Climate control system

Embedded Systems Development 600085

Group 6.1

(Ieaun Roberts)

[WC: 2010]

[Individual Report]

Contents

1 INTRODUCTION

Within this report you will find documentation supporting the individual drivers of the climate control system. They are as follows:

- LCD Panel (LCD12864)
- Temperature Sensor (DS18B20)
- Button Input (4x4 Key Matrix)
- Sounder
- Real Time Clock (DS1302)

2 ARTEFACTS PRODUCED

DESIGN

Most of the drivers created consist of an Initialization function which sets ports to either input or output, a custom delay function and functional code which makes use of each devices capabilities to perform one of the outlined features.

E.G. The typical drivers contain :

1. **Initialization function** [initThermo()]
2. **Delay function** [delayThermo()]
3. **Functional code** [mainThermo()]
4. **Ports Used** [Port A]

API DESCRIPTION

2.1.1 LCD Panel:

First 2 digits in Instruction set represent register select and read/write. The E value enables the signal/ instruction when high. The rest of the values are data buses used to transfer the data.

	Value	Description		Value	Description
RS	=0	Command is instruction not data		=1	Command is data not instruction
R/W	=0	Is write not read		=1	Is read not write
E	=1	Send Instruction/Data		=0	Do not send Instruction/Data yet

Figure 2, Explanation of RS, R/W and E values

Using Port A:
Using Port D:

Function	Description	Instruction Set
lcd_init()	Initialize function. Performs indicated instruction set tasks to allow device to be operated without errors while in use. Sets starting position for cursor.	0000110000 [Function set is 8 Bit] 0000000001 [Clear] 0000000110 [Entry Mode] 0000001100 [Display]
clear_p()	Clears the LCD panel.	0000000001 [Clear]
han_Number()	Takes 2 **Integer** parameters, display Position and a value to display respectively. Uses Send_i() to send instruction and Send_d() to display each value. Displays individual numbers on LCD in specified position.	00(DisplayPos) 10(ValueToDisplay)
han_Display()	Takes 1 **integer** for display position and a character array of items to display. Uses Send_i() to send instruction, Qushu() to send each array item 1 at a time ,Send_d() to display each char 1 at a time. Displays 1 array of size 0x5 in specified position.	00(DisplayPos) 10(ValueToDisplay) repeat for however many items in array
wr_zb()	Sets the Displays X and Y positions.	00(X) 00(Y)
send_d()	Takes a character as a parameter and after setting values such as **RS=1** R/W=0 it enables and sends the **Data** to the LCD to be display	10(Character)
send_i()	Takes a character as a parameter and after setting values such as **RS=0** R/W=0 it enables and sends the **instruction** to the LCD.	00(Character)
chk_busy()	Sets port D to input to check if the LCD is already busy with another process. Sets RS=0 R/W=1 and then enables. If the 8th bit is not 1 then the system is busy	01[Read busy flag]
qushu()	Send each character of a given input to the send_d() function 1 character at a time.	
delay()	Delay Function	

Figure 3, LCD driver methods

```
#ifndef LCD_H
#define LCD_H
//LCD
void lcd_init();                    //LCD init
void clear_p();                     //clear screen
void han_DSet();
void han_Number(int DisplayPos,int NumberSelected);
void han_Display(int DisplayPos,const unsigned char *Data);
void wr_zb();                       //display setting mode.
void send_d(unsigned char x);       //send data
void send_i(unsigned char x);       //send command.
void chk_busy();                    //check busy sub.
void qushu(int counts,const unsigned char *ps);       //search table.
void delay();                       //delay func, decide the speed of display.
#ifdef  __cplusplus
extern "C" {
#endif
```

Figure 4, Header file needed for use of the LCD driver.

2.1.2 Temperature Sensor:

Using Port A:

Features: Provides a temperature reading which indicates the current temperature of the device.

Function	Description
delayThermo()	Delay Function
initThermo()	Initializes device, sets Port A to output
mainThermo()	Takes 3 pointers as parameters of type integer which will store the tens ,ones and decimal value of temperature. Calls functions initializing device, receives temperature data and then converts it into an understandable format which is then copied into the given pointers.
get_temp()	Converts data on the device into temperature units.
displayThermo()	Copies temperature values into “TempValues” array
write_byte()	Writes 1 Byte into the device
read_byte()	Reads 1 Byte from the device
Reset()	Resets device by setting port A to output, waits 503us, sets Port A to input, waits another 70us and finally if it receives a response signal it waits 430us to reset the device

Figure 5, Temperature Sensor driver methods

```
#ifndef TEMPERATURE_H
#define TEMPERATURE_H
//Thermo
void delayThermo(char x,char y);
void mainThermo(int *Tens_Temp,int *Ones_Temp,int *Decimals_Temp );
void get_temp();
void initThermo();
void displayThermo();
#ifdef  __cplusplus
extern "C" {
#endif
```

Figure 6, Header file needed for use of the Temperature driver.

2.1.3 Button Input:

Using Port A:
Using Port C:

Function	Description
keymatrix()	Takes 2 Pointer integers as parameters to store Button Pressed and active set. Calls initialization, Stores value of button pressed into first pointer and active set into second.
initkey()	initialize function. Sets Port A low 4 bits input, high 4 bits output and Port C High 4 bits input, Low 4 bits output.
scan()	Scan for a button press by sequentially setting the C3,C2,C1 and C0 bits to 0 (columns) and all others to 1 (Rows). This value is stored. The high 4 bits are then set to 1 (C3,C2,C1 and C0) and enter a conditional statement. A button press carries the charge turning the 1 into a 0. If all values are 1 then no button has been pressed. If any value is 0 then we have found the position of the button.
display()	Takes 1 integer parameter and assigns “NoPressed” to a value of 0-99 dependant on what was parsed as a parameter. Also alters “Activeset_Buttons” If applicable

Figure 7, Temperature Sensor driver methods.

```
#ifndef BUTTONS_H
#define BUTTONS_H
void  keymatrix(int *ButtonPressed_,int *ActiveSet);
void  initkey();               //I/O PORT initialize function declare
void  scan();                 //key scan function declare
void  display(int x);         //display function declare
#ifdef  __cplusplus
extern "C" {
#endif
```

Figure 8, Header file needed for use of the Button driver.

2.1.4 Sounder:

Using Port E:

Function	Description
BuzzerActivate()	Sets Port E as output and calls sound200ms(). On return sets Port E to Input.
sounddelay0()	Adjustable delay Function. 1 Variable of type char is needed in parameters. Larger value = longer delay.
sound200ms()	Sets Maximum/Minimum frequency values for buzzer and amount of times the function will loop. Cycles through the Loop which set port E1 to output turning the buzzer on. It delays for an amount of time, then Sets Port E1 as input turning the buzzer off.

Figure 9, Sounder/Buzzer driver methods

```
8  #ifndef BUZZER_H
9  #define BUZZER_H
10 void BuzzerActivate();
11 void sounddelay0(unsigned char delay_count);
12 void sound200ms();
13 #ifdef  __cplusplus
14 extern "C" {
15 #endif
16
```

Figure 10, Header file needed for use of the Sounder/Buzzer driver.

2.1.5 Real Time Clock:

Using Port B:

Function	Description
DateMain()	Takes 6 **Character** variables as parameters to store these values. 1. Year 1st digit 2. Year 2nd digit 3. Month 1st digit 4. Month 2nd digit 5. Day 1st digit 6. Day 2nd digit Calls “get_time” function to receive the current date and stores each char result in its specific pointer.
ClockMain()	Takes 6 **Character** variables as parameters to store these values. 1. Hour 1st digit 2. Hour 2nd digit 3. Min 1st digit 4. Min 2nd digit 5. Sec 1st digit 6. Sec 2nd digit Calls “get_time” function to receive the current time and stores each char result in its specific pointer.
ds1302_init()	Initializes device, set RB1 to input, all the other bits to output
time_read_1()	Reads data stored on the DS1302 one byte at a time.
set_time()	Takes 5 **Integer** variables as parameters to store these values. 1. Hour 2. Minute 3. Day 4. Month 5. Year Sets time and date by storing values into variables then writing them to the DS1302 one byte at a time.
get_time()	Set the values received by the “time_read_1” methods into an array
Insert()	Logic for deciding whether the data that has been received is stored in the Date array or the Time array.
ds_display()	Function that extracts data from DS1302 and stores it into variables.
ds_delay()	Delay Function

Figure 11, Real time clock driver methods

```
8  #ifndef DS1302_H
9  #define DS1302_H
10 void DateMain(char *Year1_Temp,char *Year2_Temp,char *Month1_Temp,char *Month2_Temp,char *Day1_Temp,char *Day2_Temp);
11 void ClockMain(char *Hour1_Temp,char *Hour2_Temp,char *Min1_Temp,char *Min2_Temp,char *Sec1_Temp,char *Sec2_Temp);
12 void ds1302_init();                    //DS1302 initilize subroutine.
13 void set_time(int Hour,int Min,int Day_st,int Month_st,int Year_st);          //set time subroutine.
14 void get_time();                       //get time subroutine.
15 void ds_display();
16 #ifdef __cplusplus
17 extern "C" {
18 #endif
19
```

Figure 12, Header file needed for use of the DS1302 driver.

CODE

Buzzer

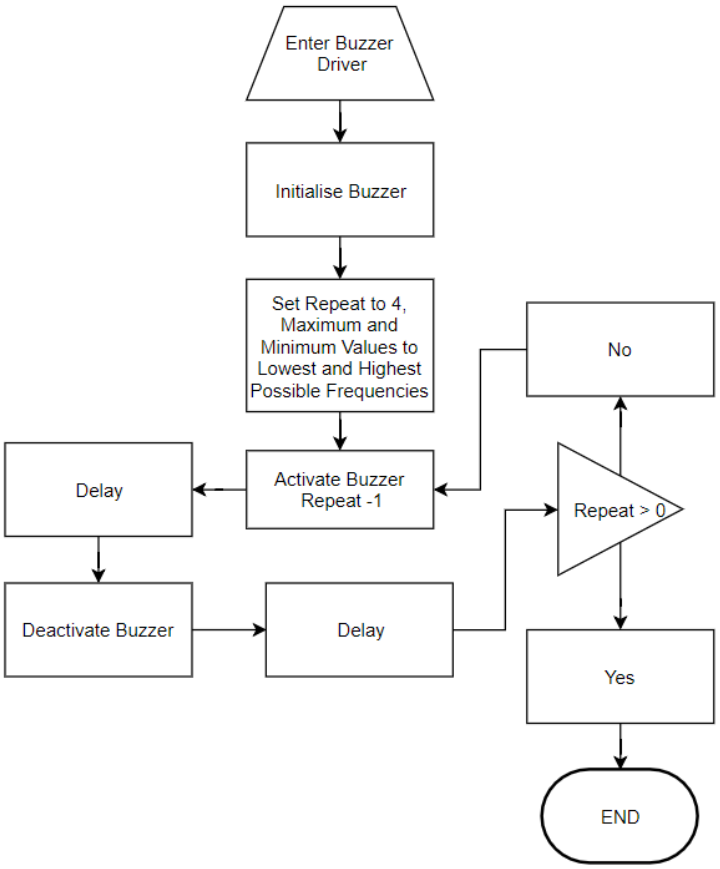

Figure 13, Flow diagram representing buzzer driver operation

Operation:

The buzzer driver initializes port E to output, sets conditional loop variable “fre_reapet” to 4 and sets max and minimum values for the sounders frequency. Proceeds to activate the buzzer and decrement the repeat value, delay and then deactivate the buzzer. This is repeated while the repeat value is larger than 0.

LCD Panel

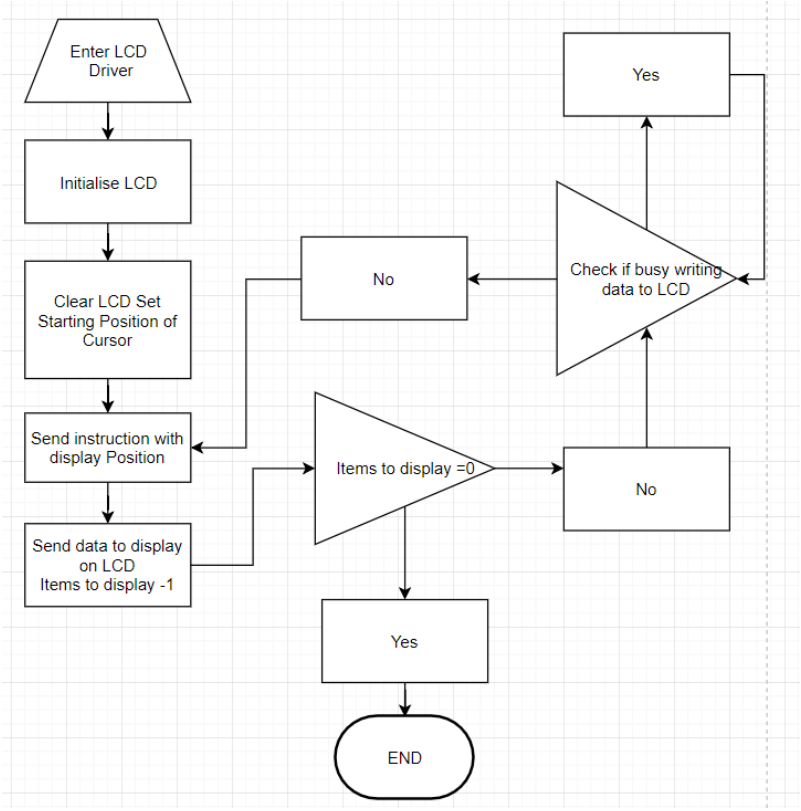

Figure 14, Flow diagram representing LCD driver operation

Operation:

The LCD driver initializes port A and D to output, clears the screen and sets starting position of cursor on display. It will then send an instruction indicating the position where future data should be displayed on the LCD panel. The data is then sent and a counter holding the total amount of values in the variable will be decremented. If it does not equal to 0 then the code should check If more data is being written to the panel. If not, then the rest of the values in the array given will be displayed on the LCD panel one at a time.

Button Input

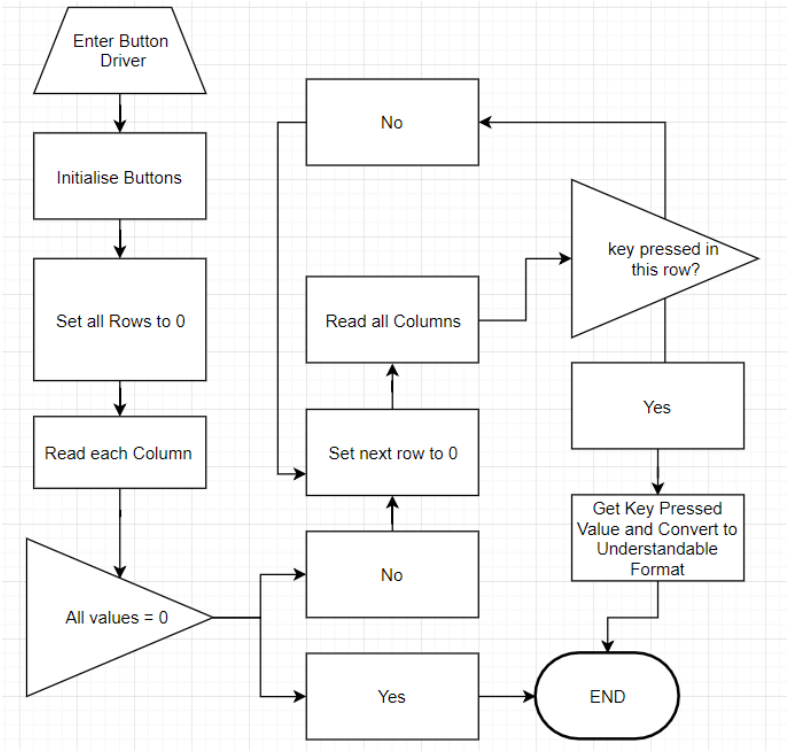

Figure 15, Flow diagram representing Button driver operation

Operation:

To Scan for a button press the driver sequentially sets the C3,C2,C1 and C0 bits to 0 (columns) and all others rows to 1. This value is stored. The value of each column is then read, if all values for Columns and rows are 1, no key has been pressed. if a column bit has a value of 0 it indicates a key press and the process continues to find the coordinates of the button and translate it into an understandable format.

```
 * Created on 05 December 2018, 12:11
 */

#include <stdio.h>
#include <stdlib.h>
#include<pic.h>              //include MCU head file
#include <xc.h>
#include "Buttons.h"

#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT enabled)
#pragma config PWRTE = ON // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial

int  result;
int ActiveSet_Buttons=0; //if 1= Time, if 2=Date, if 3=Temp, if 4=Trigger Temp, else = no entry
int NoPressed = 99;

void keymatrix();
void  initkey();               //I/O PORT initialize function declare
void  scan();                 //key scan function declare
void  display_key(int x);       //display function declare

//hardware request:SW S4 ON ,S6 ON,S5 5-6 bits ON,the others OFF
//-------------------------------------------------
//main program
void keymatrix(int *ButtonPressed_,int *ActiveSet)
{
      initkey();                 //call initialize subprogram
      scan();                  //call key scan subprogram
      display_key(result);        //call result display subprogram

      *ButtonPressed_= NoPressed;
      *ActiveSet=ActiveSet_Buttons;
      NoPressed=99;
}

//-------------------------------------------------
//initialize function
void initkey()
{
 ADCON1=0X07;                 // set A PORT general I/O PORT
 TRISA=0X0f;                  //A PORT low 4 bits INPUT,high 4 bits OUTPUT
 TRISC=0XF0;                  //C PORT high 4 bits INPUT,low 4 bits OUTPUT
}
```

```
//-----------------------------------------------------
//key scan program
void scan()
{
 PORTC=0XF7;                //C3 OUTPUT low,the other 3 bits OUTPUT high
 asm("nop");                //delay
 result=PORTC;              //read C PORT
 result=result&0xf0;        //clear low 4 bits
 if(result!=0xf0)           //judge if high 4 bits all 1(all 1 is no key press)
   {
    result=result|0x07;     //no,add low 4 bits 0x07 as key scan result
   }
 else                       //yes,change low 4 bits OUTPUT, judge if a key press again
   {
   PORTC=0XFb;              //C2 OUTPUT low,the other 3 bits OUTPUT high
   asm("nop");              //delay
   result=PORTC;            //read C PORT
   result=result&0xf0;      //clear low 4 bits
   if(result!=0xf0)         //judge if high 4 bits all 1(all 1 is no key press)
    {
    result=result|0x0b;     //no,add low 4 bits 0x0b as key scan result
    }
   else                     //yes,change low 4 bits OUTPUT, judge if a key press again
      {
      PORTC=0XFd;           //C1 OUTPUT low,the other 3 bits OUTPUT high
      asm("nop");           //delay
      result=PORTC;         //read C PORT
      }
    else                    //yes,change low 4 bits OUTPUT, judge if a key press again
        {
        PORTC=0XFd;         //C1 OUTPUT low,the other 3 bits OUTPUT high
        asm("nop");         //delay
        result=PORTC;       //read C PORT
        result=result&0xf0; //clear low 4 bits
        if(result!=0xf0)    //judge if high 4 bits all 1(all 1 is no key press)
         {
         result=result|0x0d;//no,add low 4 bits 0x0d as key scan result
         }
        else                //yes,change low 4 bits OUTPUT, judge if a key press again
           {
           PORTC=0XFe;      //C0 OUTPUT low,the other 3 bits OUTPUT high
           asm("nop");      //delay
           result=PORTC;    //read C PORT
           result=result&0xf0;//clear low 4 bits
           if(result!=0xf0) //judge if high 4 bits all 1(all 1 is no key press)
             {
              result=result|0x0e;//no,add low 4 bits 0x0e as key scan result
             }
           else             //yes,all key scan end,no key press,set no key press flag
             {
              result=0xff;  //key scan result 0xff as no key press flag
             }
           }
```

```
void display_key(int x)
  {
   switch(result)
      {
       case 0x7d:
                  NoPressed=7;break;  //K24   7
       case 0x7b:
                  NoPressed=8;break;   //K23   8
       case 0x77:
                  NoPressed=9;break;    //K22  9
       case 0xbd:
                  NoPressed=4;;break;    //K20   4
       case 0xbb:
                  NoPressed=5;break;     //K19   5
       case 0xb7:
                  NoPressed=6;break;   //K18    6
       case 0xdd:
                  NoPressed=1;break;    //K16   1
       case 0xdb:
                  NoPressed=2;break;   //K15     2
       case 0xd7:
                  NoPressed=3;break;    //K14    3
       case 0xeb:
                  NoPressed=0;break;     //K11     0
       case 0xe7:
                  NoPressed=55; break;    //K10  //Silencer
```

```
       case 0xed:
                  break;      //K12
       case 0x7e:
                  ;break;      //K25
       case 0xbe:
                  ActiveSet_Buttons=3;NoPressed=98; break;  //K21     Trigger temp
       case 0xde:
                  ActiveSet_Buttons=2;NoPressed=98;break;   //K13   Date
       case 0xee:
                  ActiveSet_Buttons=1;NoPressed=98; break;   //K17       Time
       case 0xff:
                  NoPressed=99;break;      //no key press
    }
}

//-----------------------------------------------------------------
```

Temperature sensor

```c
#include <stdio.h>
#include <stdlib.h>
#include<pic.h>                          //include MCU head file
#include <xc.h>
#include "Temperature.h"
#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT enabled)
#pragma config PWRTE = ON // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial

//Thermo
#define  uch unsigned char                    //
# define DQ RA0                               //define 18B20 data PORT
# define DQ_DIR TRISA0                        //define 18B20 D PORT direct register
# define DQ_HIGH() DQ_DIR =1                  //set data PORT INPUT
# define DQ_LOW() DQ = 0; DQ_DIR = 0          //set data PORT OUTPUT
 unsigned char  TLV=0 ;                       //temperature high byte
 unsigned char  THV=0;                        //temperature low byte
 unsigned char TZ=0;                          //temperature integer after convert
 unsigned char TX=0;                          //temperature decimal  after convert
 unsigned int wd;                             //temperature BCD code  after convert
unsigned char shi;                            //integer ten bit
unsigned char ge;                             //integer Entries bit
unsigned char shifen;                         //ten cent bit
unsigned char tablethermo[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
```

```c
uch read_byte(void);
void write_byte(uch val);
int TempValues[3];

//************************************
 //----------------------------------
//delay function
void delayThermo(char x,char y)
{
  char z;
  do{
      z=y;
      do{;}while(--z);
     }while(--x);
 }

//----------------------------------
//display function
void displayThermo()
{
 TempValues[0]=shi;
 delayThermo(10,70);
 TempValues[1]=ge;
 delayThermo(10,70);
 TempValues[2]=shifen;
 delayThermo(10,70);
```

```
65  void initThermo()
66  {
67    ADCON1=0X07;
68    TRISA=0X00;
69  }
70
71  //------------------------------
72  //reset DS18B20 function
73  void reset(void)
74  {
75    char presence=1;
76    while(presence)
77    {
78      DQ_LOW() ;
79      delayThermo(2,70);
80      DQ_HIGH();
81      delayThermo(2,8);
82      if(DQ==1) presence=1;
83      else presence=0;
84      delayThermo(2,60);
85     }
86    }
87
90  void write_byte(uch val)
91  {
92   uch i;
93   uch temp;
94   for(i=8;i>0;i--)
95   {
96     temp=val&0x01;
97     DQ_LOW();
98     NOP();
99     NOP();
100    NOP();
101    NOP();
102    NOP();
103    if(temp==1)  DQ_HIGH();
104    delayThermo(2,7);
105    DQ_HIGH();
106    NOP();
107    NOP();
108    val=val>>1;
109   }
110 }
```

```
113  //18b20 read a byte function
114  uch read_byte(void)
115  {
116   uch i;
117   uch value=0;
118   static _Bool j;
119   for(i=8;i>0;i--)
120   {
121     value>>=1;
122     DQ_LOW();
123     NOP();
124     NOP();
125     NOP();
126     NOP();
127     NOP();
128     NOP();
129     DQ_HIGH();
130     NOP();
131     NOP();
132     NOP();
133     NOP();
134     NOP();
135     j=DQ;
136     if(j) value|=0x80;
137     delayThermo(2,7);
138    }
```

```
143  //start temperature convert function
144  void get_temp()
145  {
146  int i;
147  DQ_HIGH();
148  reset();                              //reset,wait for  18b20 responsion
149  write_byte(0XCC);                     //ignore ROM matching
150  write_byte(0X44);                     //send  temperature convert command
151  for(i=20;i>0;i--)
152      {
153          displayThermo();                    //call some display function,insure the time of convert temperature
154      }
155  reset();                              //reset again,wait for 18b20 responsion
156  write_byte(0XCC);                     //ignore ROM matching
157  write_byte(0XBE);                     //send read temperature command
158  TLV=read_byte();                      //read temperature low byte
159  THV=read_byte();                      //read temperature high byte
160  DQ_HIGH();                            //release general line
161  TZ=(TLV>>4)|((THV<<4)&(0X3f));            //temperature integer
162  TX=TLV<<4;                            //temperature decimal
163  if(TZ>100) TZ/100;                    //not display hundred bit
164  ge=TZ%10;                      //integer Entries bit
165  shi=TZ/10;                     //integer ten bit
166  wd=0;
167  if (TX & 0x80) wd=wd+5000;
168  if (TX & 0x40) wd=wd+2500;
```

```
164   ge=TZ%10;                       //integer Entries bit
165   shi=TZ/10;                      //integer ten bit
166   wd=0;
167   if (TX & 0x80) wd=wd+5000;
168   if (TX & 0x40) wd=wd+2500;
169   if (TX & 0x20) wd=wd+1250;
170   if (TX & 0x10) wd=wd+625;              //hereinbefore four instructions are turn  decimal into B
171   shifen=wd/1000;                        //ten cent bit
172   NOP();
173   }
174
175   //-------------------------------------------------
176   //main function
177   void mainThermo(int *Tens_Temp,int *Ones_Temp,int *Decimals_Temp )
178   {
179         initThermo();                      //call system initialize function
180         get_temp();             //call temperature convert function
181         displayThermo();                //call display function
182         unsigned char display_number[10] ={'0','1','2','3','4','5','6','7','8','9'}; //char values
183         *Tens_Temp= display_number[TempValues[0]];
184         *Ones_Temp= display_number[TempValues[1]];
185         *Decimals_Temp= display_number[TempValues[2]];
```

Figures 16, Temperature driver code

Real Time Clock

```
7   #include<pic.h>                     //include MCU head file
8   #include <xc.h>
9   #include "ds1302.h"
10  #include "MainHead.h"
11  #pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
12  #pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT enabled)
13  #pragma config PWRTE = ON // Power-up Timer Enable bit (PWRT disabled)
14  #pragma config BOREN = OFF // Brown-out Reset Enable bit (BOR enabled)
15  #pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial
16
17
18  #define i_o   RB4                   //1302I_O
19  #define sclk  RB0                   //1302 clock
20  #define rst   RB5                   //1302 enable bit
21
22  void  ds_delay();
23
24  unsigned char time_rx;
25  void Insert(int ArrayPos,char Number_To_insert);
26  void time_write_1(unsigned char time_tx);    //write one byte subroutine.
27  unsigned char  time_read_1();          //read one byte subroutine.
28  void ds_delay();                          //delay subroutine.
29  //define the time: sec,min,hour,day,month,week,year,control word.
30  char tableds[]={0x00,0x02,0x03,0x10,0x12,0x02,0x18,0x00};
31  //define the read time and date save table.
32  char tableds1[7];
33  char EasyWayTable[96]={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x10,
34                         0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x20,
```

```
25  void Insert(int ArrayPos,char Number_To_insert);
26  void time_write_1(unsigned char time_tx);    //write one byte subroutine.
27  unsigned char  time_read_1();          //read one byte subroutine.
28  void ds_delay();                         //delay subroutine.
29  //define the time: sec,min,hour,day,month,week,year,control word.
30  char tableds[]={0x00,0x02,0x03,0x10,0x12,0x02,0x18,0x00};
31  //define the read time and date save table.
32  char tabledsl[7];
33  char EasyWayTable[96]={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x10,
34                         0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x20,
35                         0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x30,
36                         0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x40,
37                         0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x50,
38                         0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x60,
39                         0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x70,
40                         0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x80,
41                         0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x90,
42                         0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99};
43
44  char Time[6];
45  char Date[6];
46  int Time_Date;  //if 0 then time if 1 then date
47
```

```
50  //main routine.
51  void ClockMain(char *Hour1_Temp,char *Hour2_Temp,char *Min1_Temp,char *Min2_Temp,char *Sec1_Temp,char *Sec2_Temp)
52   {
53          Time_Date=0;
54          get_time();
55          ds_display();
56          *Hour1_Temp=Time[0];
57          *Hour2_Temp=Time[1];
58          *Min1_Temp=Time[2];
59          *Min2_Temp=Time[3];
60          *Sec1_Temp=Time[4];
61          *Sec2_Temp=Time[5];
62   }
63
64  void DateMain(char *Year1_Temp,char *Year2_Temp,char *Month1_Temp,char *Month2_Temp,char *Day1_Temp,char *Day2_Temp)
65  {
66   Time_Date=1;
67   get_time();
68   ds_display();
69      *Year1_Temp= Date[0];
70      *Year2_Temp= Date[1];
71      *Month1_Temp=Date[2];
72      *Month2_Temp=Date[3];
73      *Day1_Temp =Date[4];
74      *Day2_Temp =Date[5];
75  }
```

```
77  //-----------------------------------------------
78  //DS1302 initilize.
79  void ds1302_init()
80    {
81     ADCON1=0X06;                    //a port all i/o
82     TRISB=0X02;                     //rb1 input, others output
83     sclk=0;                           //pull low clock
84     rst =0;                           //reset DS1302
85     rst=1;                            //enable DS1302
86     time_write_1(0x8e);               //send control command
87     time_write_1(0);                  //enable write DS1302
88     rst=0;                            //reset
89    }
90
91  //-----------------------------------------------
```

```
92  //set time.
93  void set_time(int Hour,int Min,int Day_st,int Month_st,int Year_st)
94    {
95     int i;                              //define the loop counter.
96     rst=1;                              //enable DS1302
97     time_write_1(0xbe);                 //
98     // set time value to Hour and min
99     tableds[1]=EasyWayTable[Min];
100    tableds[2]=EasyWayTable[Hour];
101    tableds[3]=EasyWayTable[Day_st];
102    tableds[4]=EasyWayTable[Month_st];
103    tableds[6]=EasyWayTable[Year_st];
104    for(i=0;i<8;i++)                    //continue to write 8 bytes.
105      {
106        time_write_1(tableds[i]);        //write one byte
107      }
108    rst=0;                              //reset
109    }
110
```

```
111 //-----------------------------------------------
112 //get time.
113 void get_time()
114 {
115   int i;                               //set loop counter.
116   rst=1;                               //enable DS1302
117   time_write_1(0xbf);                  //
118   for(i=0;i<7;i++)                     //continue to read 7 bytes.
119     {
120        tabledsl[i]=time_read_1();       //
121     }
122    rst=0;                              //reset DS1302
123 }
```

```
127  void time_write_1(unsigned char time_tx)
128  {
129     int j;                                 //set the loop counter.
130     for(j=0;j<8;j++)                       //continue to write 8bit
131       {
132         i_o=0;                             //
133         sclk=0;                            //pull low clk
134         if(time_tx&0x01)                   //judge the send bit is 0 or 1.
135           {
136             i_o=1;                         //is 1
137           }
138         time_tx=time_tx>>1;                //rotate right 1 bit.
139         sclk=1;                            //pull high clk
140       }
141     sclk=0;                                //finished 1 byte,pull low clk
142  }
143
```

```
145  //read one byte.
146  unsigned char time_read_1()
147  {
148
149     time_rx = 0;
150    int j;                              //set the loop counter.
151    unsigned char bitmask = 0x01;
152    TRISB4=1;                           //continue to write 8bit
153    for(j=0;j<8;j++)
154      {
155        sclk=0;                          //pull low clk
156       // time_rx=time_rx>>1;              //judge the send bit is 0 or 1.
157
158        if(i_o)time_rx = time_rx | (bitmask << j);//put the received bit into the reg's highest.
159
160      //   time_rx=time_rx i_o;
161        sclk=1;                          //pull high clk
162      }
163    TRISB4=0;                           //finished 1 byte,pull low clk
164    sclk=0;
165    return(time_rx);
166  }
167
```

```
170  void ds_display()
171    {
172      int i;                            //define table variable.
173      if(Time_Date==1)                        //judge rbl.
174        {
175          tabledsl[0]=tabledsl[3];
176          tabledsl[1]=tabledsl[4];
177          tabledsl[2]=tabledsl[6];
178        }
179      i=tabledsl[0]&0x0f;              //sec's low.
180      Insert(5,i);
181      ds_delay();                       //delay some times.
182      i=tabledsl[0]&0xf0;              //sec's high
183      i=i>>4;                         //rotate right for 4 bits.
184      Insert(4,i);
185      ds_delay();                       //delay some times.
186      i=tabledsl[1]&0x0f;              //min's low.
187      Insert(3,i);
188      ds_delay();                       //delay some times.
189      i=tabledsl[1]&0xf0;              //min's high
190      i=i>>4;                         //rotate right for 4 bits.
191      Insert(2,i);
192      ds_delay();                       //delay some times.
193      i=tabledsl[2]&0x0f;              //hour's low.
194      Insert(1,i);
195      ds_delay();                       //delay some times.
```

```
190         i=i>>4;                         //rotate right for 4 bits.
191         Insert(2,i);
192         ds_delay();                       //delay some times.
193         i=tabledsl[2]&0x0f;              //hour's low.
194         Insert(1,i);
195         ds_delay();                       //delay some times.
196         i=tabledsl[2]&0xf0;              //hour's high
197         i=i>>4;                         //rotate right for 4 bits.
198         Insert(0,i);
199         ds_delay();                       //delay some times.
200     }
201
202   void Insert(int ArrayPos,char Number_To_insert)
203   {
204       unsigned char display_number_Clock[10] ={'0','1','2','3','4','5','6','7','8','9'};
205       if(Time_Date==1)
206       {
207        Date[ArrayPos]=display_number_Clock[Number_To_insert];
208       }
209       if(Time_Date==0)
210       {
211       Time[ArrayPos]=display_number_Clock[Number_To_insert];
212       }
213
214   }
```

```
216   //---------------------------------------------------------------------
217   //delay
218   void  ds_delay()                //
219       {
220        int i;                    //define variable
221        for(i=0x64;i--;)
222        {;}      //delay
223       }
224
```

Figures 17, Real Time Clock driver code

3 TESTING

VERIFICATION AND VALIDATION

Device	Case	Verification method	Validation Case	Passed
Buzzer	Does Buzzer activate when called ?	Call method **BuzzerActivate()**	Buzzer activates and beeps	✓
Button Input	Are all buttons returning appropriate values?	Call **keymatrix()** and press each button to observe results	Variable "NoPressed" is altered indicating a button was found.	✓
LCD Panel	Is the system able to display a specific character on request?	Call **han_Display()** and input a display position and character.	Character variable supplied is displayed on LCD	✓
Temperature sensor	Does the DS18B20 read the current temperature ?	Call mainThermo() and parse 3 int variables. If these are altered to the correct temperature, then the system is running.	The variables named "Shifen" , "Ge" and "Shi" are altered.	✓
Real Time Clock	What will happen if a user enters a value out of range of the 24-hour format ?	A value larger than 24 is sent as an input for hours.	The value is not retained and Invalid is displayed.	✓
Real Time Clock	What will happen if a user enters a value out of range of the date (Month) format ?	A Value of 13 is entered for month.	The value is not retained and Invalid is displayed.	✓
Real Time Clock	What will happen if a user enters a value out of range of the date (Day) format ?	A Value of 33 is entered for day.	The value is not retained and Invalid is displayed.	✓
Real Time Clock	Is the month of February able to hold a day value larger than 27 ?	A Value of 02 is entered for month and 29 for day.	The value is not retained and Invalid is displayed.	X

Figure 18, Verification and validation cases

4 CRITICAL EVALUATION

USEFULNESS OF TECHNIQUES

There is room for improvement regarding the DS1302 driver.

Specifically, the table used to convert the given reference for the Time/Date integer values into character values that follow the chips logic (e.g. 0x11 turns to 0x12). As much as I tried I was unable to create a more efficient method.

The method I created is a simple way of solving a complicated task but uses up a massive amount of memory.

```
char EasyWayTable[96]={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x10,
                       0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x20,
                       0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x30,
                       0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x40,
                       0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x50,
                       0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x60,
                       0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x70,
                       0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x80,
                       0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x90,
                       0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99};
```

The code I implemented to alter the weekday has a lot of overhead (Multiple function calls before this point + requiring the user to enter a 7th digit for the date to set weekday), compared to simply storing it and accessing it along with the other Date/Time values in the DS1302.

```
void GetWeekDay()
{
    switch(AllValues[1][6])
    {
          case  'W':  weekDay[0]='W',weekDay[1]='K',weekDay[2]='D'; break;
          case  '1':  weekDay[0]='M',weekDay[1]='O',weekDay[2]='N'; break;
          case  '2':  weekDay[0]='T',weekDay[1]='U',weekDay[2]='E'; break;
          case  '3':  weekDay[0]='W',weekDay[1]='E',weekDay[2]='D'; break;
          case  '4':  weekDay[0]='T',weekDay[1]='H',weekDay[2]='U'; break;
          case  '5':  weekDay[0]='F',weekDay[1]='R',weekDay[2]='I'; break;
          case  '6':  weekDay[0]='S',weekDay[1]='A',weekDay[2]='T'; break;
          case  '7':  weekDay[0]='S',weekDay[1]='U',weekDay[2]='N'; break;
          case  '8':  AllValues[1][6]='1';weekDay[0]='M',weekDay[1]='O',weekDay[2]='N'; break;
      default :  weekDay[0]='?',weekDay[1]='?',weekDay[2]='?'; break;
    }
```

```
132   void Nextday()
133   {
134       if((Year_ds!=Year_ds_Previous)&&(Year_ds_Previous!=0))
135       {
136             switch(AllValues[1][6])
137       {
138             case  'W':  break;
139             case  '1': AllValues[1][6]='2'; break;
140             case  '2': AllValues[1][6]='3'; break;
141             case  '3': AllValues[1][6]='4';break;
142             case  '4': AllValues[1][6]='5'; break;
143             case  '5': AllValues[1][6]='6'; break;
144             case  '6': AllValues[1][6]='7'; break;
145             case  '7': AllValues[1][6]='1'; break;
146
147          default :  weekDay[0]='?',weekDay[1]='?',weekDay[2]='?'; break;
148       }
```

Figure 18 , NextDay() method in ACW Main file

I recognize that although my functions work there were more efficient methods to solve these two issues

In contrast the LCD, Buzzer and DS18B20 can be used flawlessly to display any integer or character in a given position, emit sound when required and measure temperature as needed.

CONCLUSIONS DRAWN

Having pointed out the weaknesses of the program I am still confident that the system I have designed is a stable consumer-ready product that is able to perform all the functions outlined in the specification.

LESSONS LEARNT

- ✓ How to make use of relevant resources including datasheets and code examples to create hardware specific drivers.
- ✓ How to interpret and convert data into many different types using various methods.
- ✓ How to efficiently create programs for resource constrained devices.

5 REFERENCES

(IMB, 2018) - https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.ioaq100/ascii_table_appendix.htm

(Pic 16, 2018) http://www.pic16.com/en/Download.htm

6 APPENDIX

ASCII TABLE

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

IMB (2018), Ascii table used for conversion of values