I2C

Task program

MAIN PROGRAM:-

```
#include <xc.h> //include the neccassary header file
#include"guard.h" //include the user defined header file
#define _XTAL_FREQ 6000000 //intialize the clock speed
void update(void); //call the update function


void main()
{
    init(); //call the init function
    i2cinit(100); //cal the i2c init function
    timeset(); //call the time set function
    while(1)
    {
        lcd(); //call the lcd function
    }

}
```

GUARD PROGRAM:-

```
#ifndef XC_HEADER_TEMPLATE_H
#define XC_HEADER_TEMPLATE_H

#include <xc.h> // include processor files - each processor file is guarded.

void init(void);
void i2cinit(const unsigned long);
void i2cwrite(unsigned char);
int i2cread(int);
void i2cstart();
void i2cwait();
void i2cstop();
void lcdcmd(unsigned char);
void lcddata(unsigned char);
int bcd2dec(int);
int dec2bcd(int);
void timeset(void);
void update(void);
void lcd();
#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */
```

I2C

Task program

FUNCTION PROGRAM:

```
#include<xc.h> //include the neccessary header file
#include"guard.h" //include the user defined header file
#define _XTAL_FREQ 6000000 //intialize the clock
char msg1[5]={"TIME:"}; //declare the char array and inatialization
char msg2[5]={"DATE:"}; //decalare the char array and inatialization
char am[2]={"AM"},pm[2]={"PM"}; //decalare the char array and inatialization
int i,j,k,l,var; //declare the variable
int sec=55,min=59,hour=11,date=8,month=04,year=24,day=1; //declaration of variable
char sec1,sec2,min1,min2,hour1,hour2,date1,date2,month1,month2,year1,year2;

void init(void)
{
    TRISD=0x00; //set the portd as output
    TRISC=0x18; //set the porc input and output
    PORTD=0x00; //clear the portd
    lcdcmd(0x30); //function set  command 8bit intialization
      __delay_ms(5);
      lcdcmd(0x30); //function set  command 8bit intialization
      __delay_ms(5);
      lcdcmd(0x30); //function set  command 8bit intialization
      lcdcmd(0x38); //function set and no line in diasplay
      lcdcmd(0x0C); //display on curser and blink will turn off
      lcdcmd(0x01); //clear the display
}
void lcdcmd(unsigned char i)
{
    PORTC&=~0x04; //set Rs pin as 0
    PORTD=i; //i data will be sent to PORTD
    PORTC|=0x02; //enable set as 1
    __delay_ms(5);
    PORTC&=~0x02; //enable set as 0
    __delay_ms(5);
}
void lcddata(unsigned char i)
{
    PORTC|=0x04; //rs pin will set as 1
```

```
    PORTD=i; //i data will be sent to the port d
    PORTC|=0x02; //enable pin will be set to 1
    __delay_ms(5);
   PORTC&=~0x02; //enable pin set as 0
    __delay_ms(5);
}
void i2cinit(const unsigned long feq_k)
{
    SSPCON=0x28; //intialize the 0010 1000 to sspcon register
    SSPSTAT=0x00; //clear the sspstat register
    SSPCON2=0x00; //clear the sspcon2 register
   SSPADD = (_XTAL_FREQ/(4*feq_k*100))-1; //set the baurd rate
}
void i2cwait()
{
    while(SSPCON2 & 0x1F || SSPSTAT & 0x04); //check the comunnication line
}
void i2cstart()
{
   i2cwait(); //wait for any opration going for bus line
   SSPCON2|=0x01; //set the SEN for start bit
}
void i2cstop()
{
    i2cwait(); //wait for any opration going for bus line
    SSPCON2|=0x04; //enable the PEN for stop bit
}
void i2cwrite(unsigned char temp)
{
    i2cwait();
    SSPBUF=temp; //the data will be sent to the SSPBUF
}
int i2cread(int ack)
{
    int value; //variable declaration
    i2cwait();
```

Task program

```
    SSPCON2|=0x08; //enable reciver mode for I2c
    i2cwait();
    value=SSPBUF; //the buffer data for rtc eill store in the value
    i2cwait();
    ACKDT=(ack)?0:1; //check the ack
    SSPCON2|=0x10; //Initiate Acknowledge sequence on SDA and SCL pins
    return value; //return the value

}
int dec2bcd(int temp)
{
return ((temp/10)<<4)+(temp%10); //seprate the lsb and msb msb will be left shift
                                 //four times and add to the lsb

}
int bcd2dec(int temp)
{
return ((temp>>4)*10)+(temp&0x0F); //right shift the data four times
                                   //and operat with value of 0000 1111
                                   //add the msb and lsb
}
void timeset(void)
{
    i2cstart(); //call the start bit
    i2cwrite(0xD0); //write the address of RTC
    i2cwrite(0); //set the mode as a write for the write
    i2cwrite(dec2bcd(sec)); //convert to decimal to binary and write
    i2cwrite(dec2bcd(min)); //convert to decimal to binary and write
    i2cwrite(0x60|(dec2bcd(hour))); //convert to decimal to binary and write
                                    //and set the AM/PM 12/24
    i2cwrite(dec2bcd(day)); //convert to decimal to binary and write
    i2cwrite(dec2bcd(date)); //convert to decimal to binary and write
    i2cwrite(dec2bcd(month)); //convert to decimal to binary and write
    i2cwrite(dec2bcd(year)); //convert to decimal to binary and write
    i2cstop(); //call the stop bit
}
```

```
void update(void)
{
    i2cstart(); //call the start bit
    i2cwrite(0xD0); //set the address
    i2cwrite(0); //set the mode of write
    i2cstop(); //call the stop bit
    i2cstart(); //call the start bit
    i2cwrite(0xD1); //set the address
    sec=(bcd2dec(i2cread(1))); //read the data conv to and store in variable
    min=(bcd2dec(i2cread(1))); //read the data conv to and store in variable
    var=i2cread(1); //read the data and store in the variable
    hour=(bcd2dec(0x1F&var)); //read the data conv to and store in variable
    day=(bcd2dec(i2cread(1))); //read the data conv to and store in variable
    date=(bcd2dec(i2cread(1))); //read the data conv to and store in variable
    month=(bcd2dec(i2cread(1))); //read the data conv to and store in variable
    year=(bcd2dec(i2cread(1))); //read the data conv to and store in variable
    i2cstop(); //call the stop bit
    i2cstart(); //call the start bit
    i2cwrite(0xD1); //set the address
    i2cread(1); //call the read function ack
    i2cstop(); //call the stop bit
}
void lcd()
{
      update(); //call the update function
    sec1=sec/10; //seperate msb from the data
    sec2=sec%10; //seperate the lsb from the data
    min1=min/10; //seperate msb from the data
    min2=min%10; //seperate the lsb from the data
    hour1=hour/10; //seperate msb from the data
    hour2=hour%10; //seperate the lsb from the data
    date1=date/10; //seperate msb from the data
    date2=date%10; //seperate the lsb from the data
    month1=month/10; //seperate msb from the data
    month2=month%10; //seperate the lsb from the data
    year1=year/10; //seperate msb from the data
```

Task program

```
        year2=year%10;  //seperate the lsb from the data

        lcdcmd(0x80); //set the location of the lcd
        for(i=0;i<5;i++)
            lcddata(msg1[i]); //print "TIME" in lcd
        lcddata(hour1+'0'); //print hour on lcd
        lcddata(hour2+'0'); //print hour on lcd
        lcddata(0x2D); //print '-' symbol
        lcddata(min1+'0'); //print minute on lcd
        lcddata(min2+'0'); //print minute on lcd
        lcddata(0x2D); //print '-' symbol
        lcddata(sec1+'0'); //print second on lcd
        lcddata(sec2+'0'); //print second on lcd
        lcddata(' '); //print the white space
        if (var&0x20) //check for AM/PM
        {
            for(i=0;i<2;i++)
            {
                lcddata(pm[i]); //print the pm
            }
        }
        else
        {
            for(i=0;i<2;i++)
            {
                lcddata(am[i]); //print am
            }
        }
        lcdcmd(0xC0); //set the location of the lcd
        for(i=0;i<5;i++)
            lcddata(msg2[i]); //print the "TIME:" in lcd
        lcddata(date1+'0'); //print the data on lcd
        lcddata(date2+'0'); //print the data on lcd
        lcddata(0x2D); //print the '-' symbol
        lcddata(month1+'0'); //print the data on lcd
        lcddata(month2+'0'); //print the data on lcd
```

```
        lcddata(0x2D); //print the '-' symbol
        lcddata(year1+'0'); //print the data on lcd
        lcddata(year2+'0'); //print the data on lcd
        lcddata(' '); //print the space
        switch(day){
                case 1:lcddata('S'); //switch case for DAY print in the lcd
                break;
                case 2:lcddata('M');
                break;
                case 3:lcddata('T');
                break;
                case 4:lcddata('W');
                break;
                case 5:lcddata('T');
                break;
                case 6:lcddata('F');
                break;
                case 7:lcddata('S');
                break;
        }
    __delay_ms(500);
}
```

Task program

I²C debugger:-

```
 71.721ms  78.061ms S D0 A 00 A 55 A 59 A 71 A 01 A 08 A 04 A 24 A P
 78.088ms  78.410ms S D0 A 00 A P
 78.441ms  80.665ms S D1 A 55 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
416.368ms 416.690ms S D0 A 00 A P
416.721ms 418.944ms S D1 A 55 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
754.647ms 754.969ms S D0 A 00 A P
755.001ms 757.224ms S D1 A 55 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  1.093 s   1.093 s S D0 A 00 A P
  1.093 s   1.096 s S D1 A 56 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  1.431 s   1.432 s S D0 A 00 A P
  1.432 s   1.434 s S D1 A 56 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  1.769 s   1.770 s S D0 A 00 A P
  1.770 s   1.772 s S D1 A 56 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  2.108 s   2.108 s S D0 A 00 A P
  2.108 s   2.110 s S D1 A 57 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  2.446 s   2.446 s S D0 A 00 A P
  2.446 s   2.449 s S D1 A 57 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  2.784 s   2.785 s S D0 A 00 A P
  2.785 s   2.787 s S D1 A 57 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  3.123 s   3.123 s S D0 A 00 A P
  3.123 s   3.125 s S D1 A 58 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  3.461 s   3.461 s S D0 A 00 A P
  3.461 s   3.463 s S D1 A 58 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  3.799 s   3.799 s S D0 A 00 A P
  3.800 s   3.802 s S D1 A 58 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  4.137 s   4.138 s S D0 A 00 A P
  4.138 s   4.140 s S D1 A 59 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  4.476 s   4.476 s S D0 A 00 A P
  4.476 s   4.478 s S D1 A 59 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  4.814 s   4.814 s S D0 A 00 A P
  4.814 s   4.817 s S D1 A 59 A 59 A 71 A 01 A 08 A 04 A 24 A 00 N P
  5.152 s   5.153 s S D0 A 00 A P
  5.153 s   5.155 s S D1 A 00 A 00 A 52 A 02 A 09 A 04 A 24 A 00 N P
  5.490 s   5.491 s S D0 A 00 A P
  5.491 s   5.493 s S D1 A 00 A 00 A 52 A 02 A 09 A 04 A 24 A 00 N P
  5.829 s   5.829 s S D0 A 00 A P
  5.829 s   5.831 s S D1 A 00 A 00 A 52 A 02 A 09 A 04 A 24 A 00 N P
  6.167 s   6.167 s S D0 A 00 A P
  6.167 s   6.169 s S D1 A 01 A 00 A 52 A 02 A 09 A 04 A 24 A 00 N P
  6.505 s   6.505 s S D0 A 00 A P
  6.505 s   6.507 s S D1 A 01 A 00 A 52 A 02 A 09 A 04 A 24 A 00 N P
```