/*FARM AUTOMATION MAIN SOURCE CODE*/

```
/*********************************************************************
```

- Controller: ATmega32 (8 MHz)
- Compiler: AVR STUDIO 4.5
- Date: 27 April 2011

```
*********************************************************************/
#include <avr/io.h>
#include <util/delay.h>

#include "I2C.h"
#include "lcd.h"
#include "ds1307.h"

void ShowMainMenu();
void SetTime();
void SetonTime();
void SetoffTime();
void SetDate();
static void InitADC(void);
static long ReadTemp(void);
static void DisplayReading(long Reading);
unsigned char flag1=0,check=0,i=0;

/*********************************************************************
Function Name: InitADC
Purpose      : Initializes ADC
*********************************************************************/
static void InitADC(void)
{

         // 1. ADC Enable
        // 2. ADC Prescaler Select Bits: ADPS2 ADPS1 ADPS0 => 1 1 1 =>
Prescaler = 128

      ADCSRA = _BV(ADEN) |_BV(ADPS2) |_BV(ADPS1) |_BV(ADPS0);
         ADMUX=(1<<REFS0);
            ADMUX=(1<<REFS1);

      ADCSRA |= _BV(ADSC);
      while(!(ADCSRA & (1<<ADIF)));

      return;
}

/*********************************************************************
Function Name: ReadTemp
Purpose      : Returns ADC value after conversion
*********************************************************************/
static long ReadTemp(void)
{
      ADCSRA |= _BV(ADSC);
      //Wait for conversion to complete
        while(!(ADCSRA & (1<<ADIF)));
```

```
       //_delay_ms(5000);
       //Clear ADIF by writing one to it
       ADCSRA|=(1<<ADIF);
      return ADC;
}

/*************************************************************************
Function Name: DisplayReading
Purpose      : Displays voltage reading on LCD
*************************************************************************/
static void DisplayReading(long Reading)
{
      int d2, d3, d4;
      int voltage;
      unsigned char a[2]={0xDF,'\0'};
      voltage= Reading/4;
      d4=voltage%10;
      voltage=voltage/10;
      d3=voltage%10;
      d2=voltage/10;
      if((d4<=3)&&(d3<=3)&&(d2==0))
            {
                  DDRB|=(1<<PB5)|(1<<PB3);//sprinkler
                   PORTB &=0xD7;
            }

   if((d4>3)&&(d3=3)&&(d2==0))
      {
            check=PINC;
            check &= (0b00111000);

              if(check==0b00000000)//HIGH LEVEL

              {
               DDRB|=(1<<PB5);//sprinkler
               PORTB|=(1<<5);
               PORTB &=0xF7;
               LCDClear();
             LCDWriteString("SPRINKLER'S ON");
                _delay_ms(2000);
         _delay_ms(2000);
            _delay_ms(2000);
              }
              else if(check==0b00100000)
              {
               DDRB|=(1<<PB3);//drip

               PORTB|=(1<<3);
               PORTB &=0xDF;
               LCDClear();
             LCDWriteString("DRIP SYSTEM ON");
                _delay_ms(2000);
                _delay_ms(2000);
                _delay_ms(2000);
```

```
            }
           else if(check==0b00110000)
            {
             DDRB|=(1<<PB5)|(1<<PB3);//_BV(3);
             PORTB &=0xD7;
            LCDClear();
               LCDWriteString("Message!!!");
           LCDWriteStringXY(0,1,"LOW WATER LEVEL");
             _delay_ms(2000);
             _delay_ms(2000);
             _delay_ms(2000);
            }
            for(i=0;i<5;i++)
          _delay_ms(1000);
         LCDClear();
       }

     LCDWriteString(" T:");
     LCDWriteInt(d2,1);
     LCDWriteInt(d3,1);
     LCDWriteInt(d4,1);
   LCDWriteString(a);
     LCDWriteString("C");
   _delay_ms(500);

     return;
}


uint8_t PREV_PINB=0xFF;

/*
Function to test the current status of keys(0 to 2)

returns
0 if NOT pressed
1 if Pressed
*/
uint8_t GetKeyStatus(uint8_t key)
{
     return (!(PINB & (1<<key)));
}

/*
Function to test the previous status of keys(0 to 2)

returns
0 if NOT pressed
1 if Pressed
*/
uint8_t GetPrevKeyStatus(uint8_t key)
{
     return (!(PREV_PINB & (1<<key)));
}
```

```
unsigned char
key_flag=0,stage,to1,to2,ts1,ts2,to3,to4,to5,to6,ts3,ts4,ts5,ts6,dt1,mn1,yr
1,do1,do2,do3,do4,do5,do6;

int main()
{
      //Wait Util Other device startup
      _delay_ms(50);

      //Initialize the LCD Module
     LCDInit(0x00);
     DDRC|=_BV(5)|_BV(3)|_BV(4);
     PORTC|=_BV(3)|_BV(4)|_BV(5);
    // DDRB|=_BV(3)|_BV(4);
     //PORTB&=0xE7;

     DDRC|=~_BV(5)|~_BV(3)|~_BV(4);
     PINC |=_BV(3)|_BV(4)|_BV(5);   //input pins
       DDRC|=_BV(6);                      //output pin
       PORTC &=0xBF;

       //Initialize I2C Bus
       I2CInit();
       InitADC();


       //Enable Pull ups on keys
       PORTB|=0xFF;

    /*Initializing RTC with intial conditions*/

       #define CH 7

       uint8_t temp;
       DS1307Read(0x00,&temp);
       temp&=(~(1<<CH));                          //Clear CH bit of RTC
       DS1307Write(0x00,temp);


       DS1307Read(0x02,&temp);
       temp|=(0b00000000);                        //Set 24Hour BIT
       DS1307Write(0x02,temp);                   //Write Back to DS1307


       DS1307Read(0x04,&temp);                              //Set dATE Mode
       temp=(0b00010101);
       DS1307Write(0x04,temp);                   //Write Back to DS1307


       DS1307Read(0x05,&temp);                              //Set month Mode
       temp=(0b00000101);
       DS1307Write(0x05,temp);                   //Write Back to DS1307
```

```
  DS1307Read(0x06,&temp);                //Set year Mode
  temp=(0b00010001);
  DS1307Write(0x06,temp);               //Write Back to DS1307

 DS1307Read(0x03,&temp);                //Set day Mode
  temp=(0b00000111);
  DS1307Write(0x03,temp);               //Write Back to DS1307


  LCDClear();

  LCDWriteString("   WELCOME!!!");
  _delay_ms(5000);
  _delay_ms(5000);
  _delay_ms(5000);
  _delay_ms(5000);

 LCDClear();
       //LCDLoading();

 LCDWriteString(" RTC BASED ");
  LCDWriteStringXY(0,1,"AUTOMATED FARM");
  _delay_ms(5000);
  _delay_ms(5000);
  _delay_ms(5000);
  _delay_ms(5000);


  LCDClear();
//   LCDLoading();

  char Time[12],Time1[10];   //hh:mm:ss AM/PM
  unsigned int t,flag=0;
  //Now Read and format time
  uint8_t data;

  while(1)
  {

       DS1307Read(0x00,&data);                  //Reading sec
       Time[8]='\0';
       Time[7]=0x30+(data & 0b00001111);
       Time[6]=0x30+((data & 0b01110000)>>4);
       Time[5]=':';

       DS1307Read(0x01,&data);               //Reading minute

       Time[4]=0x30+(data & 0b00001111);
       Time[3]=0x30+((data & 0b01110000)>>4);
       Time[2]=':';

       DS1307Read(0x02,&data);                  //Reading hr

       Time[1]=0x30|(data & 0b00001111);
```

```
          Time[0]=0x30|((data & 0b00110000)>>4);



if((Time[1]==ts2)&&(Time[0]==ts1)&&(Time[6]==ts5)&&(Time[3]==ts3)&&(Time[4]
==ts4)&&(Time[7]==ts6))                              //Edited  @2
            {
              DDRC|=(1<<PC6);
               PORTC &=0xBF;
              flag=1;
               LCDClear();
               LCDWriteString("   Motor off!!!");
               _delay_ms(5000);
               _delay_ms(5000);
            }

          DS1307Read(0x04,&data);                    //DATE
         Time1[1]=(0x30+(data & 0b00001111));
          Time1[0]=(48+((data & 0b00110000)>>4));


        //month
          DS1307Read(0x05,&data);
         Time1[4]=(0x30+(data & 0b00001111));
          Time1[3]=(0x30+((data & 0b00010000)>>4));
          Time1[2]='.';

        //Year
          DS1307Read(0x06,&data);
         Time1[9]=(0x30+(data & 0b00001111));
          Time1[8]=(0x30+((data & 0b11110000)>>4));
        Time1[5]='.';
          Time1[6]=50;
          Time1[7]=48;

if((Time[1]==to2)&&(Time[0]==to1)&&(Time[6]==to5)&&(Time[3]==to3)&&(Time[4]
==to4)&&(Time[7]==to6))                              //Edited  @2
            {

              DDRC|=(1<<PC6);
               PORTC|=0xff;
              flag=1;
               LCDClear();
               LCDWriteString("   Motor on!!!");
               _delay_ms(5000);
               _delay_ms(5000);

            }
        LCDClear();
        LCDWriteString(Time);
          DisplayReading(ReadTemp());
        LCDWriteStringXY(0,1,Time1);
```

```
        DS1307Read(0x03,&data);
          t=((data & 0b00000111));
          switch(t)
             {

               case 1:LCDWriteStringXY(10,1," MON  ");
                         break;

               case 2:LCDWriteStringXY(10,1," TUE  ");
                         break;

                    case 3:LCDWriteStringXY(10,1," WED  ");
                         break;

               case 4:LCDWriteStringXY(10,1," THU  ");
                        break;

               case 5:LCDWriteStringXY(10,1," FRI  ");
                        break;

                          case 6:LCDWriteStringXY(10,1," SAT  ");
                        break;

                         case 7 :LCDWriteStringXY(10,1," SUN  ");
                        break;

             }

        _delay_ms(5000);


          uint8_t i;
          for(i=0;i<20;i++)
          {

               if(GetKeyStatus(2))
               {
                    //Go To Main Menu
                    ShowMainMenu();
               _delay_loop_2(0);
                    _delay_loop_2(0);
                    _delay_loop_2(0);


               }
               _delay_loop_2(5000);
          }
     }


return 0;
}

void ShowMainMenu()
```

```
{
	//The Main Menu
	char *menu_items[]={  "Set Time",
				"Set Date",
					"Set On Time",
					"Set off Time",
					"Quit"
				};
	uint8_t menu_count=5;
	uint8_t selected=0;

	while(1)
	{
		LCDClear();
		LCDWriteString("    Main Menu  ");
		LCDWriteStringXY(2,1,menu_items[selected]);
		LCDWriteStringXY(0,1,"<");
		LCDWriteStringXY(15,1,">");
		_delay_ms(5000);
		_delay_ms(5000);
	  _delay_ms(5000);

			if(GetKeyStatus(1))
		{
			//Left Key(No 1) is pressed
			//Check that it was not pressed previously
			if(!GetPrevKeyStatus(1))
			{
				if(selected !=0)
					selected--;
			}
		}

		if(GetKeyStatus(0))
		{
			//Right Key(No 0) is pressed
			//Check that it was not pressed previously
			if(!GetPrevKeyStatus(0))
			{
				if(selected !=(menu_count-1))
					selected++;
			}
		}

		if(GetKeyStatus(2))
		{
			//Enter Key Pressed
			//Check that it was not pressed previously
			if(!GetPrevKeyStatus(2))
			{
				//Call Appropriate Function
				switch (selected)
				{
					case 0:
```

```
                                    SetTime();
                                  //key_flag=1;
                                    break;

                            case 1: SetDate();
                                      break;

                            case 2: SetonTime();
                                    //SetonDate();
                                      break;
                            case 3: SetoffTime();
                                      break;
                            case 4: flag1=1;
                                    LCDClear();
                                    LCDWriteString("Thank You!!");
                                       _delay_ms(5000);
                            _delay_ms(5000);
                                  _delay_ms(5000);
                                       //_delay_ms(500);
                                        break;
                                    //Quit
                        }

                    }
            }
            if(flag1==1)
            {
              flag1=0;
              break;
            }

            PREV_PINB=PINB;

            _delay_ms(50);
      }
}

void SetTime()
{

      uint8_t hr,min,sec,temp;//am_pm

      //Read the Second Register
      DS1307Read(0x00,&temp);
      sec=(((temp & 0b01110000)>>4)*10)+(temp & 0b00001111);

      //Read the Minute Register
      DS1307Read(0x01,&temp);
      min=(((temp & 0b01110000)>>4)*10)+(temp & 0b00001111);

      //Read the Hour Register
      DS1307Read(0x02,&temp);
      hr=(((temp & 0b00110000)>>4)*10)+(temp & 0b00001111);
```

```
//If Hour Register is 0 make it 24
if(hr==0) hr=24;

uint8_t sel=0;

while(1)
{
	LCDClear();

	LCDWriteString("00:00:00<OK>");
	LCDWriteIntXY(0,0,hr,2);
	LCDWriteIntXY(3,0,min,2);
	LCDWriteIntXY(6,0,sec,2);


	//Draw Pointer
LCDWriteStringXY(sel*3,1,"^^");

	//Input Up key
	if(GetKeyStatus(1))
	{
		if(!GetPrevKeyStatus(1))
		{
			if(sel==0)
			{
				//Hour
				if(hr==23)
				{
					hr=0;
				}
				else
				{
					hr++;
				}
			}

			if(sel==1)
			{
				//Min
				if(min==59)
				{
					min=0;
				}
				else
				{
					min++;
				}
			}

			if(sel==2)
			{
				//Sec
```

```
                if(sec==59)
                {
                      sec=0;
                }
                else
                {
                      sec++;
                }
          }


          if(sel==3)
          {
                //OK
                //break;
          }
     }
}

//Input Down
if(GetKeyStatus(0))
{
     if(!GetPrevKeyStatus(0))
     {
          if(sel==0)
          {
                //Hour
                if(hr==0)
                {
                      hr=23;
                }
                else
                {
                      hr--;
                }
          }

          if(sel==1)
          {
                //Min
                if(min==0)
                {
                      min=59;
                }
                else
                {
                      min--;
                }
          }

          if(sel==2)
          {
                //Sec
                if(sec==0)
```

```
                        {
                              sec=59;
                        }
                        else
                        {
                              sec--;
                        }
                  }


                  if(sel == 3)
                  {
                        //OK
                        break;
                  }
            }
      }

      if(GetKeyStatus(2))
      {
            if(!GetPrevKeyStatus(2))
            {
                  //Change Selection
                  if(sel==3)
                        sel=0;
                  else
                  sel++;
            }
      }

      PREV_PINB=PINB;

      _delay_loop_2(30000);

  }

  //Now write time back to RTC Module
  temp=((sec/10)<<4)|(sec%10);
  DS1307Write(0x00,temp);

  temp=((min/10)<<4)|(min%10);
  DS1307Write(0x01,temp);

  temp=((hr/10)<<4)|(hr%10);
  temp|=0b00000000; //24 Hr

  DS1307Write(0x02,temp);
flag1=1;
  LCDClear();
  LCDWriteString("  Message !");
  LCDWriteStringXY(0,1,"Main Time Set");
_delay_ms(5000);
  _delay_ms(5000);
_delay_ms(5000);
```

```
      uint8_t i;
      for(i=0;i<10;i++)
            _delay_loop_2(0);
      return;
   }

void SetonTime()
{

      uint8_t hr1,min1,sec1,temp;

      //Read the Second Register
      DS1307Read(0x00,&temp);
      sec1=(((temp & 0b01110000)>>4)*10)+(temp & 0b00001111);

      //Read the Minute Register
      DS1307Read(0x01,&temp);
      min1=(((temp & 0b01110000)>>4)*10)+(temp & 0b00001111);

      //Read the Hour Register
      DS1307Read(0x02,&temp);
      hr1=(((temp & 0b00110000)>>4)*10)+(temp & 0b00001111);


      //If Hour Register is 0 make it 24
      if(hr1==0) hr1=24;

      uint8_t sel=0;


      while(1)
      {
            LCDClear();

            LCDWriteString("00:00:00<OK>");
            LCDWriteIntXY(0,0,hr1,2);
            LCDWriteIntXY(3,0,min1,2);
            LCDWriteIntXY(6,0,sec1,2);


            //Draw Pointer
      LCDWriteStringXY(sel*3,1,"^^");

            //Input Up key
            if(GetKeyStatus(1))
            {
                  if(!GetPrevKeyStatus(1))
                  {
                        if(sel==0)
                        {
                              //Hour
                              if(hr1==23)
                              {
                                    hr1=0;
```

```
                }
                else
                {
                        hr1++;
                }
        }

        if(sel==1)
        {
                //Min
                if(min1==59)
                {
                        min1=0;
                }
                else
                {
                        min1++;
                }
        }

        if(sel==2)
        {
                //Sec
                if(sec1==59)
                {
                        sec1=0;
                }
                else
                {
                        sec1++;
                }
        }


        if(sel==3)
        {
                //OK
                //break;
        }
    }
}

//Input Down
if(GetKeyStatus(0))
{
    if(!GetPrevKeyStatus(0))
    {
        if(sel==0)
        {
                //Hour
                if(hr1==0)
                {
                        hr1=23;
                }
```

```
                else
                {
                        hr1--;
                }
        }

        if(sel==1)
        {
                //Min
                if(min1==0)
                {
                        min1=59;
                }
                else
                {
                        min1--;
                }
        }

        if(sel==2)
        {
                //Sec
                if(sec1==0)
                {
                        sec1=59;
                }
                else
                {
                        sec1--;
                }
        }


        if(sel == 3)
        {
                //OK
                break;
        }
    }
}

if(GetKeyStatus(2))
{
    if(!GetPrevKeyStatus(2))
    {
        //Change Selection
        if(sel==3)
            sel=0;
        else
        sel++;
    }
}

PREV_PINB=PINB;
```

```
            _delay_loop_2(30000);

      }

     to2= 0x30|(hr1%10);
      to1= 0x30|(hr1/10);
      to4= 0x30|(min1%10);
     to3= 0x30|(min1/10);
      to6= 0x30|(sec1%10);
     to5= 0x30|(sec1/10);
      flag1=1;
      LCDClear();
      LCDWriteString("  Message !");
      LCDWriteStringXY(0,1,"On Time Set!!");
     _delay_ms(5000);
      _delay_ms(5000);
     _delay_ms(5000);
      _delay_ms(500);
      _delay_ms(500);
     _delay_ms(500);
      uint8_t i;
      for(i=0;i<10;i++)
            _delay_loop_2(0);
      return;
}

void SetoffTime()
{

      uint8_t hr2,min2,sec2,temp;

      //Read the Second Register
      DS1307Read(0x00,&temp);
      sec2=(((temp & 0b01110000)>>4)*10)+(temp & 0b00001111);

      //Read the Minute Register
      DS1307Read(0x01,&temp);
      min2=(((temp & 0b01110000)>>4)*10)+(temp & 0b00001111);

      //Read the Hour Register
      DS1307Read(0x02,&temp);
      hr2=(((temp & 0b00110000)>>4)*10)+(temp & 0b00001111);

      //If Hour Register is 0 make it 24, as 00:00:00 invalid time
      if(hr2==0) hr2=24;

      uint8_t sel=0;

      while(1)
      {
            LCDClear();

            LCDWriteString("00:00:00<OK>");
```

```
      LCDWriteIntXY(0,0,hr2,2);
      LCDWriteIntXY(3,0,min2,2);
      LCDWriteIntXY(6,0,sec2,2);

      //Draw Pointer
LCDWriteStringXY(sel*3,1,"^^");

      //Input Up key
      if(GetKeyStatus(1))
      {
            if(!GetPrevKeyStatus(1))
            {
                  if(sel==0)
                  {
                        //Hour
                        if(hr2==23)
                        {
                              hr2=0;
                        }
                        else
                        {
                              hr2++;
                        }
                  }

                  if(sel==1)
                  {
                        //Min
                        if(min2==59)
                        {
                              min2=0;
                        }
                        else
                        {
                              min2++;
                        }
                  }

                  if(sel==2)
                  {
                        //Sec
                        if(sec2==59)
                        {
                              sec2=0;
                        }
                        else
                        {
                              sec2++;
                        }
                  }


                  if(sel==3)
                  {
```

```
                //OK
                //break;
            }
        }
}

//Input Down
if(GetKeyStatus(0))
{
      if(!GetPrevKeyStatus(0))
      {
            if(sel==0)
            {
                  //Hour
                  if(hr2==0)
                  {
                        hr2=23;
                  }
                  else
                  {
                        hr2--;
                  }
            }

            if(sel==1)
            {
                  //Min
                  if(min2==0)
                  {
                        min2=59;
                  }
                  else
                  {
                        min2--;
                  }
            }

            if(sel==2)
            {
                  //Sec
                  if(sec2==0)
                  {
                        sec2=59;
                  }
                  else
                  {
                        sec2--;
                  }
            }

            if(sel == 3)
            {
                  //OK
                  break;
```

```
                        }
                  }
            }

            if(GetKeyStatus(2))
            {
                  if(!GetPrevKeyStatus(2))
                  {
                        //Change Selection
                        if(sel==3)
                              sel=0;
                        else
                        sel++;
                  }
            }

            PREV_PINB=PINB;

            _delay_loop_2(30000);

      }

      ts2= 0x30|(hr2%10);
      ts1= 0x30|(hr2/10);
    ts4= 0x30|(min2%10);
    ts3= 0x30|(min2/10);
      ts6= 0x30|(sec2%10);
    ts5= 0x30|(sec2/10);
      flag1=1;
      LCDClear();
      LCDWriteString("  Message !");
      LCDWriteStringXY(0,1,"Off Time Set!!");
    _delay_ms(5000);
      _delay_ms(5000);
    _delay_ms(5000);
      _delay_ms(5000);
      //_delay_ms(5000);
    //_delay_ms(500);
      uint8_t i;
      for(i=0;i<10;i++)
            _delay_loop_2(0);
      return;
}

void SetDate()
{
      uint8_t dt,mn,yr,temp;
      //Read the Second Register
      DS1307Read(0x04,&temp);
      dt=(((temp & 0b00110000)>>4)*10)+(temp & 0b00001111);

      //Read the Minute Register
      DS1307Read(0x05,&temp);
      mn=(((temp & 0b00010000)>>4)*10)+(temp & 0b00001111);
```

```
//Read the Hour Register
DS1307Read(0x06,&temp);
yr=(((temp & 0b11110000)>>4)*10)+(temp & 0b00001111);

uint8_t sel=0;

while(1)
{
     LCDClear();

     LCDWriteString("00:00:00<OK>");
     LCDWriteIntXY(0,0,dt,2);
     LCDWriteIntXY(3,0,mn,2);
     LCDWriteIntXY(6,0,yr,2);

     //Draw Pointer
    LCDWriteStringXY(sel*3,1,"^^");

     //Input Up key
     if(GetKeyStatus(1))
     {
          if(!GetPrevKeyStatus(1))
          {
               if(sel==0)
               {
                    //Date
                    if(dt==31)
                    {
                         dt=0;
                    }
                    else
                    {
                         dt++;
                    }
               }
               if(sel==1)
               {
                    //Month
                    if(mn==12)
                    {
                         mn=0;
                    }
                    else
                    {
                         mn++;
                    }
               }
               if(sel==2)
               {
                    //year
                    if(yr==99)
                    {
                         yr=0;
```

```
                }
                else
                {
                        yr++;
                }
            }
            if(sel==3)
            {
                //OK
                //break;
            }
        }
}

//Input Down
if(GetKeyStatus(0))
{
    if(!GetPrevKeyStatus(0))
    {
        if(sel==0)
        {
            //Day
            if(dt==0)
            {
                dt=31;
            }
            else
            {
                dt--;
            }
        }

        if(sel==1)
        {
            //Month
            if(mn==0)
            {
                mn=12;
            }
            else
            {
                mn--;
            }
        }

        if(sel==2)
        {
            //Year
            if(yr==0)
            {
                yr=99;
            }
            else
            {
```

```
                              yr--;
                        }
                  }


                  if(sel == 3)
                  {
                        //OK
                        break;
                  }
            }
      }

      if(GetKeyStatus(2))
      {
            if(!GetPrevKeyStatus(2))
            {
                  //Change Selection
                  if(sel==3)
                        sel=0;
                  else
                  sel++;
            }
      }

      PREV_PINB=PINB;

      _delay_loop_2(30000);
   }
   flag1=1;

   temp=((yr/10)<<4)|(yr%10);
   DS1307Write(0x06,temp);

   temp=((mn/10)<<4)|(mn%10);
   DS1307Write(0x05,temp);

   temp=((dt/10)<<4)|(dt%10);
   DS1307Write(0x04,temp);
   LCDClear();
   LCDWriteString("  Message !");
   LCDWriteStringXY(0,1,"main Date Set!!");

   _delay_ms(5000);
   _delay_ms(5000);
  _delay_ms(5000);
   _delay_ms(500);
   _delay_ms(500);
  _delay_ms(500);
   uint8_t i;
   for(i=0;i<10;i++)
         _delay_loop_2(0);
   return;
}
```

/*DS1307 SOURCE FILE */

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

#include "I2C.h"
#include "ds1307.h"

/*************************************************************************

Function To Read Internal Registers of DS1307
-------------------------------------------------------------------------
address : Address of the register
data: value of register is copied to this.

Returns:
0= Failure
1= Success
*************************************************************************/
extern void LCDByte(uint8_t c,uint8_t isdata);

uint8_t DS1307Read(uint8_t address,uint8_t *data)
{
     uint8_t res;    //result

                              //Start
     _delay_loop_2(15);
     I2CStart();

     //SLA+W (for dummy write to set register pointer)
     res=I2CWriteByte(0b11010000);   //DS1307 address + W


     //Error
     if(!res)   return FALSE;

//   LCDByte('B',1);

     //Now send the address of required register
     res=I2CWriteByte(address);

     //Error
     if(!res)   return FALSE;

     //Repeat Start
     I2CStart();

     //SLA + R
     res=I2CWriteByte(0b11010001);   //DS1307 Address + R

     //Error
     if(!res)   return FALSE;

     //Now read the value with NACK
```

```
	res=I2CReadByte(data,0);

	//Error
	if(!res)   return FALSE;

	//STOP
	I2CStop();

	return TRUE;
}

/*************************************************************************

Function To Write Internal Registers of DS1307
-------------------------------------------------------------------------

address : Address of the register
data: value to write.

Returns:
0= Failure
1= Success
*************************************************************************/

uint8_t DS1307Write(uint8_t address,uint8_t data)
{
	uint8_t res;    //result

	//Start
	I2CStart();

	//SLA+W
	res=I2CWriteByte(0b11010000);   //DS1307 address + W

	//Error
	if(!res)   return FALSE;

	//Now send the address of required register
	res=I2CWriteByte(address);

	//Error
	if(!res)   return FALSE;

	//Now write the value
	res=I2CWriteByte(data);

	//Error
	if(!res)   return FALSE;

	//STOP
	I2CStop();

	return TRUE;
}
```

/*I2C SOURCE FILE*/

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

#include "I2C.h"

void I2CInit()
{
     //Set up TWI Module
     TWBR = 2;
     TWSR |=((1<<TWPS1)|(1<<TWPS0));

     //Enable the TWI Module
     TWCR|=(1<<TWEN);
}

void I2CClose()
{
     //Disable the module
     TWCR&=(~(1<<TWEN));
}

void I2CStart()
{
     //Put Start Condition on Bus
     TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTA);

     //Poll Till Done
     while(!(TWCR & (1<<TWINT)));
}

void I2CStop()
{
     //Put Stop Condition on bus
     TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO);

     //Wait for STOP to finish
     while(TWCR & (1<<TWSTO));
     //_delay_loop_2(250);
}

uint8_t I2CWriteByte(uint8_t data)
{
     TWDR=data;
     //Initiate Transfer
     TWCR=(1<<TWEN)|(1<<TWINT);
```

```
     //Poll Till Done
     while(!(TWCR & (1<<TWINT)));

     //Check Status
     if((TWSR & 0xF8) == 0x18 || (TWSR & 0xF8) == 0x28 || (TWSR &
0xF8) == 0x40)
     {
          //SLA+W Transmitted and ACK received
          //or
          //SLA+R Transmitted and ACK received
          //or
          //DATA Transmitted and ACK recived

          return TRUE;
     }
     else
          return FALSE;  //Error
}

uint8_t I2CReadByte(uint8_t *data,uint8_t ack)
{
     //Set up ACK
     if(ack)
     {
          //return ACK after reception
          TWCR|=(1<<TWEA);
     }
     else
     {
          //return NACK after reception
          //Signals slave to stop giving more data
          //usually used for last byte read.
          TWCR&=(~(1<<TWEA));
     }

     //Now enable Reception of data by clearing TWINT
     TWCR|=(1<<TWINT);

     //Wait till done
     while(!(TWCR & (1<<TWINT)));

     //Check status
     if((TWSR & 0xF8) == 0x58 || (TWSR & 0xF8) == 0x50)
     {
          //Data received and ACK returned
          //   or
          //Data received and NACK returned

          //Read the data
```

```
          *data=TWDR;
          return TRUE;
     }
     else
          return FALSE;  //Error
}
```

/*LCD HEADER FILE*/

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

#include "myutils.h"

#ifndef _LCD_H
#define _LCD_H
/*_____________________________________________________________________*/

/*********************************************************************
     LCD CONNECTIONS
*********************************************************************/

#define LCD_DATA D    //Port PD0-PD3 are connected to D4-D7

#define LCD_E D //Enable OR strobe signal
#define LCD_E_POS     PD6  //Position of enable in above port

#define LCD_RS D
#define LCD_RS_POS PD4

#define LCD_RW D
#define LCD_RW_POS PD5

/*********************************************************************
                          F U N C T I O N S
*********************************************************************/

void LCDInit(uint8_t style);
void LCDWriteString(const char *msg);
void LCDWriteInt(int val,unsigned int field_length);
void LCDGotoXY(uint8_t x,uint8_t y);
//Low level
void LCDByte(uint8_t,uint8_t);
#define LCDCmd(c) (LCDByte(c,0))
#define LCDData(d) (LCDByte(d,1))

void LCDBusyLoop();

/*********************************************************************
                          F U N C T I O N S     E N D
*********************************************************************/
```

```
/*************************************************************************
                              M A C R O S
*************************************************************************/
#define LCDClear() LCDCmd(0b00000001);
#define LCDHome() LCDCmd(0b00000010);

#define LCDWriteStringXY(x,y,msg) {\
 LCDGotoXY(x,y);\
 LCDWriteString(msg);\
}

#define LCDWriteIntXY(x,y,val,fl) {\
 LCDGotoXY(x,y);\
 LCDWriteInt(val,fl);\
}
/*************************************************************************/

/*_____________________________________________________________________________*/
#endif
```

/*LCD SOURCE FILE*/

```
#include <avr/io.h>
#include <inttypes.h>
#include <util/delay.h>

#include "lcd.h"

#define LCD_DATA_PORT      PORT(LCD_DATA)
#define LCD_E_PORT         PORT(LCD_E)
#define LCD_RS_PORT   PORT(LCD_RS)
#define LCD_RW_PORT   PORT(LCD_RW)

#define LCD_DATA_DDR  DDR(LCD_DATA)
#define LCD_E_DDR          DDR(LCD_E)
#define LCD_RS_DDR         DDR(LCD_RS)
#define LCD_RW_DDR         DDR(LCD_RW)

#define LCD_DATA_PIN  PIN(LCD_DATA)

#define SET_E() (LCD_E_PORT|=(1<<LCD_E_POS))
#define SET_RS() (LCD_RS_PORT|=(1<<LCD_RS_POS))
#define SET_RW() (LCD_RW_PORT|=(1<<LCD_RW_POS))

#define CLEAR_E() (LCD_E_PORT&=(~(1<<LCD_E_POS)))
#define CLEAR_RS() (LCD_RS_PORT&=(~(1<<LCD_RS_POS)))
#define CLEAR_RW() (LCD_RW_PORT&=(~(1<<LCD_RW_POS)))

void LCDByte(uint8_t c,uint8_t isdata)
{
//Sends a byte to the LCD in 4bit mode
//cmd=0 for data
```

```
//cmd=1 for command

//NOTE: THIS FUNCTION RETURS ONLY WHEN LCD HAS PROCESSED THE COMMAND

uint8_t hn,ln;             //Nibbles
uint8_t temp;

hn=c>>4;
ln=(c & 0x0F);

if(isdata==0)
     CLEAR_RS();
else
     SET_RS();

_delay_us(0.500);
SET_E();

//Send high nibble

temp=(LCD_DATA_PORT & 0XF0)|(hn);
LCD_DATA_PORT=temp;

_delay_us(1);              //tEH

//Now data lines are stable pull E low for transmission

CLEAR_E();

_delay_us(1);

//Send the lower nibble
SET_E();

temp=(LCD_DATA_PORT & 0XF0)|(ln);

LCD_DATA_PORT=temp;

_delay_us(1);              //tEH

//SEND

CLEAR_E();

_delay_us(1);              //tEL

LCDBusyLoop();
}

void LCDBusyLoop()
{
     //This function waits till lcd is BUSY

     uint8_t busy,status=0x00,temp;
```

```
	//Change Port to input type because we are reading data
	LCD_DATA_DDR&=0xF0;

	//change LCD mode
	SET_RW();	//Read mode
	CLEAR_RS();		//Read status

	//Let the RW/RS lines stabilize

	_delay_us(0.5);	//tAS


	do
	{
		SET_E();

		//Wait tDA for data to become available
		_delay_us(0.5);

		status=LCD_DATA_PIN;
		status=status<<4;

		_delay_us(0.5);

		//Pull E low
		CLEAR_E();
		_delay_us(1);	//tEL

		SET_E();
		_delay_us(0.5);

		temp=LCD_DATA_PIN;
		temp&=0x0F;

		status=status|temp;

		busy=status & 0b10000000;

		_delay_us(0.5);
		CLEAR_E();
		_delay_us(1);	//tEL
	}while(busy);

	CLEAR_RW();		//write mode
	//Change Port to output
	LCD_DATA_DDR|=0x0F;

}

void LCDInit(uint8_t style)
{
	//After power on Wait for LCD to Initialize
	_delay_ms(30);
```

```
      //Set IO Ports
      LCD_DATA_DDR|=(0x0F);
      LCD_E_DDR|=(1<<LCD_E_POS);
      LCD_RS_DDR|=(1<<LCD_RS_POS);
      LCD_RW_DDR|=(1<<LCD_RW_POS);

      LCD_DATA_PORT&=0XF0;
      CLEAR_E();
      CLEAR_RW();
      CLEAR_RS();


      _delay_us(0.3);

      SET_E();
      LCD_DATA_PORT|=(0b00000010);
      _delay_us(1);
      CLEAR_E();
      _delay_us(1);

      //Wait for LCD to execute the Functionset Command
      LCDBusyLoop();

      //Now the LCD is in 4-bit mode
      LCDCmd(0b00001100|style);  //Display On,cursor off,blink off is
style=0x00;
      LCDCmd(0b00101000);             //function set 4-bit,2 line 5x8 dot
format
    LCDCmd(0b00000001);
}
void LCDWriteString(const char *msg)
{
      /*This function Writes a given string to lcd at the current cursor
      location.
      Arguments: msg: a null terminated string to print */
 while(*msg!='\0')
 {
      LCDData(*msg);
      msg++;
 }
}

void LCDWriteInt(int val,unsigned int field_length)
{
      /* This function writes a integer type value to LCD module
    Arguments:
      1)int val: Value to print
      2)unsigned int field_length :total length of field in which the value
is printed
      must be between 1-5 if it is -1 the field length is no of digits in
the val */

      char str[5]={0,0,0,0,0};
```

```
      int i=4,j=0;
      while(val)
      {
      str[i]=val%10;
      val=val/10;
      i--;
      }
      if(field_length==-1)
            while(str[j]==0) j++;
      else
            j=5-field_length;

      if(val<0) LCDData('-');
      for(i=j;i<5;i++)
      {
      LCDData(48+str[i]);
      }
}
void LCDGotoXY(uint8_t x,uint8_t y)
{
 if(x<16)
 {
  if(y) x|=0b01000000;
  x|=0b10000000;
  LCDCmd(x);
  }
}
```