate time
00000003      00089 ADJDAY  equ    D'3'        ; nth's subtracted each 1/2 day rollover
00090 ;
000000FE      00091 DISP1   equ    B'11111110'
000000FD      00092 DISP2   equ    B'11111101' ; Mapping of Active Display Selection (PORT_A)
000000FB      00093 DISP3   equ    B'11111011'
000000F7      00094 DISP4   equ    B'11110111'
000000FF      00095 DISPOFF equ    H'FF'
0000000E      00096 SWITCH  equ    B'00001110' ; Activate RB1-3 for switch inputs
00097 ;
00098 ; Flag bit assignments
00000000      00099 SEC     equ    H'0'        ; update time display values for sec, min, or hours
00000001      00100 MIN     equ    H'1'
00000002      00101 HRS     equ    H'2'
00000003      00102 CHG     equ    H'3'        ; a change has occurred on a switch or to a potentially displayed value

00000004      00103 SW1     equ    H'4'        ; Flag bit assignments - switches that are on = 1
00000005      00104 SW2     equ    H'5'        ; SW1 is Seconds-minutes, SW2-hours, SW3-mode
00000006      00105 SW3     equ    H'6'
00000007      00106 SW_ON   equ    H'7'        ; a switch has been pressed
00107 ;
00108 ; VARIABLES
00000008      00109 keys    equ    H'08'       ; variable location - which keys are pressed? bit0/sw1...
00000009      00110 flags   equ    H'09'       ; bit flags; 0-SEC, 1-MIN, 2-HRS, 3-CHG, 4-SW1, 5-SW2, 6-SW3
00111 ;      equ    H'0A'       ; Not Used
0000000B      00112 display equ    H'0B'       ; variable location - which display to update
0000000C      00113 digit1  equ    H'0C'       ; Rightmost display value
0000000D      00114 digit2  equ    H'0D'       ; Second display from right
0000000E      00115 digit3  equ    H'0E'       ; Third      "      "
0000000F      00116 digit4  equ    H'0F'       ; Fourth (and Leftmost)
00117 ;
00118 ;      timer variables start at a number that allows rollover in sync with time rollover,
00119 ;      i.e. seconds starts at decimal 195 so that sixty 1-second increments causes 0.
00000010      00120 sec_nth equ    H'10'       ; seconds, fractional place
00000011      00121 seconds equ    H'11'       ; seconds
00000012      00122 minutes equ    H'12'       ; minutes
00000013      00123 hours   equ    H'13'       ; hours
00000014      00124 var     equ    H'14'       ; variable for misc math computations
00000015      00125 count   equ    H'15'       ; loop counter variable
00000016      00126 count2  equ    H'16'       ; 2nd loop counter for nested loops
00127
00128 ;
00129 ;*****

```

```

00130 ;
00131 ; Initialize Ports all outputs, blank display
00132 ;
0000 0C03 00133 START    movlw   H'03'           ; set option register, transition on clock,
0001 0002 00134         option                ; Prescale TMR0, 1:16
00135 ;
0002 0C00 00136         movlw   0
0003 0005 00137         tris    PORT_A         ; Set all port pins as outputs
0004 0006 00138         tris    PORT_B
0005 0C00 00139         movlw   BLANK
0006 0026 00140         movwf   PORT_B         ; Blank the display
0007 04C3 00141         bcf     STATUS,PA1
0008 04A3 00142         bcf     STATUS,PA0
00143 ;
00144 ; initialize variables
0009 0C01 00145         movlw   H'01'
000A 0021 00146         movwf   TMR0           ; set TMR0 above zero so initial wait period occurs
000B 0CFE 00147         movlw   H'FE'
000C 002B 00148         movwf   display        ; initializes display selected to first display.
000D 0C00 00149         movlw   BLANK         ; put all displays to blank, no visible segments
000E 002C 00150         movwf   digit1
000F 002D 00151         movwf   digit2
0010 002E 00152         movwf   digit3
0011 002F 00153         movwf   digit4
0012 0C0C 00154         movlw   MAXNTHS       ; set timer variables to initial values
0013 0030 00155         movwf   sec_nth
0014 0CC4 00156         movlw   MAXSECS
0015 0031 00157         movwf   seconds
0016 0CC4 00158         movlw   MAXMINS
0017 0032 00159         movwf   minutes
0018 0CFF 00160         movlw   H'FF'         ; hours start at 12 which is max at FF
0019 0033 00161         movwf   hours
001A 0C00 00162         movlw   H'00'
001B 0029 00163         movwf   flags
00164 ;
00165 ;? call converts for minutes and hours to initialize display Variables
00166 ;
001C      00167 MAIN
00168 ;
00169 ; wait for TMR0 to roll-over
001C      00170 TMR0_FILL
001C 0201 00171         movf     TMR0,0
001D 0743 00172         btfss   STATUS,Z       ; note, TMR0 is left free running to not lose clock cycles on writes
001E 0A1C 00173         goto    TMR0_FILL
00174 ;
001F 03F0 00175         incfsz  sec_nth,1      ; add 1 to nth, n X nth = 1 sec, n is based on prescaler
0020 0A52 00176         goto    TIME_DONE

```

```

0021 0C0C      00177      movlw   MAXNTHS
0022 0030      00178      movwf   sec_nth      ; restore sec_nth variable for next round
              00179      ;
0023          00180 CHECK_SW
0023 07E9      00181      btfs    flags,SW_ON    ; if no switches press, bypass this
0024 0A3A      00182      goto    SET_TIME
0025 0689      00183      btfs    flags,SW1
0026 0A3A      00184      goto    SET_TIME      ; if seconds display is pressed, do not change time
0027 0CC4      00185      movlw   MAXSECS
0028 0031      00186      movwf   seconds      ; reset seconds to zero when setting clock
0029 0C7F      00187      movlw   H'7F'
002A 0030      00188      movwf   sec_nth      ; advance second timer 1/2 second to speed time setting
002B 07A9      00189      btfs    flags,SW2
002C 0A33      00190      goto    HOURSET      ; minutes do not need changing, check hours
002D 0CAF      00191      movlw   H'AF'
002E 0030      00192      movwf   sec_nth      ; advances timer faster when setting minutes
002F 03F2      00193      incfsz  minutes,1
0030 0A33      00194      goto    HOURSET
0031 0CC4      00195      movlw   MAXMINS
0032 0032      00196      movwf   minutes
              00197      ;
0033 06A9      00198 HOURSET btfs    flags,SW2
0034 0A5E      00199      goto    CHECK_TIME    ; not changing hours
0035 03F3      00200      incfsz  hours,1
0036 0A5E      00201      goto    CHECK_TIME
0037 0CF4      00202      movlw   MAXHRS
0038 0033      00203      movwf   hours
0039 0A5E      00204      goto    CHECK_TIME    ; since no timing is required, go to display changes
              00205      ;
003A          00206 SET_TIME
003A 0509      00207      bsf     flags,SEC      ; seconds, if displayed, should be updated
003B 0569      00208      bsf     flags,CHG      ; a flag change was made.
003C 03F1      00209      incfsz  seconds,1      ; add 1 to seconds
003D 0A52      00210      goto    TIME_DONE
003E 0CC4      00211      movlw   MAXSECS
003F 0031      00212      movwf   seconds      ; restore seconds variable for next round
              00213      ;
0040 0529      00214      bsf     flags,MIN      ; minutes, if displayed, should be updated
0041 0569      00215      bsf     flags,CHG
0042 0C09      00216      movlw   ADJMIN
0043 00B0      00217      subwf   sec_nth,1      ; subtraction needed adjustment for each minute
0044 03F2      00218      incfsz  minutes,1      ; add 1 to minutes
0045 0A52      00219      goto    TIME_DONE
0046 0CC4      00220      movlw   MAXMINS
0047 0032      00221      movwf   minutes      ; restore minutes variable for next hour countdown
              00222      ;
0048 0549      00223      bsf     flags,HRS

```

```

0049 0569      00224      bsf      flags,CHG
004A 0C22      00225      movlw    ADJHR
004B 01F0      00226      addwf    sec_nth,1      ; add needed adjustment for each hour
004C 03F3      00227      incfsz   hours,1      ; add 1 to hours
004D 0A52      00228      goto     TIME_DONE
004E 0CF4      00229      movlw    MAXHRS
004F 0033      00230      movwf    hours      ; restore hours variable for next round
0050 0C03      00231      movlw    ADJDAY
0051 00B0      00232      subwf    sec_nth,1      ; subtraction adjustment for each 1/2 day rollover
                00233      ;
0052           00234      TIME_DONE
0052 0769      00235      btfss    flags,CHG      ; if no switches or potentially displayed numbers were
0053 0A8F      00236      goto     CYCLE      ; changed, then skip updating display variables
                00237      ;
                00238      ;
0054           00239      CHECK_SECONDS
                00240      ; if seconds is button was pushed and not mode display seconds
0054 0789      00241      btfss    flags,SW1
0055 0A5E      00242      goto     CHECK_TIME
0056 0C00      00243      movlw    H'00'
0057 002D      00244      movwf    digit2      ; 3rd digit variable used to store temp hex value for hours display
0058 002E      00245      movwf    digit3
0059 002F      00246      movwf    digit4
005A 0CC4      00247      movlw    MAXSECS
005B 0091      00248      subwf    seconds,0
005C 002C      00249      movwf    digit1      ; 1st digit variable temporarily holds hex value for seconds display
005D 0A67      00250      goto     SPLIT_HEX
                00251      ;
005E           00252      CHECK_TIME
005E 0C00      00253      movlw    H'00'
005F 002F      00254      movwf    digit4      ; zero out tens places in case there is no tens increment
0060 002D      00255      movwf    digit2
0061 0CF3      00256      movlw    MINHRS
0062 0093      00257      subwf    hours,0
0063 002E      00258      movwf    digit3      ; 3rd digit variable temporarily holds hex value for hours
0064 0CC4      00259      movlw    MAXMINS
0065 0092      00260      subwf    minutes,0
0066 002C      00261      movwf    digit1      ; 1st digit temporarily holds hex value for minutes
                00262      ;
                00263      ;
                00264      ;
0067           00265      SPLIT_HEX      ; split into two hex display variables and write
                00266      ;
0067 0C02      00267      movlw    H'02'
0068 0035      00268      movwf    count      ; loop to convert each number - seconds - or minutes and hours
                00269      ;
                00270      ;1st time through, FSR = digit1, 2nd time FSR = digit3

```



```

0069 0C0C      00271      movlw   digit1      ;
006A 0024      00272      movwf   FSR          ; address of digit1 into File Select Register enables POINTER
006B 0A6E      00273      goto    LOOP        ; this loop is used to modify the minutes/seconds place
00274 ;
006C 0C0E      00275 LOOP2   movlw   digit3
006D 0024      00276      movwf   FSR          ; this loop is used to modify the hours place
00277 ;
006E           00278 LOOP
006E 0C0A      00279      movlw   D'10'
006F 00A0      00280      subwf   POINTER,1      ; find out how many tens in number,
0070 0603      00281      btfsc   STATUS,C      ; was a borrow needed?
0071 0A74      00282      goto    INCREMENT_10S ; if not, add 1 to tens position
0072 01E0      00283      addwf   POINTER,1      ; if so, do not increment tens place, add ten back on to get 1s
0073 0A78      00284      goto    NEXT_DIGIT
00285 ;
0074           00286 INCREMENT_10S
0074 02A4      00287      incf     FSR,1          ; bump address pointed to from 1st position to 10s
0075 02A0      00288      incf     POINTER,1      ; add 1 to 10s position as determined by previous subtract
0076 00E4      00289      decf     FSR,1          ; put POINTER value back to 1s place for next subtraction
0077 0A6E      00290      goto    LOOP        ; go back and keep subtracting until finished
00291 ;
0078           00292 NEXT_DIGIT
0078 02F5      00293      decfsz   count,1
0079 0A6C      00294      goto    LOOP2
00295 ;
007A           00296 CONVERT_HEX_TO_DISPLAY      ; converts hex number in digit variables to decimal display code
007A 0C0C      00297      movlw   digit1
007B 0024      00298      movwf   FSR          ; put the address of the first digit into the FSR to enable POINTER
007C 0C04      00299      movlw   H'04'
007D 0035      00300      movwf   count      ; prepare count variable to loop for all four displays
007E           00301 NEXT_HEX
007E 0200      00302      movf     POINTER,0      ; get the hex value of the current digit variable
007F 09C0      00303      call    RETURN_CODE    ; call for the hex to decimal display conversion
0080 0020      00304      movwf   POINTER      ; put the returned display code back into the digit variable
0081 02A4      00305      incf     FSR,1          ; increment the pointer to the next digit variable address
0082 02F5      00306      decfsz   count,1      ; allow only count(4) times through loop
0083 0A7E      00307      goto    NEXT_HEX
00308 ;
0084           00309 FIX_DISPLAY
0084 0C7E      00310      movlw   ZERO
0085 008F      00311      subwf   digit4,0
0086 0743      00312      btfss   STATUS,Z
0087 0A8A      00313      goto    FIX_SEC
0088 0C00      00314      movlw   BLANK
0089 002F      00315      movwf   digit4
00316
008A 0789      00317 FIX_SEC btfss   flags,SW1

```

```

008B 0A8D      00318      goto      CLEAR_FLAGS
008C 002E      00319      movwf    digit3
                   00320      ;
008D           00321      CLEAR_FLAGS
008D 0CF0      00322      movlw    H'F0'
008E 0169      00323      andwf    flags,1      ; clear the lower 4 flag bits to show update status
                   00324      ;
008F           00325      CYCLE
                   00326      ;
008F 0CFF      00327      movlw    DISPOFF
0090 0025      00328      movwf    PORT_A      ; Turn off LED Displays
0091 0C0E      00329      movlw    SWITCH
0092 0006      00330      tris     PORT_B      ; Set some port B pins as switch inputs
0093 0C0F      00331      movlw    H'0F'
0094 0169      00332      andwf    flags,1      ; reset switch flags to zero
0095 0000      00333      nop              ; nop may not be needed, allows old outputs to bleed
0096 0000      00334      nop              ; off through 10k R before reading port pins
0097 0000      00335      nop
0098 0206      00336      movf     PORT_B,0
0099 0034      00337      movwf    var
009A 0734      00338      btfss    var,1
009B 0A9F      00339      goto     SWITCH2
009C 0569      00340      bsf      flags,CHG
009D 0589      00341      bsf      flags,SW1
009E 05E9      00342      bsf      flags,SW_ON
009F 0754      00343      SWITCH2 btfss    var,2
00A0 0AA4      00344      goto     SWITCH3
00A1 0569      00345      bsf      flags,CHG
00A2 05A9      00346      bsf      flags,SW2
00A3 05E9      00347      bsf      flags,SW_ON
00A4 0774      00348      SWITCH3 btfss    var,3
00A5 0AA9      00349      goto     SETPORT
00A6 0569      00350      bsf      flags,CHG
00A7 05C9      00351      bsf      flags,SW3
00A8 05E9      00352      bsf      flags,SW_ON
                   00353      ;
00A9 0C00      00354      SETPORT movlw    H'00'
00AA 0006      00355      tris     PORT_B
00AB 0C00      00356      movlw    BLANK
00AC 0026      00357      movwf    PORT_B
                   00358      ;
                   00359      ; determine which display needs updating and cycle it on
00AD 070B      00360      btfss    display,0      ; if 1st display, get 1st digit
00AE 020F      00361      movf     digit4,0
00AF 072B      00362      btfss    display,1      ; if 2nd display, get 2nd digit
00B0 020E      00363      movf     digit3,0
00B1 074B      00364      btfss    display,2      ; if 3rd display, get 3rd digit

```

```

00B2 020D      00365      movf    digit2,0
00B3 076B      00366      btfss   display,3      ; if 4th display, get 4th digit
00B4 020C      00367      movf    digit1,0
00B5 0026      00368      movwf   PORT_B      ; put the number out to display
00B6 06F0      00369      btfsc   sec_nth,7
00B7 0506      00370      bsf     PORT_B,0      ; sets colon decimal on %50 duty using highest bit
00B8 020B      00371      movf    display,0      ; get display needing cycle on
00B9 0025      00372      movwf   PORT_A      ; enables proper display
00BA 002B      00373      movwf   display      ; returns old w if not done, new w if resetting display
00BB 036B      00374      rlf     display,1      ; rotate display "on" bit to next position
00BC 050B      00375      bsf     display,0      ; assures a 1 on lowest position since rotated carry is zero
00BD 078B      00376      btfss   display,4      ; check if last display was already updated
00BE 040B      00377      bcf     display,0      ; if it was, set display back to 1st (bit 0 set)
                   00378 ;
                   00379 ;
                   00380 ;
00BF 0A1C      00381      goto    MAIN
                   00382 ;
00C0           00383      RETURN_CODE
                   00384 ;
00C0 01E2      00385      addwf   PC,1
00C1 087E      00386      retlw   ZERO
00C2 080C      00387      retlw   ONE
00C3 08B6      00388      retlw   TWO
00C4 089E      00389      retlw   THREE
00C5 08CC      00390      retlw   FOUR
00C6 08DA      00391      retlw   FIVE
00C7 08FA      00392      retlw   SIX
00C8 080E      00393      retlw   SEVEN
00C9 08FE      00394      retlw   EIGHT
00CA 08CE      00395      retlw   NINE
                   00396 ;
                   00397 ;
01FF           00398      org     PIC54
01FF 0A00      00399      goto    START
                   00400 ;
                   00401      END

```

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

```

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXX-----
01C0 : -----X

```

All other memory blocks unused.

Program Memory Words Used: 204  
Program Memory Words Free: 308

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

---

# WORLDWIDE SALES & SERVICE

---

## AMERICAS

### Corporate Office

Microchip Technology Inc.  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 602-786-7200 Fax: 602-786-7277  
Technical Support: 602 786-7627  
Web: <http://www.microchip.com>

### Atlanta

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

### Boston

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

### Chicago

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

### Dallas

Microchip Technology Inc.  
14651 Dallas Parkway, Suite 816  
Dallas, TX 75240-8809  
Tel: 972-991-7177 Fax: 972-991-8588

### Dayton

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

### Los Angeles

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

### New York

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

### San Jose

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

### Toronto

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

## ASIA/PACIFIC

### Hong Kong

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

### India

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

### Korea

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

### Shanghai

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

### Singapore

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

### Taiwan, R.O.C

Microchip Technology Taiwan  
10F-1C 207  
Tung Hua North Road  
Taipei, Taiwan, ROC  
Tel: 886 2-717-7175 Fax: 886-2-545-0139

## EUROPE

### United Kingdom

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

### France

Arizona Microchip Technology SARL  
Zone Industrielle de la Bonde  
2 Rue du Buisson aux Fraises  
91300 Massy, France  
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

### Germany

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

### Italy

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

## JAPAN

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

5/8/97



# MICROCHIP

All rights reserved. © 1997, Microchip Technology Incorporated, USA. 6/97

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