Climate control system

Embedded Systems Development 600085

Group 6.1

(Ieaun Roberts)

[WC : 2133]

[Group Report]

Contents

1 INTRODUCTION

Once a luxury item, Climate control systems are devices that are common place in most homes all around modern day UK that are used to measure several environmental values using numerous sensors all working in unison.

Here-in you will find the documentation to support the Embedded Systems Development 600085 Coursework.

2 PROJECT MANAGEMENT

Group organization:

Interface:	**Ieaun Roberts**
Real time clock	✓
Temperature Sensor	✓
LCD panel	✓
Input switches	✓
Sounder	✓
I/O ports	✓

Figure 1, Interface allocation

3 DRIVERS

3.1 SOUNDER:

Using Port E:

Function	**Description**
han_Display_Results()	Displays all current values for temperature, trigger temperature, date and time. Once validation criteria has been met indicating a temperature bellow the trigger temperature and in the heating output time schedule, method BuzzerActivate() is called to start the sounder .This function is continuously called and will read updated values and continue to activate the sounder at the end of each loop if criteria is met, if criteria is not met then sounder is turned off.

Figure 2, Usability of sounder driver

3.2 REAL TIME CLOCK:

Using Port B:

Function	Description
Main()	Initializes the DS1302. If the time and date have both been soft set (both entered and saved to variables but not yet sent to chip), calls CalculateTime(), CalculateDate() and then calls SetTimeMethod() this hard sets the time on the chip and time will begin to pass on LCD. ClockMain() and DateMain() are then called to **read** time and date from DS1302 and store it into the 6 addresses specified by the ClockMain() function and the 6 addresses specified by the DateMain() function. *If the time/date triggers a change in day this function is called to adjust weekday. These updated time, date and weekday values are then displayed using the han_Display_Results() function.
CalculateTime()	Converts 4 **character** values for time into **integer** (Global variables). Combines 2 **integer** values for hour (Tens, Ones) and 2 **integer** values for minute (Tens, Ones) to return time in Hour and Min as **integers**.
CalculateDate()	Same as CalculateTime() but with Date variables. 2 for year, 2 for month, 2 for day.
SetTimeMethod()	Calls CalculateTime() and SetTime().
SetTime()	Sets the time and date on the DS1302 chip. Takes 5 **Integer** values in this sequence representing the following: 1) Hour 2) Minute 3) Year 4) Month 5) Day e.g. set_time(Hour ,Min ,Year_ds ,Month_ds ,Day_ds) *Time and date go hand in hand and if Time is not set then Date will not pass and vice versa. **Both values must be set for either to start.** After initial setup time and date values can be change independently of each other.
ClockMain()	Sends the addresses of 6 char variables to the DS1302 driver where they are updated with the new time. on each loop of the main function this function will be called constantly updating the time and date variables. e.g. ClockMain(&Hour1 ,&Hour2 ,&Min1 ,& Min2 ,&Sec1, &Sec2)
DateMain()	Same definition as ClockMain() but with date. e.g. ClockMain(&Year1 ,&Year2 ,&Month1 ,&Month2 ,&Day1, &Day2)
GetWeekDay()	When date is being set 6 digits represent date and the 7th represents the weekday. This function is used to convert the given number entered as the 7th digit in the range of 1-7, into a 3-letter code representing the day (e.g. Wed ,Fri)
Nextday()	Cycles to the next day after the current weekday. E.g. Wed turns into Thu.

Figure 3, Usability of real time clock driver

3.3 BUTTON INPUT:

Using Port A:
Using Port C:

Function	Description
Main()	This following method is continuously called. To read if a button is being pressed a call is made to method ‘keymatrix()’. Keymatix requires that 2 **integer** variable addresses are parsed as parameters to store the values for the following respectively: **1) Button pressed** **2) Active set** E.g. “keymatrix(&ButtonPressed,&ActiveSet)” EnterData() is then called.
EnterData()	Processes data obtained and calls appropriate initialize, Store in alternate location and display functions according to the following: 1) The Button pressed number: • 0-9 indicates button pressed value • 55 indicates silencer state change (Loud or no buzzer output) • 98 indicates that the user has requested to change a trigger temperature, date or time value. This alters the Active set value. • 99 indicates no button pressed 2) The active set number which indicates which value will be altered if any: • 1=Time • 2=Date • 3=Trigger Temperature • 0=None • Uses Function GetValue() To store appropriate amount of data in each field. E.g. “An active set of 1 indicates a Time change. Therefore, the next button pressed will be used as the 1st hour value if the next button pressed is in the range of 0-9. 2 would equal 2H:MM”
GetValue()	Determines where to store data that is being input according to the active set value and amount of allowed values within the field (E.G. time only allows 4 for HH:MM, Trigger temperature only allows 2 TT)

Figure 4, Usability of button driver

3.4 TEMPERATURE SENSOR:

Using Port A:

Function	Description
Main()	Initializes the DS18B20
han_Display_Results()	**[Line 102]** Calls mainThermo(&**character** tens ,& **character** ones ,& **character** decimals) to update each variable to 1 digit of the current temperature value being read by the temperature sensor. E.G. Given a current temperature of 21.5 the following values will be stored in the given **character** variables in respective order (tens, ones, decimals). 1) 2 2) 1 3) 5 After the value is updated it is then displayed. Explained in detail in LCD driver.

Figure 5, Usability of Temperature Sensor driver

3.5 LCD PANEL:

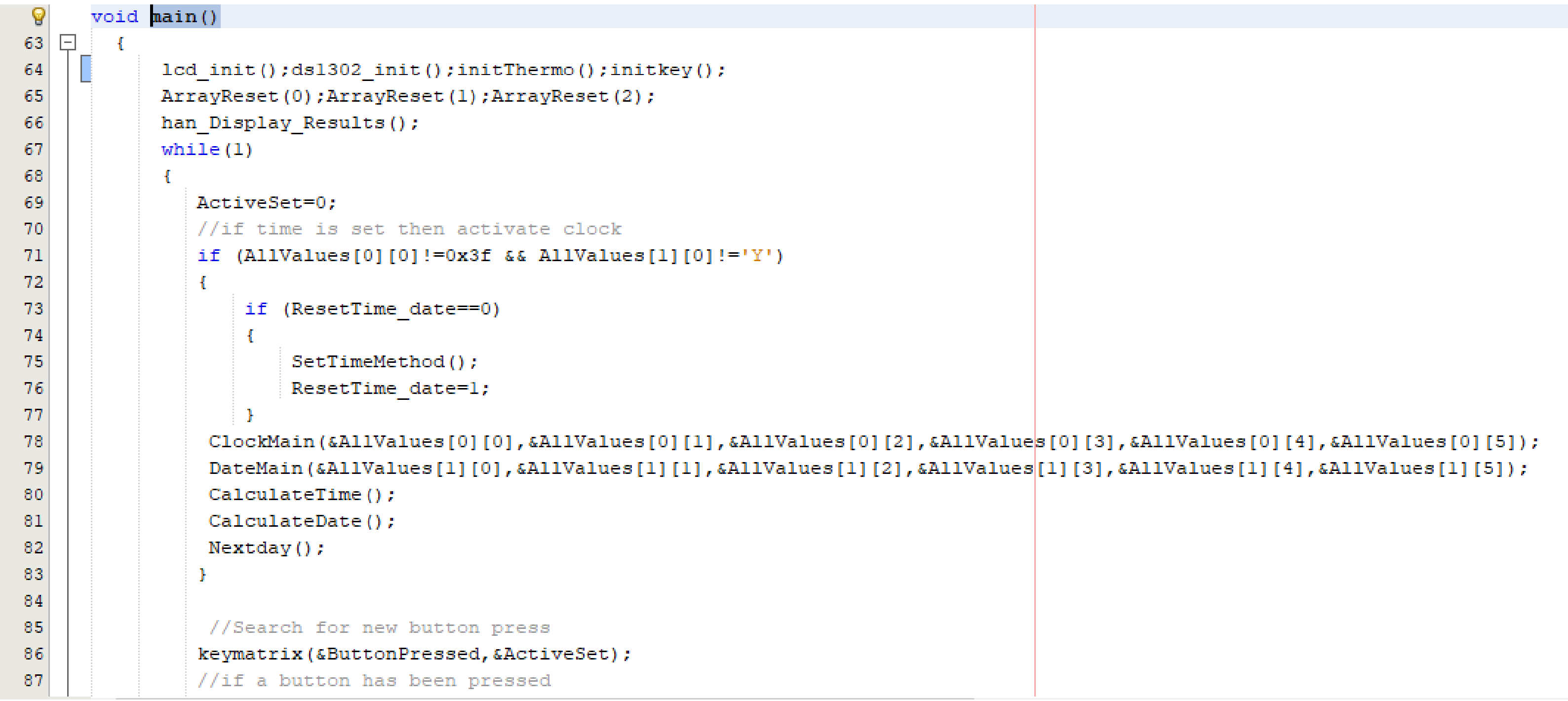
Using Port A:
Using Port D:

Function	Description
Main()	Calls Lcdinit() which initializes the LCD driver and clears the display. Enters infinite loop and Calls han_Display_Results().
han_Display_Results()	**[Line 104]** Predefined text is sent to the LCD driver along with the values for temperature, trigger temperature, date and time to be displayed by making use of the following functions in the following sequence with the following parameters: 1) han_Display(**character** Location on screen, **character** data to display) 2) send_d(“ ”) insert a space. 3) send_d(**character** single **character** data to display). 4) send_d(**character** Special Character to display) • Repeat steps 2, 3 and 4 according to what needs to be displayed This function is repeatedly called, rewriting new values from given variables each time if they have been updated.

Figure 6, Usability of LCD driver.

4 OVERALL SYSTEM FUNCTION

The main method initializes all drivers and the array "AllVallues[][]" which holds the values for Temperature ,Trigger temperature and Date/Time on first time execution.

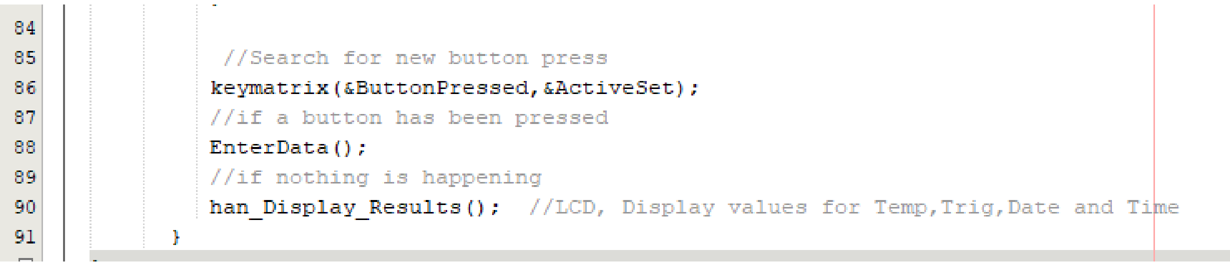


```
void main()
 {
      lcd_init();ds1302_init();initThermo();initkey();
      ArrayReset(0);ArrayReset(1);ArrayReset(2);
      han_Display_Results();
      while(1)
      {
         ActiveSet=0;
         //if time is set then activate clock
         if (AllValues[0][0]!=0x3f && AllValues[1][0]!='Y')
         {
             if (ResetTime_date==0)
             {
                 SetTimeMethod();
                 ResetTime_date=1;
             }
          ClockMain(&AllValues[0][0],&AllValues[0][1],&AllValues[0][2],&AllValues[0][3],&AllValues[0][4],&AllValues[0][5]);
          DateMain(&AllValues[1][0],&AllValues[1][1],&AllValues[1][2],&AllValues[1][3],&AllValues[1][4],&AllValues[1][5]);
          CalculateTime();
          CalculateDate();
          Nextday();
         }

          //Search for new button press
         keymatrix(&ButtonPressed,&ActiveSet);
         //if a button has been pressed
```

Main Infinte Loop

The system checks if the initialized value for time and date have been changed

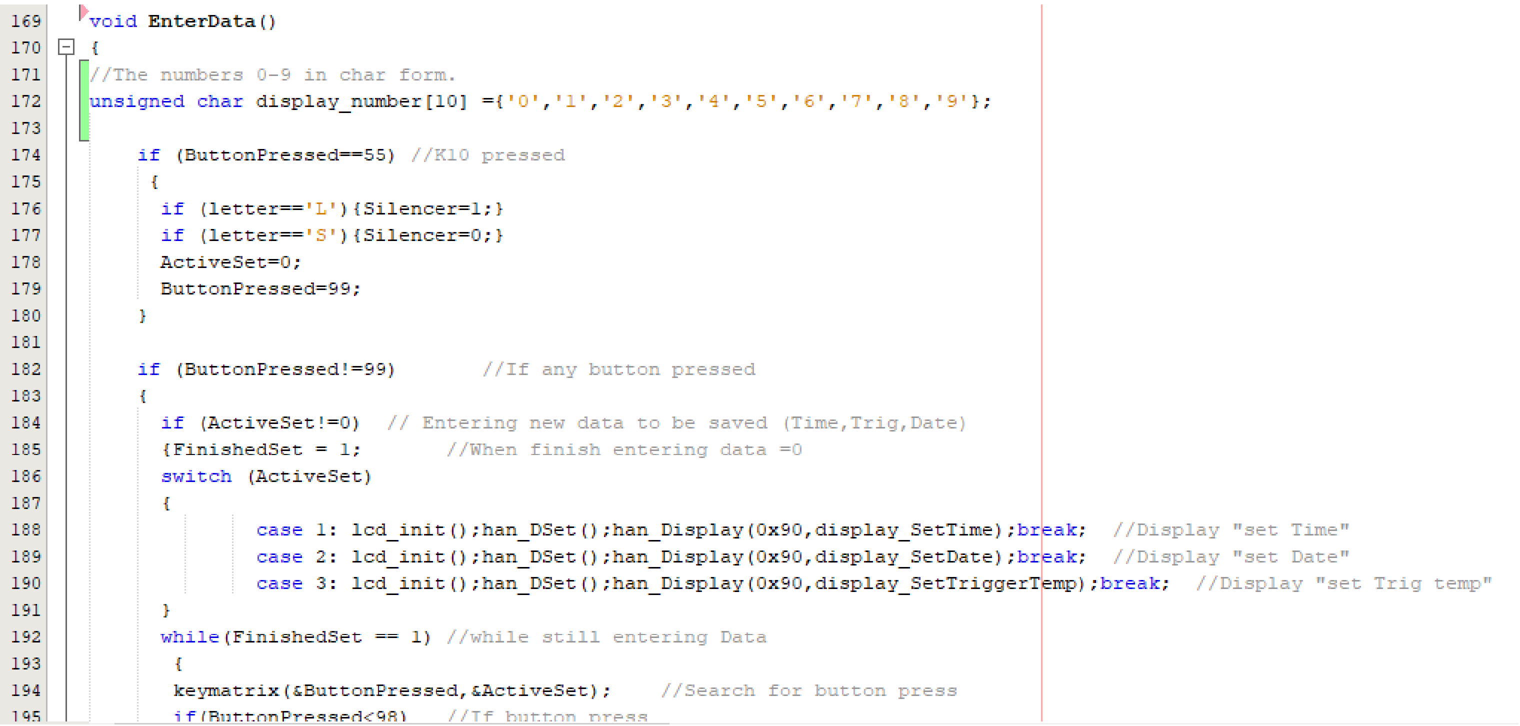
1. On first time execution the conditional if statement is bypassed as the values would still be at default. The program then scans for a button press using **keymatrix(&ButtonPressed,&ActiveSet)**.



```

          //Search for new button press
         keymatrix(&ButtonPressed,&ActiveSet);
         //if a button has been pressed
         EnterData();
         //if nothing is happening
         han_Display_Results();  //LCD, Display values for Temp,Trig,Date and Time
      }
```

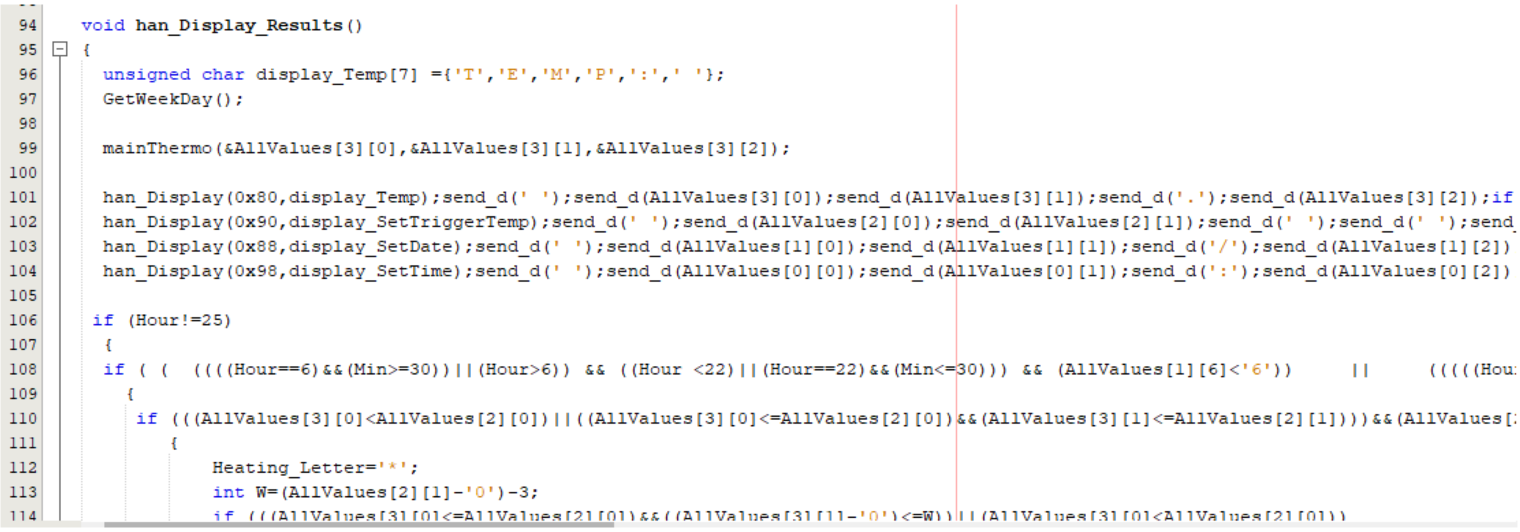
The variables "ActiveSet" and "ButtonPressed" are crucial in the operation of the system. "ButtonPressed" holds the value of the last button pressed and holds either 0-9,98 if an active set value was selected or 99 if no button pressed, "ActiveSet" is used to signal when a value change (Trigger temperature, Date, Time) is requested. ActiveSet = 1(Time), =2(Date), =3(Trigger Temperature).

EnterData() is then called which compares the value stored in “ButtonPressed” against various possibilities to understand what type of data has been entered. If “ActiveSet” is not 0 then a value is being requested to change. The system prompts the user to enter the data for said field and saves each digit to the array “AllValues[][]” continues to search for values until the required amount has been captured E.G time requires 4 digits (21:00).



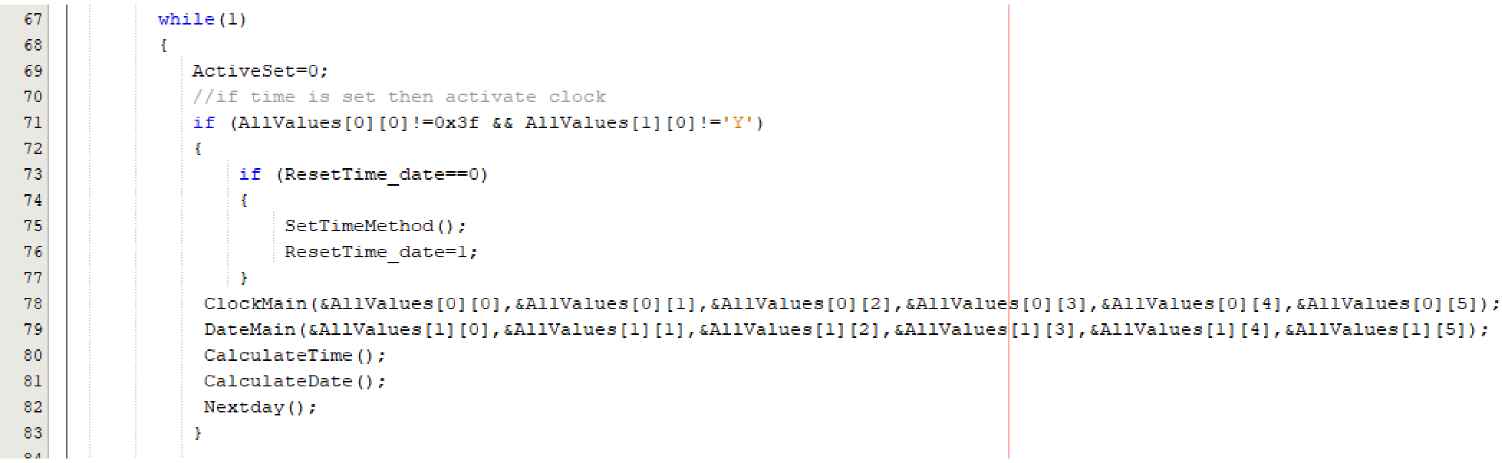
```c
void EnterData()
{
//The numbers 0-9 in char form.
unsigned char display_number[10] ={'0','1','2','3','4','5','6','7','8','9'};

    if (ButtonPressed==55) //K10 pressed
     {
      if (letter=='L'){Silencer=1;}
      if (letter=='S'){Silencer=0;}
      ActiveSet=0;
      ButtonPressed=99;
    }

    if (ButtonPressed!=99)       //If any button pressed
    {
      if (ActiveSet!=0)  // Entering new data to be saved (Time,Trig,Date)
      {FinishedSet = 1;        //When finish entering data =0
      switch (ActiveSet)
      {
              case 1: lcd_init();han_DSet();han_Display(0x90,display_SetTime);break;  //Display "set Time"
              case 2: lcd_init();han_DSet();han_Display(0x90,display_SetDate);break;  //Display "set Date"
              case 3: lcd_init();han_DSet();han_Display(0x90,display_SetTriggerTemp);break;  //Display "set Trig temp"
      }
      while(FinishedSet == 1) //while still entering Data
       {
       keymatrix(&ButtonPressed,&ActiveSet);     //Search for button press
       if(ButtonPressed<98)    //If button press
```

Han_Display_Results() is then called to display all the values stored in “AllValues[][]” and activate the heating output if in the correct time period. This function also calls **MainThermo()** which updates the values of the current temperature.

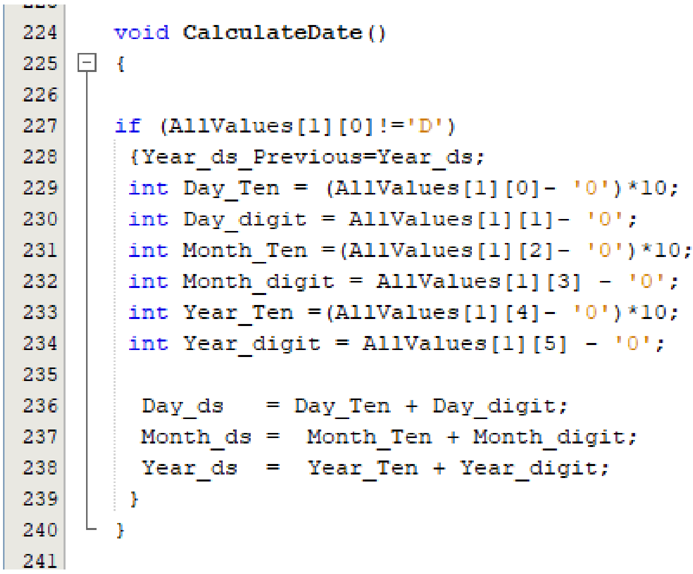


```c
void han_Display_Results()
{
 unsigned char display_Temp[7] ={'T','E','M','P',':',' '};
 GetWeekDay();

 mainThermo(&AllValues[3][0],&AllValues[3][1],&AllValues[3][2]);

 han_Display(0x80,display_Temp);send_d(' ');send_d(AllValues[3][0]);send_d(AllValues[3][1]);send_d('.');send_d(AllValues[3][2]);if
 han_Display(0x90,display_SetTriggerTemp);send_d(' ');send_d(AllValues[2][0]);send_d(AllValues[2][1]);send_d(' ');send_d(' ');send_
 han_Display(0x88,display_SetDate);send_d(' ');send_d(AllValues[1][0]);send_d(AllValues[1][1]);send_d('/');send_d(AllValues[1][2])
 han_Display(0x98,display_SetTime);send_d(' ');send_d(AllValues[0][0]);send_d(AllValues[0][1]);send_d(':');send_d(AllValues[0][2])

if (Hour!=25)
 {
 if ( (  ((((Hour==6)&&(Min>=30))||(Hour>6)) &&  ((Hour <22)||(Hour==22)&&(Min<=30))) &&  (AllValues[1][6]<'6'))      ||      (((((Hou
   {
    if (((AllValues[3][0]<AllValues[2][0])||((AllValues[3][0]<=AllValues[2][0])&&(AllValues[3][1]<=AllValues[2][1])))&&(AllValues[
      {
          Heating_Letter='*';
          int W=(AllValues[2][1]-'0')-3;
          if (((AllValues[3][0]<=AllValues[2][0])&&((AllValues[3][1]-'0')<=W))||(AllValues[3][0]<AllValues[2][0]))
```

2. The System then returns to the beginning of the **Main()** conditional Loop. Assuming the values for Date and Time have now been set and stored in “AllVallues[][]” they are now sent to the Real Time Clock to set the time and date on first operation. Hereafter the values are parsed and updated using **ClockMain**(), **DateMain**() and the calculate functions to convert the characters values to integer for Date/Time.

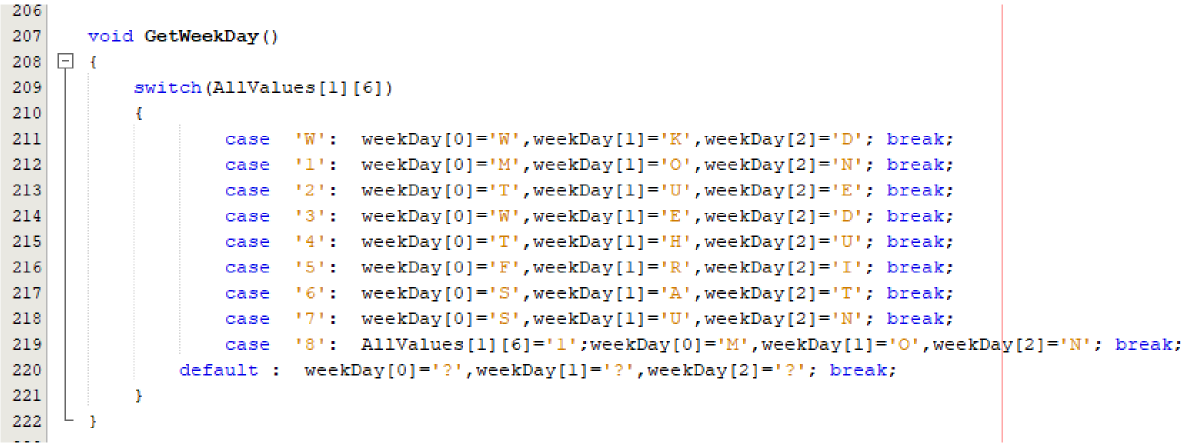


```
    while(1)
    {
      ActiveSet=0;
      //if time is set then activate clock
      if (AllValues[0][0]!=0x3f && AllValues[1][0]!='Y')
      {
          if (ResetTime_date==0)
          {
              SetTimeMethod();
              ResetTime_date=1;
          }
       ClockMain(&AllValues[0][0],&AllValues[0][1],&AllValues[0][2],&AllValues[0][3],&AllValues[0][4],&AllValues[0][5]);
       DateMain(&AllValues[1][0],&AllValues[1][1],&AllValues[1][2],&AllValues[1][3],&AllValues[1][4],&AllValues[1][5]);
       CalculateTime();
       CalculateDate();
       Nextday();
      }
```



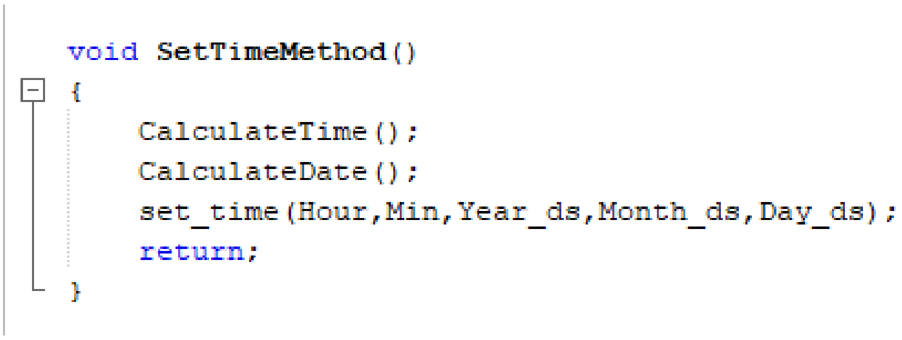
```
void CalculateDate()
{

if (AllValues[1][0]!='D')
 {Year_ds_Previous=Year_ds;
 int Day_Ten = (AllValues[1][0]- '0')*10;
 int Day_digit = AllValues[1][1]- '0';
 int Month_Ten =(AllValues[1][2]- '0')*10;
 int Month_digit = AllValues[1][3] - '0';
 int Year_Ten =(AllValues[1][4]- '0')*10;
 int Year_digit = AllValues[1][5] - '0';

  Day_ds   = Day_Ten + Day_digit;
  Month_ds =  Month_Ten + Month_digit;
  Year_ds  =  Year_Ten + Year_digit;
 }
}
```

GetWeekDay() is Used to convert the 7th digit entered when setting the date into the weekday.



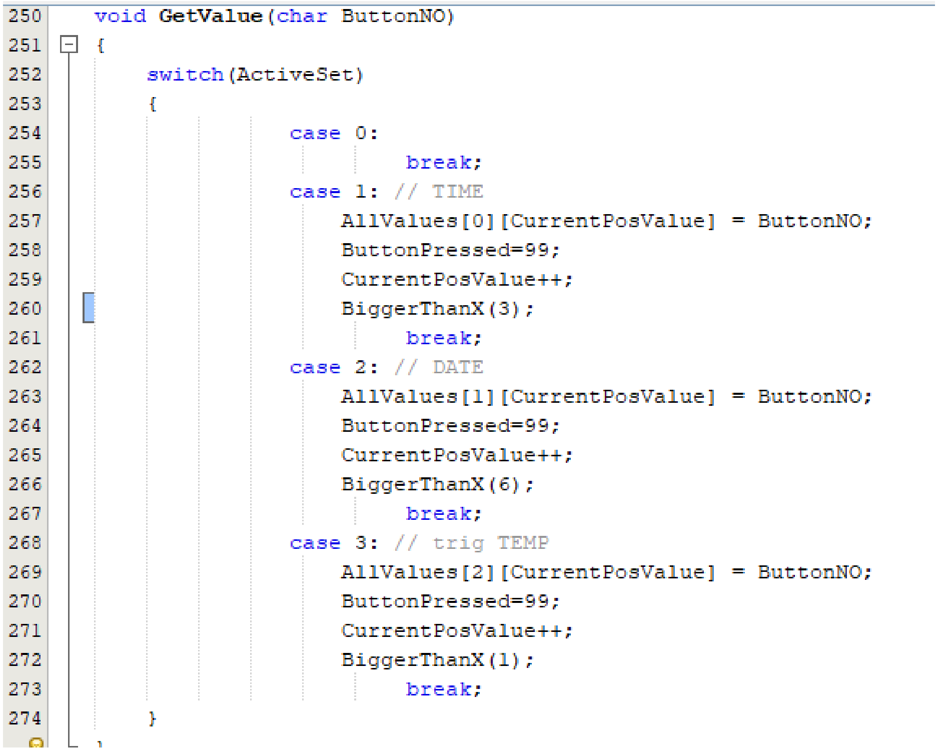
```
void GetWeekDay()
{
    switch(AllValues[1][6])
    {
            case  'W':  weekDay[0]='W',weekDay[1]='K',weekDay[2]='D'; break;
            case  '1':  weekDay[0]='M',weekDay[1]='O',weekDay[2]='N'; break;
            case  '2':  weekDay[0]='T',weekDay[1]='U',weekDay[2]='E'; break;
            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;
    }
}
```

SetTIme() sends altered integer versions of values in “AllValues[][]” to the Set_time() method to save values to the Real Time clock device for processing.

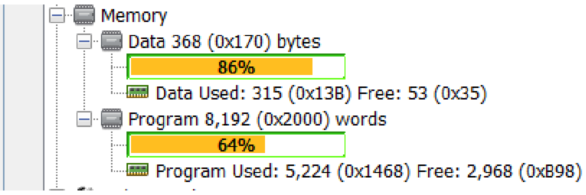


```
void SetTimeMethod()
{
    CalculateTime();
    CalculateDate();
    set_time(Hour,Min,Year_ds,Month_ds,Day_ds);
    return;
}
```

GetValue() Stores data into the correct position in array.



```
void GetValue(char ButtonNO)
{
    switch(ActiveSet)
    {
                case 0:
                        break;
                case 1: // TIME
                    AllValues[0][CurrentPosValue] = ButtonNO;
                    ButtonPressed=99;
                    CurrentPosValue++;
                    BiggerThanX(3);
                        break;
                case 2: // DATE
                    AllValues[1][CurrentPosValue] = ButtonNO;
                    ButtonPressed=99;
                    CurrentPosValue++;
                    BiggerThanX(6);
                        break;
                case 3: // trig TEMP
                    AllValues[2][CurrentPosValue] = ButtonNO;
                    ButtonPressed=99;
                    CurrentPosValue++;
                    BiggerThanX(1);
                        break;
    }
}
```



```
Memory
  Data 368 (0x170) bytes
    86%
    Data Used: 315 (0x13B) Free: 53 (0x35)
  Program 8,192 (0x2000) words
    64%
    Program Used: 5,224 (0x1468) Free: 2,968 (0xB98)
```

Figure 7, System memory usage

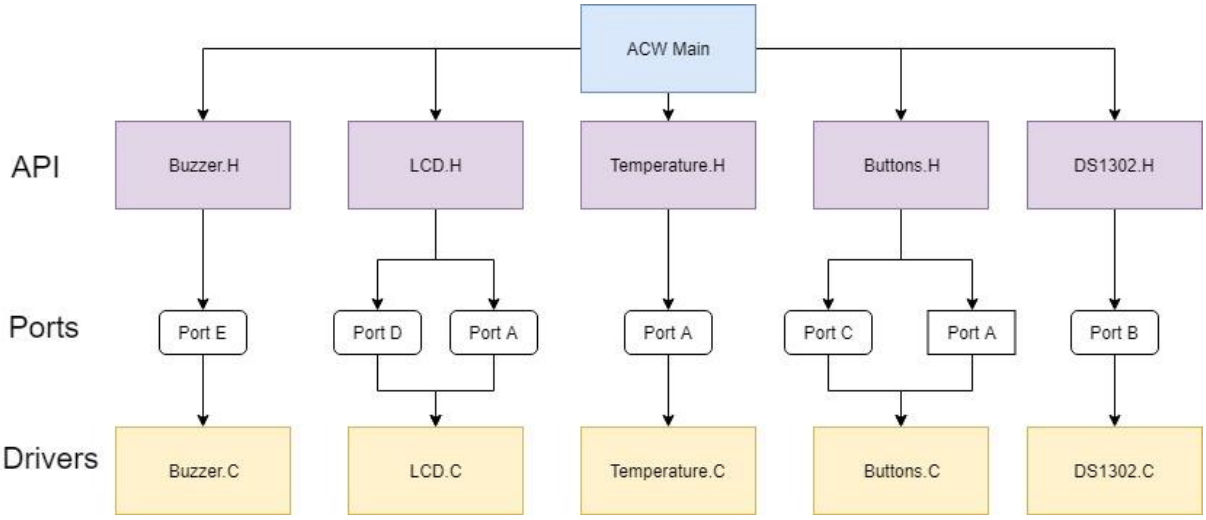


Figure 8, Diagram representing relationships and ports by of system

5 USER MANUAL

Here-in we will discuss the separate segments of the device relevant to the user, the system functions when in normal use, the layout of the user interface and finally FAQs & Troubleshooting.

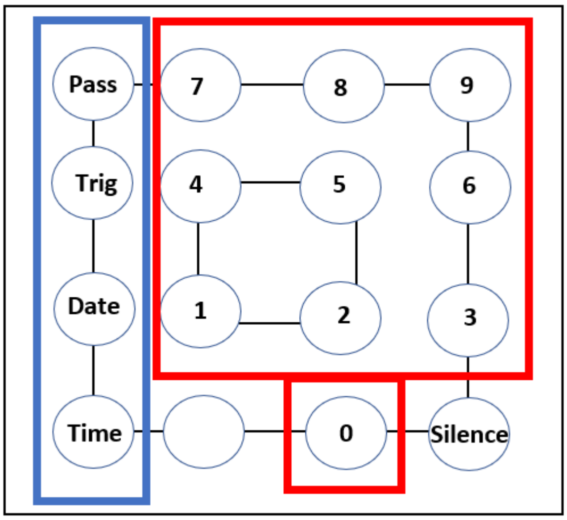
5.1 COMPONENTS

To get started using your new climate system, you must first familiarize yourself with several aspects of the device, namely the LCD display, Buttons (4x4 Key Matrix) and the reset button.

Buttons: The buttons can be split up into 3 sections according to functionality.

1) Section 1 (Red/ Number Pad)
   a) Buttons used to input numerical data into the device, used in conjunction with the Set Functions.
2) Section 2 (Blue/ Set Functions)
   a) These buttons are used to set data to the device.
   b) E.g. Press and hold the time button to enter 'time set mode' and by making use of the Number pad enter your time.
3) Section 3 (Silence Button)
   a) This button controls the activation of 'Silent mode' which restricts the device from outputting any audio to the Buzzer.

Figure 9, Button Layout of Device



LCD: The Display is used to output all data/options to the user and can be split into 3 sections according to what they display:

1) System Values (Purple Arrows)
2) Weekday Value (Red Arrow)
   a) E.g. (Mon, Tue, Wed)
3) Silent mode (Blue Arrow)
   a) L if 'Loud mode' is active
   b) S if 'Silent mode' is active

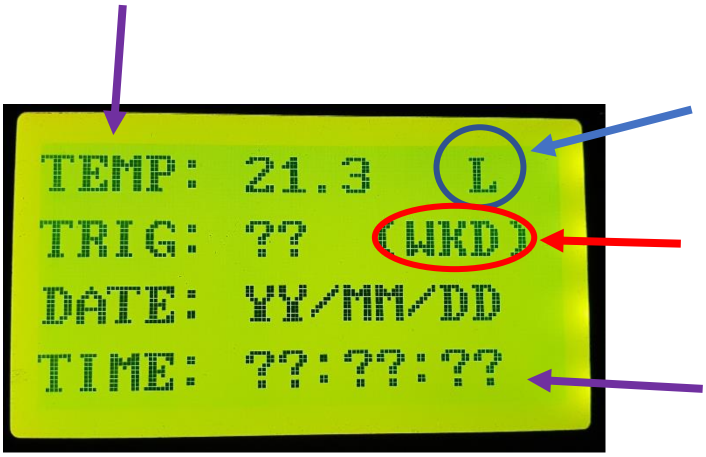


Figure 10, Interface of device

Reset Button: Used to completely reset device to default settings.

5.2 System Functions

Set Time:

To set the time on the device the user must first press the set time button (see components) followed by inputting the specified time by making use of the numerical pad. Users must then enter 4 values representing the time in the 24-hour clock format E.G. 23:35. The system will not proceed until 4 values have been entered.

Set Trigger Temperature:

Similarly setting the trigger temperature only requires a user to press the set trigger temperature button (see components) and key in a 2-digit value on the number pad. The system will not proceed until 2 values have been entered.

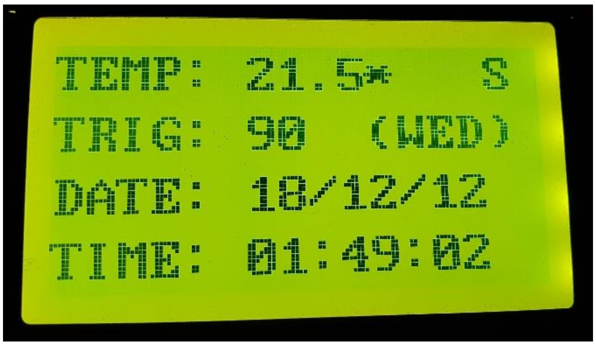
Set Date and Weekday:

Following the same trend as the others, press the set date button (see components), key in 7 values using the key pad. These values represent the following in order: YYMMDDW where Y=Year M=Month D=Day

Enable/Disable Silent Mode:

Toggling between the two modes of the device will allow the user to enter states where the system will either output audio from the buzzer once it activates its heating output or it will not output any audio regardless of the heating output activation. Press the silence button (see components) to toggle between Loud and Silent mode, indicated by an L or S.

5.3 User Interface



```
TEMP:  21.5*    S
TRIG:  90   (WED)
DATE:  18/12/12
TIME:  01:49:02
```

Temperature = 21.5 degrees Celsius
Trigger Temperature = 90 degrees Celsius
Date = Year/Month/Day (18/12/12)
Time = 01:49 am
S = Silent Mode active
(WED) = Wednesday

Figure 11, User interface

6 TESTING

To fully understand the summarized testing results, please refer to Appendix 1 (Testing) which contains outlined system requirements and Test Cases.

Traceability matrix

Requirement Traceability Matrix

System Requirement ID	Technical requirement ID	Test Case ID
S1	T1,T2	1
S2	T1,T2	2
S3	T1,T2	3
S4	T1,T2	4
S5	T4	5
S6	T4	6
S7	T4	7
S8	T4	8
S9	T4	9
S10	T4	10
S11	T2	11
S12	T4	12
S13	T1,T2	13
S14	T1,T2	14

Technical requirements

Tec ID	Description	Requirements
T1	Date/ Time/ Temperature must be Set	Date,Time,Temp!= null
T2	Required devices are connected and functional	DS18B20, LCD , Real Time Clock ,Sounder , Buttons, EEPROM = Connected
T3	Trigger temperature must be set	TriggerTemp !=null

T4	T1, T2, T3 encapsulation	Date,Time,Temp!= null DS18B20, LCD , Real Time Clock ,Sounder , 4x4 key matrix = Connected TriggerTemp !=null

7 Critical Evaluation

7.1 Individual contribution

As I was tasked with completing this group work ACW on my own, every deliverable was created solely by myself or was altered from code already supplied by PIC16. (Pic 16, 2018)

The 2 largest problems I faced while completing this ACW were 1) The amount of knowledge I had to have of the inner workings of the device drivers and 2) the Sheer amount of work in units of hours it took to complete this project alone (over 150 hours completing this project).

7.2 Conclusions Drawn

I am confident that the create climate control system I have designed for the QL200 board is a stable consumer-ready product that is able to perform all the functions outlined in the specification being:

- Setting and displaying of fields.
- Time specific audio and visual heating output.

All by making use of separate drivers for individual devices and APIs to access them.

With this device a user will be able to measure, process and output Date/Time/Temperature/Trigger Temperature values.

7.3 Lessons Learnt

- Make use of relevant resources including datasheets and code examples.
- Begin coursework early in its project lifecycle to allow ample time to solve any problems that you may face.
- Make use of good coding practices when using the C language.
- How to efficiently create programs for resource constrained devices.
- When working with drivers ensure to add relevant code to the specific driver.

8 References

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

9 APPENDIX

9.1 APPENDIX 1 (TESTING)

System Requirements

SR ID	Module name	Description
S1	Set Date/Time	Users will be able to enter the current Date, Time and Weekday.
S2	Display Date/Time	The system should be able to display the current time, date and week day
S3	Set Trigger Temperature	A User will be able to enter their required trigger temperature.
S4	Display Current Temperature	The system should be able to display the current temperature
S5	Below Temp	Once the system reads temperatures below the trigger value the heating output will be triggered. Display that the temperature is below the set trig and activate buzzer alarm.
S6	Deactivate Alarm	The system should turn the Buzzer off once a button is pressed.
S7	Above Temp	Temperature is higher than trigger temp, so buzzer and heating output are turned off.
S8	At Trig Temp	Turn heating output on and display temp.
S9	Heating behaviour on	Systems heating functions should output low temps during these times 6:30-22:30 weekdays 7:00-23:00 weekends.
S10	Heating behaviour off	System should not output low temps during these times 22:31-06:29 weekdays 23:01-06:59 weekends.
S11	Password Protection	Require a password to change trigger temp.
S12	Persistent settings	Using the QL200s non-volatile memory [EEPROM] store settings already entered by user regarding date/time and trigger temp.
S13	Out of range time	The system can only store time in a 24-hour clock format 00:00 – 23:59. Any values above this are invalid and treated as such.
S14	Out of range date	Similarly, to the time, the date field does not accept values over a certain amount and will display “invalid” and discard the data if values over a certain amount for each field are entered. MAX values Year = 99 Month=12 Day =31 Weekday=7

Test Case

Test Case ID	SR #	TR #	Test Case	Test steps	Test data	Expected result
1	S1	T1 T2	Set Date/Time/ Weekday	1) Press Date Set Button 2) Enter Date 3) Press Time Set Button 4) Enter Time 5) Observe Output	Date=2018/01/01 Weekday = wed Time=22:00	Date/Time are visible on LCD, along with the weekday.
2	S2	T4	Display fields	Test case 1 + 6) Press Trig Button 7) Enter Trigger temperature	Test case 1 + Trig Temp = 80	Date/Time/ Weekday fields are visible on LCD.
3	S3	T2 T3	Set Trigger Temperature	1) Press Trig Button 2) Enter Trig temperature	Trig=90	The trigger temperature will be visible underneath the current temperature.
4	S4	T2	Display Current Temperature	1) Power on device	Room temperature is 21 degrees	The current temperature is shown on LCD
5	S5	T4	Below Temp	Test case 2 + 8) Ensure loud mode is active (indicated by L)	Test case 2	First an asterisk is shown next to the current temp indicating that it is below the trigger temp then If it is 2 degrees below the trigger temp the Buzzer is activated.
6	S6	T4	Deactivate Alarm	Test case 5 + 9) Press Silent button	Test case 5	Pressing the silent button (Forcing device into silent mode) deactivates the alarm.
7	S7	T4	Above Temp	Test case 2	Test case 2	Once the device reads a temperature above the trigger temperature the heating output is switched off
8	S8	T4	At Trig Temp	Test case 1 + 6) Press Trig Button 7) Enter Trigger temperature	Test case 1 + Trig Temp = 80	An asterisk is displayed next to the current temperature value to indicate it is equal or within 2 degrees from the trigger temp

9	S9	T4	Heating behaviour on	Test Case 2 + 8) Time is within bounds of weekday/weekend cycle	Time = 06:30 - 22:30 weekdays Or Time = 07:00 - 23:00 weekends.	Device will output any heating operations during this time
10	S10	T4	Heating behaviour off	Test Case 2 + 8) Time is out of bounds of weekday/weekend cycle	Time = 22:31-06:29 weekdays Time = 23:01-06:59 weekends.	Device will not output any heating operations during this time
11	S11	T2	Password Protection			
12	S12	T2	Persistent settings			
13	S13	T1 T2	Out of range time	1) Press Time Button 2) Enter Time	Time = 25:00	‘Invalid’ will be displayed on the LCD following an attempt at setting an out of range time. The result will not be saved.
14	S14	T1 T2	Out of range Date/Weekday	1) Press Date Set Button 2) Enter Date	Year = 99 Month=12 Day =31 Weekday=7	‘Invalid’ will be displayed on the LCD following an attempt at setting an out of range date. The result will not be saved.

9.2 Appendix 2 (Drivers)

9.2.1 A) Real Time Clock

Function	Description	Parameters
DateMain()	Calls “get_time” function to receive the current date and stores each char result in its specific pointer.	• char *Year1_Temp • char *Year2_Temp • char *Month1_Temp • char *Month2_Temp • char *Day1_Temp • char *Day2_Temp
ClockMain()	Calls “get_time” function to receive the current time and stores each char result in its specific pointer.	• char *Hour1_Temp • char *Hour2_Temp • char *Min1_Temp • char *Min2_Temp • char *Sec1_Temp • char *Sec2_Temp
ds1302_init()	Initializes device, set RB1 to input, rest to output	
time_read_1()	Reads data stored on the DS1302 one byte at a time.	
set_time()	Sets time and date by storing values into variables then writing them to the DS1302 one byte at a time.	• int Hour • int Min • int Day_st • int Month_st • int Year_st
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	

9.2.2 B) Sounder

Function	Description	Parameters
BuzzerActivate()	Sets Port E as output and calls sound200ms(). On return sets Port E to Input.	
sounddelay0()	Adjustable delay Function	• unsigned char delay_count
sound200ms()	Sets Maximum/Minimum frequency values for buzzer. Cycles through a Loop which set port E to output turning the buzzer on,delays for an amount of time, then Sets Port E as input turning the buzzer off.	

9.2.3 C) Button Input

Function	Description	Parameters
keymatrix()	Calls initialization, Stores value of the button pressed into *ButtonPressed	• int *ButtonPressed_, • int *ActiveSet
initkey()	initialize function	
scan()	Search for a Button press.	
display()	Assign the button pressed to a numerical value based on its position.	• int x

9.2.4 D) Temperature Sensor

Function	Description	Parameters
delayThermo()	Delay Function	• char x • char y
initThermo()	Initializes device, sets Port A and D to output	
mainThermo()	Calls functions initializing device, receives temperature data and then converts it into an understandable format which is then copied into the given pointers.	• int *Tens_Temp • int *Ones_Temp • int *Decimals_Temp
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	• uch val
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	

9.2.5 E) LCD Panel

Function	Description	Parameters
lcd_init()	initialize function	
clear_p()	Clears the Lcd panel	
han_Number()	Displays individual numbers on LCD in specified position.	• int DisplayPos • int NumberSelected
han_Display()	Displays 1 array of size 0x5 in specified position.	• int DisplayPos • const unsigned char *Data
wr_zb()	Sets the displays X and Y positions.	
send_d()	Send 1 character of output to the LCD display	• unsigned char x
send_i()	Send 1 character of input to the LCD display	• unsigned char x
chk_busy()	Check if the lcd is already busy with another process.	
qushu()	Send each character of a given input which contains multiple characters to the send_d() function 1 character at a time.	• int counts • const unsigned char *ps
delay()	Delay Function	

9.3 APPENDIX 4 (OBJECTIVES AND TASKS)

Objectives and Tasks:

The objectives and their tasks and subtasks are as follows:

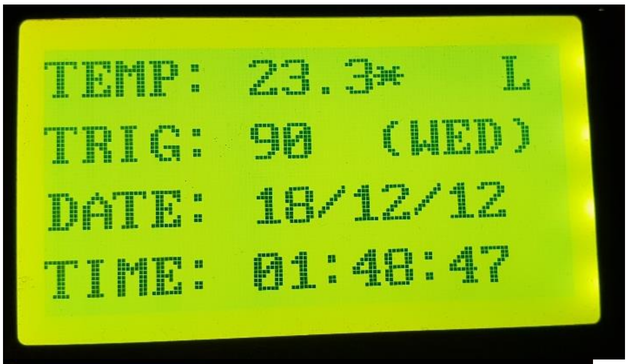
1) Produce a fully functioning climate control system by making use of the PIC QL200 development board with the following features:
   a) Setting of current time, date and week day
   b) Display of current time, date and week day
   c) Setting of a trigger temperature
   d) Display of current temperature
   e) Output control for a time-based heating circuit

2) Demonstrate the overall functionality of the created system during all conditions.(**25%**)
   a) Answer any questions from the audience.

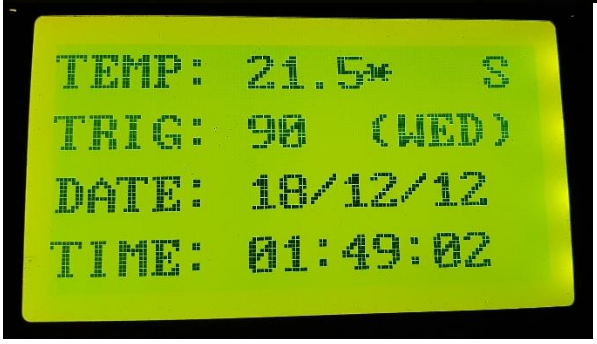
3) Write a group report discussing the following. (**50%**)
   a) The overall system
   b) How the drivers are integrated
   c) Driver specifications
   d) User manual
   e) Testing

4) Each member must write an individual report discussing their contribution to the system development. (**25%**)
   a) Artefacts produced: design, code and test output that you were involved with.
   b) Testing performed: how the systems that you created were verified and validated.
   c) Critical Evaluation: Conclusions drawn, usefulness of techniques, lessons learnt.

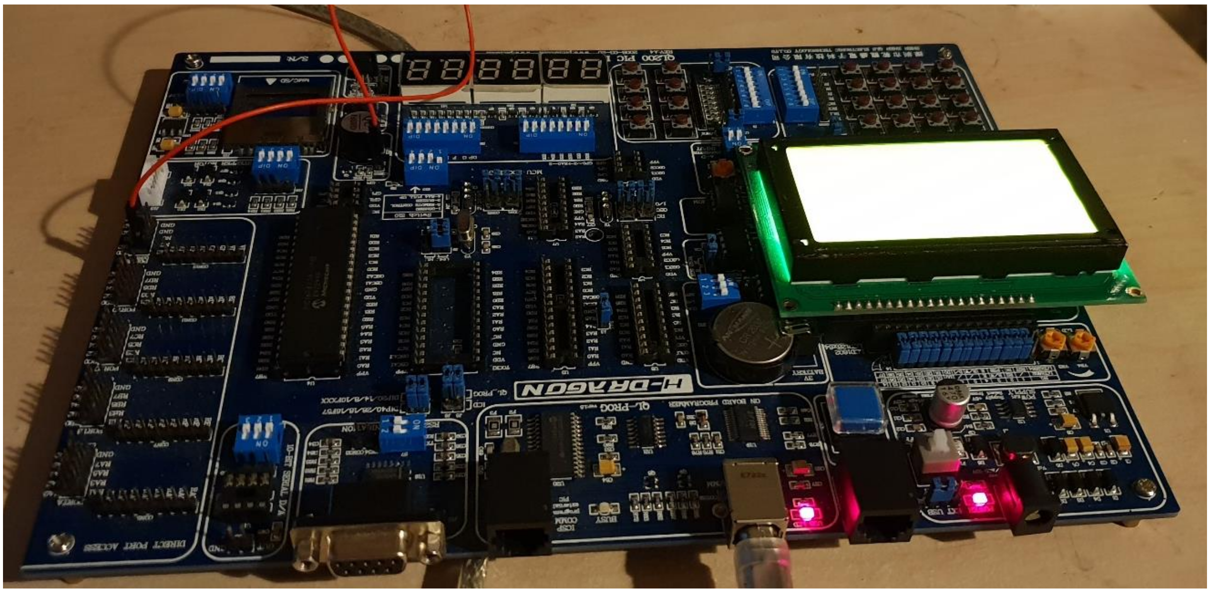
9.4 APPENDIX 4 (ADDITIONAL IMAGES)

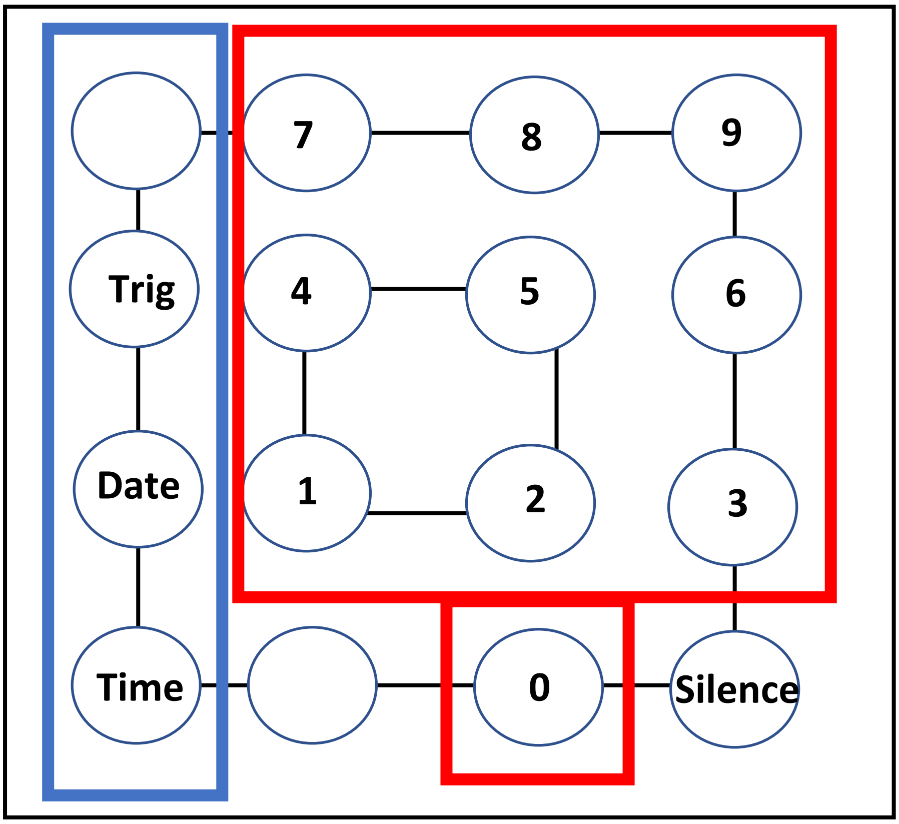


```
TEMP: 23.3*   L
TRIG: 90  (WED)
DATE: 18/12/12
TIME: 01:48:47
```



```
TEMP: 21.5*   S
TRIG: 90  (WED)
DATE: 18/12/12
TIME: 01:49:02
```





9.5 Appendix (FAQs & Troubleshooting)

"My device seems unresponsive as there is nothing being displayed on the screen"

Cause – 9/10 times this is caused by either:

1) The DS18B20 module not being found (faulty QL200 board)
2) The User has not entered enough data to fulfil the needs of the field they are trying to set.

Fix:

1) Gently wiggle the DS18B20 module into place until a reading Is picked up.
2) Continue to enter more data using the number pad, if the result is not satisfactory then simply restart the Set function you were attempting to complete. (If the time is incorrect simply set the time again)

“When the heating output should be enabled my temperature, readings show the value for current temp as 00.0 and the buzzer does not output any audio.”

Cause – Wire for buzzer is connected to thermometer causing the thermometer value to change to 0 when the output is activated.

Fix : Unplug wire from thermometer and plug into buzzer