UNIVERSITY OF KHARTOUM

Faculty of Engineering

Department of Electrical and Electronic Engineering

Microprocessor system design

Semester Project:

MOSQUE WATCH

Group Members:

1.	Israa Mohamed Hamid	144013
2.	Ruba Mutasim Haroun	144031
3.	Fay Majid Ahmed	144050
4.	Moayad Hassan Mohamed	144058
5.	Mohamed Kamal Ahmed	144070

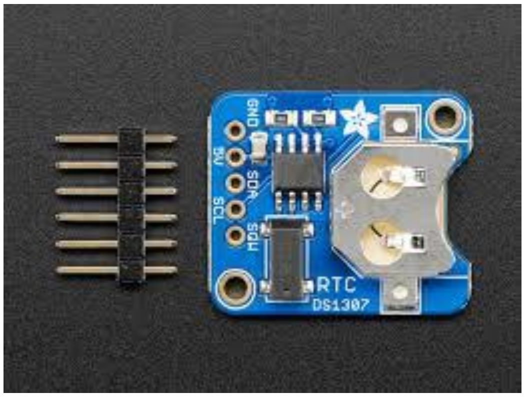
Functional Requirements :

The software does the following:

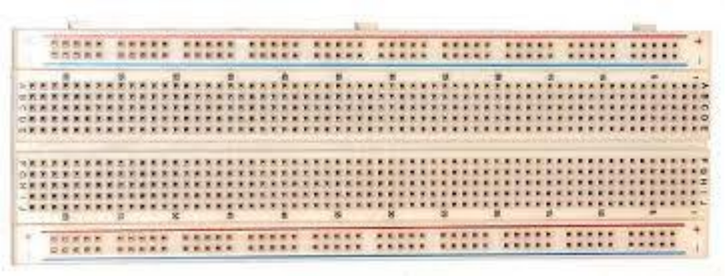
- Displays the current Date (Year/Month/Day)
- Displays the current time (Hour : Minute : Second).
- Displays the next Prayer’s name and time
- Alarms the user when it’s time for a prayer

Technical Requirements:

AVR microcontroller (ATMEGA32)



DS1307(SERIAL REAL TIME CLOCK)



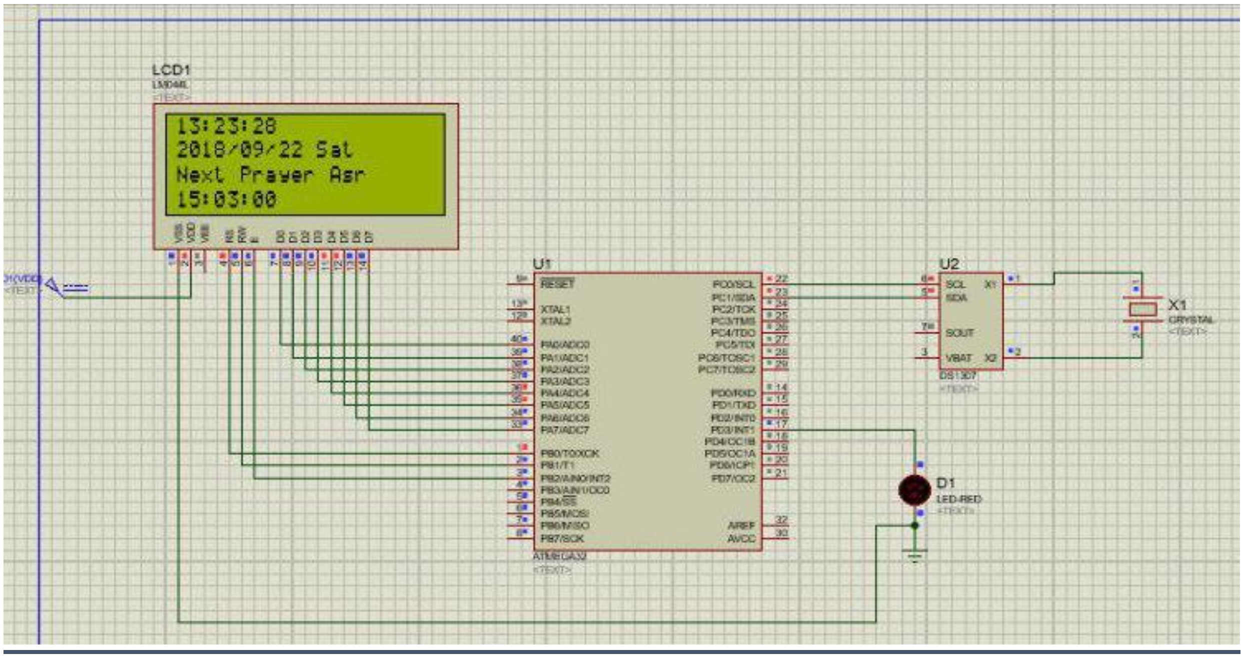
Bread Board

AVR chip programmer

LM041L(16x4 Alphanumerical LCD)

LED(Light Emitting Diode)

- **Design**

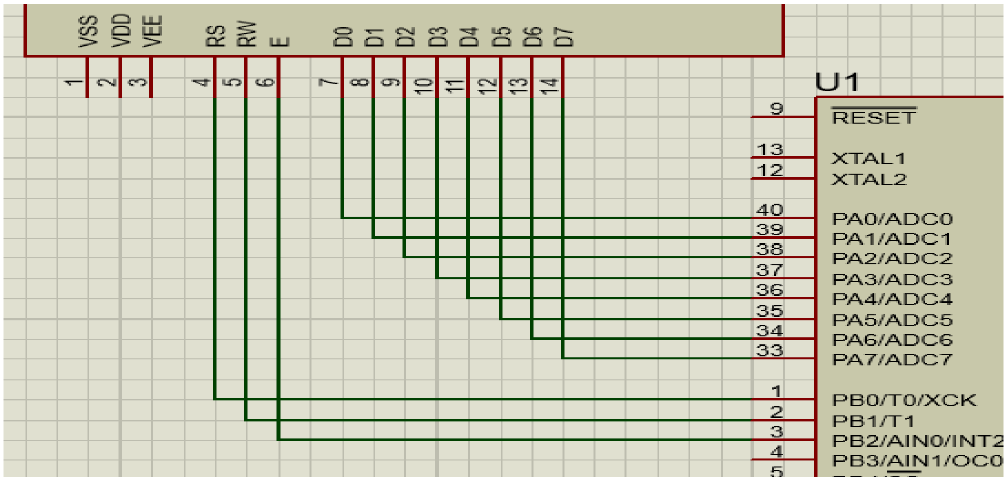


○ *Components:*

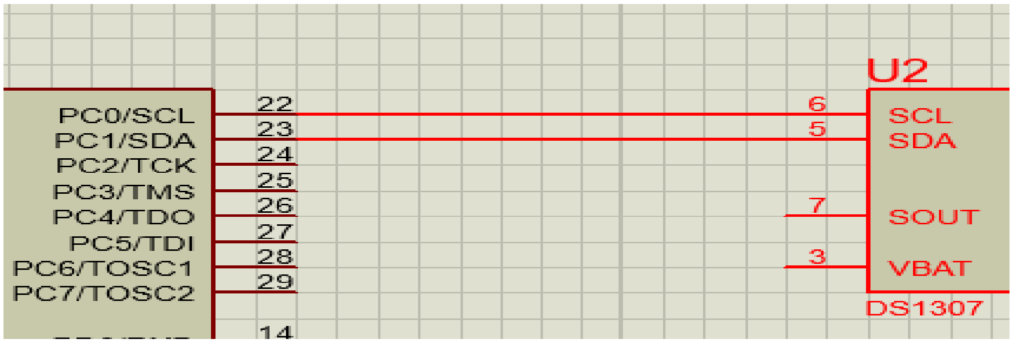
1. ATMEGA32(AVR Microcontroller)
2. CRYSTAL (Quartz Crystal)
3. DS1307(Serial Real Time Clock)
4. LM041L (16x4 Alphanumeric LCD)
5. LED(LED-GREEN)

- ## Implementation outline

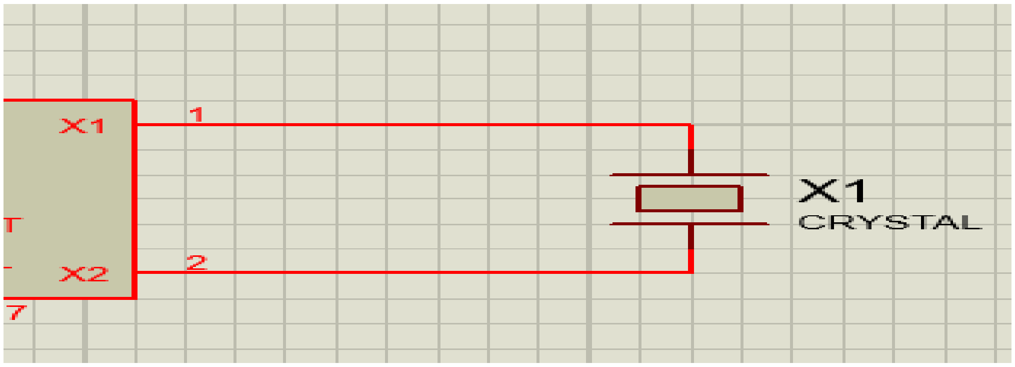
  - Connected the LCD data lines to port A in the atmega32
  - Control lines (RS, RW, EN) connected to port B in the atmega32.



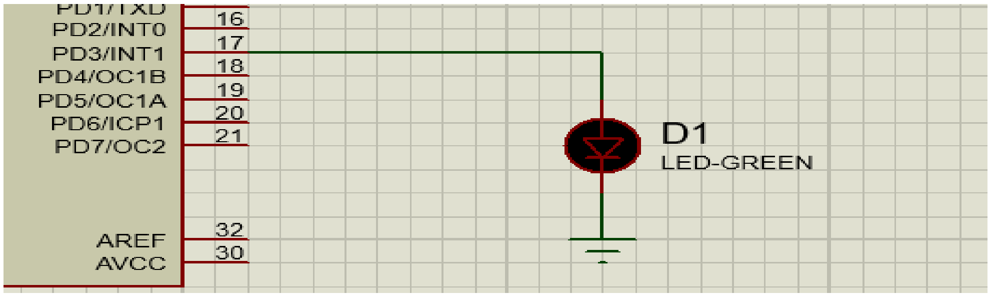
  - We then connected the DS1307(Serial real time clock) data bus (SCL, SDA) with port C (PC0, PC1) respectively



  - DS1307 is connected to the crystal frequency.



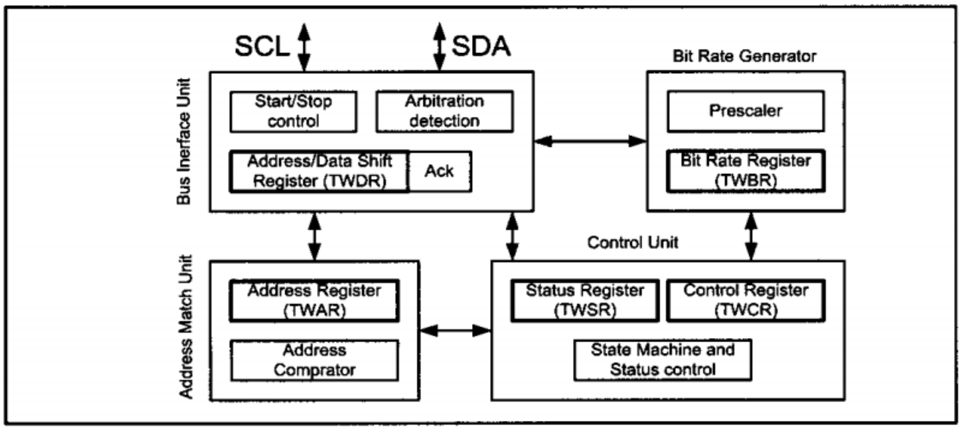
- LED is connected to port D (PD3).

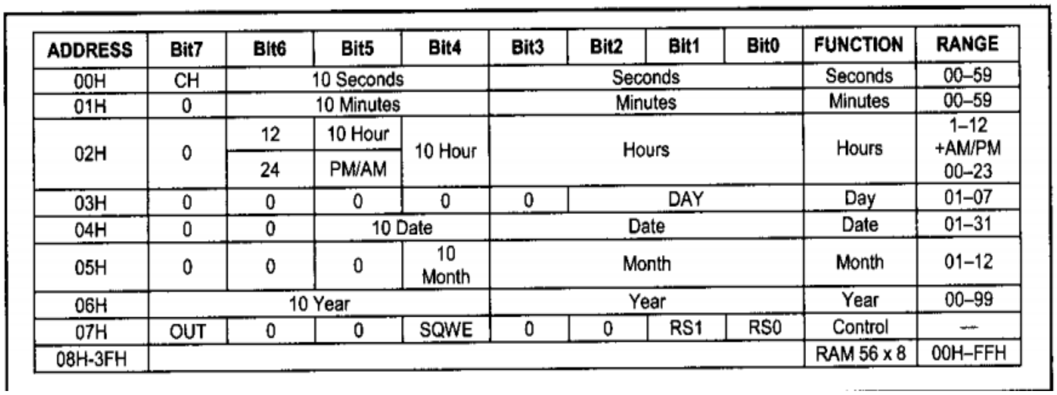


○ *Datasheets:*

- **DS1307(Serial Real Time Clock)** datasheet used:

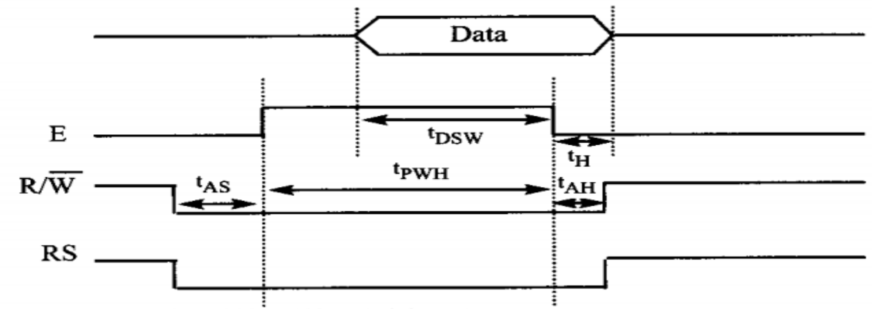
Registers:





ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/AM							
03H	0	0	0	0	0	DAY			Day	01–07
04H	0	0	10 Date		Date				Date	01–31
05H	0	0	0	10 Month	Month				Month	01–12
06H	10 Year				Year				Year	00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H–FFH

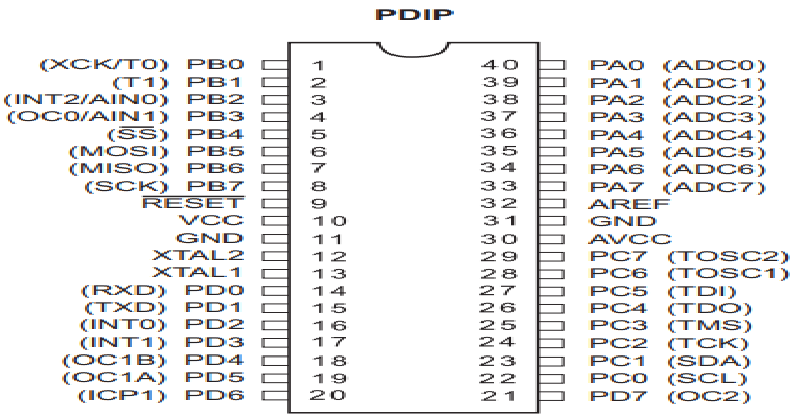
- **<u>LCD datasheet (Timing diagram)</u>**



t_{PWH} = Enable pulse width = 450 ns (minimum)
t_{DSW} = Data setup time = 195 ns (minimum)
t_H = Data hold time = 10 ns (minimum)
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

Figure 12-5. LCD Timing for Write (H-to-L for E line)

- **<u>ATMEGA32 Datasheet</u>**



- **Code**

```
#include <aver/io.h>
#include <util/delay.h>

#define LCD_DPRT PORTA
#define LCD_DDDR DDRA
#define LCD_CDDR DDRB
#define LCD_DPIN PINA
#define LCD_CPRT PORTB
#define LCD_CPIN PINB
#define LCD_RS 0
#define LCD_RW 1
#define LCD_EN 2
/////////////////////I2C Functions
//////// Initialize the I2C communication protocol
void i2c_init(void)
{
  TWSR = 0X00 ; // Pre-scalar = 0
  TWBR = 0X47 ; // frequency = 50k
  TWCR = 0X04 ; // TWEN = ON // Two Wire Enable
}

////////// Start the I2C
void i2c_start(void)
{
  TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN) ; /// clear the interrupt flag, signal a start and enable the I2C
  while (!(TWCR & (1 << TWINT))); // wait untill the interrupt is reset again // meaning the current master has control over the bus
}
///// wirte to the I2C bus
void i2c_write(unsigned char data)
{
  TWDR = data ;// Data to be written to I2C
  TWCR = (1 << TWINT) | (1 << TWEN) ; // clear the interrupt and enable the bus
  while (!(TWCR & (1 << TWINT))); // go if no itnerrupt
}
//// Read from the I2C bus
unsigned char i2c_read(unsigned char ackval)
{
  TWCR = (1 << TWINT) | (1 << TWEN) | (ackval << TWEA) ; // same. if ackva = 1 meaning nect package is to be read, if == 0 then reading is finished
  while (!(TWCR & (1 << TWINT)));
  return TWDR ; // return the data read from the I2C
}
//Stopping control  on the I2C
void i2c_stop(void)
{
  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO) ;
```

```

    //DELAY
    _delay_ms(1);
}
///////////////////////////////////////////
//////////////RTC Functions
///////////////////////initilaize the RTC
void rtc_init(void)
{
    i2c_init();      // initialize I2C
    i2c_start();     // transmit START CONDITION
    i2c_write(0xD0); // DS1307 address
    i2c_write(0x07); // pointer to control Reg TWCR
    i2c_write(0x00); // TWCR = 0
    i2c_stop();           // transmit STOP CONDITION
}
///////////////set the time that RTC starts counting from
void rtc_setTime (unsigned char h, unsigned char m, unsigned char s )
{
    i2c_start();    //I2C START CONDITION
    i2c_write(0xD0);  //Location of the first bit of the RTC
    i2c_write(0x0);   // pointert time Reg // loc of seconds register
    i2c_write(s);   // write seconds
    i2c_write(m);   // write minutes
    i2c_write(h) ;         // write hours
    i2c_stop();            // I2C STOP CONDITION
}
//////////////set the date I2C starts from
void rtc_setDate (unsigned char y, unsigned char m, unsigned char d )
{
    i2c_start();
    i2c_write(0xD0);
    i2c_write(0x04); // location of day register
    i2c_write(d);
    i2c_write(m);
    i2c_write(y);
    i2c_stop();
}

void rtc_getTime (unsigned char *h, unsigned char *m, unsigned char *s )
{
    i2c_start();
    i2c_write(0xD0);
    i2c_write(0x0);
    i2c_stop();

    i2c_start();
    i2c_write(0xD1);
    *s = i2c_read(1);
    *m = i2c_read(1);
    *h = i2c_read(0);
    i2c_stop();
    /////////this is the reading from RTC sequence, it's start (loc of
start + 0 to write) start (loc of start + 1 to read) untill finish
```

```
}


void rtc_getDtae (unsigned char *dn, unsigned char *y, unsigned char *m,
unsigned char *d )
{
  i2c_start();
  i2c_write(0xD0);
  i2c_write(0x03);
  i2c_stop();

  i2c_start();
  i2c_write(0xD1);
  *dn = i2c_read(1);
  *y = i2c_read(1);
  *m = i2c_read(1);
  *d = i2c_read(0);
  i2c_stop();
}
///////////////////////////////////////////////////
//////////////LCD Functions
////////////////////////////////////
void delay_us(unsigned int d){
         while (0 < d)
         {
               _delay_us(1);
               --d;
         }
}
void lcdCommand(unsigned char cmnd){
  LCD_DPRT=cmnd ;
  LCD_CPRT &= ~ (1<<LCD_RS);
  LCD_CPRT &= ~ (1<<LCD_RW);
  LCD_CPRT |= (1<<LCD_EN);
  delay_us(1);
  LCD_CPRT &= ~ (1<<LCD_EN);
  delay_us(100);
}

void lcd_init() {
  LCD_DDDR = 0xff ;
  LCD_CDDR = 0xff;

  LCD_CPRT &= ~ (1<<LCD_EN);
  delay_us(2000);
  lcdCommand(0x38);
  lcdCommand(0x0C);
  lcdCommand(0x01);
  delay_us(2000);
  lcdCommand(0x06);
}

void lcd_gotoxy(unsigned char x , unsigned char y){

  unsigned char firstchar[] ={0x80,0xC0,0x94,0x4D};
  lcdCommand(firstchar[y-1] + x-1);
  delay_us(100);
}

void lcd_Data(unsigned char data) {
  LCD_DPRT = data ;
  LCD_CPRT |= (1<<LCD_RS);
  LCD_CPRT &= ~ (1<<LCD_RW);
  LCD_CPRT |= (1<<LCD_EN);
  delay_us(1);
  LCD_CPRT &= ~(1<<LCD_EN);
  delay_us(100);
}
void lcd_print (char *str) {
  unsigned char i=0;
  while (str[i]!=0)
  {
         lcd_Data(str[i]);
         i++;
  }

}

int main(void)
{
  unsigned char hour,minute,second, year, month, day, WeekDay;
  unsigned char PH[5] = {0x04, 0x11, 0x15, 0x17, 0x18};
  unsigned char PM[5] = {0x27, 0x43, 0x03, 0x47, 0x59};
  unsigned char PN[5] = {'F', 'D', 'A', 'M', 'E'};
         unsigned char i = 0 ;

  lcd_init();
  rtc_init();
  rtc_setTime(0x19,0x41,0x55);
  while(1){
  ////////////display Time
  rtc_getTime(&hour , &minute ,&second);
  /////////// hours
                lcdCommand(0x80);
                lcd_Data('0'+(hour>>4));
                lcdCommand(0x81);
                lcd_Data('0'+(hour&0x0f));
                //
                lcd_Data(':') ;
  /////////////minustes
                       lcd_Data('0'+(minute>>4));
                       lcd_Data('0'+(minute&0x0f));
                       //
                       lcd_Data(':') ;
  ///////// seconds
         lcd_Data('0'+(second>>4));

lcd_Data('0'+(second&0x0f));
       ////////////display Date
rtc_getDtae(&WeekDay, &day, &month, &year);
lcdCommand(0xC0);
lcd_Data('2');
lcd_Data('0');
/////////// years
lcd_Data('0'+(year>>4));
lcd_Data('0'+(year&0x0f));
//
lcd_Data('/') ;
/////////////months
lcd_Data('0'+(month>>4));
lcd_Data('0'+(month&0x0f));
//
lcd_Data('/') ;
///////// days
lcd_Data('0'+(day>>4));
lcd_Data('0'+(day&0x0f));
//
lcd_Data(' ');
/////////WEEKDAY
switch (WeekDay)
{
       case 0x01:
              lcd_print("Sun");
              break;
       case 0x02:
              lcd_print("Mon");
       break;
       case 0x03:
       lcd_print("Tues");
       break;
       case 0x04:
       lcd_print("Wed");
       break;
       case 0x05:
       lcd_print("Thu");
       break;
       case 0x06:
       lcd_print("Fri");
       break;
       case 0x07:
       lcd_print("Sat");
       break;
}


///////////////next prayer mteen
lcdCommand(0x94);
for (i=0;i<=4;i++) {
       if(PH[i] > hour) {

```

```

lcd_Data('0'+(PH[i]>>4));
lcd_Data('0'+(PH[i]&0x0f));
//
lcd_Data(':') ;
lcd_Data('0'+(PM[i]>>4));
lcd_Data('0'+(PM[i]&0x0f));
//
lcd_Data(':') ;
lcd_Data('0') ;
lcd_Data('0') ;
break;
}else if (PH[i]== hour)
{
if (minute< PM[i]){

        lcd_Data(' ');
        lcd_Data('0'+(PH[i]>>4));
        lcd_Data('0'+(PH[i]&0x0f));
        //
        lcd_Data(':') ;
        lcd_Data('0'+(PM[i]>>4));
        lcd_Data('0'+(PM[i]&0x0f));
        //
        lcd_Data(':') ;
        lcd_Data('0') ;
        lcd_Data('0') ;
        break;
        }else if(minute == PM[i]) {
                DDRD = 0xff ;
                PORTD = 1<<3 ;
                if (second >0x5)
                PORTD = 0x00 ;
        }
        else{
        i++;
        lcd_Data('0'+(PH[i]>>4));
        lcd_Data('0'+(PH[i]&0x0f));
        //
        lcd_Data(':') ;
        lcd_Data('0'+(PM[i]>>4));
        lcd_Data('0'+(PM[i]&0x0f));
        //
        lcd_Data(':') ;
        lcd_Data('0') ;
        lcd_Data('0') ;
        break;
        }
```

- `}`
- `}`
- `}`
- `While(1);`
- `}`