2017

Computer Engineering Design 2 Phase 3 Report

UNIVERSITY OF KWAZULU-NATAL ™

INYUVESI YAKWAZULU-NATALI

Keshav Jeewanlall

213508238

10/26/2017

Abstract

This report entails the design, development and implementation of Project 5 in Computer Engineering Design 2. The Project chosen is a digital hexadecimal converter. This report includes well detailed explanations and in-depth discussions on the project.

The digital hexadecimal converter uses a microcontroller to control the system. The hexadecimal converter counts from 0 to 15 and displays both the four-digit binary code and the two-digit hexadecimal number using six multiplexed seven-segment displays. The hexadecimal converter can run in two modes i.e. “Count Up” mode where it counts up from 0000 (0) 0 to 1111 (F); and “Count Down” mode where the it counts down from 1111 (F) to 0000 (0). An audio alarm is sounded at certain intervals as the hexadecimal converter counts. The alarm sounds for two seconds when the count is between 0001 (1) and 0010 (2), the alarm sounds for five seconds when the count is 0011 (3) and for ten seconds when the count is 1111 (F). When the hexadecimal converter is not in one of its stop modes, it displays the current time of day. The time of day is controlled using a Real-Time Clock (RTC). The time is displayed on the six multiplexed Seven-Segment Displays (SSDs), showing the hours, minutes and seconds.

The system was designed and debugged on MPLAB X IDE. The system was implemented using In-Circuit Serial Programming (ICSP) and a PicKit 3 was used to program the PIC16F690 microcontroller used in this project.

Contents

Digital Hexadecimal Converter

1. Introduction

The objective of this project is to design and build a microcontroller based Digital hexadecimal converter which counts from 0 to 15 showing the 4-digit binary code and the 2-digit hexadecimal number. The hexadecimal converter must also have start, stop and reset buttons as well as an audio alarm which sounds when the count is at the value; 1,2,3 and 15. Furthermore, a Real-Time Clock (RTC) must be interfaced with the microcontroller to display the current time of day when the digital converter is not in one of its stop modes. The microcontroller used for this project is the PIC16F690. This report contains complete details of the design and development of the Digital Hexadecimal converter as well as all source codes.

2. Specifications

- This Design must be a PIC based digital binary converter that can be used to display two digit numbers.

- The display must be implemented using six SSDs, four for the 4-digit binary code and 2 for the two-digit number.

  Example: 0010 02

- The converter should have 2 modes:
  - a) Up-count: 0 to 15. (i.e. 0000, 0001, to 1111)
  - b) Down-count: 15 to 0. (i.e. 1111, 110, to 0000)

- A switch is to be used to select between the two modes.

- Push buttons must be used to implement the following:
  - a) To start and stop the displaying of the binary code and hexadecimal number.
  - b) To reset the user set binary code (to 0000).

- A buzzer should be used to beep 5 seconds when the binary code is 0011, every 2 seconds when binary code between 0001 to 0010 and within 10 seconds when the binary code is 1111

- Furthermore, when the digital Hexadecimal converter is not in one of the stop modes, it should display the time of the day in the format (HHMMSS) using an external Real-Time Clock (RTC), which is to be interfaced with the microcontroller.

3. Design and configuration of Components

3.1. Components Used

- PIC16F690 microcontroller
- 74HC514 Decoder/Demultiplexer
- 6 x Common anode seven-segment displays
- 7 x Current limiting resistors (220Ω)
- 2 x Pull up resistors (220Ω)
- 2x SPST Push button switches
- 1x Toggle switch
- 1x Piezo buzzer
- 1x DS1305 Real-time clock
- 1x 3V lithium battery
- 1x 32.768kHz crystal Oscillator
- 6x BC237 N-P-N transistors
- 6x 1KΩ resistors

3.2. The Display

Due to the limited number of pins on the PIC16F690, the six SSDs need to be multiplexed to display the data. Pin a-g of each of the SSDs will be connected to PORTC<0:6> respectively. The enable pin of each SSD will be connected separately because two SSDs cannot both be enabled at any given time.

To multiplex six SSDs, six pins are needed to enable each SSD. Due to the limited number of pins on the microcontroller; a 74HC514 decoder is used for the enabling of each SSD during multiplexing.
The decoder receives 3 inputs from the microcontroller and depending on the input, produces six outputs with only one of the outputs being high at a time. Therefore, only three pins from the microcontroller will be connected to the decoder gate circuit and each output of the decoder will be connected to a SSD.

Since all six SSDs is being multiplexed from one port of the microcontroller, N-P-N transistors are used to increase brightness of each SSD.

3.3. Switch and Button Setup

The push button switches were set up as pull up switches. When the switch is pressed, the microcontroller will receive a low signal. The “start/stop” button is connected to pin RA2 (PORTA<2>) which is also an interrupt pin. The reason for choosing pin RA2 is because the start/stop button is setup as an external interrupt.

The “reset” button is connected to pin RA3 (PORTA<3>), which is also the MCLR (Master clear reset) pin. This pin was chosen due to the lack of pins on the microcontroller, and since this pin can only be used as an input it made sense to connect a button to it. The MCLR pin is an active low pin therefore when a logic 0 is applied to it, it resets the microcontroller hence the reset button is connected to this pin.

A toggle switch is used to determine whether the stopwatch is in “count up” or “count down” mode. When the switch is opened, a low signal is sent. When the switch is closed, a high signal is sent. This switch is connected to pin RA0 (PORTA<0>). When RA0 is low, the digital hexadecimal converter will be in “count up” mode, when RA0 is high, the digital hexadecimal converter will be in “count down” mode.

3.4. The Real-time Clock (RTC)

The RTC and the microcontroller communicate using the Synchronous Serial Port (SSP) module of the microcontroller. The microcontroller is configured as the Master device of the communication bus as it controls all actions and the RTC is configured as the Slave. In the PIC16F690, there are two communication methods for this module; Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I^2C).

In the PIC16F690, I^2C mode only supports Master mode in firmware. This means all Start and Stop bits have to be physically implemented in the source code. All attempts for this means of communication have resulted in failure thus SPI communication was used instead.

In SPI mode, the microcontroller enables a slave, then reads and writes data synchronously between itself and the slave. With each byte of data transmitted, the MSB of the data is shifted out first, followed by the rest with one bit per clock pulse.

The RTC chosen for this project is the DS1305 Serial Alarm Real-Time Clock. The reason for this choice is that it supports SPI communication. The DS1305 provides a full clock calendar valid up to the year 2100 and includes leap year compensation. The clock can operate in 12- hour mode or 24-hour mode (in this project, 24-hour mode is used). All registers are in Binary Coded Decimal (BCD) format. The device is also flexible when it comes to clock polarity as it samples the clock pulse when enabled thus allowing for the clock polarity to be determined by the RTC itself.

3.5. RTC Pinout

The DS1305 can be backed up by a non-rechargeable 3V lithium battery. The backup battery is used to increment the counters when the system is turned off. A 32.768 kHz crystal is connected to the RTC to drive the oscillator. The load capacitance of the crystal should be 6pF for accurate timing. The CE (Chip Enable) pin is used to initiate and terminate data transfer. Pin RA5 is used to enable the RTC in this project.

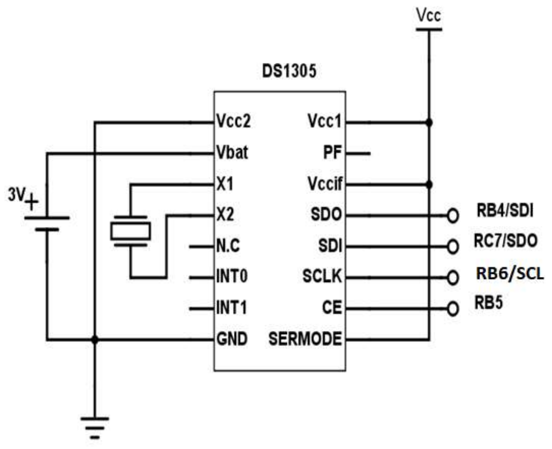
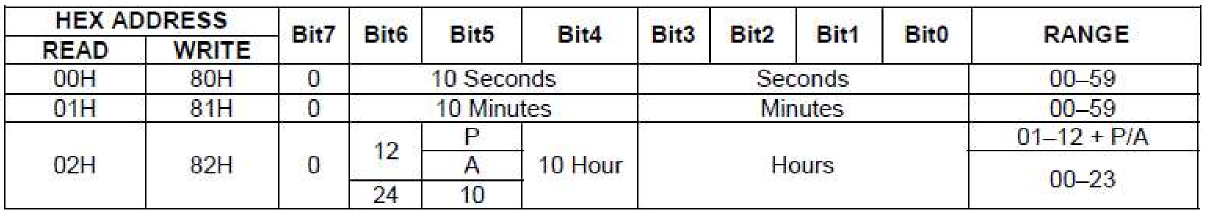


Figure 1: Pinout of DS1305 RTC to PIC16F690

3.6. RTC Registers

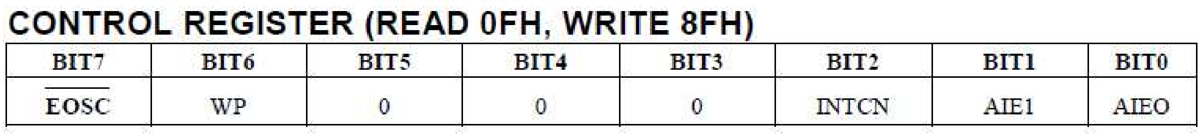
The DS1305 contains two different addresses for each register; one address for reading and the other for writing. All data in the registers are in BCD format. Bit 6 in the Hours register determines whether the clock is in 12-hourmode or 24-hour mode. When low, the RTC is in 24-hour mode. When in 24-hour mode, bit 5 and 4 represent the ‘tens’ of the hour; for example, if the register contained 00100001, this shows that the hour is 21. Another example; if the register contained 00011001, this shows that the hour is 19.



HEX ADDRESS		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	RANGE
READ	WRITE									
00H	80H	0	10 Seconds			Seconds				00–59
01H	81H	0	10 Minutes			Minutes				00–59
02H	82H	0	12	P	10 Hour	Hours				01–12 + P/A
				A						00–23
			24	10						

Figure 2: RTC Registers

On initial start-up of the DS1305, the oscillator is disabled and the “write protection” feature is enabled. The write protection feature prevents unwanted signals from communicating with the RTC and changing the data in its registers. When initializing the RTC, bit 7 ($\overline{\text{EOSC}}$) needs to be cleared in order to enable the oscillator. Bit 6 also needs to be cleared in order to disable the write protection.



CONTROL REGISTER (READ 0FH, WRITE 8FH)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
$\overline{\text{EOSC}}$	WP	0	0	0	INTCN	AIE1	AIEO

Figure 3: RTC Control Register

3.7. Writing to the RTC

When the CE (Chip enable) pin is driven high, an address byte should immediately follow, followed by the data that needs to be transmitted/received.

With the PIC16F690 in Master mode, data transmission/reception is initiated by writing to the SSPBUF register. When the transmission/reception is complete, the BF bit in the SSPSTAT register will set to high. This bit will be polled in order to wait for data communication to complete.

To perform a single byte write to the RTC, the following steps are followed:

- Enable the RTC by sending a high to the CE pin of the RTC.
- Write the address (of the register to be written to in the RTC) to the SSPBUF register.
- Wait for data to transmit.
- Write the required data to the SSPBUF register.
- Wait for data to transmit.
- Disable the RTC by sending a low to the CE pin of the RTC

3.8. Reading from the RTC

To perform a single byte read of the RTC, the following steps are followed:

- Enable the RTC by sending a high to the CE pin of the RTC.
- Write the address (of the register to read from the RTC) to the SSPBUF register.
- Wait for data to transmit.
- Write the 'dummy' data to the SSPBUF register. While this data is shifted out, the RTC will send the required data (the data of the register that was previously addressed) which will then be stored in the SSPBUF register.
- Wait for data to transmit.
- Read the SSPBUF register.
- Disable the RTC by sending a low to the CE pin of the RTC

Refer to Appendix A for the Circuit Schematic

4. Code Structure and Implementation

14 general file registers are used in this project, namely:

• tempW	: Used for contents saving when interrupt is triggered
• current_sec	: Stores the seconds that's read from the RTC
• current_min	: Stores the minutes that's read from the RTC
• current_hr	: Stores the hours that's read from the RTC
• tens	: Stores the tens value that's displayed on the SSDs
• units	: Stores the units' value that's displayed on the SSDs
• temp	: Temporary register used various purposes
• control_count	: used for controlling the starting and stopping of the count
• alarm_value1	: Contains the 1st count value for the alarm to sound
• alarm_value2	: Contains the 2nd count value for the alarm to sound
• alarm_value3	: Contains the 3rd count value for the alarm to sound
• alarm_value4	: Contains the 4th count value for the alarm to sound
• count	: Used to store the count value
• disp_freq	: used as a delay variable to delay the display

control_count, controls which mode the digital hexadecimal converter is in and indicates when the.

7	6	5	4	3	2	1	0
-	-	-	-	-	DOWN	UP	24Hr

Figure 4: control_count register

DOWN:	1 – Count Down mode enabled 0 – Count Down mode disabled
UP:	1 – Count Up mode enabled 0 – Count Up mode disabled
24Hr:	1 – Do not display the time of day 0 – Display the time of day

4.1. Setup Phase

In the setup phase, all ports and general file registers are cleared thus initializing them.

- BANK 0
    - PORTA, PORTB, PORTC is cleared to initialize the ports
    - PORTA<0>, PORTA<2> and PORTA<3> is set. These ports are used for button inputs.
    - 00100000 is loaded into SSPCON. This sets the microcontroller to SPI Master Mode at a speed of Fosc/4. Enables all serial ports and sets the idle state of the clock pulse at a low.
- BANK 1
    - Set the necessary ports as inputs and outputs.
    - 11000000 is loaded into SSPSTAT. Input data sampled at end of clock pulse for SPI mode. Data transmitted on rising edge of clock pulse.
    - IRCF<2:0> of OSCCON = 111. Sets the oscillator to run at 8MHz.
    - OPTION_REG<6> is set to allow interrupt on RA2 to occur on falling edge of signal.
    - The Global interrupts and external interrupt services are enabled by setting INTCON<7> and INTCON<4>
- Bank 2
    - ANSEL and ANSELH is cleared to configure all ports as digital I/O.

4.2. Display Loop

The subroutine Display_Loop is an endless loop that is mainly used for checking for button presses and to determine whether to display the binary to hexadecimal conversion or the current time.

- Reads the RTC's hours, minutes and seconds registers and stores them in the appropriate file registers (at this moment the data is in BCD format).
- If the "24Hr mode" is enabled, then it displays the current time of day.
- If the "24Hr mode" is disabled, then it displays the binary to hexadecimal conversion
- If the start button is pressed, the subroutine Ctrl_Start_Btn is called.
- If the reset button is pressed, the subroutine Ctrl_Reset_Btn is called.
- If Count up mode is enabled, call Count_Up subroutine.
- If Count down mode is enabled, call Count_Down subroutine.

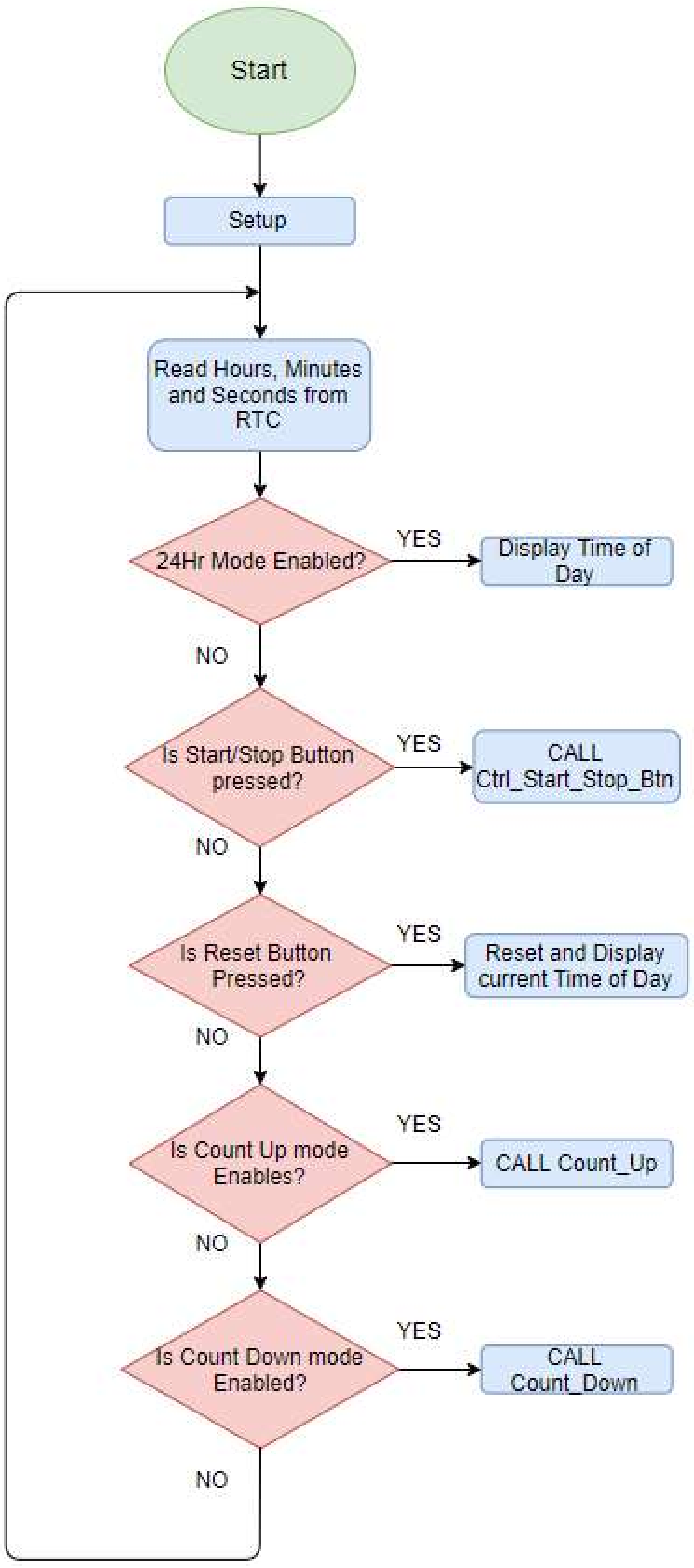

Start
Setup
Read Hours, Minutes and Seconds from RTC
24Hr Mode Enabled?
YES
Display Time of Day
NO
Is Start/Stop Button pressed?
YES
CALL Ctrl_Start_Stop_Btn
NO
Is Reset Button Pressed?
YES
Reset and Display current Time of Day
NO
Is Count Up mode Enables?
YES
CALL Count_Up
NO
Is Count Down mode Enabled?
YES
CALL Count_Down
NO

Figure 5:Display_Loop Flow chart

4.4. Reset Button

This button is connected to Pin RA3 which is the MCLR pin. Therefore, when this button is pressed, the microcontroller resets, hence no actual coding of this button was necessary

4.3. Start/Stop Button

The subroutine Ctrl_Start_Stop_Btn is used to start or stop the hexadecimal converter. With each press of the start button, the 24Hr mode is disabled and the hexadecimal converter is toggled between starting and stopping. When a start is initiated, it checks if whether the hexadecimal converter should be in Count up or Count down mode and then starts the appropriate count mode. This is done by checking if the toggle switch on pin RA0 is low or high. If low, start or stop a count up sequence, which is done by branching to subroutine Start_Up_Count. If high start or stop a down count sequence which is done by branching to subroutine Start_Down_Count.

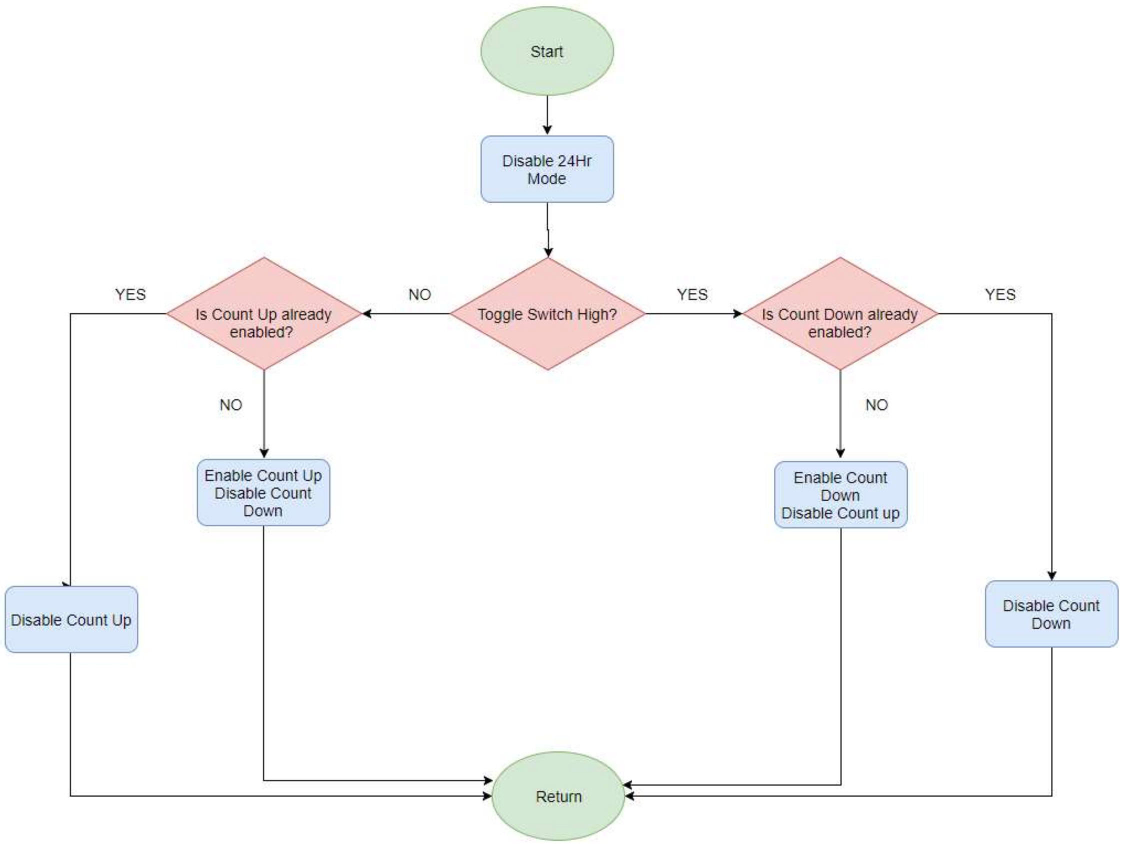
4.4. Start Up Count

- Checks if Count up mode is already active, if it is, it means that the hexadecimal converter should stop therefore disable Count up mode.
- Otherwise, Enable Count up mode.
- Disable Count down mode.

4.5. Start Down Count

- Checks if Count down mode is already active, if it is, it means that the hexadecimal converter should stop, therefore disable Count down mode.
- Otherwise, enable count down mode.
- Disable count up mode.

All checks are done by testing bits 1 and 2 of the control_count register. If bit 1 is set, then count up is enabled. If bit 2 is set, then count down mode is enabled



Start
Disable 24Hr Mode
Toggle Switch High?
NO
YES
Is Count Up already enabled?
YES
NO
Enable Count Up Disable Count Down
Disable Count Up
Is Count Down already enabled?
YES
NO
Enable Count Down Disable Count up
Disable Count Down
Return

Figure 6: Start/Stop Flowchart

4.6. Displaying the Counting and Converting to Hexadecimal

- A toggle switch is used to change between up or down counting mode.
- The count increased or decreased every second.
- If counting up, the subroutine Count_up increases the count register by 1. After each increase, bit 5 of the count register is checked. If it is set this means that the count is beyond 15 and should reset to zero.
- If counting down, the subroutine Count_down decreases the count register by 1. After each decrease, bit 7 of the count register is checked. If it is set this means that the count is below 0 and should reset to 15
- Each of the first 4 LSB of the count register is tested to determine if it is a 1 or 0.
- The respective 1 or 0 is displayed on each corresponding SSD.
- The number is displayed as a hexadecimal by obtaining the corresponding value from the lookup table, SSD_HEX_Table

4.7. Sounding the Alarm

- The subroutine Alarm takes the value in registers; alarm_value1, alarm_value2, alarm_value3 and alarm_value4, which contains values 1, 2, 3, 15 respectively.

- For the values of 1 and 2, it then calls the subroutine Check_sound_2sec, which checks if the value loaded when subtracted from the current count value results in a zero, if so sound the buzzer for 2 seconds.

- For the value of 3, it then calls the subroutine Check_sound_5sec, which checks if the value loaded when subtracted from the current count value results in a zero, if so sound the buzzer for 5 seconds

- For the value 15, it then calls the subroutine Check_sound_10sec, which checks if the value loaded when subtracted from the current count value results in a zero, if so sound the buzzer for 10 seconds

The subroutine Alarm is called in both Count_Up and Count_Down subroutines to sound the alarm when necessary during a change in value of the count register.

4.8. Setting the Time on the RTC and Reading the Current Time

Initial write to the RTC was 00000000_2 to its control register (address $8F_{16}$) to initiate the RTC.

The time of day when initiating the RTC was 23:45, thus to set the current time, 01000101 (45 in BCD) was written to its minutes' register (address 81_{16}) and 00100011 (23 in BCD) was written to its hours' register (address 82_{16}). Bit 6 in the hours' register must be maintained clear in order for the RTC to operate in 24 Hr mode.

To read and display the current time, the values stored in the Hours, Minutes and Seconds registers of the RTC are required. The microcontroller sends the address of the required register, the value in the corresponding register is then stored in the GPRs (current_Hr, current_Min and current_sec) of the microcontroller and is then displayed. Microcontroller sends 'dummy' data while receiving data from RTC.

Refer to Figure. 2 for RTC Registers

Refer to Appendix B for the Source code

5. Code Testing

The MPLAB X simulator was used to ensure all control register were correctly configured. System testing was done on a PIC16F690 microcontroller with all the components connected to it.

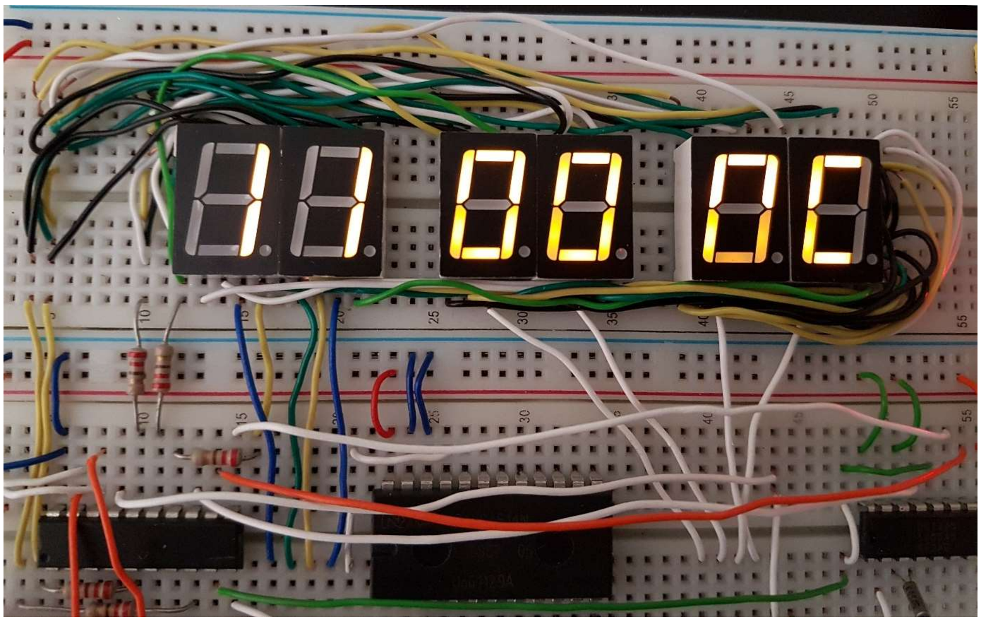


Figure 7: Circuit in count mode showing the 4-Digit binary code and the Hexadecimal Value of 12

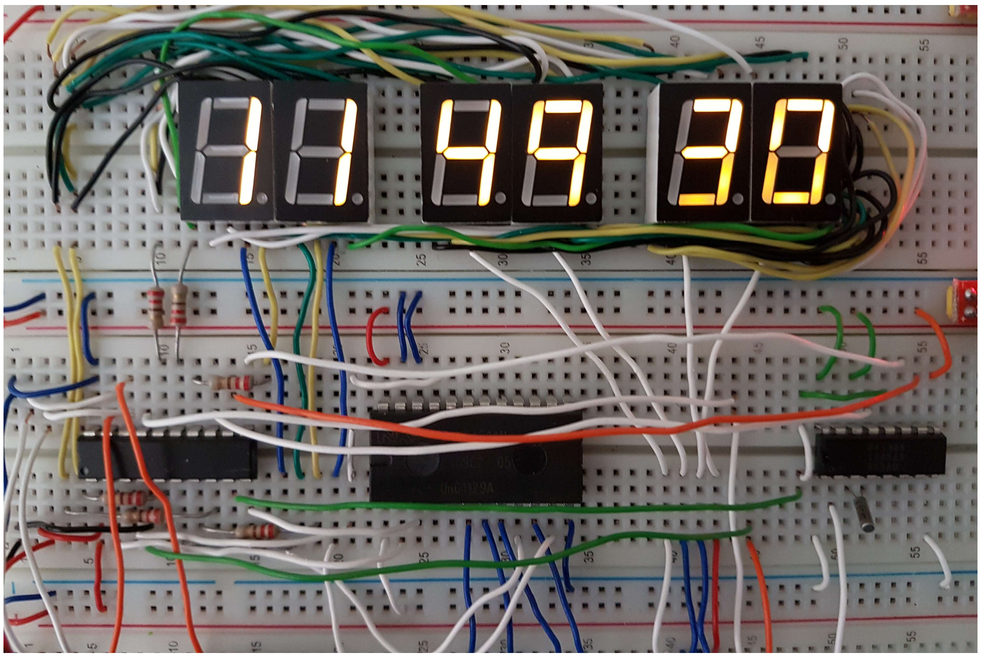


Figure 8: Circuit in 24Hr Mode showing time of day

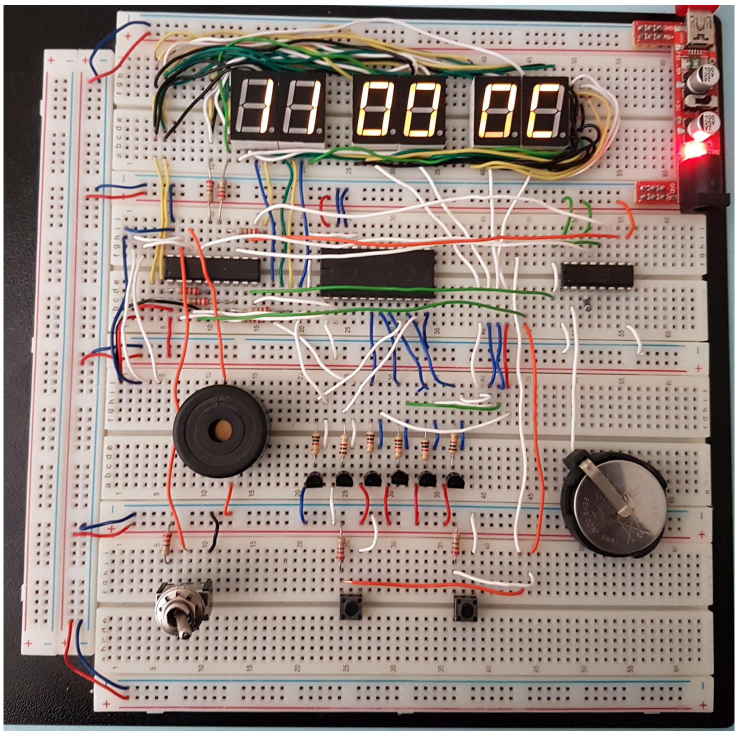


Figure 9: Full Circuit Design

6. Code Statistics

- 14 general file registers were used as well as the W register.
- The project consists of 463 lines of code excluding labels and comments.

7. Conclusion

The contents of this report convey the solutions and details regarding the design and development of this project. Many different design solutions and source codes have been drafted out until an efficient solution which met all requirements was designed. Lessons leant during this course include the importance of modular coding and how advantageous it can be. Working with microcontrollers has given a broader understanding of memory usage and efficiency in coding. Efficient code is more desirable as memory is limited in these devices. The system has been put through numerous test cases and all were a success. Overall, system specifications were met and alternative solutions were explored to determine the most feasible design.

8. References

[1] Picnote.blogspot.co.za. (2017). *6 Digits LED 7-Segment Multiplexing*. [online] Available at: http://picnote.blogspot.co.za/2008/11/6-digit-led-7-segment-multiplexing.html [Accessed 30 Sep. 2017].

[2] Picbasic.co.uk. (2017). *Time display on 7-seg 4 digits*. [online] Available at: http://www.picbasic.co.uk/forum/showthread.php?t=5623&highlight=digit+display [Accessed 30 Sep. 2017].

[3] Digital Clock using PIC Microcontroller and the DS1307 Real Time Clock &#8211; MikroCPosted Bitahwa Bindu on Saturday, T. and Bindu, B. (2017). *Digital Clock using PIC Microcontroller and the DS1307 Real Time Clock – MikroC*. [online] Student Companion SA. Available at: http://www.studentcompanion.co.za/digital-clock-using-pic-microcontroller-and-the-ds1307-real-time-clock-mikroc/ [Accessed 12 Oct. 2017].

[4] YouTube. (2017). *David's Lab: 7 Segment Multiplexing LED Display PIC Assembly Language Tutorial*. [online] Available at: https://www.youtube.com/watch?v=ZWdhEmrdozk [Accessed 12 Oct. 2017].

Appendix A – Circuit Schematic

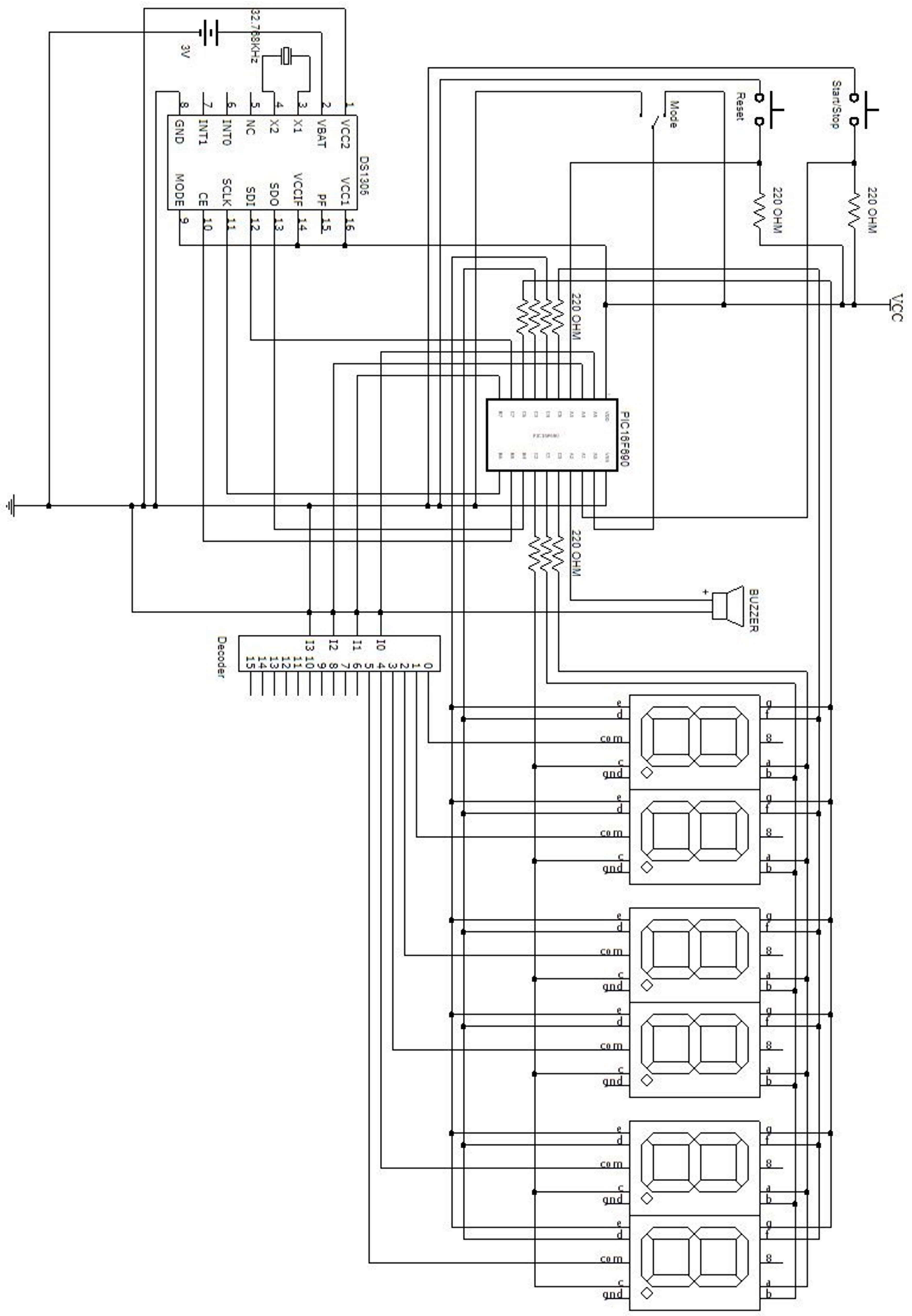


Figure 10: Circuit Schematic

Appendix B – Source Code

```
;*******************************************************************************
;
;    Student Name          : Keshav Jeewanlall
;    Student Number        : 213508238
;    Description           : Project 5 - 5.1. Digital Hexadecimal Converter
;
;*******************************************************************************
    List p=16f690
#include <p16F690.inc>
errorlevel  -302
    __CONFIG    _CP_OFF & _CPD_OFF & _BOR_OFF & _MCLRE_ON & _WDT_OFF & _PWRTE_ON &
_INTRC_OSC_NOCLKOUT & _FCMEN_OFF & _IESO_OFF

    UDATA
temp1              RES 1           ;Temporary register used various purposes
tempW              RES 1           ;Used for contents saving when interrupt is triggered
disp_freq                 RES 1    ;used as a delay variable to delay the display
count                     RES 1    ;Used to store the count value
alarm_value1      RES 1     ;Contains the 1st count value for the alarm to sound
alarm_value2      RES 1     ;Contains the 2nd count value for the alarm to sound
alarm_value3      RES 1     ;Contains the 3rd count value for the alarm to sound
alarm_value4      RES 1     ;Contains the 4th count value for the alarm to sound
current_sec               RES 1    ;Stores the seconds that's read from the RTC
current_min               RES 1    ;Stores the minutes that's read from the RTC
current_hr            RES 1        ;Stores the hours that's read from the RTC
tens                  RES 1        ;Stores the tens value that's displayed on the SSDs
units                 RES 1        ;Stores the units value that's displayed on the SSDs
control_count     RES 1     ;used for controlling the starting and stopping of the count

RESET ORG 0x00              ;Reset vector, PIC starts here on power up and reset
GOTO Setup
 ORG 0x04                   ;The PIC will come here on an interrupt
                            ;This is our interrupt routine that we want
                            ;the PIC to do when it receives an interrupt

;*******************************INTERRUPT ROUTINE*******************************
Count_Up_Interrupt
    MOVWF tempW                     ;temporarily stores value in W register when
                            ;interrupt occurs
    CALL Ctrl_Start_Stop_Btn
    BCF INTCON,1            ;enable the interrupt again
    MOVFW tempW                     ;restore W register value from before the interrupt
    RETFIE                  ;This tells the PIC that the interrupt routine
                            ;has finished and the PC will point back to the
                            ;main program

;***************************SETUP AND CONFIGURATION*****************************
Setup
                        ;select Bank 0
    BCF STATUS,5
    BCF STATUS,6
    CLRF PORTC              ;Initialize PORTC
    CLRF PORTB              ;Initialize PORTB
    CLRF PORTA              ;Initialize PORTA

    BSF PORTA,2                     ;Used as input for start/stop button
    BSF PORTA,3                     ;Used as input for reset button
    BSF PORTA,0                     ;Used as input for switch mode button

    MOVLW b'00100000'
    MOVWF SSPCON            ;Enable serial port pins, idle for clock is low,
                            ;SPI master mode @ Fosc / 4


                        ;Select Bank 1
    BSF STATUS,5

    CLRF TRISC          ;set PORTC as output for ssds
    BCF TRISA,5                 ;Set RA5 as output to decoder used for multiplexing
    BCF TRISA,4                 ;Set RA4 as output to decoder used for multiplexing
    BCF TRISB,7                 ;Set RB7 as output to decoder used for multiplexing
    BCF TRISA,1               ;Set RA1 as output for buzzer
    BSF TRISB,4               ;set RB4/SDI as input
    BCF TRISB,5               ;set RB5 as output to control the RTC enable pin
    BCF TRISB,6               ;set RB6/SCL as output

   MOVLW b'11000000'
   MOVWF SSPSTAT       ;Input data sampled at end of CP,data transmitted on
                       ;rising edge of CP

    BSF OSCCON,6
    BSF OSCCON,5
    BSF OSCCON,4       ;Set Fosc at 8 MHz

    BCF OPTION_REG, 6
    BSF INTCON,7       ;enable Global Interrupt
    BSF INTCON,4       ;enable External Interrupt

                       ;select Bank 2
    BCF STATUS,5
    BSF STATUS,6



    CLRF ANSEL         ;enable digital I/O on ports
    CLRF ANSELH

                       ;select Bank 0
    BCF STATUS,6

                       ;clear GPRs
    CLRF temp1
    CLRF disp_freq
    CLRF count
    CLRF alarm_value1
    CLRF alarm_value2
    CLRF alarm_value3
    CLRF alarm_value4
    CLRF control_count
    CLRF tempW

    GOTO Initialize_Loop

CODE

;**********************************DISPLAY LOOP*********************************
 Initialize_Loop

    BSF PORTB,5
    MOVLW 0x8F
    MOVWF SSPBUF
    CALL RTC_wait
    MOVLW b'00000000'
    MOVWF SSPBUF          ;New data to xmit
    CALL RTC_wait
    BCF PORTB,5

Display_Loop

   CALL Read_sec           ;Reads the seconds register in the RTC
   CALL Read_min           ;Reads the minutes register in the RTC
   CALL Read_hour          ;Reads the hours register in the RTC

   BTFSS control_count,0    ;Is 24Hr mode enabled? If so, display the time of day
   CALL Display_Time
   BTFSC control_count,0    ;Else display binary to hexadecimal conversion
   CALL Display_Hex_Value

                           ;selects count mode

    BTFSC control_count,2   ;If Count down mode enabled, count up
    CALL Count_Down
    BTFSC control_count,1   ;If Count up mode enabled, count up
    CALL Count_Up

    GOTO Display_Loop
```

```
;*****CODE FOR DISPLAYING THE 4-DIGIT BINARY CODE AND THE HEXADECIMAL VALUE*****

Display_Hex_Value
;This subroutine is for displaying the 4-digit binary code and the hexadecimal value

    CALL Turn_off_SSDs
    MOVLW 0x40              ;keeps tens Hex SSD at zero
    CALL Turn_on_SSD_5
    MOVWF PORTC

    CALL Multiplex_Delay    ;delay for multiplexing SSDs
    CALL Update_Hex_Digit   ;update units Hex value SSD
    CALL Multiplex_Delay    ;delay for multiplexing SSDs
    CALL Update_Binary_4    ;Display binary bit 0
    CALL Multiplex_Delay    ;delay for multiplexing SSDs
    CALL Update_Binary_3    ;Display binary bit 1
    CALL Multiplex_Delay    ;delay for multiplexing SSDs
    CALL Update_Binary_2    ;Display binary bit 2
    CALL Multiplex_Delay    ;delay for multiplexing SSDs
    CALL Update_Binary_1    ;Display binary bit 3
    CALL Multiplex_Delay    ;delay for multiplexing SSDs

    INCFSZ disp_freq        ;delay used for displaying count
    GOTO Display_Hex_Value

    RETURN

Update_Hex_Digit
;This subroutine updates and displays the required hexadecimal value on the SSD

    CALL Turn_off_SSDs      ;subroutine to turn off SSDs
    MOVLW 0x0F
    ANDWF count,W           ;Masks last 4 bits
    CALL SSD_HEX_Table      ;get required value to be displayed from lookup table
    CALL Turn_on_SSD_6      ;subroutine to turn on SSD 2
    MOVWF PORTC                     ;Moves value to PORTC
    RETURN

Update_Binary_1
;This subroutine checks if bit 0 of the count register is a 1 or 0 and displays
;the appropriate digit

    CALL Turn_off_SSDs      ;subroutine to turn off SSDs
    MOVLW 0x0F
    ANDWF count,W           ;Masks last 4 bits
    BTFSS count,3           ;Test if bit is 1, if not display 0
    CALL Display_0          ;subroutine to display 0
    BTFSC count,3           ;Test if bit is 0, if not display 1
    CALL Display_1          ;subroutine to display 1
    CALL Turn_on_SSD_1      ;subroutine to turn on SSD 2
    RETURN

Update_Binary_2
;This subroutine checks if bit 1 of the count register is a 1 or 0 and displays
;the appropriate digit

    CALL Turn_off_SSDs      ;subroutine to turn off SSDs
    MOVLW 0x0F
    ANDWF count,W           ;Masks last 4 bits
    BTFSS count,2           ;Test if bit is 1, if not display 0
    CALL Display_0          ;subroutine to display 0
    BTFSC count,2           ;Test if bit is 0, if not display 1
    CALL Display_1          ;subroutine to display 1
    CALL Turn_on_SSD_2      ;subroutine to turn on SSD 2
    RETURN

Update_Binary_3
;This subroutine checks if bit 2 of the count register is a 1 or 0 and displays
;the appropriate digit

    CALL Turn_off_SSDs      ;subroutine to turn off SSDs
    MOVLW 0x0F
    ANDWF count,W           ;Masks last 4 bits
    BTFSS count,1           ;Test if bit is 1, if not display 0
    CALL Display_0          ;subroutine to display 0
    BTFSC count,1           ;Test if bit is 0, if not display 1
    CALL Display_1          ;subroutine to display 1
    CALL Turn_on_SSD_3     ;subroutine to turn on SSD 3
    RETURN

Update_Binary_4
;This subroutine checks if bit 3 of the count register is a 1 or 0 and displays
;the appropriate digit

    CALL Turn_off_SSDs     ;subroutine to turn off SSDs
    MOVLW 0x0F
    ANDWF count,W          ;Masks last 4 bits
    BTFSS count,0          ;Test if bit is 1, if not display 0
    CALL Display_0         ;subroutine to display 0
    BTFSC count,0          ;Test if bit is 0, if not display 1
    CALL Display_1         ;subroutine to display 1
    CALL Turn_on_SSD_4     ;subroutine to turn on SSD 4
    RETURN

Display_0
;This subroutine displays a 0 on the SSD

    MOVLW 0x40
    MOVWF PORTC
    RETURN

 Display_1
 ;This subroutine displays a 1 on the SSD

    MOVLW 0x79
    MOVWF PORTC
    RETURN

Multiplex_Delay
;This subroutine is used as a delay for multiplexing the SSDs

    MOVLW 0x01
    MOVF temp1
Multiplex_loop
    DECFSZ temp1,1
    GOTO Multiplex_loop
    RETURN

;*************SSD LOOKUP TABLE TO DISPLAY VALUES ON COMMON ANODE SSDs***********

    SSD_HEX_Table
                         ;These HEX values are required because common anode SSDs
                         ;are being used
    ADDWF PCL,F
    RETLW 0x40           ;displays number 0 on SSD
    RETLW 0x79           ;displays number 1 on SSD
    RETLW 0x24           ;displays number 2 on SSD
    RETLW 0x30           ;displays number 3 on SSD
    RETLW 0x19           ;displays number 4 on SSD
    RETLW 0x12           ;displays number 5 on SSD
    RETLW 0x02           ;displays number 6 on SSD
    RETLW 0x78           ;displays number 7 on SSD
    RETLW 0x00           ;displays number 8 on SSD
    RETLW 0x18           ;displays number 9 on SSD
    RETLW 0x08           ;displays number A on SSD
    RETLW 0x03           ;displays number b on SSD
    RETLW 0x46           ;displays number C on SSD
    RETLW 0x21           ;displays number d on SSD
    RETLW 0x06           ;displays number E on SSD
    RETLW 0x0E           ;displays number F on SSD


;*************************CODE FOR COUNTIN UP OR DOWN***************************

 Count_Up
 ;This is the subroutine for Counting up

    CALL  Alarm                 ;Subroutine to sound the Alarm
    INCF count            ;increases count by 1
    BTFSC count,5         ;if bit 5 is set, count goes beyond 15, therefore
                          ;reset to 0
    CLRF count
    RETURN
 Count_Down
 ;This is the subroutine for Counting down

    CALL  Alarm                   ;Subroutine to sound the Alarm
    DECF count,1          ;decreases count by 1
    BTFSS count,7         ;If -1 occurs, bit 7 will be set, reset count to 15
    GOTO No_Reset         ;else skip
    MOVLW 0x0F
    MOVWF count
  No_Reset
    RETURN

;**********************CODE FOR CONTROLLING THE BUTTONS***********************

Ctrl_Start_Stop_Btn
;This subroutine controls the starting and stopping of the count

    BSF control_count,0
    BTFSS PORTA,0         ;Check if switch is up
    GOTO Start_Down_Count   ;If switch is down, Start a down count, else Start up count
    BTFSC PORTA,0
    GOTO Start_Up_Count

Start_Down_Count
    BTFSS control_count,2   ;If down already active, stop timer
    GOTO $+3
    BCF control_count,2           ;Disable Down count
    RETURN
    BCF control_count,1           ;Disable Up
    BSF control_count,2           ;Enable Down
    RETURN

Start_Up_Count
    BTFSS control_count,1   ;If up already active, stop timer
    GOTO $+3
    BCF control_count,1           ;Disable Up
    RETURN
    BSF control_count,1           ;Enable Up
    BCF control_count,2           ;Disable Down
    RETURN

;*************************CODE FOR SOUNDING THE ALARM*************************

Alarm
;This subroutine tests when to sound the alarm. The alarm is sounded when the
;count is at the values of 1,2,3 and 15

    MOVLW 0x01
    MOVF alarm_value1
    CALL Check_Sound_2sec   ;if count is 1 sound for 2 seconds
    MOVLW 0x02
    MOVF alarm_value2
    CALL Check_Sound_2sec   ;if count is 2 sound for 2 seconds
    MOVLW 0x03
    MOVF alarm_value3
    CALL Check_Sound_5sec   ;if count is 3 sound for 5 seconds
    MOVLW 0x0F
    MOVF alarm_value4
    CALL Check_Sound_10sec  ;if count is 15 sound for 10 seconds
    RETURN

 Check_Sound_2sec
 ;This subroutine subracts the value of 1 or 2 from the current count value,
 ;if a zero occurs then sound the alarm for 2 seconds

    BCF STATUS,2    ;clear bit 2 of STATUS register
    SUBWF count,0   ;subtracts value in WREG from current count value
    BTFSS STATUS,2  ;if a zero occurs then bit 2 of STATUS register will be set
    RETURN

Sound_Buzzer_2sec
;This subroutine sounds the alarm for 2 seconds

    BSF PORTA,1
    CALL Display_Hex_Value  ;display count whilst sounding alarm
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    RETURN

Check_Sound_5sec
;This subroutine subracts the value of 3 from the current count value,
;if a zero occurs then sound the alarm for 5 seconds

    BCF STATUS,2    ;clear bit 2 of STATUS register
    SUBWF count,0   ;subtracts value in WREG from current count value
    BTFSS STATUS,2  ;if a zero occurs then bit 2 of STATUS register will be set
    RETURN

Sound_Buzzer_5sec
;This subroutine sounds the alarm for 5 seconds

    BSF PORTA,1
    CALL Display_Hex_Value  ;display count whilst sounding alarm
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    RETURN

Check_Sound_10sec
;This subroutine subracts the value of 15 from the current count value,
;if a zero occurs then sound the alarm for 10 seconds

    BCF STATUS,2    ;clear bit 2 of STATUS register
    SUBWF count,0   ;subtracts value in WREG from current count value
    BTFSS STATUS,2  ;if a zero occurs then bit 2 of STATUS register will be set
    RETURN

Sound_Buzzer_10sec
;This subroutine sounds the alarm for 10 seconds

    BSF PORTA,1
    CALL Display_Hex_Value  ;display count whilst sounding alarm
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    CALL Display_Hex_Value
    BSF PORTA,1
    CALL Display_Hex_Value
    BCF PORTA,1
    RETURN

;**********************CODE FOR READING THE TIME FROM THE RTC*****************
Read_sec
    BCF SSPCON,7              ;Clear the WCOL bit
    BSF PORTB,5                      ;Enable the RTC
    MOVLW 0x00
    MOVWF SSPBUF              ;Tells the RTC that the seconds register is to be read
    CALL RTC_wait             ;Wait for communication to end
    MOVLW 0x00                ;'dummy' data sent. While sending, the required data will be
received
    MOVWF SSPBUF
    CALL RTC_wait             ;Wait for communication to end
    MOVFW SSPBUF              ;Get data that was received
    BCF PORTB,5                      ;Disable the RTC
    MOVWF current_sec         ;Store as current_sec
    RETURN

Read_min
    BCF SSPCON,7              ;Clear the WCOL bit
    BSF PORTB,5                      ;Enable the RTC
    MOVLW 0x01
    MOVWF SSPBUF              ;Tells the RTC that the minutes register is to be read
    CALL RTC_wait             ;Wait for communication to end
    MOVLW 0x57                ;'dummy' data sent. While sending, the required data will be
received
    MOVWF SSPBUF
    CALL RTC_wait             ;Wait for communication to end
    MOVFW SSPBUF              ;Get data that was received
    BCF PORTB,5                      ;Disable the RTC
    MOVWF current_min         ;Store as current_min
    RETURN

Read_hour
    BCF SSPCON,7              ;Clear the WCOL bit
    BSF PORTB,5                      ;Enable the RTC
    MOVLW 0x02
    MOVWF SSPBUF              ;Tells the RTC that the hours register is to be read
    CALL RTC_wait             ;Wait for communication to end
    MOVLW 0x20
    MOVWF SSPBUF              ;'dummy' data sent. While sending, the required data will be
received
    CALL RTC_wait             ;Wait for communication to end
    MOVFW SSPBUF              ;Get data that was received
    BCF PORTB,5                      ;Disable the RTC
    ANDLW b'00111111'         ;Remove the unwanted data from the register
    MOVWF current_hr          ;Store as current_hr
    RETURN

;*********************CODE FOR DISPLAYING THE CURRENT TIME********************

Display_sec
    CALL Convert_to_Tens_and_Units  ;Stores it to Tens
    CALL SSD_HEX_Table            ;Gets code for displaying the tens value
    CALL Turn_on_SSD_5            ;Turns on Seconds Tens SSD
    MOVWF PORTC                          ;Display Seconds Tens value
    CALL Multiplex_Delay          ;Delay for multiplexing
    CALL Turn_off_SSDs            ;Disable Seconds Tens SSD

    MOVFW units
    CALL SSD_HEX_Table            ;Gets code for displaying the Units value
    CALL Turn_on_SSD_6            ;Enable Seconds units SSD
    MOVWF PORTC                          ;Displays Seconds units value
    CALL Multiplex_Delay          ;Delay for multiplexing
    CALL Turn_off_SSDs            ;Disable Seconds units SSD
    RETURN

Display_min
    CALL Convert_to_Tens_and_Units  ;Stores it to Tens
    CALL SSD_HEX_Table            ;Gets code for displaying the tens value
    CALL Turn_on_SSD_3            ;Turns on Minutes Tens SSD
    MOVWF PORTC                           ;Display Minutes Tens value
    CALL Multiplex_Delay          ;Delay for multiplexing
    CALL Turn_off_SSDs            ;Disable Minutes Tens SSD

    MOVFW units
    CALL SSD_HEX_Table            ;Gets code for displaying the Units value
    CALL Turn_on_SSD_4            ;Enable Minutes units SSD
    MOVWF PORTC                           ;Displays Minutes units value
    CALL Multiplex_Delay          ;Delay for multiplexing
    CALL Turn_off_SSDs            ;Disable Minutes units SSD
    RETURN

 Display_hour
    CALL Convert_to_Tens_and_Units  ;Stores it to Tens
    CALL SSD_HEX_Table            ;Gets code for displaying the tens value
    CALL Turn_on_SSD_1            ;Turns on Hours Tens SSD
    MOVWF PORTC                           ;Display Hours Tens value
    CALL Multiplex_Delay          ;Delay for multiplexing
    CALL Turn_off_SSDs            ;Disable Hours Tens SSD

    MOVFW units
    CALL SSD_HEX_Table            ;Gets code for displaying the Units value
    CALL Turn_on_SSD_2            ;Enable Hours units SSD
    MOVWF PORTC                           ;Displays Hours units value
    CALL Multiplex_Delay          ;Delay for multiplexing
    CALL Turn_off_SSDs            ;Disable Hours units SSD
    RETURN

Display_Time
    MOVFW current_sec
    CALL Display_sec              ;Displays current second
    MOVFW current_min
    CALL Display_min              ;Displays current minute
    MOVFW current_hr
    CALL Display_hour             ;Displays current hour
    RETURN


Convert_to_Tens_and_Units
;This subroutine splits the value into tens and units

    MOVWF tens
    ANDLW 0x0F          ;b'00001111 , clears upper nibble of BCD number
    MOVWF units                 ;stores the value as the units
    SWAPF tens,1        ;swaps the nibbles of the BCD number
    MOVFW tens
    ANDLW 0x0F          ;b'00001111, clears the high nibble to get tens value
    MOVWF tens          ;stores value in tens register
    RETURN

RTC_wait
    BANKSEL SSPSTAT;WRITE Address
    BTFSS SSPSTAT, BF ;Has data been received(transmit complete)?
    GOTO RTC_wait ;No
    BANKSEL 0x00
    RETURN

;********************CODE FOR ENABLING AND DISABLING SSDs*********************

Turn_off_SSDs
;This subroutine turns off all SSDs

    BSF PORTA,5 ;A0
    BSF PORTB,7 ;A1
    BSF PORTA,4 ;A2
    RETURN
```

```
Turn_on_SSD_1
;This subroutine turns on SSD 1

    BCF PORTA,5 ;A0
    BCF PORTB,7 ;A1
    BCF PORTA,4 ;A2
    RETURN

Turn_on_SSD_2
;This subroutine turns on SSD 2

    BSF PORTA,5 ;A0
    BCF PORTB,7 ;A1
    BCF PORTA,4 ;A2
    RETURN

Turn_on_SSD_3
;This subroutine turns on SSD 3

    BCF PORTA,5 ;A0
    BSF PORTB,7 ;A1
    BCF PORTA,4 ;A2
    RETURN

Turn_on_SSD_4
;This subroutine turns on SSD 4

    BSF PORTA,5 ;A0
    BSF PORTB,7 ;A1
    BCF PORTA,4 ;A2
    RETURN

Turn_on_SSD_5
;This subroutine turns on SSD 5

    BCF PORTA,5 ;A0
    BCF PORTB,7 ;A1
    BSF PORTA,4 ;A2
    RETURN

Turn_on_SSD_6
;This subroutine turns on SSD 6

    BSF PORTA,5 ;A0
    BCF PORTB,7 ;A1
    BSF PORTA,4 ;A2
    RETURN

END
```