
Clock Design Using Low Power/Cost Techniques

*Author: John Day
Sr., Field Application Engineer, Boston*

INTRODUCTION

Typical embedded control applications place demands such as low power consumption, small size, low cost and reduced component count onto the microcontroller. This application note implements a 24-hour digital clock, alarm and 99 minute 59 second count down timer, yet operates on two "AA" batteries. The PIC16C54A is perfect for this application, due to its small size, high current I/Os with direct LED drive, low cost, fast instruction throughput and low frequency/current operation.

System cost

The objective of this design was to implement the maximum number of features with the least expensive and smallest device. The PIC16C54A is Microchip's lowest cost microcontroller and it has 12 I/O lines, each capable of sinking 25 mA and sourcing 20 mA. High efficiency common cathode LED displays were chosen for their 3.5 mA current requirement, eliminating the need for any external transistors for display drive. A low impedance direct drive piezo buzzer was chosen and its tone is generated by the software of the PIC16C54A to further reduce system cost.

Operating power

In battery powered applications, the operating current determines the lifetime of the batteries. There are many ways to reduce the operating current of any application, including low frequency operation and the use of sleep mode. Since the clock has to keep track of time, SLEEP mode could not be used and the processor must be kept running all of the time. The PIC16C54A supports the 32.768 kHz "watch" crystal and typically consumes less than 15 μ A of current in this configuration. Since the PIC16C54A executes instructions in one cycle and its instruction set is very efficient, this application was able to be implemented using a low frequency crystal. Another solution to this problem comes with the PIC16C74/73/65/63 in its Timer1 module. Timer1 will run when the device is asleep, so it could have been used to keep track of time, simplifying the software.

Clock system

A 32.768 kHz crystal was chosen for the clock due to the low power and cost requirements of this design. The four internal phases of this input clock create an internal instruction cycle. Therefore, the instruction time is calculated as follows:

$$\text{Instruction rate} = \frac{1}{(\text{CLKIN}/4)} = \frac{1}{32,768/4} = \frac{1}{8,192}$$

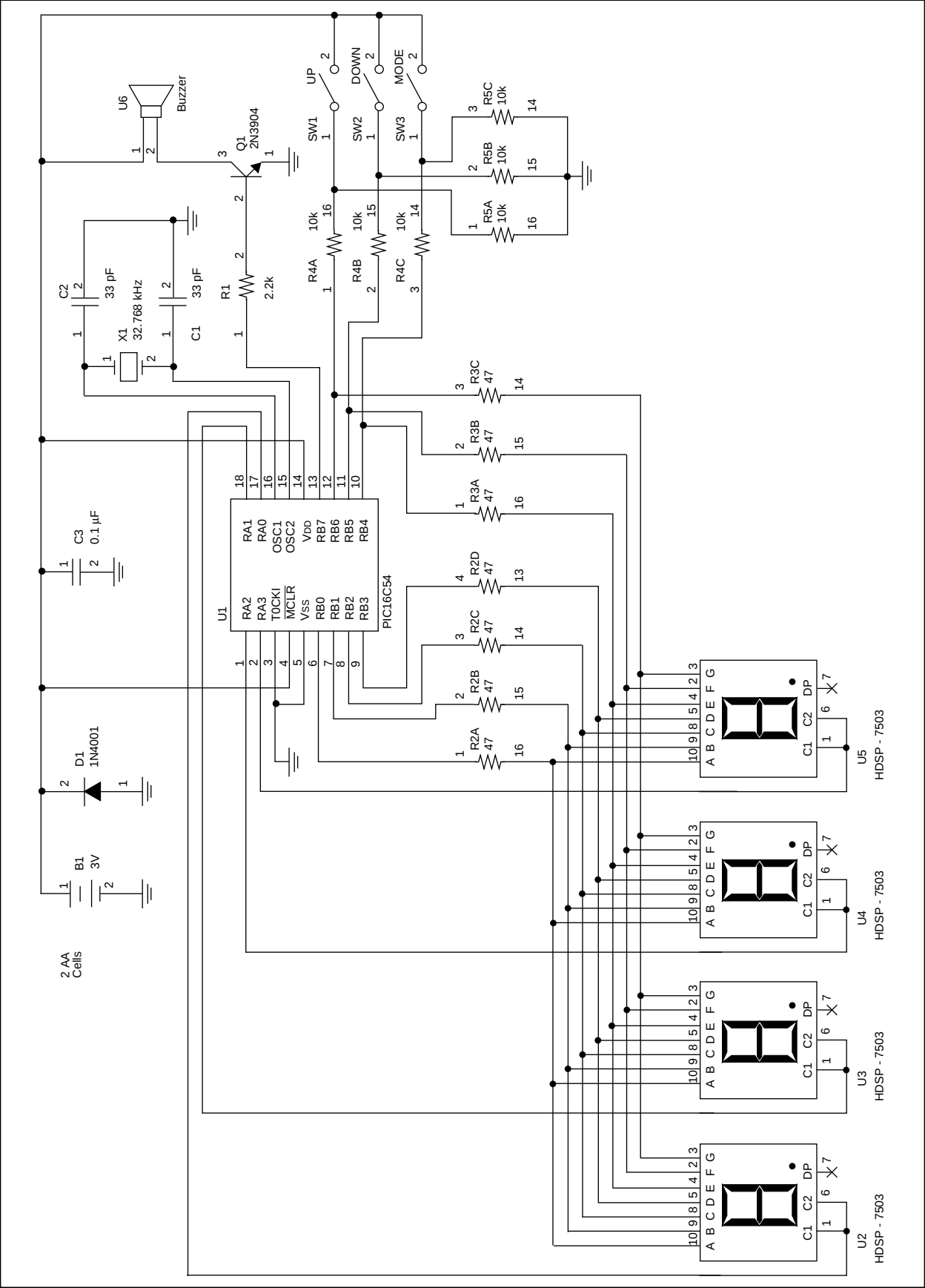
$$\text{Instruction cycle} = 122.07 \mu\text{s}$$

This means that every instruction executes in 122.07 μ s or we execute exactly 8,192 instructions per second.

Display and keypad multiplexing

The display contains four digits with seven segments each; therefore a multiplexing scheme was used to reduce the number of I/O lines needed to drive the displays. There are 4 common cathode display connections (one for each display digit and connected to PORTA for convenience so that rotates and moves can be used) and 7 segments (connected to PORTB for convenience so that moves can be used) for a total of 11 I/O lines needed for the display. Common Cathode displays were chosen, since the PIC16C54A can sink 5 mA more current than it can source. The last I/O line (RB7) was used to drive the buzzer. The three keys for setting the time are multiplexed onto the LED display segments to eliminate the need for additional I/O lines.

FIGURE 1: CLOCK DESIGN SCHEMATIC



SOFTWARE IMPLEMENTATION

The main loop of the software must perform the following tasks to implement the clock's functionality:

1. Determine when one second has passed (when bit7 on the TMR0 register changes state 4 times), increment the current time and (if enabled) decrement the countdown timer.
2. Determine if any of the alarms (countdown timer or the alarm time) are currently alarming or should be alarming. If so, the buzzer is buzzed.
3. Check for any keys that are pressed. If MODE is pressed, the current mode is incremented and if UP or DOWN is pressed, the time displayed is modified.
4. Automatically turn the displays on/off for power management.
5. Multiplex the displays every 3.9 ms (32 instruction cycles).

General purpose registers are defined and used for the following purposes:

- **DISPSEGS_A** through **DISPSEGS_D** store the bit pattern that is to be displayed on each of the four 7-segment displays.
- **CLK_SEC** stores the second counter for the current clock time (values from 0-59 decimal are stored).
- **CLK_MIN_LD**, **CLK_MIN_HD** store the upper and lower minute digit of the current time.
- **CLK_HOUR_LD**, **CLK_HOUR_HD** store the upper and lower hour digit of the current time.
- **ALM_MIN_LD**, **ALM_MIN_HD** store the upper and lower minute digit of the alarm time.
- **ALM_HOUR_LD**, **ALM_HOUR_HD** store the upper and lower hour digit of the alarm time.
- **TMR_SEC_LD**, **TMR_SEC_HD** store the upper and lower second digit of the countdown timer.
- **TMR_MIN_LD**, **TMR_MIN_HD** store the upper and lower minute digit of the countdown timer.
- **KEYPAT** stores a pattern showing the currently pressed keys:
 - UP = bit6
 - DOWN = bit5
 - MODE = bit4
- **FLAGS** stores key flag bits such as the current mode, display on, alarm on, etc.
- **PREVTMR0** stores previous TMR0 values so that the differences can be detected the next time the TMR0 is polled.
- **TEMP** is a temporary register used for various routines.
- **DISPONCNT** stores the remaining number of seconds the displays should be on.

- **MODE_COUNT** stores the number of 1/2 seconds the MODE and UP or DOWN buttons are pressed. Used to switch from setting minutes to hours.
- **ALARMCNT** stores the number of beeps remaining to be driven into the buzzer.

FLAGS Register

Most designs require flag or state bits to indicate current modes or the state of a software routines. In this design, the FLAGS register is defined as follows:

Bits 0,1 - indicates the current operating mode (changed by pressing and releasing the MODE button):

00 - Display OFF

01 - Display/Set countdown timer

10 - Display/Set alarm time

11 - Display/Set clock (current time)

Bit 3 - indicates if (alarm time) = (current clock time)

Bit 4 - indicates if (count down timer) = 0

Bits 5,6,7 -Used as a divide by four counter to keep track of seconds

The software is broken up into the following routines for modularity:

buzz_now routine - *Output buzzing tone during alarm for 156 ms.*

Buzzers are available in self-oscillating and direct drive models. To save cost, a low impedance, direct drive model was selected. The buzz_now routine is called by the main_loop and it chirps the buzzer for 156 ms at a 1638 Hz frequency. This routine first turns off the LEDs (by clearing PORTB) and then uses TEMP to count for 256 pulses. The pulse is sent to the buzzer by the BSF BUZZEROUT and BCF BUZZEROUT instructions. This routine returns once 256 pulses are sent to the buzzer. This is necessary, since the controller cannot buzz the buzzer and keep track of time at the same time (running at such a low frequency), so these two functions are multiplexed.

task_scan routine - *multiplex LEDs to display the next digit, only one digit is lit at a time).*

The PIC16C5X family is designed for polled I/O applications and does not contain a hardware interrupt structure. To achieve the lowest cost design, the PIC16C54A was selected and all modules are written to CALL this task_scan routine within the multiplexing time frame of 3.9 ms or 32 instruction cycles. This routine first synchronizes itself with the TMR0 register, bit0 to ensure that the scanning occurs at the same point in time, regardless of when the routine is called. Next, PREVSCAN is rotated, setting up the CARRY bit correctly. The bit pattern for the next digit to be displayed is then moved into the W register. The display is blanked, PORTA is rotated (to select the next digit) and the next display bit pattern is moved to PORTB to display it. For ESD integrity, PORTA is later restored from the PREVSCAN register. This routine takes a total of 21 cycles (including the CALL and RETLW instructions) to execute and the displays are scanned every 3.9 ms (32 instruction cycles); therefore, this routine needs to be called after every 11 instruction cycles from every routine to maintain proper display multiplexing.

disp_value routine - *Update the display registers with the bitmap of what digits are to be displayed next.*

Indirect addressing is used here to reduce the amount of code needed and to simplify the routine. Since the clock, alarm or countdown time could each be displayed, the W register contains the base address (in the register file) of the four digits that are to currently be displayed. The W register is first moved to the FSR register so that the indirect address register contains the first digit to be displayed. The first digit is first converted into the segment bit pattern by calling the led_lookup table and then the bit pattern is moved to DISPSEGS_A. The FSR register is incremented (moving to the next digit) and the process is repeated for the remaining 3 digits. To maintain proper multiplexing, task_scan is called throughout this routine.

turnon_scan routine - *Turns on the LEDs and restores a legal scan position.*

To save battery power, the displays are automatically shut off after 8 seconds when no buttons are pressed. The DISPON bit is used to preset the remaining display on time to 8 seconds. This routine sets this flag (to later turn on the displays) and then checks to see if the PREVSCAN register contains a legal value (an illegal value of 'FFh' is used to turn off all of the displays) and it restores a legal value if the displays were off.

scan_keys routine - *Turns off LEDs for a moment and scans the push-button inputs.*

To reduce the number of I/Os needed by this application, the three user input keys are multiplexed onto the LED display segments through PORTB. First, the PORTB is cleared and PORTA is set to '0Fh', turning off the LED displays. Next, PORTB is set up with bits 4,5 and 6 as inputs to read the keys. TEMP is then loaded with the keys that have changed state (to detect the fall-

ing edge of a key press) and KEYPAT is loaded with a pattern ('0' = not pressed, '1' = pressed) for the keys that are pressed. Lastly, PORTB is restored to all outputs and the current multiplex scan is restored to PORTA.

check_time routine - *Checks for alarm or countdown timer expiration.*

Each second, alarm conditions must be detected and the buzzer sounded if an alarm condition is true. ALARMNOW and EGGNOW are flag bits that are used by the main program to sound the buzzer if they are set. This routine starts by setting both ALARMNOW and EGGNOW. Next, the current time hours and minutes are compared (through a subtraction and a test of the STATUS register Z bit) with the alarm time. If there is any miscompare, the ALARMNOW bit is cleared. To finish, the countdown timer time minutes and seconds digits are each compared with zero. If there is any miscompare, the EGGNOW bit is cleared. To maintain proper multiplexing, the task_scan routine is regularly called throughout this routine.

inc_time routine - *Adds one second, minute or hour to the clock, alarm or timer.*

Every second, inc_time is called by main_loop to increment the seconds count for the clock. This routine is also called when the "UP" key is pressed and "MODE" key is held down to adjust the current time, alarm time or set the countdown timer. This routine uses indirect addressing to reduce the amount of code and simplify it's operation. Before this routine is called, the W register is loaded with the address of the clock second register and the routine is called. The FSR register is loaded with this value and the indirect address register is incremented (effectively incrementing the seconds counter).

Once the second counter is incremented, this register is checked for overflow (greater than 59 seconds) and if no overflow occurred, the routine returns. If an overflow happened, the second counter is cleared and the minute low digit is incremented. This register is then checked for an overflow (greater than 9 minutes) and so on until the all digits are updated.

This routine can also be called from multiple points. If called with the label inc_min_Id, only the minutes (and hours if an overflow occurs) will be incremented. Additionally, calling inc_hour_Id will increment only the hour digits. These features are used when setting the clock or alarm function. The FLAGS register (bits 0 and 1) is used to determine the current mode (clock, alarm or countdown timer) and ensure proper overflow calculations. To maintain proper multiplexing, the task_scan routine is regularly called throughout this routine.

dec_time routine - *Subtracts one second, minute or hour from the clock, alarm or timer.*

If the countdown timer is enabled, dec_time is called by the main loop to decrement the seconds count for the countdown timer. This routine is also called by the main loop when the "DOWN" key is pressed and "MODE" key is held down to adjust the current time, alarm time or set the countdown timer. This routine uses indirect addressing to reduce the amount of code and simplify it's operation. Before this routine is called, the W register is loaded with the address of the countdown timer's second register and the routine is called. The FSR register is loaded with this value and the indirect address register is incremented (effectively incrementing the seconds counter).

Once the second counter LSD is decremented, this register is checked for underflow (less than 0 seconds) and if no underflow occurred, the routine returns. If an underflow happened, the second counter LSD is set to 9 and the second MSD is decremented. This register is then checked for an underflow (less than 0 seconds) and so on until all digits are updated.

This routine also can be called from multiple points. If called with the label dec_hour_id_vec, only the hour digits (or minutes if it is the countdown timer) will be decremented. This feature is used when setting the clock or alarm function. The FLAGS register (bits 0 and 1) is used to determine the current mode (clock, alarm or countdown timer) and ensure proper underflow calculations. To maintain proper multiplexing, the task_scan routine is regularly called throughout this routine.

main_loop routine - *Calls the above routines as needed and keeps track of when to increment the clock or decrement the countdown timer.*

The main_loop calls all of the previous routines as necessary to maintain time, LED multiplexing, alarming and setting each function. The OPTION register is loaded with a 03h value to set up a Divide by 16 prescaler for the TMR0 register and internal instruction cycle increment. The instruction cycle is 122.07 μ s; therefore, bit0 changes every $(122 \mu\text{s} \cdot 16) = 1.953 \text{ ms}$ and bit7 changes every $(122.07 \mu\text{s} \cdot 16 \cdot 128) = 250 \text{ ms}$. Bits 5 and 6 of the FLAGS register are used to divide this 250 ms event by 4 to call inc_time every second.

The check_time routine is called after calling inc_time (every second), setting the EGGNOW or ALARMNOW flag bits. If the alarm is enabled, the buzzer is buzzed by calling buzz_now; however, the main timer updates need to occur in between buzzer beeps to keep track of time.

Every 500 ms, the keys are scanned and the edges on the MODE key are detected. Pressing the UP or DOWN key will shut off the buzzer (clearing the enable bits) and pressing the MODE key will advance the current mode. The mode is a 4-state state machine, revolving between the following states:

1. Display OFF - saves battery power - defaults to this mode if no keys are pressed for 8 seconds.
2. Display or Set countdown timer (holding MODE key allows setting).
3. Display or Set Alarm time (holding MODE key allows setting).
4. Display or Set Clock time (holding MODE key allows setting).

Next, the UP and DOWN keyscan values are tested and if the MODE and UP are both pressed, the currently displayed mode time is incremented or decremented. If MODE is not pressed and UP or DOWN is pressed, the displays are turned on, but the displayed time is not altered.

DISPNCNT is used to keep track of how long the displays have been on once all buttons are released. After 8 seconds, the displays are automatically turned off to save power. MODE_COUNT is used to switch from setting the right hand displays (minutes or seconds) to the left hand displays (hours or minutes). When the UP or DOWN button is held with mode for more than 4 seconds consecutively, MODE_COUNT reaches zero, switching from the right to left hand displays.

Finally, the main_loop finishes by updating the display registers by calling disp_value and if DISPNCNT has decreased to zero, the displays are turned off.

Lookup Tables - *Convert a number into a bit pattern or RAM address.*

There are three lookup tables used in this design for BCD to 7-Segment decoding, manufacturing diagnostics and looking up the address of the currently displayed mode.

- **mode_timer** - look-up the address of the clock, Alarm or Timer data storage RAM.
- **led_lookup** - look-up table contains the bitmap display pattern for displaying digits 0-9.
- **mfg_led_lookup** - look-up table contains the bitmap display pattern used for manufacturing mode. Only one segment is lit at a time.

Miscellaneous routines used for initialization and manufacturing test:

- **init** - Initializes all of RAM to zero, sets up the I/O ports and sets default time values.
- **mfg_selftest** - Used in manufacturing mode only - tests each LED segment, push-button, buzzer and display separately to expose bad keys, connections, buzzer or displays.

CONCLUSION

The implementation of this application highlights the PIC16C54's highly efficient instruction set, low power and frequency operation, high current direct LED drive capability and high performance instruction execution. Many of the routines used in this application note apply to a variety embedded control applications.

Ram Used: 25 Bytes

Code Space

Used: 444 Words (without manufacturing diagnostics)

510 Words (including manufacturing diagnostics)

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

APPENDIX A: CODE

MPASM 01.21.03 Intermediate

CLK8.ASM

8-21-1995

9:17:56

PAGE 1

LOC OBJECT LINE SOURCE TEXT
VALUE CODE

```

00001 ; *****
00002 ; *      PIC Egg Timer Give-Away      *
00003 ; *                                  *
00004 ; * Author:   John Day                *
00005 ; *          Sr. Field Applications Engineer *
00006 ; *          Northeast Region          *
00007 ; *                                  *
00008 ; * Revision: 1.2                    *
00009 ; * Date      September 22, 1994    *
00010 ; * Part:     PIC16C54-LP/P or PIC16LC54A/P *
00011 ; * Fuses:    OSC:  LP                *
00012 ; *          WDT:  OFF                *
00013 ; *          Port: OFF                *
00014 ; *          CP:   OFF                *
00015 ; *****
00016 ;
00017 ; This program is intended to run on a 32 Khz watch crystal and
00018 ; connects to four multiplexed seven segment displays. It displays the
00019 ; current time, alarm time and egg count down timers. There are
00020 ; switches that allow the user to set the alarm, timer and clock functions.
00021 ;
00022 LIST F=INHX8M,P=16C54
00023 INCLUDE "p16C5X.inc"
00001      LIST
00002 ; P16C5X.INC Standard Header File, Version 2.02 Microchip Technology, Inc.
00143      LIST
00FF  00F8 00024      __FUSES __CP_OFF&__WDT_OFF&__LP_OSC
00025
0007 00026      ORG 07h
00027 ; *****
00028 ; * Static RAM Register File Definitions *
00029 ; *****
00000000 00030 INDADDR      EQU    0      ; Indirect address register
00000007 00031 DISPSEGS_A      EQU    07h     ; Current Display A segment bit pattern
00000008 00032 DISPSEGS_B      EQU    08h     ; Current Display B segment bit pattern
00000009 00033 DISPSEGS_C      EQU    09h     ; Current Display C segment bit pattern
0000000A 00034 DISPSEGS_D      EQU    0Ah     ; Current Display D segment bit pattern
0000000B 00035 CLK_SEC      EQU    0Bh     ; Clock second counter (0-59)
0000000C 00036 CLK_MIN_LD      EQU    0Ch     ; Clock minute low digit counter (0-9)
0000000D 00037 CLK_MIN_HD      EQU    0Dh     ; Clock minute high digit counter (0-5)
0000000E 00038 CLK_HOUR_LD      EQU    0Eh     ; Clock hour low digit counter (0-9)
0000000F 00039 CLK_HOUR_HD      EQU    0Fh     ; Clock hour high digit counter (0-2)
00000010 00040 ALM_MIN_LD      EQU    10h     ; Alarm minute low digit counter (0-9)
00000011 00041 ALM_MIN_HD      EQU    11h     ; Alarm minute high digit counter (0-5)
00000012 00042 ALM_HOUR_LD      EQU    12h     ; Alarm hour low digit counter (0-9)
00000013 00043 ALM_HOUR_HD      EQU    13h     ; Alarm hour high digit counter (0-2)
00000014 00044 TMR_SEC_LD      EQU    14h     ; Timer second low digit counter (0-9)
00000015 00045 TMR_SEC_HD      EQU    15h     ; Timer second high digit counter (0-5)
00000016 00046 TMR_MIN_LD      EQU    16h     ; Timer hour low digit counter (0-9)
00000017 00047 TMR_MIN_HD      EQU    17h     ; Timer hour high digit counter (0-2)
00000018 00048 KEYPAT      EQU    18h     ; Currently pressed key bits
00000019 00049 FLAGS      EQU    19h     ; Status of alarms, display on, etc.
0000001A 00050 PREVTMR0      EQU    1Ah     ; Used to determine which TMR0 bits changed
0000001B 00051 PREVSCAN      EQU    1Bh     ; Store Common Cathode display scan state
0000001C 00052 TEMP      EQU    1Ch     ; Temporary storage

```

AN615

```
0000001D    00053 DISPONCNT    EQU    1Dh    ; Time the displays have been on
0000001E    00054 MODE_COUNT    EQU    1Eh    ; Current mode state
0000001F    00055 ALARMCNT    EQU    1Fh    ; Time the alarm has been sounding
00056 ; *****
00057 ; * Flag and state bit definitions *
00058 ; *****
00059 #define    SECBIT    TEMP,7    ; Bit to spawn 1/4 second count
00060 #define    SCANBIT    TMR0,0    ; Bit to spawn display MUX
00061 #define    MODEKEY    KEYPAT,4    ; Bit for MODEKEY pressed
00062 #define    UPKEY    KEYPAT,6    ; Bit for UPKEY pressed
00063 #define    DOWNKEY    KEYPAT,5    ; Bit for DOWNKEY pressed
00064 #define    MODEKEYCHG    TEMP,4    ; Bit for delta MODEKEY
00065 #define    TIMENOW    FLAGS,7    ; Flag to indicate 1 second passed
00066 #define    ALARMNOW    FLAGS,3    ; Flag to indicate wakeup alarm
00067 #define    EGGNOW    FLAGS,4    ; Flag to indicate egg timer alarm
00068 #define    ALARMOK    STATUS,PA0    ; Flag to enable wakeup alarm
00069 #define    EGGOK    STATUS,PA1    ; Flag to enable timer alarm
00070 #define    BUZZEROUT    PORTB,7    ; Pin for pulsing the buzzer
00071 #define    DISPON    DISPONCNT,4    ; Bit to turn on LED displays
00072
00073 ; *****
00074 ; * Various Constants used throughout the program *
00075 ; *****
0000003C    00076 SEC_MAX    EQU    .60    ; Maximum value for second counter
0000000A    00077 MIN_LD_MAX    EQU    .10    ; Maximum value for low digit of minute
00000006    00078 MIN_HD_MAX    EQU    .6    ; Maximum value for high digit of minute
00000004    00079 HOUR_LD_MAX    EQU    .4    ; Maximum value for low digit of hour
00000002    00080 HOUR_HD_MAX    EQU    .2    ; Maximum value for high digit of hour
00000003    00081 OPTION_SETUP    EQU    b'00000011'    ; TMR0 - internal, /16 prescale
00000007    00082 BUZINITVAL    EQU    7    ;
00000008    00083 INIT_MODE_COUNT    EQU    8    ; Digit counts to move to hour digits
00000028    00084 ALARMCYCCNT    EQU    .40    ; Alarm for 10 seconds (ALARMCYCCNT/4)
00085
01FF    00086    ORG    01FFh    ; The PIC5X reset vector is at end of memory
01FF    00087 reset_vector
01FF 0BA8    00088    GOTO    init    ; Jump to the initialization code
00089
0000    00090    ORG    0
00091 ; *****
00092 ; * Current mode look-up table *
00093 ; *****
0000    00094 mode_timer
0000 0E03    00095    ANDLW    3    ; Mask off upper bits just in case
0001 01E2    00096    ADDWF    PCL,F    ; Jump to one of 4 look-up entries
0002 0814    00097    RETLW    TMR_SEC_LD    ; Return the address of the 99 min timer RAM
0003 0810    00098    RETLW    ALM_MIN_LD    ; Return the address of the alarm RAM
0004 080C    00099    RETLW    CLK_MIN_LD    ; Return the address of the clock RAM
0005 080C    00100    RETLW    CLK_MIN_LD    ; Return the address of the clock RAM
00101
00102 ; *****
00103 ; * Buzz the buzzer for 1/8 second *
00104 ; *****
0006    00105 buzz_now
0006 0066    00106    CLRF    PORTB    ; Shut off the segments
0007    00107 buzz_now_dispon
0007 007C    00108    CLRF    TEMP    ; Buzz for 256 pulses
0008    00109 loop_buz
0008 05E6    00110    BSF    BUZZEROUT    ; Send out pulse
0009 04E6    00111    BCF    BUZZEROUT    ; Clear out the pulse
000A 02FC    00112    DECFSZ    TEMP,F    ; Decrement counter and skip when done
000B 0A08    00113    GOTO    loop_buz    ; Go back and send another pulse
000C 0800    00114    RETLW    0    ; We are done so come back!
00115
00116 ; *****
00117 ; * Mux drive the next LED display digit *
00118 ; *****
```

```

000D      00119 task_scan    ; (19 (next_scan) + 2 = 21 cycles - must be called every 11 cy)
000D 0601  00120          BTFSC  SCANBIT    ; Synch up with 3.9 mS timer bit
000E 0A0D  00121          GOTO   task_scan   ; Jump back until bit is clear
00122
000F      00123 next_scan    ; (15 + 2 call + 2 return = 19 cycles)
000F 035B  00124          RLF     PREVSCAN,W  ; Move to the next digit select into C
0010 073B  00125          BTFSS  PREVSCAN,1   ; 0 Check if display A was on before
0011 0209  00126          MOVF   DISPSEGS_C,W  ; Place display B value into W
0012 071B  00127          BTFSS  PREVSCAN,0   ; 1 Check if display B was on before
0013 0208  00128          MOVF   DISPSEGS_B,W  ; Place display C value into W
0014 077B  00129          BTFSS  PREVSCAN,3   ; 2 Check if display C was on before
0015 0207  00130          MOVF   DISPSEGS_A,W  ; Place display D value into W
0016 075B  00131          BTFSS  PREVSCAN,2   ; 3 Check if display D was on before
0017 020A  00132          MOVF   DISPSEGS_D,W  ; Place display A value into W
0018 0066  00133          CLRF    PORTB       ; Turn off all segments
0019 037B  00134          RLF     PREVSCAN,F   ; Move to the next digit
001A 0365  00135          RLF     PORTA,F     ; Move port to the next digit
001B 0026  00136          MOVWF   PORTB       ; Place next segment value on PORTB
001C 021B  00137          MOVF   PREVSCAN,W   ; Restore the port in case it is wrong
001D 0025  00138          MOVWF   PORTA       ; Restore the port
001E 0800  00139          RETLW   0           ; Display is updated - now return
00140
00141
00142 ; *****
00143 ; * Move new digit display info out to display *
00144 ; *****
001F      00145 disp_value
001F 0024  00146          MOVWF   FSR         ; Place W into FSR for indirect addressing
0020 090D  00147          CALL    task_scan   ; Scan the next LED digit.
0021 0200  00148          MOVF   INDADDR,W   ; Place display value into W
0022 0937  00149          CALL    led_lookup  ; Look up seven segment value
0023 0027  00150          MOVWF   DISPSEGS_A   ; Move value out to display register A
0024 02A4  00151          INCF    FSR,F      ; Go to next display value
0025 090D  00152          CALL    task_scan   ; Scan the next LED digit.
0026 0200  00153          MOVF   INDADDR,W   ; Place display value into W
0027 0937  00154          CALL    led_lookup  ; Look up seven segment value
0028 0028  00155          MOVWF   DISPSEGS_B   ; Move value out to display register B
0029 02A4  00156          INCF    FSR,F      ; Go to next display value
002A 090D  00157          CALL    task_scan   ; Scan the next LED digit.
002B 0200  00158          MOVF   INDADDR,W   ; Place display value into W
002C 0937  00159          CALL    led_lookup  ; Look up seven segment value
002D 0029  00160          MOVWF   DISPSEGS_C   ; Move value out to display register C
002E 02A4  00161          INCF    FSR,F      ; Go to next display value
002F 090D  00162          CALL    task_scan   ; Scan the next LED digit.
0030 0200  00163          MOVF   INDADDR,W   ; Place display value into W
0031 0643  00164          BTFSC  STATUS,Z    ; ZBLANK - Check for a zero
0032 0240  00165          COMF    INDADDR,W   ; ZBLANK - Clear digit with FF if leading 0
0033 0937  00166          CALL    led_lookup  ; Look up seven segment value
0034 002A  00167          MOVWF   DISPSEGS_D   ; Move value out to display register D
0035 090D  00168          CALL    task_scan   ; Scan the next LED digit.
0036 0800  00169          RETLW   0
00170
00171 ; *****
00172 ; * Convert display value into segments *
00173 ; *****
0037      00174 led_lookup
0037 0E0F  00175          ANDLW   0Fh         ; Strip off upper digits
0038 01E2  00176          ADDWF   PCL,F      ; Jump into the correct location
0039 083F  00177          RETLW   b'00111111' ; Bit pattern for a Zero
003A 0806  00178          RETLW   b'00000110' ; Bit pattern for a One
003B 085B  00179          RETLW   b'01011011' ; Bit pattern for a Two
003C 084F  00180          RETLW   b'01001111' ; Bit pattern for a Three
003D 0866  00181          RETLW   b'01100110' ; Bit pattern for a Four
003E 086D  00182          RETLW   b'01101101' ; Bit pattern for a Five
003F 087D  00183          RETLW   b'01111101' ; Bit pattern for a Six
0040 0807  00184          RETLW   b'00000111' ; Bit pattern for a Seven

```

```

0041 087F    00185    RETLW    b'01111111'    ; Bit pattern for a Eight
0042 086F    00186    RETLW    b'01101111'    ; Bit pattern for a Nine
0043 0800    00187    RETLW    0                ; Turn display off - ILLEGAL VALUE
0044 0800    00188    RETLW    0                ; Turn display off - ILLEGAL VALUE
0045 0800    00189    RETLW    0                ; Turn display off - ILLEGAL VALUE
0046 0800    00190    RETLW    0                ; Turn display off - ILLEGAL VALUE
0047 0800    00191    RETLW    0                ; Turn display off - ILLEGAL VALUE
0048 0800    00192    RETLW    0                ; Turn display off - ILLEGAL VALUE
00193
00194 ; *****
00195 ; * Convert display value into single segment ON for manufacturing diags *
00196 ; *****
0049 00197 mfg_led_lookup
0049 0E07    00198    ANDLW    07h                ; Strip off upper digits
004A 01E2    00199    ADDWF    PCL,F                ; Jump into the correct location
004B 0801    00200    RETLW    b'00000001'    ; Bit pattern for segment A on only
004C 0802    00201    RETLW    b'00000010'    ; Bit pattern for segment B on only
004D 0804    00202    RETLW    b'00000100'    ; Bit pattern for segment C on only
004E 0808    00203    RETLW    b'00001000'    ; Bit pattern for segment D on only
004F 0810    00204    RETLW    b'00010000'    ; Bit pattern for segment E on only
0050 0820    00205    RETLW    b'00100000'    ; Bit pattern for segment F on only
0051 0840    00206    RETLW    b'01000000'    ; Bit pattern for segment G on only
0052 087F    00207    RETLW    b'01111111'    ; Bit pattern for all segments on
00208
00209 ; *****
00210 ; * Wake-up and turn on the displays *
00211 ; *****
0053 00212 turnon_scan
0053 059D    00213    BSF        DISPON                ; Set display ON bit
0054 0CEE    00214    MOVLW    b'11101110'    ; Place digit 0 scan pattern in W
0055 019B    00215    XORWF    PREVSCAN,W        ; See if this is the current scan
0056 0643    00216    BTFSC    STATUS,Z        ; Skip if this is not the current scan
0057 0800    00217    RETLW    0                ; Legal scan value - we are done!
0058 0CDD    00218    MOVLW    b'11011101'    ; Place digit 1 scan pattern in W
0059 019B    00219    XORWF    PREVSCAN,W        ; See if this is the current scan
005A 0643    00220    BTFSC    STATUS,Z        ; Skip if this is not the current scan
005B 0800    00221    RETLW    0                ; Legal scan value - we are done!
005C 0CBB    00222    MOVLW    b'10111011'    ; Place digit 2 scan pattern in W
005D 019B    00223    XORWF    PREVSCAN,W        ; See if this is the current scan
005E 0643    00224    BTFSC    STATUS,Z        ; Skip if this is not the current scan
005F 0800    00225    RETLW    0                ; Legal scan value - we are done!
0060 0C77    00226    MOVLW    b'01110111'    ; Place digit 3 scan pattern in W
0061 019B    00227    XORWF    PREVSCAN,W        ; See if this is the current scan
0062 0643    00228    BTFSC    STATUS,Z        ; Skip if this is not the current scan
0063 0800    00229    RETLW    0                ; Legal scan value - we are done!
0064 0CEE    00230    MOVLW    0EEh                ; Move digit 0 scan value into W
0065 003B    00231    MOVWF    PREVSCAN                ; Move it into scan pattern register
00232
00233 ; *****
00234 ; * Scan for pressed keys *
00235 ; *****
0066 00236 scan_keys
0066 0066    00237    CLRF        PORTB                ; Turn off all of the segments
0067 0CFF    00238    MOVLW    0FFh                ; Place FF into W
0068 0025    00239    MOVWF    PORTA                ; Make PORT A all ones
0069 0C70    00240    MOVLW    b'01110000'    ; Place 70 into W
006A 0006    00241    TRIS        PORTB                ; Make RB4,5,6 inputs others outputs
006B 0206    00242    MOVF        PORTB,W        ; Place keyscan value into W
006C 0198    00243    XORWF    KEYPAT,W        ; Place Delta key press into W
006D 003C    00244    MOVWF    TEMP                ; Place Delta key press into TEMP
006E 01B8    00245    XORWF    KEYPAT,F        ; Update KEYPAT reg to buttons pressed
006F 0040    00246    CLRW                ; Place 0 into W
0070 0006    00247    TRIS        PORTB                ; Make PORT B outputs
0071 021B    00248    MOVF        PREVSCAN,W        ; Place previous scan value into W
0072 0025    00249    MOVWF    PORTA                ; Turn on the scan
0073 0800    00250    RETLW    0

```

```

00251 ; *****
00252 ; * Check if alarm or timer is expired *
00253 ; *****
0074 00254 check_time
0074 090D 00255 CALL task_scan ; Scan the next LED digit.
0075 0579 00256 BSF ALARMNOW ; Set the alarm bit
0076 0599 00257 BSF EGGNOW ; Set the Egg timer alarm bit
0077 0210 00258 MOVF ALM_MIN_LD,W ; Place alarm minute counter into W
0078 008C 00259 SUBWF CLK_MIN_LD,W ; CLK_MIN_LD - W -> W
0079 0743 00260 BTFSS STATUS,Z ; Skip if they are equal
007A 0479 00261 BCF ALARMNOW ; They are not equal so clear alarm bit
007B 0211 00262 MOVF ALM_MIN_HD,W ; Place alarm minute counter into W
007C 008D 00263 SUBWF CLK_MIN_HD,W ; CLK_MIN_HD - W -> W
007D 0743 00264 BTFSS STATUS,Z ; Skip if they are equal
007E 0479 00265 BCF ALARMNOW ; They are not equal so clear alarm bit
007F 090D 00266 CALL task_scan ; Scan the next LED digit.
0080 0212 00267 MOVF ALM_HOUR_LD,W ; Place alarm hour counter into W
0081 008E 00268 SUBWF CLK_HOUR_LD,W ; CLK_HOUR_LD - W -> W
0082 0743 00269 BTFSS STATUS,Z ; Skip if they are equal
0083 0479 00270 BCF ALARMNOW ; They are not equal so clear alarm bit
0084 0213 00271 MOVF ALM_HOUR_HD,W ; Place alarm hour counter into W
0085 008F 00272 SUBWF CLK_HOUR_HD,W ; CLK_HOUR_LD - W -> W
0086 0743 00273 BTFSS STATUS,Z ; Skip if they are equal
0087 0479 00274 BCF ALARMNOW ; They are not equal so clear alarm bit
0088 090D 00275 CALL task_scan ; Scan the next LED digit.
0089 0214 00276 MOVF TMR_SEC_LD,W ; Set the Z bit to check for zero
008A 0743 00277 BTFSS STATUS,Z ; Skip if this digit is zero
008B 0499 00278 BCF EGGNOW ; Timer is not zero so clear egg alarm bit
008C 0215 00279 MOVF TMR_SEC_HD,W ; Set the Z bit to check for zero
008D 0743 00280 BTFSS STATUS,Z ; Skip if this digit is zero
008E 0499 00281 BCF EGGNOW ; Timer is not zero so clear egg alarm bit
008F 0216 00282 MOVF TMR_MIN_LD,W ; Set the Z bit to check for zero
0090 0743 00283 BTFSS STATUS,Z ; Skip if this digit is zero
0091 0499 00284 BCF EGGNOW ; Timer is not zero so clear egg alarm bit
0092 090D 00285 CALL task_scan ; Scan the next LED digit.
0093 0217 00286 MOVF TMR_MIN_HD,W ; Set the Z bit to check for zero
0094 0743 00287 BTFSS STATUS,Z ; Skip if this digit is zero
0095 0499 00288 BCF EGGNOW ; Timer is not zero so clear egg alarm bit
0096 0799 00289 BTFSS EGGNOW ; Skip if we are still at EGG Time
0097 05C3 00290 BSF EGGOK ; If we are not at EGG time, re-set egg alarm
0098 0779 00291 BTFSS ALARMNOW ; Skip if we are still at Alarm time
0099 05A3 00292 BSF ALARMOK ; If we are not at Alarm time, re-set alarm
009A 090D 00293 CALL task_scan ; Scan the next LED digit.
009B 0800 00294 RETLW 0
00295
00296 ; *****
00297 ; * Increment the clock, timer or alarm *
00298 ; *****
009C 00299 inc_time
009C 0024 00300 MOVWF FSR ; Add one to clock second counter
009D 090D 00301 CALL task_scan ; Scan the next LED digit.
009E 02A0 00302 INCF INDADDR,f ; Add one to minute lower digit
009F 0C3C 00303 MOVLW SEC_MAX ; Place second max value into w
00A0 0080 00304 SUBWF INDADDR,W ; CLOCK_SEC - SEC_MAX -> W
00A1 0703 00305 BTFSS STATUS,C ; Skip if there is an overflow
00A2 0800 00306 RETLW 0 ; We are done so let's get out of here!
00A3 006B 00307 CLRF CLK_SEC ; Clear CLK_second counter
00A4 02A4 00308 INCF FSR,F ; Move to the next digit
00A5 02A0 00309 INCF INDADDR,F ; Add 1 to minute LOW digit
00A6 0AA9 00310 GOTO skip_min_fsr ; Jump to the next digit
00A7 00311 inc_min_ld
00A7 0024 00312 MOVWF FSR
00A8 02A0 00313 INCF INDADDR,F ; Add 1 to minute LOW digit
00A9 00314 skip_min_fsr
00A9 090D 00315 CALL task_scan ; Scan the next LED digit.
00AA 0C0A 00316 MOVLW MIN_LD_MAX ; Place minute lower digit max value into W

```

AN615

```
00AB 0080      00317      SUBWF      INDADDR,W      ; CLK_MIN_LD - MIN_LD_MAX -> W
00AC 0703      00318      BTFSS      STATUS,C      ; Skip if there is an overflow
00AD 0800      00319      RETLW      0      ; We are done so let's get out of here!
00AE 0060      00320      CLRF      INDADDR      ; Clear CLK minute low digit
00AF 02A4      00321      INCF      FSR,F      ; Move to the minute high digit
00B0 02A0      00322      INCF      INDADDR,F      ; Add one to minute high digit
00B1      00323      inc_min_hd
00B1 090D      00324      CALL      task_scan      ; Scan the next LED digit.
00B2 0C06      00325      MOVLW      MIN_HD_MAX      ; Place minute high digit max value into W
00B3 0080      00326      SUBWF      INDADDR,W      ; CLK_MIN_HD - MIN_HD_MAX -> W
00B4 0703      00327      BTFSS      STATUS,C      ; Skip if there is an overflow
00B5 0800      00328      RETLW      0      ; We are done so let's get out of here!
00B6 0060      00329      CLRF      INDADDR      ; Clear CLK minute high digit
00B7 02A4      00330      INCF      FSR,F      ; Move to the hour low digit
00B8 02A0      00331      INCF      INDADDR,F      ; Add one to hour low digit
00B9 0ABE      00332      GOTO      skip_hour_fsr      ; Jump to the next digit
00BA      00333      inc_hour_ld
00BA 0024      00334      MOVWF      FSR
00BB 02A4      00335      INCF      FSR,F
00BC 02A4      00336      INCF      FSR,F
00BD 02A0      00337      INCF      INDADDR,F      ; Add 1 to minute LOW digit
00BE      00338      skip_hour_fsr
00BE 090D      00339      CALL      task_scan      ; Scan the next LED digit.
00BF 0C0A      00340      MOVLW      MIN_LD_MAX      ; Place hour lower digit max value into W
00C0 0080      00341      SUBWF      INDADDR,W      ; CLK_HOUR_LD - HOUR_LD_MAX -> W
00C1 0703      00342      BTFSS      STATUS,C      ; Skip if there is an overflow
00C2 0AC7      00343      GOTO      check_inc      ; We need to check for overflow
00C3 0060      00344      CLRF      INDADDR      ; Clear CLK hour low digit
00C4 02A4      00345      INCF      FSR,F      ; Move to the hour high digit
00C5 02A0      00346      INCF      INDADDR,F      ; Add one to hour high digit
00C6 0AC8      00347      GOTO      inc_hour_hd
00C7      00348      check_inc
00C7 02A4      00349      INCF      FSR,F      ; Move to hour high digit
00C8      00350      inc_hour_hd
00C8 090D      00351      CALL      task_scan      ; Scan the next LED digit.
00C9 0C02      00352      MOVLW      HOUR_HD_MAX      ; Place hour high digit max value into W
00CA 0639      00353      BTFSC      FLAGS,1
00CB 0ACE      00354      GOTO      off_mode1
00CC 0619      00355      BTFSC      FLAGS,0
00CD 0C09      00356      MOVLW      MIN_LD_MAX-1
00CE      00357      off_mode1
00CE 0080      00358      SUBWF      INDADDR,W      ; CLK_HOUR_HD - HOUR_HD_MAX -> W
00CF 0703      00359      BTFSS      STATUS,C      ; Skip if there is an overflow
00D0 0800      00360      RETLW      0      ; We are done so let's get out of here!
00D1 00E4      00361      DECF      FSR,F      ; Move to the hour low digit
00D2 090D      00362      CALL      task_scan      ; Scan the next LED digit.
00D3 0C04      00363      MOVLW      HOUR_LD_MAX      ; Place hour high digit max value into W
00D4 0639      00364      BTFSC      FLAGS,1
00D5 0AD8      00365      GOTO      off_mode2
00D6 0619      00366      BTFSC      FLAGS,0
00D7 0C00      00367      MOVLW      0      ; Clear W
00D8      00368      off_mode2
00D8 0080      00369      SUBWF      INDADDR,W      ; CLK_HOUR_HD - HOUR_HD_MAX -> W
00D9 0703      00370      BTFSS      STATUS,C      ; Skip if there is an overflow
00DA 0800      00371      RETLW      0      ; We are done so let's get out of here!
00DB 090D      00372      CALL      task_scan      ; Scan the next LED digit.
00DC 0060      00373      CLRF      INDADDR      ; Clear hour high digit
00DD 0639      00374      BTFSC      FLAGS,1
00DE 0AE0      00375      GOTO      off_mode3
00DF 0719      00376      BTFSS      FLAGS,0
00E0      00377      off_mode3
00E0 0000      00378      NOP
00E1 02A4      00379      INCF      FSR,F      ; Move to the hour high digit
00E2 0060      00380      CLRF      INDADDR      ; Clear one hour low digit
00E3 090D      00381      CALL      task_scan
00E4 0800      00382      RETLW      0      ; We are done so let's get out of here!
```

```

00383
00E5      00384 dec_hour_ld
00E5 0AF9      00385      GOTO    dec_hour_ld_vect ; ran out of CALL space....
00386
00387 ; *****
00388 ; * Decrement the clock, alarm or timer *
00389 ; *****
00E6      00390 dec_time
00E6      00391 dec_min_ld
00E6 0024      00392      MOVWF   FSR                ; Set up pointer for indirect address
00E7 090D      00393      CALL    task_scan          ; Scan the next LED digit.
00E8 00E0      00394      DECF    INDADDR,F          ; Subtract one from CLK_MIN_LD
00E9 0240      00395      COMF    INDADDR,W          ; Set the Z bit to check for zero
00EA 0743      00396      BTFSS   STATUS,Z          ; Skip if CLK_MIN_LD is zero
00EB 0800      00397      RETLW   0                  ; We are done... Let's get out of here
00EC 0C09      00398      MOVLW   MIN_LD_MAX - 1      ; Place minute lower digit max value into W
00ED 0020      00399      MOVWF   INDADDR          ; MIN_LD_MAX -> CLK_MIN_LD
00EE      00400 dec_min_hd
00EE 090D      00401      CALL    task_scan          ; Scan the next LED digit.
00EF 02A4      00402      INCF    FSR,F              ; Move the pointer to Min HIGH DIGIT
00F0 00E0      00403      DECF    INDADDR,F          ; Subtract one from CLK_MIN_HD
00F1 0240      00404      COMF    INDADDR,W          ; Set the Z bit to check for zero
00F2 0743      00405      BTFSS   STATUS,Z          ; Skip if CLK_MIN_LD is zero
00F3 0800      00406      RETLW   0                  ; We are done... Let's get out of here
00F4 0C05      00407      MOVLW   MIN_HD_MAX - 1      ; Place minute lower digit max value into W
00F5 0020      00408      MOVWF   INDADDR          ; MIN_HD_MAX -> CLK_MIN_HD
00F6 090D      00409      CALL    task_scan          ; Scan the next LED digit.
00F7 02A4      00410      INCF    FSR,F              ; Move the pointer to Hour LOW DIGIT
00F8 0AFD      00411      GOTO    skip_dhour_fsr      ; Jump to the next digit
00F9      00412 dec_hour_ld_vect
00F9 0024      00413      MOVWF   FSR
00FA 02A4      00414      INCF    FSR,F
00FB 02A4      00415      INCF    FSR,F
00FC 090D      00416      CALL    task_scan          ; Scan the next LED digit.
00FD      00417 skip_dhour_fsr
00FD 00E0      00418      DECF    INDADDR,F          ; Subtract one from CLK_HOUR_LD
00FE 0240      00419      COMF    INDADDR,W          ; Set the Z bit to check for zero
00FF 0743      00420      BTFSS   STATUS,Z          ; Skip if CLK_MIN_LD is zero
0100 0B06      00421      GOTO    check_hour
0101 0C09      00422      MOVLW   MIN_LD_MAX - 1      ; Place minute lower digit max value into W
0102 0020      00423      MOVWF   INDADDR          ; MIN_LD_MAX -> CLK_HOUR_LD
0103 02A4      00424      INCF    FSR,F
0104 00E0      00425      DECF    INDADDR,F          ; Subtract one from CLK_HOUR_HD
0105 0B07      00426      GOTO    dec_hour_hd
0106      00427 check_hour
0106 02A4      00428      INCF    FSR,F              ; Point to hour high digit
0107      00429 dec_hour_hd
0107 090D      00430      CALL    task_scan          ; Scan the next LED digit.
0108 0240      00431      COMF    INDADDR,W
0109 0743      00432      BTFSS   STATUS,Z
010A 0800      00433      RETLW   0
010B 090D      00434      CALL    task_scan          ; Scan the next LED digit.
010C 00E4      00435      DECF    FSR,F
010D 0C09      00436      MOVLW   .9                  ; Reset digit to 9
010E 0080      00437      SUBWF   INDADDR,W
010F 0743      00438      BTFSS   STATUS,Z          ; Skip if CLK_MIN_LD is zero
0110 0800      00439      RETLW   0                  ; We are done... Let's get out of here
0111 090D      00440      CALL    task_scan          ; Scan the next LED digit.
0112 02A4      00441      INCF    FSR,F
0113 0C02      00442      MOVLW   HOUR_HD_MAX          ; Place minute lower digit max value into W
0114 0739      00443      BTFSS   FLAGS,1          ; Skip if CLOCK or ALARM mode
0115 0C09      00444      MOVLW   .9                  ; Reset digit to 9
0116 0020      00445      MOVWF   INDADDR          ; HOUR_HD_MAX -> CLK_HOUR_HD
0117 0C03      00446      MOVLW   HOUR_LD_MAX - 1      ; Place minute lower digit max value into W
0118 0739      00447      BTFSS   FLAGS,1          ; Skip if CLOCK or ALARM mode
0119 0C09      00448      MOVLW   .9                  ; Reset digit to 9

```

AN615

```
011A 00E4      00449      DECF    FSR,F          ; Move the pointer to Min LOW DIGIT
011B 0020      00450      MOVWF   INDADDR       ; HOUR_LD_MAX -> CLK_HOUR_LD
011C 090D      00451      CALL    task_scan     ; Scan the next LED digit.
011D 0800      00452      RETLW   0              ; We are done... Let's get out of here
               00453
00454 ; *****
00455 ; * Main loop calls all tasks as needed *
00456 ; *****
011E           00457 main_loop
011E 090D      00458      CALL    task_scan     ; Scan the next LED digit.
011F 0201      00459      MOVF    TMR0,W         ; Place current TMR0 value into W
0120 019A      00460      XORWF   PREVTMR0,W      ; Lets see which bits have changed...
0121 003C      00461      MOVWF   TEMP          ; All changed bits are placed in temp for test
0122 01BA      00462      XORWF   PREVTMR0,F      ; Update Previous TMR0 value.
0123 07FC      00463      BTFSS   SECBIT        ; Skip if it is not time to increment second
0124 0B1E      00464      GOTO    main_loop     ; Go back to main loop if 250 mS not passed
0125 0C20      00465      MOVLW   b'00100000'    ; Bits 6 and 5 of FLAGS used as divide by 4
0126 01F9      00466      ADDWF   FLAGS,F         ; Add one to bit 5
0127 07F9      00467      BTFSS   TIMENOW       ; Check bit 7 - if four adds occur, skip
0128 0B38      00468      GOTO    skip_timer    ; One second has not passed - skip timers
0129 090D      00469      CALL    task_scan     ; Scan the next LED digit.
012A 04F9      00470      BCF     TIMENOW       ; Clear out second passed flag
012B 0C0B      00471      MOVLW   CLK_SEC        ; Place pointer to increment clock
012C 099C      00472      CALL    inc_time       ; Increment the clock
012D 0974      00473      CALL    check_time     ; Check for alarm or timer conditions
012E 0699      00474      BTFSC   EGGNOW        ; Do NOT decrease timer if zero
012F 0B38      00475      GOTO    skip_timer    ; Jump out if egg timer is zero
0130 06D8      00476      BTFSC   UPKEY         ; Skip if UP key is NOT pressed
0131 0B38      00477      GOTO    skip_timer    ; Jump out if UP key is pressed
0132 06B8      00478      BTFSC   DOWNKEY      ; Skip if DOWN key is NOT pressed
0133 0B38      00479      GOTO    skip_timer    ; Jump out if DOWN key is pressed
0134 0C14      00480      MOVLW   TMR_SEC_LD     ; Place pointer to decrement timer
0135 09E6      00481      CALL    dec_time       ; Decrement countdown timer
0136 0C28      00482      MOVLW   ALARMCYCCNT    ; Place the number of alarm beeps into W
0137 003F      00483      MOVWF   ALARMCNT     ; Move beep count to ALARMCNT
0138           00484 skip_timer
0138 07A3      00485      BTFSS   ALARMOK        ; Skip if this is the first pass into alarm
0139 0B3F      00486      GOTO    skip_wakeup    ; Second pass - do not re-init ALARMCNT
013A 0779      00487      BTFSS   ALARMNOW      ; Skip if this is alarm pass
013B 0B3F      00488      GOTO    skip_wakeup    ; Countdown timer - do not re-init ALARMCNT
013C 0C28      00489      MOVLW   ALARMCYCCNT    ; Place the number of alarm beeps into W
013D 003F      00490      MOVWF   ALARMCNT     ; Move beep count to ALARMCNT
013E 04A3      00491      BCF     ALARMOK        ; Clear flag for second pass
013F           00492 skip_wakeup
013F 090D      00493      CALL    task_scan     ; Scan the next LED digit.
0140 0679      00494      BTFSC   ALARMNOW      ; Skip if alarm clock is not set
0141 0B45      00495      GOTO    send_alarm     ; Blast out a beep
0142 0699      00496      BTFSC   EGGNOW        ; Skip if countdown timer is not alarming
0143 0B45      00497      GOTO    send_alarm     ; Blast out a beep
0144 0B4A      00498      GOTO    skip_alarm     ; Skip beeping and continue
0145           00499 send_alarm
0145 021F      00500      MOVF    ALARMCNT,W     ; Place ALARMCNT into W
0146 0643      00501      BTFSC   STATUS,Z       ; Skip if not zero
0147 0B4A      00502      GOTO    skip_alarm     ; We are done beeping - skip and continue
0148 02FF      00503      DECF    ALARMCNT,F     ; Decrement beep count and skip when zero
0149 0906      00504      CALL    buzz_now      ; Blast out the beep!!!
014A           00505 skip_alarm
014A 07B9      00506      BTFSS   FLAGS,5        ; Skip if it is time to scan the keys 1/2 sec
014B 0B9A      00507      goto    finish_update  ; Jump to finish updates - don't scan
014C 0966      00508      CALL    scan_keys     ; Scan the keys and load value into KEYPAT
014D 090D      00509      CALL    task_scan     ; Scan the next LED digit.
014E 0798      00510      BTFSS   MODEKEY        ; Skip if the MODEKEY is pressed
014F 0B55      00511      GOTO    same_mode      ; Not pressed so it is the same mode...
0150 079C      00512      BTFSS   MODEKEYCHG     ; Skip if the is pressing edge
0151 0B55      00513      GOTO    same_mode      ; Button is held so it is the same mode...
0152 02B9      00514      INCF    FLAGS,F         ; Advance the mode by incrimting bits 0,1
```

```

0153 0459      00515      BCF      FLAGS,2      ; Force mode to wrap-around by clearing bit 2
0154 0953      00516      CALL     turnon_scan    ; Mode button pressed - must turn on LEDs
0155           00517
0155           00518 same_mode
0155 090D      00519      call     task_scan      ; Scan the next LED digit.
0156 06D8      00520      BTFSC    UPKEY          ; Skip if the UP key is not pressed
0157 0B66      00521      GOTO     serve_up_key    ; UP key is pressed - jump to serve it!
0158 06B8      00522      BTFSC    DOWNKEY        ; Skip if the DOWN key is not pressed
0159 0B81      00523      GOTO     serve_down_key  ; DOWN key is pressed - jump to serve it!
015A 0C08      00524      MOVLW    INIT_MODE_COUNT ; UP and DOWN not pressed - re-init mode count
015B 003E      00525      MOVWF    MODE_COUNT      ; Change back to lower digits for setting
015C 023D      00526      MOVF     DISPONCNT,F      ; Update Z bit in STATUS reg display on time
015D 0743      00527      BTFSS    STATUS,Z        ; Skip if displays should be OFF
015E 00FD      00528      DECF     DISPONCNT,F      ; Decrement display ON counter
015F 0743      00529      BTFSS    STATUS,Z        ; Skip if displays should be OFF
0160 0B9A      00530      GOTO     finish_update   ; Displays are ON - jump to finish updates
0161 0419      00531      BCF      FLAGS,0          ; Restore the mode to displays OFF
0162 0439      00532      BCF      FLAGS,1          ; Restore the mode to displays OFF
0163 0066      00533      CLRF     PORTB           ; Clear out segment drives on PORTB
0164 0065      00534      CLRF     PORTA           ; Clear out common digit drives on PORTA
0165 0B9A      00535      GOTO     finish_update   ; Jump to finish updates
0166           00536 serve_up_key
0166 090D      00537      call     task_scan      ; Scan the next LED digit.
0167 0619      00538      BTFSC    FLAGS,0          ; Skip if not in TIMER or CLOCK mode
0168 0B6D      00539      GOTO     no_up_display    ; Currently in TIMER or CLOCK - keep mode
0169 0639      00540      BTFSC    FLAGS,1          ; Skip if not in ALARM mode
016A 0B6D      00541      GOTO     no_up_display    ; Currently in ALARM - keep mode
016B 0519      00542      BSF      FLAGS,0          ; Set to CLOCK mode
016C 0539      00543      BSF      FLAGS,1          ; Set to CLOCK mode
016D           00544 no_up_display
016D 007F      00545      CLRF     ALARMCNT        ; A key was pressed, so turn off alarm
016E 0953      00546      call     turnon_scan    ; Turn on the LEDs
016F 0798      00547      BTFSS    MODEKEY        ; Skip if MODE is pressed as well
0170 0B9A      00548      GOTO     finish_update   ; MODE is not pressed - jump to finish update
0171 021E      00549      MOVF     MODE_COUNT,W      ; Update STATUS Z bit for mode count
0172 0743      00550      BTFSS    STATUS,Z        ; Skip if we have counted down to zero
0173 00FE      00551      DECF     MODE_COUNT,F      ; Decrement the mode count
0174 090D      00552      call     task_scan      ; Scan the next LED digit.
0175 021E      00553      MOVF     MODE_COUNT,W      ; Update the Z bit to check for zero
0176 0743      00554      BTFSS    STATUS,Z        ; Skip if we have incremented for 7 times
0177 0B7C      00555      GOTO     serve_min_up    ; Increment the minutes digits
0178 00D9      00556      DECF     FLAGS,W          ; Place current mode into W
0179 0900      00557      CALL     mode_timer      ; Look-up register RAM address for current mode
017A 09BA      00558      CALL     inc_hour_ld      ; Add one hour to the current display
017B 0B9A      00559      GOTO     finish_update   ; Jump to finish updates
017C           00560 serve_min_up
017C 090D      00561      call     task_scan      ; Scan the next LED digit.
017D 00D9      00562      DECF     FLAGS,W          ; Place current mode into W
017E 0900      00563      CALL     mode_timer      ; Look-up register RAM address for current mode
017F 09A7      00564      CALL     inc_min_ld      ; Add one minute to the current display
0180 0B9A      00565      GOTO     finish_update   ; Jump to finish updates
0181           00566 serve_down_key
0181 090D      00567      call     task_scan      ; Scan the next LED digit.
0182 0619      00568      BTFSC    FLAGS,0          ; Skip if not in TIMER or CLOCK mode
0183 0B88      00569      GOTO     no_dn_display    ; Currently in TIMER or CLOCK - keep mode
0184 0639      00570      BTFSC    FLAGS,1          ; Skip if not in ALARM mode
0185 0B88      00571      GOTO     no_dn_display    ; Currently in ALARM - keep mode
0186 0519      00572      BSF      FLAGS,0          ; Set to CLOCK mode
0187 0539      00573      BSF      FLAGS,1          ; Set to CLOCK mode
0188           00574 no_dn_display
0188 007F      00575      CLRF     ALARMCNT        ; A key was pressed, so turn off alarm
0189 0953      00576      CALL     turnon_scan    ; Turn on the LEDs
018A 0798      00577      BTFSS    MODEKEY        ; Skip if MODE is pressed as well
018B 0B9A      00578      GOTO     finish_update   ; MODE is not pressed - jump to finish update
018C 021E      00579      MOVF     MODE_COUNT,W      ; Update STATUS Z bit for mode count
018D 0743      00580      BTFSS    STATUS,Z        ; Skip if we have counted down to zero

```

AN615

```
018E 00FE      00581      DECF    MODE_COUNT,F    ; Decrement the mode count
                                00582
018F 090D      00583      call    task_scan        ; Scan the next LED digit.
0190 021E      00584      MOVF    MODE_COUNT,W    ; Update the Z bit to check for zero
0191 0743      00585      BTFSS   STATUS,Z        ; Skip if we have incremented for 7 times
0192 0B97      00586      GOTO    serve_min_down    ; Decrement the minutes digits
0193 00D9      00587      DECF    FLAGS,W        ; Place current mode into W
0194 0900      00588      CALL    mode_timer        ; Look-up register RAM address for current mode
0195 09E5      00589      CALL    dec_hour_ld       ; Subtract one hour from the current display
0196 0B9A      00590      GOTO    finish_update     ; Jump to finish updates
0197           00591      serve_min_down
0197 00D9      00592      DECF    FLAGS,W        ; Place current mode into W
0198 0900      00593      CALL    mode_timer        ; Look-up register RAM address for current mode
0199 09E6      00594      CALL    dec_min_ld        ; Subtract one minute from the current display
019A           00595      finish_update
019A 090D      00596      call    task_scan        ; Scan the next LED digit.
019B 0619      00597      BTFSC   FLAGS,0        ; Skip if in mode OFF or ALARM
019C 0BA4      00598      GOTO    new_display       ; Jump to update LED display registers
019D 0639      00599      BTFSC   FLAGS,1        ; Skip if in mode OFF
019E 0BA4      00600      GOTO    new_display       ; Jump to update LED display registers
019F 0067      00601      CLRF    DISPSEGS_A       ; Clear display regs to Shut off LED display
01A0 0068      00602      CLRF    DISPSEGS_B       ; Clear display regs to Shut off LED display
01A1 0069      00603      CLRF    DISPSEGS_C       ; Clear display regs to Shut off LED display
01A2 006A      00604      CLRF    DISPSEGS_D       ; Clear display regs to Shut off LED display
01A3 0B1E      00605      GOTO    main_loop        ; We are done - go back and do it again!
01A4           00606      new_display
01A4 00D9      00607      DECF    FLAGS,W        ; Move current mode state into W
01A5 0900      00608      CALL    mode_timer        ; Look-up register address of value to display
01A6 091F      00609      CALL    disp_value       ; Update display registers with new values
01A7 0B1E      00610      GOTO    main_loop        ; We are done - go back and do it again!
                                00611
00612 ; *****
00613 ; * Set up and initialize the processor *
00614 ; *****
01A8           00615      init
01A8 0C03      00616      MOVLW   OPTION_SETUP    ; Place option reg setup into W
01A9 0002      00617      OPTION   ; Set up OPTION register
01AA 0C05      00618      MOVLW   PORTA          ; Place beginning of RAM/Port location into W
01AB 0024      00619      MOVWF   FSR            ; Now initialize FSR with this location
01AC           00620      clear_mem
01AC 0060      00621      CLRF    INDADDR        ; Clear the FSR pointed memory location
01AD 03E4      00622      INCFSZ   FSR,F          ; Point to the next location
01AE 0BAC      00623      GOTO    clear_mem        ; Jump back to clear memory routine
01AF 0572      00624      BSF     ALM_HOUR_LD,3    ; Place 8:00 into alarm register
01B0 02AE      00625      INCF    CLK_HOUR_LD,F    ; Place 1:00 into clock register
01B1 0CEE      00626      MOVLW   0EEh           ; Turn on display A scan line, others off
01B2 003B      00627      MOVWF   PREVSCAN        ;
01B3 0040      00628      CLRW          ;
01B4 0006      00629      TRIS     PORTB          ; Make all Port B pins outputs.
01B5 0005      00630      TRIS     PORTA          ; Make all Port A pins outputs.
01B6 0539      00631      BSF     FLAGS,1        ; Set up current mode to CLOCK, display ON
01B7 0519      00632      BSF     FLAGS,0        ;
01B8 04A3      00633      BCF     ALARMOK        ; Don't want to trigger alarms
01B9 04C3      00634      BCF     EGGOK          ;
01BA 059D      00635      BSF     DISPON         ; Turn on the displays
01BB           00636      mfg_checkkey
01BB 0966      00637      CALL    scan_keys        ; Lets see what is pressed
01BC 07D8      00638      BTFSS   UPKEY          ; Goto self-test if UP key is pressed at pwr up
01BD 0B1E      00639      GOTO    main_loop        ; Normal operation - Jump to the main loop
                                00640
00641 ; *****
00642 ; * Self-test code for manufacturing only - test buttons and LEDs *
00643 ; *****
01BE           00644      mfg_selftest
01BE 0C70      00645      MOVLW   b'01110000'     ; Place all key on pattern into W
01BF 002D      00646      MOVWF   CLK_MIN_HD      ; Use CLK_MIN_HD for keystuck ON test
```

```

01C0 006F      00647      CLRF      CLK_HOUR_HD      ; Use CLK_HOUR_HD for keystuck OFF test
01C1           00648 mfg_display
01C1 020B      00649      MOVF      CLK_SEC,W          ; Current segment display count -> W
01C2 0949      00650      CALL      mfg_led_lookup      ; Look-up the next segment pattern to display
01C3 0026      00651      MOVWF     PORTB              ; Move the pattern to PORT B to display it
01C4           00652 mfg_timer
01C4 0201      00653      MOVF      TMR0,W              ; Place current TMR0 value into W
01C5 019A      00654      XORWF     PREVTMR0,W          ; Lets see which bits have changed...
01C6 003C      00655      MOVWF     TEMP              ; All changed bits are placed in temp for test
01C7 01BA      00656      XORWF     PREVTMR0,F          ; Update Previous TMR0 value.
01C8 07FC      00657      BTFSS     TEMP,7              ; Skip if it is not time to increment second
01C9 0BC4      00658      GOTO      mfg_timer            ; It is not time to move to next digit - go back
01CA 02AB      00659      INCF      CLK_SEC,F          ; Move to the next display pattern
01CB           00660 mfg_check_digit
01CB 07AB      00661      BTFSS     CLK_SEC,5          ; Skip if we have timed out waiting for button
01CC 0BD5      00662      GOTO      mfg_doneclk         ; Jump to check for the next button press
01CD           00663 mfg_nextdigit
01CD 006B      00664      CLRF      CLK_SEC              ; Clear out timer
01CE 0906      00665      CALL      buzz_now           ; Send out a buzzer beep!
01CF 077B      00666      BTFSS     PREVSCAN,3          ; Skip if we have NOT tested the last digit
01D0 0BE5      00667      GOTO      finish_mfg_test      ; Jump to the end after last digit tested
01D1 035B      00668      RLF      PREVSCAN,W          ; Select the next digit through a rotate..
01D2 037B      00669      RLF      PREVSCAN,F
01D3 021B      00670      MOVF      PREVSCAN,W          ; Place next digit select into W
01D4 0025      00671      MOVWF     PORTA              ; Update port A to select next digit
01D5           00672 mfg_doneclk
01D5 0966      00673      CALL      scan_keys           ; Scan the keys to see what is pressed...
01D6 0218      00674      MOVF      KEYPAT,W              ; Place pattern into W
01D7 016D      00675      ANDWF     CLK_MIN_HD,F          ; Make sure keys are not stuck ON
01D8 012F      00676      IORWF     CLK_HOUR_HD,F          ; Make sure each key is pressed at least once
01D9 077B      00677      BTFSS     PREVSCAN,3          ; Skip if we are NOT at the last digit
01DA 05F8      00678      BSF      KEYPAT,7              ; Set flag bit to indicate we are done!
01DB 0C08      00679      MOVLW     .8              ; Place 8 into W
01DC 008B      00680      SUBWF     CLK_SEC,W              ; CLK_SEC - W => W
01DD 0703      00681      BTFSS     STATUS,C
01DE 0078      00682      CLRF      KEYPAT
01DF 03B8      00683      SWAPF     KEYPAT,F
01E0 025B      00684      COMF      PREVSCAN,W
01E1 0158      00685      ANDWF     KEYPAT,W
01E2 0743      00686      BTFSS     STATUS,Z
01E3 0BCD      00687      GOTO      mfg_nextdigit
01E4 0BC1      00688      GOTO      mfg_display
01E5           00689 finish_mfg_test
01E5 022D      00690      MOVF      CLK_MIN_HD,F
01E6 0743      00691      BTFSS     STATUS,Z
01E7 0BEF      00692      GOTO      bad_switch
01E8 020F      00693      MOVF      CLK_HOUR_HD,W
01E9 0F70      00694      XORLW     070h
01EA 0743      00695      BTFSS     STATUS,Z
01EB 0BEF      00696      GOTO      bad_switch
01EC           00697 mfg_cleanup
01EC 006F      00698      CLRF      CLK_HOUR_HD          ; Restore temp registers to zero
01ED 006D      00699      CLRF      CLK_MIN_HD          ; Restore temp registers to zero
01EE 0B1E      00700      GOTO      main_loop          ; Jump to main loop
01EF           00701 bad_switch
01EF 026D      00702      COMF      CLK_MIN_HD,F
01F0 038D      00703      SWAPF     CLK_MIN_HD,W
01F1 0038      00704      MOVWF     KEYPAT
01F2 05EF      00705      BSF      CLK_HOUR_HD,7
01F3 038F      00706      SWAPF     CLK_HOUR_HD,W
01F4 0178      00707      ANDWF     KEYPAT,F
01F5 0C7F      00708      MOVLW     07Fh
01F6 0026      00709      MOVWF     PORTB
01F7 006C      00710      CLRF      CLK_MIN_LD
01F8 05AC      00711      BSF      CLK_MIN_LD,5
01F9           00712 loop_bad_sw

```

AN615

```
01F9 0907      00713      CALL    buzz_now_dispon ; Beep the buzzer constantly for a few secs
01FA 02EC      00714      DECFSZ  CLK_MIN_LD,F      ; Decrement counter and skip when done
01FB 0BF9      00715      GOTO    loop_bad_sw      ; Not done buzzing - go back and do it again
01FC 0BEC      00716      GOTO    mfg_cleanup      ; Done buzzing - clean-up and run clock
                        00717      END
```

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

```
0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0F80 : -----
0FC0 : -----X
```

All other memory blocks unused.

```
Errors      :      0
Warnings    :      0
Messages    :      0
```

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

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

Atlanta

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

Boston

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

Chicago

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

Dallas

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

Dayton

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

Los Angeles

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

New York

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

San Jose

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

Toronto

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

ASIA/PACIFIC

Hong Kong

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

India

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

Korea

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

Shanghai

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

Singapore

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

Taiwan, R.O.C

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

EUROPE

United Kingdom

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

France

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

Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 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.