

Specification for a digital alarm clock

Date: June 28, 2006

Author: Jonathan T Cotton

Introduction:

This specification describes the design of a digital alarm clock that has a simple input design using a numeric keypad. This system monitors and maintains the current time, as well as an alarm time setting. The current time will be displayed in the standard 12 hour time format for the user to view. The current time as well as the alarm time is programmable using the numeric keypad. The alarm can be enable to run or turned off completely.

System Specifications:

System Components:

The parts utilized for this system are listed below. Next to each part number is a description of the part.

1 x PIC16F877	Microcontroller
4 x LDS-C302RI	7-Segment Display
4 x 74LS47	BCD to 7-segment Decoder
1 x 7805	5 volt Voltage Regulator
1 x GH5003-ND	4 x 4 Keypad
1 x PT-2130PQ	Piezo Buzzer
1 x ECS-40-20-1	4 MHz crystal HC-49/UA

16 x 330 Ω Resistor
2 x 0.01 μ F Capacitor
2 x 22 pF Capacitor
6 x LEDS

System Inputs:

The system uses a sixteen key numeric keypad for input. The keypad contains the decimal digits zero through nine as well as the letters A through D and the special symbols * and #. The keypad is configured as a 4x4 matrix which requires four input pins and four output pins. These inputs are used to set the current time, alarm times, disable the alarm after it has been triggered, as well as select the current operating mode.

These four input lines are assigned to RB0 through RB3. These four input pins are setup to be active high by connecting the pins to voltage high using four 330 Ω resistors.

There is an external clock source utilizing a 4 MHz crystal that uses the OSC1 and OSC2 input pins. The clock utilizes two 22 pF capacitors.

A simple SPST switch is used to enable / disable the alarm function. When this is disabled, the system will track the time but will not check for alarm occurrences. This will be assigned to RE2 and it is tied to voltage high when the alarm is enabled and it is voltage low when it is disabled. The switch is tied to active high when open using a 330 Ω resistor.

System Outputs:

There are four LEDs that are used to display the current mode the system is in. These LEDs use four 330 Ω resistors tied to voltage high on the anode side of the LED. The cathode sides are connected to RA0 through RA3.

The numeric keypad requires four outputs in addition to the four inputs described earlier. The output signals for the keyboard matrix are assigned to RB4 through RB7.

The system uses four 7-segment displays to show the current time and input feedback. These 7-segment displays are driven using four 74LS47 BCD to 7-segment decoders. Each 7-segment display has a common anode that is connected to voltage high using a 330 Ω resistor.

Each 74LS47 has a 4-bit input that is used to drive the 7-segment displays. In order to drive all four displays, a total of sixteen output lines are required to display all of the data. These sixteen output signals are assigned to RC0 through RC7 and RD0 through RD7. The hours are assigned to PORT C and the minutes are assigned to PORT D. Internally, the hours and minutes are represented as an 8-bit binary number. When outputting to PORTS C and D, these numbers have to be split into the tens digit and the ones digit, where the tens digit is the high nibble and the ones digit is the low nibble.

There is an AM/PM indicator LED. This LED is assigned to RE0. The AM/PM LED is lit when the current time represents AM. The anode side is connected through a 330 Ω resistor to RE0 and the cathode side is connected to voltage low.

The alarm buzzer requires a single output pin. This is assigned to RE1. There is also one LED that is connected to the alarm buzzer to add a visual indicator that the alarm has been triggered. The anode of this LED is connected through a 330 Ω resistor to RE1 and the cathode is connected to voltage low.

System Operation:

This system is directly affected by any change or variation in clock frequency. Even though small clock frequency changes may not be noticeable over a few minutes, hours, or even days, the long term effect can cause inaccurate time keeping. The time keeping and input functionality is primarily an interrupt driven system while the information processing is primarily found in a repetitive looping structure.

Interrupts:

This system relies heavily on the interrupt sequence for its functionality. The interrupt sequence is setup to occur every 100ms. At this time, multiple processes occur.

The first thing to occur in the interrupt is the saving of the STATUS and W registers to temporary registers. This ensures that the system functions the same before and after the interrupt.

After this, the system checks to see if 10 100ms interrupts have occurred. This indicates when 1 second has occurred. If 1 second has occurred, the system increments the seconds register and then checks for overflow (beyond 60). If an overflow occurs, the seconds are set to 0 and the minutes register is incremented. This too is then checked for an overflow (60 minutes). If this overflow occurs, the minutes are set to 0 and the hours are incremented. At this point, two "overflow" checks are made. The first one is to see if we have moved from 11 to 12. At this point, the AM/PM bit needs to be changed. The second check is to see if we have moved from 12 to 13. If this occurs, the hours are set back to 1 as this system is keeping time using a 12 hour system.

The next stage of the interrupt is the alarm checking portion. This portion compares each of the current time registers with the stored alarm time registers to see if they match. If they all match, then the alarm bit is set. Otherwise, nothing is changed.

The final step of the interrupt sequence is the keypad scanning. In this part, each output bit on port B is individually turned off (active low) and then each input line (RB0-RB3) is then checked to see if a connection has been made. If a connection is detected, the system jumps to the function for the key associated with the current input and output pin combination and saves this key's value into the KEYPRESS register. It then sets the flag indicating a key was pressed and goes to the end of the interrupt routine. If no keys were detected, then it continues to the end of the interrupt routine without changing anything.

At the end of the interrupt routine, all of the outputs on port B are reset to high and the interrupt flag is cleared. The STATUS and W registers are then restored and the interrupt is ended.

Looping Modes:

There are four separate modes of operation for this system outside of the interrupt sequence. Each of these modes represents a different stage in the alarm process. These modes include the mode when there is not an alarm triggered, the mode when the alarm has been triggered, the time setup mode, and the alarm setup mode. There is also a loop that updates the displays and checks for changes in the mode.

The first mode is the "Run" mode. This mode begins by checking for a change in the mode. If the mode has changed, it then redirects to the proper change mode routine. Otherwise, it updates the displayed time based on the information stored in the registers and then returns to the top of the loop.

The second mode is the time setup mode. This mode is activated by pressing the "A" key while in mode 1. In this mode, the system will wait for a key to be pressed on the keypad. Once a key has been pressed, this mode will check to see if it is an "A". If it is, the mode is ended and the contents of ports C, D, and E are stored into the hours, minutes, and AM/PM registers. This mode also checks to see if "#" was pressed. If it was, port E bit 0 is complemented. This effectively lets the user set the AM/PM setting for the current time. If it is not an "A" or "#", the contents of port C and D are shifted to the left and the key that was pressed is appended to the right. This effectively shifts user input through ports C and D.

The third mode is the alarm setup mode. This mode is activated by pressing the "B" key while in mode 1. This mode functions nearly identically to mode 2. There are a few differences however. The first difference is that upon starting mode 3, the current values in the alarm storage registers, timerminutes and timerhours, are displayed on ports C, D, and E. This displays the current alarm setting for user verification. After that, this mode functions identically to mode 2 with the exception of the fact that at the end, the data from the ports is saved to the alarm registers instead of the current time registers.

The fourth mode is the alarm triggered mode. This mode is triggered in two different ways. The normal way it is triggered is by the interrupt routine. The interrupt routine compares the current time with the alarm time and if they match, sets a flag indicating that it is time for an alarm. When this happens, mode 1 detects the flag and changes to mode 4. Now mode 4 checks to see if the "disarm" or "snooze" button has been pressed. It does this by watching for a "D" on the keypad. If a "D" is detected, it clears the alarm flag and returns to mode 1. If a "D" is not detected, it then proceeds to produce an oscillating frequency on port E bit 1 in order to drive the piezo buzzer. This signal also drives the alarm LED. The frequency is controlled by a counter loop that decrements a counter each time the mode 4 routine is called. Once it is 0, it complements RE1 and resets the count which effectively oscillates the signal.

At the end of each mode, there is a GOTO directive that loops back to the top. This returns the system to the beginning of the loop where the current mode is the re-checked and the system continues again based on the current mode.

Processor Choice:

Due to the need for a large number of input and output pins, the PIC16F877 processor is used for this project. The PIC16F877 has an acceptable amount of memory for this project. The clocking frequency is set to 4 MHz and is done using a crystal oscillator which is well within the 20 MHz range of this processor. A ceramic resonator was used in the initial design but its inaccuracy led to the time being off after a number of hours. By using a high quality crystal instead, this problem is resolved.

I/O Assignments:

PORTA – 0	MODE LED 0	
PORTA – 1	MODE LED 1	
PORTA – 2	MODE LED 2	
PORTA – 3	MODE LED 3	
PORTA – 4	NOT USED	
PORTA – 5	NOT USED	
PORTB – 0	KMI0 Keypad Matrix Input 0	
PORTB – 1	KMI1 Keypad Matrix Input 1	
PORTB – 2	KMI2 Keypad Matrix Input 2	
PORTB – 3	KMI3 Keypad Matrix Input 3	
PORTB – 4	KMO0 Keypad Matrix Output 0	Active High
PORTB – 5	KMO1 Keypad Matrix Output 1	Active High
PORTB – 6	KMO2 Keypad Matrix Output 2	Active High
PORTB – 7	KMO2 Keypad Matrix Output 3	Active High
PORTC – 0	D00 7-segment Display BCD 0 digit 0	Active High
PORTC – 1	D01 7-segment Display BCD 0 digit 1	Active High
PORTC – 2	D02 7-segment Display BCD 0 digit 2	Active High
PORTC – 3	D03 7-segment Display BCD 0 digit 3	Active High
PORTC – 4	D10 7-segment Display BCD 1 digit 0	Active High
PORTC – 5	D11 7-segment Display BCD 1 digit 1	Active High
PORTC – 6	D12 7-segment Display BCD 1 digit 2	Active High
PORTC – 7	D13 7-segment Display BCD 1 digit 3	Active High
PORTD – 0	D20 7-segment Display BCD 2 digit 0	Active High
PORTD – 1	D21 7-segment Display BCD 2 digit 1	Active High
PORTD – 2	D22 7-segment Display BCD 2 digit 2	Active High
PORTD – 3	D23 7-segment Display BCD 2 digit 3	Active High

PORTD – 4	D30	7-segment Display BCD 3 digit 0	Active High
PORTD – 5	D31	7-segment Display BCD 3 digit 1	Active High
PORTD – 6	D32	7-segment Display BCD 3 digit 2	Active High
PORTD – 7	D33	7-segment Display BCD 3 digit 3	Active High
PORTE – 0	AP	AM/PM LED	Active Low
PORTE – 1	BZ	Buzzer	Active High
PORTE – 2	ENBL	Alarm Enable Input	

Additional Information:

The system uses a 5V voltage regulator, in order to provide acceptable voltage to the external ICs. The power for this system can be supplied by a 9volt battery power source as well as by an AC-DC wall power source (between 6 and 10 volts is acceptable).

How to Operate:

The alarm clock system automatically starts up with a default time of 12:00 AM and a default alarm setting of 12:00 AM.

To Set The Time:

Press the “A” button and verify that the second mode indicator LED is lit. You are now in the time setting mode. Now type in the new time using the numeric keypad. For example, type in 0532 for 5:32 or 1234 for 12:34. Note that the leading zero is required for single digit hours. To set the AM/PM bit, press the “#” key. This will toggle the AM/PM bit and the AM/PM LED will reflect this change. Once you have set the time, press the “A” button again and the new time will be saved.

To Set The Alarm:

Press the “B” button and verify that the third mode indicator LED is lit. The currently saved alarm time will be displayed on the 7-segment displays. You are now in the alarm setting mode. Type in the desired alarm time using the keypad. For example, type in 0532 for 5:32 or 1234 for 12:34. Note that the leading zero is required for single digit hours. To set the AM/PM bit, press the “#” key. This will toggle the AM/PM bit and the AM/PM LED will reflect this change. Once you have set the time, press the “B” button again and the new alarm will be saved. The current time will now be displayed again.

To Turn Off a Triggered Alarm:

If the alarm has been triggered, press the “D” key to turn off the alarm. This does not disable the alarm. The alarm will trigger again the next day when the alarm matches the time again.

Future Improvements:

Overall, when constructed using a fairly accurate crystal that produces a 4 MHz frequency, this system is capable of keeping accurate time. This time is however, slightly off due to the design of the hardware and software. Using a different frequency of crystal, this could be improved upon so that the actual internal delays are exact instead of simply being close enough to not notice the difference.

Another future improvement would be the addition of a real time clock, such as the Dallas DS1307. This would allow extremely accurate time keeping. This would also extend the functionality of the system by adding date, day of week, and various other options. Additionally, this would relieve a large part of the coding needed and would also remove the microcontroller's dependence on an accurate clock frequency. A simple RC timing network could be easily used as a replacement for the crystal used now.

Another potential improvement would be the inclusion of an LCD display instead of the 7-segment displays. Not only would this offer greater flexibility in the actual display of data, but it would also require fewer pins than the 7-segment displays do. Using an LCD such as the HITACHI HD44780U Dot Matrix Liquid Crystal Display, the number of I/O pins needed could be reduced to as low as 7 pins from the currently used 16 pins. These displays can be found relatively inexpensively on eBay and from other sources as well.

Appendix B – Source Code Listing

```

*****
This file is the complete specification of a simple digital alarm *
clock. This alarm clock uses a 4x4 keypad for data input and *
uses various LEDs as well as a speaker for output. It keeps *
track of the current time in seconds, minutes, and hours, and *
displays the minutes and hours on 4 7-segment displays. The *
processor used is the PICmicro PIC16F877. *
*
Interrupts are used to update the current time as well as to scan *
for new keypad inputs. The information processing occurs in a *
repetitive looping structure. *
*****

Filename:          alarmclock.asm *
Date:              June 27, 2006 *
File Version:     1.0 *

Author:           Jonathan Cotton *
Company:          CS3371 Texas Tech University *
*****

I/O Assignments: *

PORTA - 0        MODE LED 0 *
PORTA - 1        MODE LED 1 *
PORTA - 2        MODE LED 2 *
PORTA - 3        MODE LED 3 *
PORTA - 4        NOT USED *
PORTA - 5        NOT USED *

PORTB - 0        KMI0    Keypad Matrix Input 0 *
PORTB - 1        KMI1    Keypad Matrix Input 1 *
PORTB - 2        KMI2    Keypad Matrix Input 2 *
PORTB - 3        KMI3    Keypad Matrix Input 3 *
PORTB - 4        KMO0    Keypad Matrix Output 0 *
PORTB - 5        KMO1    Keypad Matrix Output 1 *
PORTB - 6        KMO2    Keypad Matrix Output 2 *
PORTB - 7        KMO2    Keypad Matrix Output 3 *

PORTC - 0        D00     7-segment Display BCD 0 digit 0 *
PORTC - 1        D01     7-segment Display BCD 0 digit 1 *
PORTC - 2        D02     7-segment Display BCD 0 digit 2 *
PORTC - 3        D03     7-segment Display BCD 0 digit 3 *
PORTC - 4        D10     7-segment Display BCD 1 digit 0 *
PORTC - 5        D11     7-segment Display BCD 1 digit 1 *
PORTC - 6        D12     7-segment Display BCD 1 digit 2 *
PORTC - 7        D13     7-segment Display BCD 1 digit 3 *

PORTD - 0        D20     7-segment Display BCD 2 digit 0 *
PORTD - 1        D21     7-segment Display BCD 2 digit 1 *
PORTD - 2        D22     7-segment Display BCD 2 digit 2 *
PORTD - 3        D23     7-segment Display BCD 2 digit 3 *
PORTD - 4        D30     7-segment Display BCD 3 digit 0 *
PORTD - 5        D31     7-segment Display BCD 3 digit 1 *
PORTD - 6        D32     7-segment Display BCD 3 digit 2 *
PORTD - 7        D33     7-segment Display BCD 3 digit 3 *

PORTE - 0        AP      AM/PM LED *
PORTE - 1        BZ      Buzzer *
PORTE - 2        ENBL    Alarm Enable Input *
*****

Boilerplate Setup Code
*****

LIST                P=16f877 ; define processor
#include "p16f877.inc" ; variable definitions

; '__CONFIG' directive is used to embed configuration data
; within .asm file. The labels following the directive are
; located in the respective .inc file. See the respective
; data sheet for additional information on configuration word.

__CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_OFF & _XT_OSC & _WRT_ENABLE_ON & _LVP_OFF & _DEBUG_OFF & _CPD_OFF
*****

Equates
*****

; DEBUG
DEBUG EQU 1 ; debug = 1 mode => timer much faster
DEBUG EQU 0

w_temp EQU 0x70 ; variable for saving W register
s_temp EQU 0x71 ; variable for saving STATUS register

TICK EQU 0 ; 100 ms tick flag
KEYFLAG EQU 1 ; Flag to indicate if a key has been pressed
MODE1 EQU 2 ; Mode 1 flag - Run Mode.
MODE2 EQU 3 ; Update current time on the display
; Mode 2 flag - Set Time Mode.
; Set the current time using the

keypad
MODE3 EQU 4 ; Mode 3 flag - Set Alarm Mode.

```

```

; Set the time the alarm should go
off at
MODE4 EQU 5 ; Mode 4 flag - Alarm triggered Mode.
; Continue here until snooze

has been hit.
SHOWTIMER EQU 6 ; Flag to indicate the timer time needs to be displayed
NEXTMIN EQU 7 ; Keep alarm from getting stuck

IF DEBUG == 1
OFBASE EQU D'2' ; overflow counts fast!
TEN EQU D'2'
ELSE
OFBASE EQU D'195' ; overflow counts in 100.12 ms
TEN EQU D'10' ; based on 4.00 MHz clock
ENDIF

*****
;
; Macro definitions
;
*****

RAM VARIABLES
*****

CBLOCK 20 ; directive to start our variable declaration block
Wcopy ; saved W register for interrupt handler
STcopy ; saved status register for int handler
cfgFlag ; Configuration Flags
tFlags ; Timer Flags, shown as TMASK
gFlags ; Go Flags. Keypress signals
ovrCnt ; 100 ms timebase counter
ovrCnt2 ; 200 ms timebase counter
secCnt ; one second count
SECONDS ; keep track of seconds
MINUTES ; keep track of minutes
HOURS ; keep track of hours
AP ; keep track of AM/PM
TIMERMINUTES ; keep track of timer minutes
TIMERHOURS ; keep track of timer hours
TIMERAP ; keep track of timer AM/PM
TMP ; Temporary Variable
TMP2 ; Temporary Variable
TMP3 ; Temporary Variable
KEYPRESS ; The key that was last pressed
INPUTCOUNT ; The count to make sure the correct # of inputs was made

ENDC ; directive to end our variable declaration block

*****
;
; MAIN PROGRAM
;
*****

ORG 0x000 ; Start program at address 0
CLRFB ; ensure page bits are cleared
GOTO MAIN ; go to beginning of program

*****
;
; BEGINNING OF TIMER INTERRUPT HANDLING CODE
;
*****

ORG 0x004 ; interrupt vector location

*****
;
; Save current values of W and STATUS
;
*****

MOVWF w_temp ; save off current W register contents
MOVF STATUS, w ; move status register into W register
MOVWF s_temp ; save off contents of STATUS register

*****
;
; ISR code goes here
;
*****

BTFSB INTCON, TOIF ; see if it is a timer interrupt
; by comparing timer 0 interrupt flag with
; interrupt register contents
GOTO GOTICK ; if timer interrupt, process it
GOTO IntExit ; if not timer interrupt, exit from interrupt
; handling code since we only care about
; timer interrupts in this demo

GOTICK
BTFSB ovrCnt, 0
INCF TMRO, 1

DECFSZ ovrCnt, f ; decrement the overflow counter
GOTO TintDone ; if not 0, then goto TimrDone
; otherwise overflow counter is zero, so

MOVLW OFBASE ; preset overflow counter from constant OFBASE
MOVWF ovrCnt ; reload the overflow counter

```

TESTTICK

```

*****
100ms tick has occurred...
*****
        DECFSZ   secCnt, f      ; decrement 1s counter
        GOTO    EOF            ; not zero

*****
1s tick has occurred...
*****
        MOVLW   d' 10'
        MOVWF   secCnt         ; reset 1s counter

SECOND
        INCF    SECONDS, 1     ; increment the seconds count
        MOVLW   d' 60'
        SUBWF   SECONDS, 0     ; load 60 into the W register
        BTFSZ   STATUS, Z     ; subtract 60 from the seconds count
        GOTO    EOF            ; check if it is zero.
                                ; if not 0, then it isn't time to increment minutes

MI NUTE
        MOVLW   0
        MOVWF   SECONDS       ; reset the seconds count to 0
        INCF    MINUTES, 1    ; increment the minutes count
        BSF     gFlags, NEXTMIN ; used to keep the alarm from getting stuck
        MOVLW   d' 60'
        SUBWF   MINUTES, 0    ; load 60 into the W register
        BTFSZ   STATUS, Z     ; subtract 60 from the minutes count
        GOTO    EOF            ; check if it is zero.
                                ; if not 0, then it isn't time to increment hours

HOUR
        MOVLW   0
        MOVWF   MINUTES       ; reset the minutes count to 0
        INCF    HOURS, 1     ; increment the hours count

        MOVLW   d' 12'
        SUBWF   HOURS, 0      ; move 12 into the W register
        BTFSZ   STATUS, Z     ; subtract 12 from the current hours
        GOTO    HOURSOVFLW    ; check for hours overflow past 12
                                ; Go to the function to deal with the overflow

        MOVLW   d' 13'
        SUBWF   HOURS, 0      ; move 12 into the W register
        BTFSZ   STATUS, Z     ; subtract 12 from the current hours
        GOTO    HOURSOVFLW2   ; check for hours overflow past 12
                                ; Go to the function to deal with the overflow

        GOTO    EOF            ; Otherwise, we are done.

HOURSOVFLW
        BTFSZ   AP, 0
        GOTO    SETAMP        ; Switch AM/PM. Check to see if AP,0 is set
        GOTO    CLEARAMP     ; If it isn't, then set it (AM)
                                ; otherwise, clear it (PM)

HOURSOVFLW2
        MOVLW   d' 1'
        MOVWF   HOURS        ; Move 1 into the W register
        GOTO    EOF            ; Set the Hours to 1

SETAMP
        BSF     AP, 0
        GOTO    EOF            ; Set the AP,0 bit (AM)

CLEARAMP
        BCF     AP, 0
        GOTO    EOF            ; clear the AP, 0 bit (PM)

EOF

; see if the alarm is enabled.
BTFSZ   PORTE, 2             ; check PORTE,2. This is the alarm enable pin (active low)
GOTO    NOALARM              ; if it is low, that means the alarm is disabled.
                                ; skip to the end.

; otherwise, the alarm is enabled. check for an alarm condition.
BTFSZ   gFlags, NEXTMIN
GOTO    NOALARM

MOVWF   MINUTES              ; save the current minutes into the W register
SUBWF   TIMERMINUTES, 0     ; subtract the current minutes from the timer
BTFSZ   STATUS, Z           ; check to see if the result was 0
GOTO    NOALARM              ; if it wasn't that means the minutes don't match.
                                ; no alarm.

MOVWF   HOURS                ; save the current hours into the W register
SUBWF   TIMERHOURS, 0       ; subtract the current hours from the timer hours.
BTFSZ   STATUS, Z           ; check to see if the result was 0
GOTO    NOALARM              ; if it wasn't, that means the hours don't match.
                                ; no alarm.

BTFSZ   AP, 0
GOTO    CHECKAPSET          ; test to see if it is AM or PM
GOTO    CHECKAPCLEAR       ; if it is AM
                                ; if it is PM

CHECKAPSET
        BTFSZ   TIMERAP, 0    ; check to see if TIMER AP is PM
        GOTO    ALARM         ; if it is, we match. Go to ALARM
        GOTO    NOALARM      ; otherwise, no match = no alarm

CHECKAPCLEAR
        BTFSZ   TIMERAP, 0    ; check to see if TIMERAP is PM
        GOTO    ALARM         ; if it is, we match. Go to ALARM

```

```

GOTO    NOALARM                ; otherwise, no match = no alarm
; otherwise, if we got here, that means everything matches

ALARM
MOVLW   b'11000011'
ANDWF   gFlags, 1              ; clear all of the other mode flags.
BSF     gFlags, MODE4          ; set the mode to mode 4

NOALARM
; no alarm. continue on with the interrupt processing
; only process keypad input every other timer interrupt.
; this helps avoid key bounce.

DECFSZ  ovrCnt2, f             ; decrement the 2nd overflow counter
GOTO    TintDone               ; if not 0, then goto TimrDone
; otherwise overflow counter is zero, so

MOVLW   d'2'                   ; preset overflow counter from constant OFBASE
MOVWF   ovrCnt2                ; reload the overflow counter

BTFSC   gFlags, KEYFLAG        ; check to see if a key is already being pressed.
; If it is, skip to the end.
GOTO    DONE                   ; otherwise, Scan for a new key press.

SCAN
BCF     PORTB, 4                ; set the first control line to low

BTFSS   PORTB, 0                ; check the first signal line
GOTO    KEY1                   ; and go to the section for this key if pressed

BTFSS   PORTB, 1                ; check the second signal line
GOTO    KEY2                   ; and go to the section for this key if pressed

BTFSS   PORTB, 2                ; check the third signal line
GOTO    KEY3                   ; and go to the section for this key if pressed

BTFSS   PORTB, 3                ; check the fourth signal line
GOTO    KEYA                   ; and go to the section for this key if pressed

BSF     PORTB, 4                ; set the first control line to high
BCF     PORTB, 5                ; set the second control line to low

BTFSS   PORTB, 0                ; check the first signal line
GOTO    KEY4                   ; and go to the section for this key if pressed

BTFSS   PORTB, 1                ; check the second signal line
GOTO    KEY5                   ; and go to the section for this key if pressed

BTFSS   PORTB, 2                ; check the third signal line
GOTO    KEY6                   ; and go to the section for this key if pressed

BTFSS   PORTB, 3                ; check the fourth signal line
GOTO    KEYB                   ; and go to the section for this key if pressed

BSF     PORTB, 5                ; set the second control line to high
BCF     PORTB, 6                ; set the third control line to low

BTFSS   PORTB, 0                ; check the first signal line
GOTO    KEY7                   ; and go to the section for this key if pressed

BTFSS   PORTB, 1                ; check the second signal line
GOTO    KEY8                   ; and go to the section for this key if pressed

BTFSS   PORTB, 2                ; check the third signal line
GOTO    KEY9                   ; and go to the section for this key if pressed

BTFSS   PORTB, 3                ; check the fourth signal line
GOTO    KEYC                   ; and go to the section for this key if pressed

BSF     PORTB, 6                ; set the third control line to high
BCF     PORTB, 7                ; set the fourth control line to low

BTFSS   PORTB, 0                ; check the first signal line
GOTO    KEYS                   ; and go to the section for this key if pressed

BTFSS   PORTB, 1                ; check the second signal line
GOTO    KEYO                   ; and go to the section for this key if pressed

BTFSS   PORTB, 2                ; check the third signal line
GOTO    KEYP                   ; and go to the section for this key if pressed

BTFSS   PORTB, 3                ; check the fourth signal line
GOTO    KEYD                   ; and go to the section for this key if pressed

BSF     PORTB, 7                ; set the fourth control line to high
GOTO    DONE

; The Key Press Functions. When called, save the appropriate value to the
; KEYPRESS register and then set the 0 bit in the gFlags register to 1.

KEY0
MOVLW   d'0'
MOVWF   KEYPRESS
BSF     gFlags, KEYFLAG
GOTO    DONE

KEY1
MOVLW   d'1'
MOVWF   KEYPRESS
BSF     gFlags, KEYFLAG
GOTO    DONE

KEY2
MOVLW   d'2'
MOVWF   KEYPRESS
BSF     gFlags, KEYFLAG
GOTO    DONE

```

```

KEY3      MOVLW    d' 3'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEY4      MOVLW    d' 4'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEY5      MOVLW    d' 5'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEY6      MOVLW    d' 6'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEY7      MOVLW    d' 7'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEY8      MOVLW    d' 8'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEY9      MOVLW    d' 9'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEYA      MOVLW    d' 10'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEYB      MOVLW    d' 11'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEYC      MOVLW    d' 12'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEYD      MOVLW    d' 13'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEYS      MOVLW    d' 14'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

KEYP      MOVLW    d' 15'
          MOVWF    KEYPRESS
          BSF      gF1ags, KEYFLAG
          GOTO     DONE

DONE      ; The end of the scanning function
          BSF      PORTB, 4      ; Re-set all of the control signals to high.
          BSF      PORTB, 5
          BSF      PORTB, 6
          BSF      PORTB, 7

```

```

TIntDone
BCF      INTCON, TOIF      ; clear RTCC int mask

```

IntExit

```

*****
: Restore values of W and STATUS to their pre-interrupt states *
:
*****
          MOVF     s_temp, w      ; retrieve copy of STATUS register
          MOVWF    STATUS        ; restore pre-isr STATUS register
          SWAPF   w_temp, f      ; restore pre-isr W register
          SWAPF   w_temp, w
          RETFIE    ; return from interrupt

*****
: THIS IS THE REAL START OF PROGRAM CODE *
:
*****

```

MAIN

```
*****
:
: SETUP ALL OF THE CONFIGURATION REGISTERS
:
*****

        BSF     STATUS, RPO           ; Select Bank 1
        BCF     STATUS, RP1

        MOVLW   b'00000000'          ; Setup the OPTION_REG register
        MOVWF   OPTION_REG           ; Enable pullups on PORTB, prescaler
                                        ; assigned to TMR0, prescaler set to 2

        MOVLW   b'00000110'          ; Setup the ADCON1 register.
        MOVWF   ADCON1              ; All PORTA and PORTE are digital now

        MOVLW   b'00000000'          ; Setup PORTA Direction
        MOVWF   TRISA               ; Set PORTA as all outputs

        MOVLW   b'00001111'          ; Setup PORTC Direction
        MOVWF   TRISB               ; Set PORTB <0:3> as inputs and <4:7> as outputs

        MOVLW   b'00000000'          ; Setup PORTC Direction
        MOVWF   TRISC               ; Set PORTC as all outputs

        MOVLW   b'00000000'          ; Setup PORTD Direction
        MOVWF   TRISD               ; Set PORTD as all outputs

        MOVLW   b'00000100'          ; Setup PORTE Direction
        MOVWF   TRISE               ; Set PORTE <0:1> as outputs and <2> as an input

        BCF     STATUS, RPO           ; Select Bank 0
        BCF     STATUS, RP1

        CLRF    gFlags               ; clear the run flags
        CLRF    tFlags               ; clear timer flags

        MOVLW   OFBASE               ; get the overflow counter value into W
        MOVWF   ovrCnt              ; write it from W into the variable

        MOVLW   d'2'                 ; set the debounce counter to 2 in W
        MOVWF   ovrCnt2             ; write it from W into the variable

        MOVLW   d'10'                ; 1 second down counter
        MOVWF   secCnt

        MOVLW   b'11111111'          ; set all PORTA to high (active low)
        MOVWF   PORTA              ; set all PORTB to high (active low)
        MOVLW   b'10100000'          ; enable interrupts, & Timer 0 overflow
        MOVWF   INTCON

        MOVLW   d'0'                 ; Set Seconds, Ten Seconds, Minutes,
        MOVWF   SECONDS             ; to 0 at startup
        MOVLW   MI NUTES
        MOVWF   T I M E R M I N U T E S

        MOVLW   d'12'                ; and Hours to 2
        MOVWF   HOURS
        MOVWF   T I M E R H O U R S

        BSF     AP, 0                 ;
        BSF     T I M E R A P, 0

        MOVLW   d'4'                 ; Set the initial input count to 4
        MOVWF   I N P U T C O U N T

        BSF     gFlags, MODE1        ; Default mode is the run mode
        MOVLW   b'11111110'          ; indicate the current mode on PORTA
        MOVWF   PORTA

*****
:
: LOOP CODE
:
*****

        GOTO    D I S P L A Y T I M E ; Start out by displaying the current time

LOOP

        BTFSC   gFlags, MODE1
        GOTO    R U N M O D E 1

        BTFSC   gFlags, MODE2
        GOTO    R U N M O D E 2

        BTFSC   gFlags, MODE3
        GOTO    R U N M O D E 3

        BTFSC   gFlags, MODE4
        GOTO    R U N M O D E 4

*****
:
: MODE 1
:
*****

RUNMODE1

CHECKMODE
; Check to see if the mode is changing.
        BTFSS   gFlags, KEYFLAG     ; Test to see if a key was pressed by checking the gFlags register
```

```

GOTO    DISPLAYTIME                ; If no key was pressed, skip to the end of the interrupt

MOVLW  d' 10'                      ; check to see if A was pressed.
SUBWF  KEYPRESS, 0
BTFSC  STATUS, Z
GOTO    CHANGEMODE2                ; if it was, change the mode

MOVLW  d' 11'                      ; check to see if B was pressed.
SUBWF  KEYPRESS, 0
BTFSC  STATUS, Z
GOTO    CHANGEMODE3                ; if it was, change the mode

MOVLW  d' 12'                      ; check to see if C was pressed.
SUBWF  KEYPRESS, 0
BTFSC  STATUS, Z
GOTO    CHANGEMODE4                ; if it was, change the mode

GOTO    ENDCHECKMODE                ; otherwise, no mode changes were made. Go to the end.

ENDCHECKMODE
BCF    gFlags, KEYFLAG              ; And finally clear the key press flag.

DISPLAYTIME
; minutes on port D
; hours on port C

MOVLW  d' 12'                      ; set the W register to 12
SUBWF  HOURS, 0                    ; subtract 12 from the Hours
BTFSC  STATUS, Z                    ; If it is 12, go to TENHOURDISP
GOTO    TENHOURDISP

MOVLW  d' 11'                      ; set the W register to 11
SUBWF  HOURS, 0                    ; subtract 11 from the Hours
BTFSC  STATUS, Z                    ; If it is 11, go to TENHOURDISP
GOTO    TENHOURDISP

MOVLW  d' 10'                      ; set the W register to 10
SUBWF  HOURS, 0                    ; subtract 10 from the Hours
BTFSC  STATUS, Z                    ; If it is 10, go to TENHOURDISP
GOTO    TENHOURDISP

ONEHOURDISP
MOVWF  HOURS                        ; move the current HOURS into the W register
MOVWF  PORTC                        ; set PORTC to the current HOURS
GOTO    DONEDISPHERS                ; skip the TENHOURDISP function

TENHOURDISP
MOVLW  d' 10'                      ; set the W register to 10
SUBWF  HOURS, 0                    ; subtract 10 from the HOURS
MOVWF  PORTC                        ; move the result to PORTC.
; This sets the low 4 bits to the correct value

BSF    PORTC, 4                    ; set the 5th bit to 1 (indicates a 1 on the hours)

DONEDISPHERS

CLRF   TMP2                        ; Setup TMP2
MOVWF  MINUTES                      ; Save MINUTES to TMP
MOVWF  TMP

DISPMINLOOP
MOVLW  d' 10'                      ; subtract 10
SUBWF  TMP, 1

BTFSC  TMP, 7                      ; test high bit
GOTO    DISPMINNOFLW                ; it is 1, went too far

; it isn't 1
INCF   TMP2, 1                      ; increment the high minutes
GOTO   DISPMINLOOP                  ; Loop

DISPMINNOFLW
RLF    TMP2, 1                      ; Rotate TMP2 to the left.
RLF    TMP2, 1
RLF    TMP2, 1
RLF    TMP2, 1

MOVLW  d' 10'                      ; add 10 back
ADDWF  TMP, 1

MOVLW  b' 11110000'                 ; set the low 4 bits to 1.
ANDWF  TMP2, 0                      ; leave the 4 high bits alone.
IORWF  TMP, 1                       ; and that with PORTD. This set the high bits.

; set that as the low minutes
MOVWF  TMP
MOVWF  PORTD
; DONE

BTFS   AP, 0
GOTO   CLEARAP
GOTO   SETAP

SETAP
BSF    PORTE, 0
GOTO   ENDDISPLAYTIME

CLEARAP
BCF    PORTE, 0

ENDDISPLAYTIME
GOTO   LOOP

```

```

;*****
;
; MODE 2
;
;*****

```

```

RUNMODE2
    BTFSS    gFlags, KEYFLAG    ; Test to see if a key was pressed by checking the gFlags register
    GOTO    LOOP                ; If no key was pressed, skip to the end of the interrupt

    MOVLW   d' 10'              ; check to see if A was pressed.
    SUBWF   KEYPRESS, 0
    BTFSC   STATUS, Z
    GOTO    ENDMODE2           ; if it was, save the new data and go back to mode 1

    MOVLW   d' 15'              ; check to see if # was pressed.
    SUBWF   KEYPRESS, 0
    BTFSC   STATUS, Z
    GOTO    APMODE2           ; if it was, change the AM/PM setting

; otherwise, rotate the data on PORTA and save the new input to the least digit
    RLF     PORTC, 1            ; rotate the last digit off.
    RLF     PORTC, 1
    RLF     PORTC, 1
    RLF     PORTC, 1

    MOVFW   PORTD              ; save the high digit on port d to the tmp variable
    MOVWF   TMP

    RRF     TMP, 1             ; rotate the hight digit in tmp to the low digit
    RRF     TMP, 1
    RRF     TMP, 1
    RRF     TMP, 1

    MOVLW   b' 00001111'      ; clear the now high digit in the tmp variable.
    ANDWF   TMP, 1

    MOVLW   b' 11110000'      ; clear the low digits on port C
    ANDWF   PORTC, 1

    MOVFW   TMP                ; move the tmp var to W
    IORWF   PORTC, 1          ; OR the tmp var and PORTC. The next digit is now in place.

    RLF     PORTD, 1          ; rotate the low digit to the 2nd digit
    RLF     PORTD, 1
    RLF     PORTD, 1
    RLF     PORTD, 1

    MOVLW   b' 11110000'      ; clear the now low digit
    ANDWF   PORTD, 1

    MOVLW   b' 00001111'      ; clear the high part of the key press
    ANDWF   KEYPRESS, 1

    MOVFW   KEYPRESS          ; move the key to W
    IORWF   PORTD, 1          ; OR the tmp var and PORTD.
                                ; Sets the new key onto the low digit of D.

    BCF     gFlags, KEYFLAG    ; clear the key press flag
    GOTO    LOOP

APMODE2
    BTFSS   PORTE, 0
    GOTO   SETAPM2
    GOTO   CLRAPM2

SETAPM2
    BSF     PORTE, 0

    BCF     gFlags, KEYFLAG    ; clear the key press flag
    GOTO    LOOP

CLRAPM2
    BCF     PORTE, 0

    BCF     gFlags, KEYFLAG    ; clear the key press flag
    GOTO    LOOP

ENDMODE2
; hours on PORTC
    MOVFW   PORTC              ; save port c to W
    MOVWF   TMP                ; save W to TMP

    MOVLW   b' 00001111'      ; clear the high 4 bits
    ANDWF   TMP, 1

    MOVFW   TMP                ; save TMP to hours. This sets HOURS to the ones place.
    MOVWF   HOURS

    MOVLW   d' 10'            ; set W to 10
    BTFSC   PORTC, 4          ; see if the 1 is set in the high digits place.
    ADDWF   HOURS, 1          ; if it is, add 10 to HOURS

; minutes on PORTD
    MOVFW   PORTD              ; save PORTD to W
    MOVWF   TMP                ; save W to TMP

    MOVLW   b' 00001111'      ; clear the high 4 bits
    ANDWF   TMP, 1

    MOVFW   TMP                ; save TMP to minutes. This sets MINUTES to the ones place.
    MOVWF   MINUTES

    MOVFW   PORTD              ; save PORTD to W
    MOVWF   TMP                ; save W to TMP

    RRF     TMP, 1             ; rotate TMP to get the ten minutes in the right spot
    RRF     TMP, 1
    RRF     TMP, 1

```

```

RRF      TMP, 1
MOV LW  b'00001111'
ANDWF   TMP, 1      ; clear the high bits in TMP
INCF    TMP, 1      ; add one to TMP
M2LOOPMIN
DECFSZ  TMP, 1      ; decrement TMP. Skip if it is zero
GOTO    M21NCMIN
GOTO    M2CHECKAP
M21NCMIN
MOV LW  d'10'
ADDWF   MI NUTES, 1 ; set W to 10
GOTO    M2LOOPMIN
M2CHECKAP
BTFS    PORTE, 0    ; see if the AM bit is set
GOTO    SETAPEM2    ; if it is set, go to the set AP part
GOTO    CLEARAPEM2  ; otherwise, go to the clr AP part
SETAPEM2
BSF     AP, 0
GOTO    ENDENDMODE2 ; set the AP bit.
CLEARAPEM2
BCF     AP, 0      ; clear the AP bit.
ENDENDMODE2
CLRF    SECONDS    ; clear SECONDS
GOTO    CHANGEMODE1 ; got back to mode 1
;*****
;
; MODE 3
;
;*****
RUNMODE3
BTFS    gFlags, SHOWTIMER ; see if we should display the timer time
GOTO    DISPLAYTIMER
GOTO    STARTMODE3
DISPLAYTIMER
; minutes on port D
; hours on port C
MOV LW  d'12'
SUBWF   TIMERHOURS, 0    ; set the W register to 12
BTFS    STATUS, Z        ; subtract 12 from the Hours
GOTO    TENHOURDISPTIMER ; If it is 12, go to TENHOURDISPTIMER
MOV LW  d'11'
SUBWF   TIMERHOURS, 0    ; set the W register to 11
BTFS    STATUS, Z        ; subtract 11 from the Hours
GOTO    TENHOURDISPTIMER ; If it is 11, go to TENHOURDISPTIMER
MOV LW  d'10'
SUBWF   TIMERHOURS, 0    ; set the W register to 10
BTFS    STATUS, Z        ; subtract 10 from the Hours
GOTO    TENHOURDISPTIMER ; If it is 10, go to TENHOURDISPTIMER
ONEHOURDISPTIMER
MOVWF   TIMERHOURS      ; move the current HOURS into the W register
MOVWF   PORTC           ; set PORTC to the current HOURS
GOTO    DONEDISPHRSTIMER ; skip the TENHOURDISPTIMER function
TENHOURDISPTIMER
MOV LW  d'10'
SUBWF   TIMERHOURS, 0    ; set the W register to 10
MOVWF   PORTC           ; subtract 10 from the HOURS
MOVWF   PORTC           ; move the result to PORTC.
; This sets the low 4 bits to the correct value
BSF     PORTC, 4        ; set the 5th bit to 1 (indicates a 1 on the hours)
DONEDISPHRSTIMER
CLRF    TMP2            ; Setup TMP2
MOVWF   TIMERMINUTES    ; Save MINUTES to TMP
MOVWF   TMP
DISPMINLOOPTIMER
MOV LW  d'10'
SUBWF   TMP, 1          ; subtract 10
BTFS    TMP, 7
GOTO    DISPMINNOFLWTIMER ; it is 1, went too far
; it isn't 1
INCF    TMP2, 1
GOTO    DISPMINLOOPTIMER ; Loop ; increment the high minutes
DISPMINNOFLWTIMER
RLF     TMP2, 1
RLF     TMP2, 1
RLF     TMP2, 1
RLF     TMP2, 1
MOV LW  d'10'
ADDWF   TMP, 1          ; add 10 back
MOV LW  b'11110000'
ANDWF   TMP2, 0
IORWF   TMP, 1          ; set the low 4 bits to 1.
; leave the 4 high bits alone.
; and that with PORTD. This set the high bits.
; set that as the low minutes
MOVWF   TMP

```

```

MOVWF PORTD
; DONE

BTFSS TIMERAP, 0
GOTO CLEARAPTIMER
GOTO SETAPTIMER

SETAPTIMER
BSF PORTE, 0
GOTO ENDDISPLAYTIMERIME

CLEARAPTIMER
BCF PORTE, 0

ENDDISPLAYTIMERIME
BCF gFlags, SHOWTIMER ; clear the flag that tells us to display the timer time
; the first time this is called.

STARTMODE3

BTFSS gFlags, KEYFLAG ; Test to see if a key was pressed by checking the gFlags register
GOTO LOOP ; If no key was pressed, skip to the end of the interrupt

MOVLW d' 11' ; check to see if B was pressed.
SUBWF KEYPRESS, 0
BTFSC STATUS, Z
GOTO ENDMODE3 ; if it was, save the new data and go back to mode 1

MOVLW d' 15' ; check to see if # was pressed.
SUBWF KEYPRESS, 0
BTFSC STATUS, Z
GOTO APMODE3 ; if it was, change the AM/PM setting

; otherwise, rotate the data on PORTA and save the new input to the least digit

RLF PORTC, 1 ; rotate the last digit off.
RLF PORTC, 1
RLF PORTC, 1
RLF PORTC, 1

MOVWF PORTD ; save the high digit on port d to the tmp variable
MOVWF TMP

RRF TMP, 1 ; rotate the high digit in tmp to the low digit
RRF TMP, 1
RRF TMP, 1
RRF TMP, 1

MOVLW b' 00001111' ; clear the now high digit in the tmp variable.
ANDWF TMP, 1

MOVLW b' 11110000' ; clear the low digits on port C
ANDWF PORTC, 1

MOVWF TMP ; move the tmp var to W
IORWF PORTC, 1 ; OR the tmp var and PORTC. The next digit is now in place.

RLF PORTD, 1 ; rotate the low digit to the 2nd digit
RLF PORTD, 1
RLF PORTD, 1
RLF PORTD, 1

MOVLW b' 11110000' ; clear the now low digit
ANDWF PORTD, 1

MOVLW b' 00001111' ; clear the high part of the key press
ANDWF KEYPRESS, 1

MOVWF KEYPRESS ; move the key to W
IORWF PORTD, 1 ; OR the tmp var and PORTD. Sets the new key onto the low digit of D.

BCF gFlags, KEYFLAG ; clear the key press flag
GOTO LOOP

APMODE3
BTFSS PORTE, 0
GOTO SETAPM3
GOTO CLRAPM3

SETAPM3
BSF PORTE, 0
BCF gFlags, KEYFLAG ; clear the key press flag
GOTO LOOP

CLRAPM3
BCF PORTE, 0
BCF gFlags, KEYFLAG ; clear the key press flag
GOTO LOOP

ENDMODE3
; hours on PORTC

MOVWF PORTC ; save port c to W
MOVWF TMP ; save W to TMP

MOVLW b' 00001111' ; clear the high 4 bits
ANDWF TMP, 1

MOVWF TMP ; save TMP to hours. This sets HOURS to the ones place.
MOVWF TIMERHOURS

MOVLW d' 10' ; set W to 10
BTFSC PORTC, 4 ; see if the 1 is set in the high digits place.
ADDWF TIMERHOURS, 1 ; if it is, add 10 to HOURS

```

```

; minutes on PORTD
MOVFW PORTD ; save PORTD to W
MOVWF TMP ; save W to TMP

MOVLW b'00001111'
ANDWF TMP,1 ; clear the high 4 bits

MOVWF TMP
MOVWF TIMERMINTUTES ; save TMP to minutes. This sets MINUTES to the ones place.

MOVWF PORTD ; save PORTD to W
MOVWF TMP ; save W to TMP

RRF TMP,1 ; rotate TMP to get the ten minutes in the right spot
RRF TMP,1
RRF TMP,1

MOVLW b'00001111'
ANDWF TMP,1 ; clear the high bits in TMP

INCF TMP,1 ; add one to TMP

M3LOOPMIN
DECFSZ TMP,1 ; decrement TMP. Skip if it is zero
GOTO M3NCMIN
GOTO M3CHECKAP

M3NCMIN
MOVLW d'10' ; set W to 10
ADDWF TIMERMINTUTES,1
GOTO M3LOOPMIN

M3CHECKAP
BTFSC PORTE,0 ; see if the AM bit is set
GOTO SETAPEM3 ; if it is set, go to the set AP part
GOTO CLEARAPEM3 ; otherwise, go to the clr AP part

SETAPEM3
BSF TIMERAP,0 ; set the AP bit.
GOTO ENDENDMODE3

CLEARAPEM3
BCF TIMERAP,0 ; clear the AP bit.

ENDENDMODE3
BCF gFlags, KEYFLAG ; clear the key press flag
MOVLW b'11000111' ; set the mode 1 flag and clear the other mode flags
ANDWF gFlags,1
MOVLW b'00000100'
IORWF gFlags,1

MOVLW b'11111110' ; indicate the current mode on PORTA
MOVWF PORTA
GOTO DISPLAYTIME ; refresh the current time and then go back to mode 1

;*****
; MODE 4
;*****

RUNMODE4
MOVLW b'11110111' ; indicate the current mode on PORTA
MOVWF PORTA

DECFSZ TMP3,1
GOTO M4NEXT

MOVLW d'20' ; reset the counter
MOVWF TMP3

BTFSS PORTE,1 ; test to see if the alarm is set
GOTO M4SET1 ; if not, set it
GOTO M4CLRE1 ; otherwise, clear it

M4SET1
BSF PORTE,1 ; set the alarm bit
GOTO M4NEXT

M4CLRE1
BCF PORTE,1 ; clear the alarm bit

M4NEXT
BTFSS gFlags, KEYFLAG ; Test to see if a keypress by checking the gFlags register
GOTO DISPLAYTIME ; If no key was pressed, skip to the end of the interrupt

MOVLW d'13' ; check to see if D was pressed.
SUBWF KEYPRESS,0
BTFSC STATUS,Z
GOTO ENDMODE4 ; if it was, turn off the alarm

BCF gFlags, KEYFLAG ; clear the key press flag
GOTO DISPLAYTIME ; return to the top

ENDMODE4
BCF gFlags, NEXTMIN ; make sure the alarm doesn't get stuck after D is hit.
BCF PORTE,1
GOTO CHANGEMODE1 ; alarm has been disabled. go back to normal running mode

;*****
; Change to MODE 1
;*****

```

```

CHANGEMODE1
    BCF     gFlags, KEYFLAG      ; clear the key press flag
    MOVLW  b'11000111'         ; set the mode 1 flag and clear the other mode flags
    ANDWF  gFlags, 1
    MOVLW  b'00000100'
    IORWF  gFlags, 1

    MOVLW  b'11111110'         ; indicate the current mode on PORTA
    MOVWF  PORTA

    GOTO   RUNMODE1           ; change modes to mode 1

```

```

*****
: Change to MODE 2
:
*****

```

```

CHANGEMODE2
    BCF     gFlags, KEYFLAG      ; clear the key press flag
    MOVLW  b'11001011'         ; set the mode 2 flag and clear the other mode flags
    ANDWF  gFlags, 1
    MOVLW  b'00001000'
    IORWF  gFlags, 1

    MOVLW  b'11111101'         ; indicate the current mode on PORTA
    MOVWF  PORTA

    GOTO   RUNMODE2           ; change modes to mode 2

```

```

*****
: Change to MODE 3
:
*****

```

```

CHANGEMODE3
    BCF     gFlags, KEYFLAG      ; clear the key press flag
    MOVLW  b'11010011'         ; set the mode 3 flag and clear the other mode flags
    ANDWF  gFlags, 1
    MOVLW  b'00010000'
    IORWF  gFlags, 1

    BSF    gFlags, SHOWTIMER    ; set the flag that tells us to display the timer time the
                                ; first time this is called.

    MOVLW  b'11111011'         ; indicate the current mode on PORTA
    MOVWF  PORTA

    GOTO   RUNMODE3           ; change modes to mode 3

```

```

*****
: Change to MODE 4
:
*****

```

```

CHANGEMODE4
    BCF     gFlags, KEYFLAG      ; clear the key press flag
    MOVLW  b'11100011'         ; set the mode 4 flag and clear the other mode flags
    ANDWF  gFlags, 1
    MOVLW  b'00100000'
    IORWF  gFlags, 1

    MOVLW  b'11110111'         ; indicate the current mode on PORTA
    MOVWF  PORTA

    MOVLW  d'20'               ; preload the counter
    MOVWF  TMP3

    GOTO   RUNMODE4           ; change modes to mode 4

```

```

*****
: End of all the code
:
*****

```

END