1. INTRODUCTION

1.1 INTRODUCTION

Classroom management system is used to know the presence of teacher in classrooms; whether to indicate the teacher is present or not. A teacher presence can be indicated using the indicators like led. If there are many classrooms for a department all the classroom status can be indicated by single panel.

Installation of the CCTV's can be done; but it costs more and its connection is difficult to give for every department. By this system Head of the department or head master or head mistress can know about the availability of teacher in particular classrooms. To avoid the misuse of the system, biometric sensors should be used.

The main use of this system is that the time can be saved. By less cost and simple components this system can be achieved.

1.2 Organization of Report

The report organized has follows. Chapter 2 provides system specifications and block diagram description. Chapter 3 contains detailed hardware description. Chapter 4 contains software design and description of the project. It also contains program code and flowchart of the system.Chapter5 gives operating instruction and conclusion of the project.

CHAPTER 2

2. SYSTEM BLOCK DIAGRAM

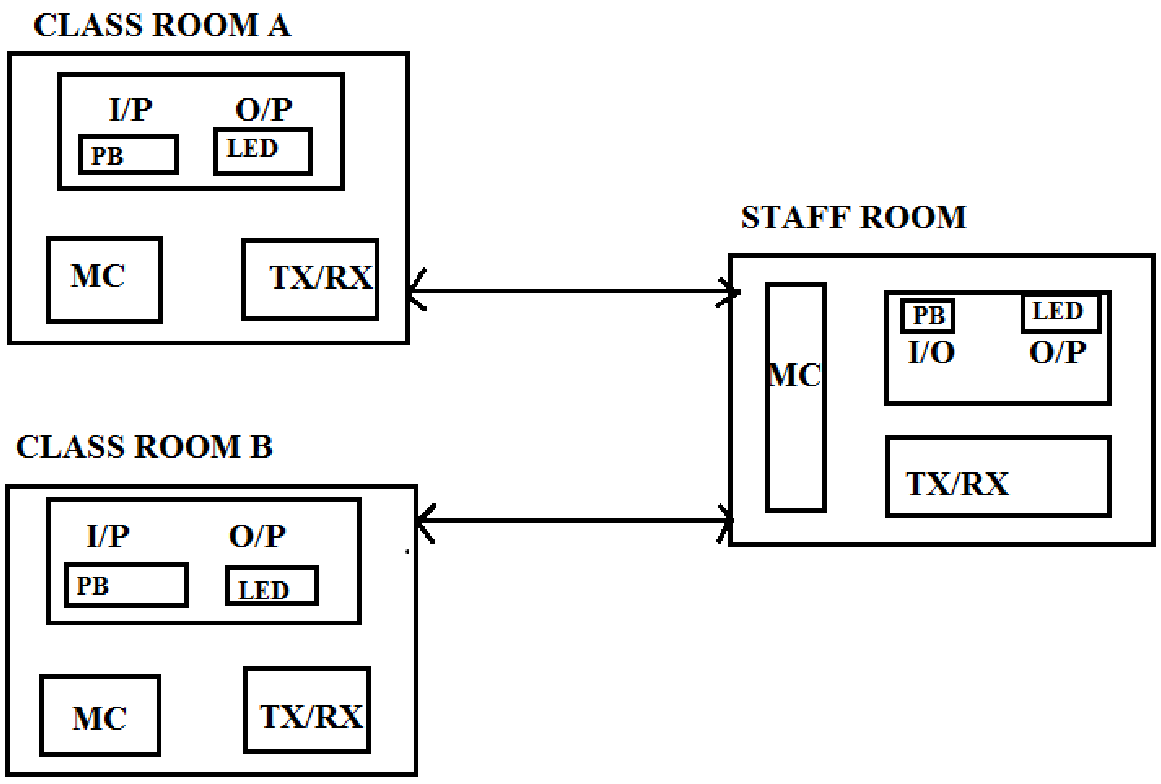
2.1 DESCRIPTION

The TXD pin of the classroom microcontroller is connected to RXD o staffroom microcontroller and vice versa .the microcontroller receives the information and works as per the program written.

Glowing of red led indicates that teacher is not present in classroom and green led indicates the presence in class room .teachers can call the class representatives from staff room by pressing particular keys provided.

Here push buttons are used as input devices and leds are used as output devices.

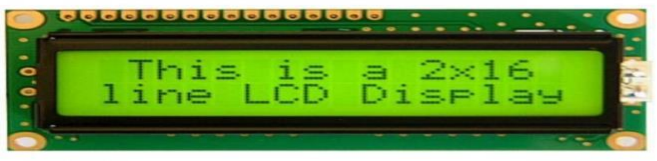
2.2 SYSTEM BLOCK DIAGRAM OF CLASS ROOM MANAGEMENT SYSTEM



CHAPTER 3

HARDWARE DESCRIPTION

3.1 LCD Display



Liquid Crystal Display which is commonly Known as LCD is an Alphanumeric Display it

means that it can display Alphabets, numbers as well as special symbols thus LCD is user friendly Display device which can be used for displaying various messages unlike seven segment display only numbers and some of the alphabets .The only disadvantage of the robust display is visualized from a longer distance as compared to LCD. Here we have used 16*2 alphanumeric displays which mean on this display we can display two lines with maximum of
16 characters in one line. This LCD has two registers, namely, command and Data.

The command register stores the command instructions given to the LCD.A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD .The data is the ASCII value of the character to be displayed on the LCD.

3.2 BUZZER

A buzzer or beeper is an audio signaling device,[1] which may be mechanical, electromechanical, or piezoelectric (piezo for short) .we have used magnetic buzzer.

Magnetic buzzer specifications:

- Current Rating- 30mA
- Frequency- 2.3 kHz
- Voltage Rating-5VDC
- Size / Dimension Circular - 12mm Dia x 9.5mm H
- Mounting Type- through Hole
- Voltage Range -3 ~ 7VDC

3.3 LED

A light-emitting diode (LED) is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated. When a suitable current is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor. LEDs are typically small (less than 1 mm2) and integrated optical components may be used to shape the radiation pattern.

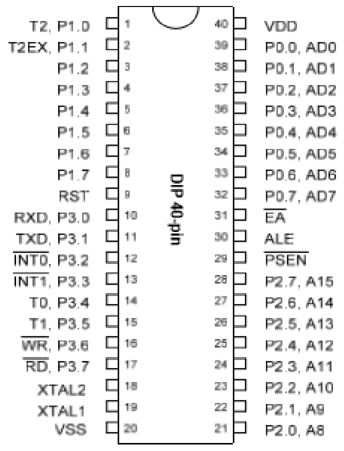
Appearing as practical electronic components in 1962, the earliest LEDs emitted low-intensity infrared light. Infrared LEDs are still frequently used as transmitting elements in remote-control circuits, such as those in remote controls for a wide variety of consumer electronics.

3.4 PUSHBUTTON

A **push-button** (also spelled **pushbutton**) or simply **button** is a simple switch mechanism to control some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed.

3.5 AT89C51

The AT89C51 is an old 8 bit microcontroller from the Atmel family. It works with the popular 8051 architecture and hence is used by most beginners till date. It is a 40 pin IC package 4kb flash memory.it has four ports and all together provide 32 programmable GPIO pins. It does not have in built ADC module and supports only USART communication although it can be interfaced with external ADC IC like the ADC084 or the ADC0808.The AT89C51 is no longer in production and Atmel does not support



New design .Instead AT89C51 is recommended for new applications .

3.5.2 Features

- 80c51 Central Processing unit
- 5V operating voltage from 0Mhz to 40Mhz
- 64kB of on-chip Flash user code memory with ISP(In-System Programming)and IAP(In- Application Programming)
- 256 bytes of on-chip scratchpad RAM
- 16K/8K/4K bytes electrically erasable/programmable Flash EPROM
- 2K bytes LDROM support ISP function (Reference Application Note)
- 64KB program memory address space
- 64KB data memory address space
- Four 8-bit bi-directional ports
- 8-sources, 4-level interrupt capability
- One extra 4-bit bit-addressable I/O port, additional INT2/INT3 (available on PQFP, PLCCand LQFP package)
- Three 16-bit timer/counters
- One full duplex serial port

3.6 MULTISIM SOFTWARE

NATIONAL INSTRUMENTS™

NI Multisim (formerly **MultiSIM**) is an electronic schematic capture and simulation program which is part of a suite of circuit design programs, along with NI Ultiboard. Multisim is one of the few circuit design programs to employ the original Berkeley SPICE based software simulation. Multisim was originally created by a company named Electronics Workbench, which is now a division of National Instruments. Multisim includes microcontroller simulation (formerly known as MultiMCU), as well as integrated import and export features to the printed circuit board layout software in the suite, NI Ultiboard**.**

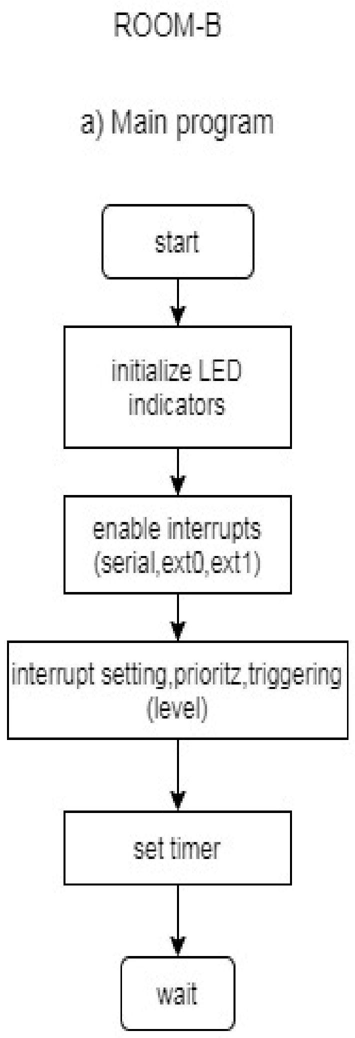
Multisim software provides SPICE simulation, analysis, and printed circuit board (PCB) tools to help you quickly iterate through designs and improve prototype performance. Move from schematic to layout seamlessly to save time and reduce prototype iterations.

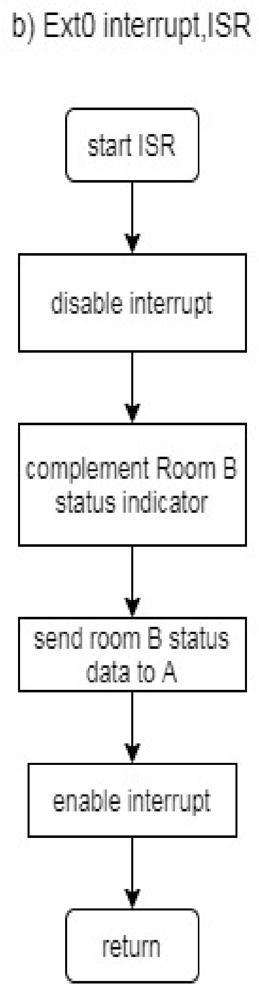
Multisim was originally called **Electronics Workbench** and created by a company called Interactive Image Technologies. At the time it was mainly used as an educational tool to teach electronics technician and electronics engineering programs in colleges and universities. National Instruments has maintained this educational legacy, with a specific version of Multisim with features developed for teaching electronics.

CHAPTER 4

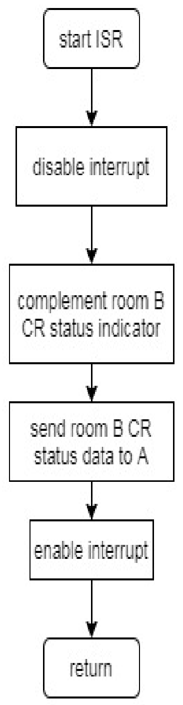
SOFTWARE DESIGN

4.1 FLOW CHART

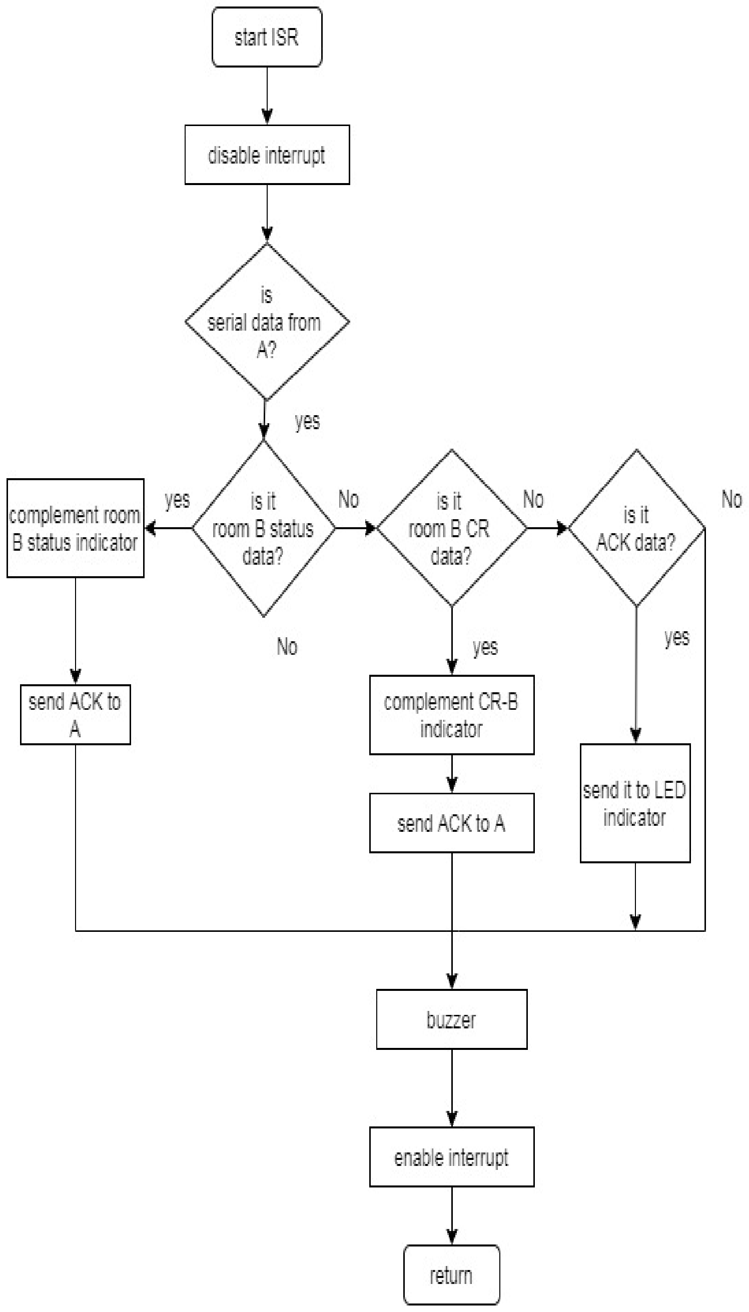

ROOM-B
a) Main program
start
initialize LED indicators
enable interrupts (serial,ext0,ext1)
interrupt setting,prioritz,triggering (level)
set timer
wait



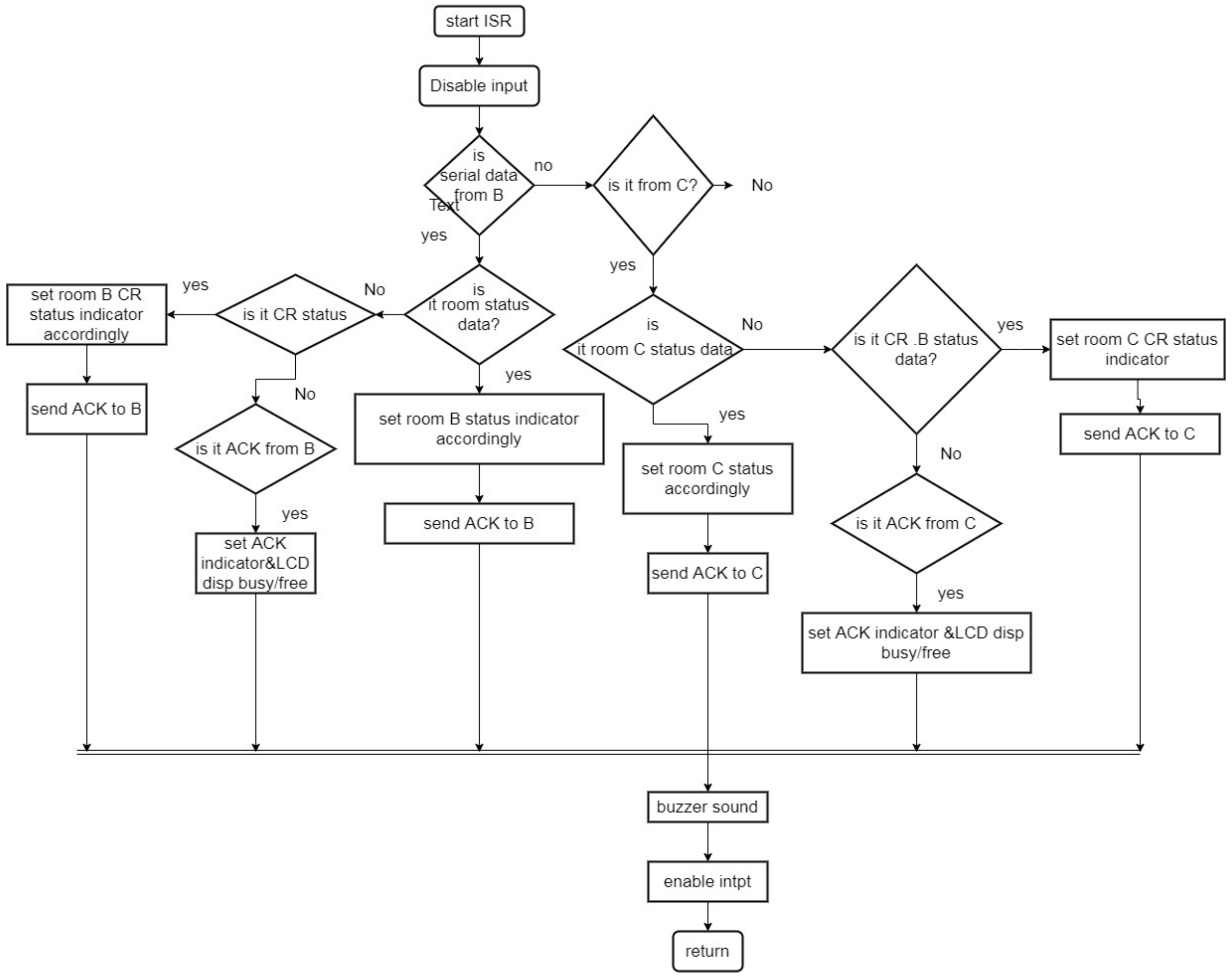
c) Ext1 interrupt ISR


start ISR
disable interrupt
complement room B CR status indicator
send room B CR status data to A
enable interrupt
return

d) Serial interrupt ISR



c) Serial input ISR



4.2 PROGRAM CODE

//Program to manage Class Room B

```
#include<htc.h>

#define LEDC0Cls P10//Classroom status

#define LEDC1Crc P11//CR call status

#define LEDC2Ack P12//Acknowledgement indicator

#define Buzz P13//Buzzer line
```

```
//const char a[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};

//void RAM_VECTOR(0x73, t2int_handler);


//Macro to create ISR for serial-port interrupt

#asm

        GLOBAL _serialisr    // ISR name = serialisr()

        PSECT vectors,ovrld

        ORG 0x23                        // Int vector

        LJMP _serialisr                 // ISR function name again with a preceding underscore
added.

        PSECT text

#endasm


//Macro to create ISR for Ext. Interrupt 0

#asm

        GLOBAL _ext0isr                 //ISR name = ext1isr()

        PSECT vectors,ovrld

        ORG 0x03                        //Int vector

        LJMP _ext0isr           //ISR function name again with a preceding underscore added.

        PSECT text

#endasm


#asm

        GLOBAL _ext1isr// ISR name = ext1isr()

        PSECT vectors,ovrld

        ORG 0x13// Int vector

        LJMP _ext1isr// ISR function name again with a preceding underscore added.
        PSECT text
#endasm


//Functions
#pragma interrupt_level 0
void reentrant delay(unsigned int msec)    //Function to provide time delay in msec for LCD.
{       unsigned int i,j ;
        for (i=0;i<msec; i++)
          for (j=0; j<100;j++);
}


//Buzzer function
#pragma interrupt_level 0
void reentrant buzzer(unsigned char Length, unsigned int msec)    //Function to provide time
delay in msec for LCD.
{       unsigned int i;
   Buzz=1;
   for (i=0;i<Length;i++)
          {
     delay(msec);
     Buzz=!Buzz;
    }
}


//ISR for serial-port interrupt
interrupt void serialisr(void)
{unsigned int i,j,Outer=50, Inner=50;
```

```
ES=0;           //Disable serial-port interrupt
RI=0;           // Clear intpt flag
ACC=SBUF; // Reading from the port buffer

if(ACC5==0&&ACC4==1)        //is it for B?
  if(ACC7==0&&ACC6==0)      //is it from A?
    {if(ACC3==0&&ACC2==0)   //Room status data?
      {LEDC0Cls=!LEDC0Cls;
       ACC7=0; ACC6=1;     //from B
       ACC5=0;ACC4=0;     // to A
       ACC3=1;ACC2=0;      //Setting Ack
       SBUF=ACC;                      //Preparing to send Ack to A
       while(TI==0);      //Sending Ack to A
       TI=0;
      }                    //Sent ack to A
     if(ACC3==0&&ACC2==1)   //CR status data?
      {LEDC1Crc=!LEDC1Crc;
       ACC7=0;ACC6=1;      // from B
       ACC5=0;ACC4=0;      // to A
       ACC3=1;ACC2=0;      //Setting Ack
       ACC0=LEDC1Crc;
       SBUF=ACC;                      //Preparing to send Ack to A
       while(TI==0);      //Sending Ack to A
       TI=0;
      }
```

```
        if(ACC3==1&&ACC2==0)   //Ack data?
        {LEDC2Ack=1;
         delay(5);
         LEDC2Ack=0;
                        buzzer(3, 2);
        }
      }
  ES=1;                                                            //Enable serial-port interrupt
 }

//ISR for Ext. Interrupt-0
interrupt void ext0isr(void)
{EX0=0; //Disable ext intpt-0
 LEDC0Cls=!LEDC0Cls; //Complementing class-room status LED
 ACC=0x00;
 ACC7=0;ACC6=1;     //from B
 ACC5=0;ACC4=0;   //to A
 ACC3=0;ACC2=0;   //Room status data
 ACC0=LEDC0Cls;   //Current room-status data in room C
 SBUF=ACC;
 while(TI==0);
 TI=0;
 EX0=1;                              //Enable external interrupt0
}

//ISR for Ext. Interrupt-1
interrupt void ext1isr(void)
```

```
{EX1=0; //Disable ext intpt-0
 LEDC1Crc=!LEDC1Crc; //Complementing class-room status LED
 ACC=0x00;
 ACC7=0;ACC6=1;      //from B
 ACC5=0;ACC4=0;   //to A
 ACC3=0;ACC2=1;   //CR status data
 ACC0=LEDC1Crc;   //Current room-status data in room C
 SBUF=ACC;
 while(TI==0);
 TI=0;
 EX1=1;                                   //Enable external interrupt0
}

//main program
void main()
{  unsigned int i;
   LEDC0Cls=1;
   LEDC2Ack=1;
   LEDC1Crc=1;
   for (i=1;i<4;i++)  //led blinking
   {LEDC0Cls=!LEDC0Cls;
    LEDC2Ack=!LEDC2Ack;
    LEDC1Crc=!LEDC1Crc;
    delay(20);
   }
   LEDC0Cls=0; //led initial setting
   LEDC2Ack=0;
```

```
  LEDC1Crc=0;
  EA=1; //interrupt settings
  ES=1;
  EX0=1;IT0=1;
  EX1=1;
  TMOD=0x20; //timer settiing
  TH1=0xFD;
  SCON=0x90;
  TR1=1;
  while(1);}
```

// Room A Programs

```
#include<htc.h>
#include <string.h>
#define LED0BCls P10        //Room B status
#define LED1CCls P11        //Room C status
#define LED2BAck P12        //Room B ack
#define LED3CAck P13        //Room C ack
#define LED4BCrc P14        //CR B status
#define LED5CCrc P34        //CR C status
#define Buzz P36      // Buzzer
#define dataport P2   //LCD 8-bit data line
#define rs P06
#define en P05
#define rw P07

//ISR macros
//Macro to create ISR for ext interrupt 0
```

```
#asm
        GLOBAL _ext0isr             // ISR name = ext0isr()
        PSECT vectors,ovrld
        ORG 0x03                    // Int vector
        LJMP _ext0isr          // ISR function name again with a preceding underscore added.
        PSECT text
#endasm

//Macro to create ISR for serial interrupt
#asm
        GLOBAL _serialisr   //ISR name = serialisr()
        PSECT vectors,ovrld
        ORG 0x23                    //Int vector
        LJMP _serialisr             //ISR function name again with a preceding underscore
added.
        PSECT text
#endasm

//Functions

//Delay function
#pragma interrupt_level 0
void reentrant delay(unsigned int msec)    //Function to provide time delay in msec for LCD.
{       unsigned int i,j ;
        for(i=0;i<msec;i++)
          for(j=0;j<10;j++);
}
```

```
//Buzzer function

#pragma interrupt_level 0

void reentrant buzzer(unsigned char Length, unsigned int msec)    //Function to provide time
delay in msec for LCD.

{         unsigned int i;

    Buzz=1;

    for (i=0;i<Length;i++)

             {

      delay(msec);

      Buzz=!Buzz;

     }

}


//LCD command write function

#pragma interrupt_level 0

void reentrant lcdcmd(unsigned char item)    //Function to send command to LCD

{         dataport = item;

          rs=0;

    rw=0;

          en=1;

          delay(1);

          en=0;

    rw=0;

}


// LCD data writing function
```

```
#pragma interrupt_level 0
void reentrant lcddata(unsigned char item)    //Function to send data to LCD
{          dataport=item;
           rs=1;
   rw=0;
           en=1;
           delay(1);
           en=0;
   rw=0;
}

//LCD Initialization function
//#pragma interrupt_level 1
void reentrant lcdini2(void)
{          lcdcmd(0x38);    //for using 8-bit 2 row mode of LCD
           delay(100);
           lcdcmd(0x0C);    //turn display ON and cursor OFF
           delay(100);
           lcdcmd(0x01);    //clear screen
           delay(100);
           lcdcmd(0x80);    //bring cursor to position 0 of line 1
           delay(100);
}

//LCD scrolling display function
//#pragma interrupt_level 1
void disp_scrol_name(char *item)
```

```
  {unsigned int lcdloc=0x80,i=0,j,k, lcdwidth=16;
   j=strlen(item);

          for(k=0;k<lcdwidth;k++) // scrolling
    {
      lcdcmd(lcdloc);
      while(i<=j)
      {if ((k+i)<lcdwidth) // cutting name tail
       lcddata(item[i]);

      i++;
      }
     lcdloc++;
     i=0;
     lcdcmd(0x01);// clearing display for every iteration
    }
  }

// LCD Normal display
#pragma interrupt_level 0
void reentrant displcd(unsigned int lcdloc, char *item)
  {unsigned int i=0,j,k, lcdwidth=16;//lcdloc=0x80,
   j=strlen(item);
   //lcdini2();
     // lcdcmd(0x01);
      lcdcmd(lcdloc);
      while(i<=j)
      {  lcddata(item[i]);
         i++;
       }

  }



//Ext Intpt 0
#pragma interrupt_level 1
interrupt void ext0isr(void)
{ EX0=0;                   //disable interrupt
  ACC=0x00;

  if (P15==0)             //if status B switch
  {LED0BCls=!LED0BCls;    //toggle Room B status
   ACC7=0;ACC6=0;         //from A
   ACC5=0;ACC4=1;               //to B
   ACC3=0;ACC2=0;         //Room status data
   ACC0=LED0BCls;}        //ACC0 current status of room B at room A

  if (P16==0)             //if status C switch
  {LED1CCls=!LED1CCls;    //toggle Room C status
   ACC7=0;ACC6=0;         //from A
   ACC5=1;ACC4=0;         //to C
   ACC3=0;ACC2=0;         //Room status data
   ACC0=LED1CCls;         //ACC0 current status of room C at room A
  }
```

```
 if (P17==0)           //if CR-B  switch
 {LED4BCrc=!LED4BCrc;   //CR-B status disp at A is complelemnted
  ACC7=0;ACC6=0;       // from A
  ACC5=0;ACC4=1;       // to B
  ACC3=0;ACC2=1;       //CR status data
  ACC0=LED4BCrc;
 }                     //CR-B status passed

 if (P35==0)            //if CR-C switch
 {LED5CCrc=!LED5CCrc;    //CR-C status disp complelemnted
  ACC7=0;ACC6=0;        // from A
  ACC5=1;ACC4=0;        //to C
  ACC3=0;ACC2=1;        // CR C status
  ACC0=LED5CCrc;        //CR-C status passed
 }

  SBUF=ACC;
  while(TI==0);            //transmitting data to B or C
  TI=0;
//buzzer(1, 4);
  EX0=1;                        //enable interrupt
}

//serial port
#pragma interrupt_level 1
interrupt void serialisr(void)
```

```
{   unsigned int i,j,Outer=10,Inner=10,lcdln1=0x80, lcdln2=0xC0;

    unsigned char *busy10="LH-10 Busy", *busy11="LH-11 Busy", *nbusy10="LH-10
Free",*nbusy11="LH-11 Free";


    ES=0;RI=0;//disable serial interrupt

         ACC=SBUF;


    if (ACC5==0&&ACC4==0)// to A?

       {

         if (ACC7==0&&ACC6==1)// is it from B

           {

            if (ACC3==0&&ACC2==0)// is it Room Status Data?

               {LED0BCls=ACC0;

                ACC7=0;ACC6=0;// from A

                ACC5=0;ACC4=1; // to B

                ACC3=1;ACC2=0;// Ack

                SBUF=ACC;

                while(TI==0); // sending ack

                TI=0;

               }


            if (ACC3==0&&ACC2==1) // is it CR status?

                {LED4BCrc=ACC0;

                 ACC7=0;ACC6=0;   // from A

                 ACC5=0;ACC4=1;   // to B

                 ACC3=1;ACC2=0;// Ack

                 SBUF=ACC;
```

```
        while(TI==0); //sending ack
        TI=0;
       }

   if (ACC3==1&&ACC2==0) // is it Ack
      { LED2BAck=1;
        //delay(1);
                                 if (LED0BCls==1)
           displcd(lcdln1,busy10);
        else
          displcd(lcdln1,nbusy10);

       LED2BAck=0;//ack indicator goes off
      }
  }

 if (ACC7==1&&ACC6==0)   // is it from C
  { if (ACC3==0&&ACC2==0)// is it Room Status Data?
       {LED1CCls=ACC0;
                                 ACC7=0;ACC6=0; //from A
        ACC5=1;ACC4=0;// to C
        ACC3=1;ACC2=0; // ack
                                SBUF=ACC;
              while(TI==0); //sending ack C
              TI=0;
       }
    if (ACC3==0&&ACC2==1)// is it CR status?
```

```
              {LED5CCrc=ACC0;
                                    ACC7=0;ACC6=0;// from A
               ACC5=1;ACC4=0;// to C
               ACC3=1;ACC2=0; // ack
                                    SBUF=ACC;
                    while(TI==0); //sending ack C
                    TI=0;
              }
            if (ACC3==1&&ACC2==0)// is it Ack
              {LED3CAck=1;
               delay(5);
                                     if (LED1CCls==1)
                            displcd(lcdln2,busy11);
                      else
                            displcd(lcdln2,nbusy11);
              LED3CAck=0;
             }
                     }
         buzzer(5, 4);
        }

  ES=1;// enable serial interrupt

}

void main()
```

```
{  unsigned int i;
   unsigned char item[]="CL-ROOM";
   buzzer(1, 1);
   LED0BCls=1;
   LED1CCls=1;
   LED2BAck=1;
   LED3CAck=1;
   LED4BCrc=1;
   LED5CCrc=1;
   lcdini2();// initializing LCD
   //disp_scrol_name(item);// scroll Displaying in LCD
   //displcd(0x08,item);// Displaying project name

   for (i=1;i<4;i++) //LEDs blinking to indicate project is powered on
   {LED0BCls=!LED0BCls;
    LED1CCls=!LED1CCls;
    LED2BAck=!LED2BAck;
    LED3CAck=!LED3CAck;
    LED4BCrc=!LED4BCrc;;
    LED5CCrc=!LED5CCrc;
    delay(2);
   }

   // interrupt and timer settings
   EA=1;EX0=1;EX1=1;ES=1;IT0=1;IT1=1;
   TMOD=0x20;
   TH1=0xFD;
   SCON=0x90;
   TR1=1;

   while(1); // keeping in microcontroller wait condition

}
```

//Program to manage Class Room C

```
#include<htc.h>
#define LEDC0Cls P10//Classroom status
#define LEDC1Crc P11//CR call status
#define LEDC2Ack P12//Acknowledgement indicator
#define Buzz P13//Buzzer line



//Macro to create ISR for serial-port interrupt
#asm
        GLOBAL _serialisr    // ISR name = serialisr()
        PSECT vectors,ovrld
        ORG 0x23                     // Int vector
        LJMP _serialisr              // ISR function name again with a preceding underscore
added.
        PSECT text
#endasm

//Macro to create ISR for Ext. Interrupt 0
#asm
```

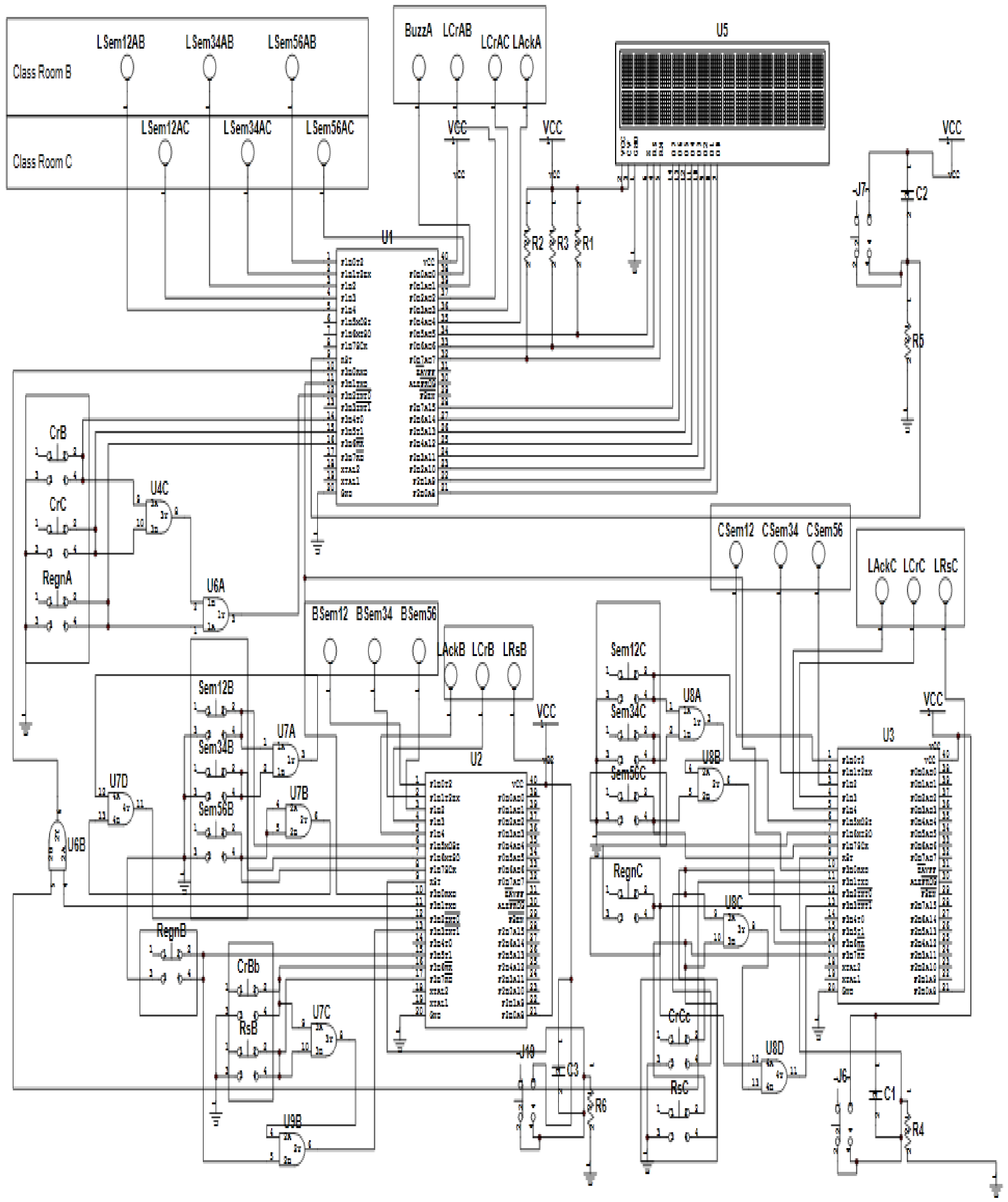
```
        GLOBAL _ext0isr             //ISR name = ext1isr()

        PSECT vectors,ovrld

        ORG 0x03                    //Int vector

        LJMP _ext0isr          //ISR function name again with a preceding underscore added.

        PSECT text

#endasm


#asm

        GLOBAL _ext1isr// ISR name = ext1isr()

        PSECT vectors,ovrld

        ORG 0x13// Int vector

        LJMP _ext1isr// ISR function name again with a preceding underscore added.

        PSECT text

#endasm


//Functions

#pragma interrupt_level 0

void reentrant delay(unsigned int msec)    //Function to provide time delay in msec for LCD.

{       unsigned int i,j ;

        for(i=0;i<msec;i++)

          for(j=0;j<100;j++);

}


//Buzzer function

#pragma interrupt_level 0

void reentrant buzzer(unsigned char Length, unsigned int msec)    //Function to provide time
delay in msec for LCD.
```

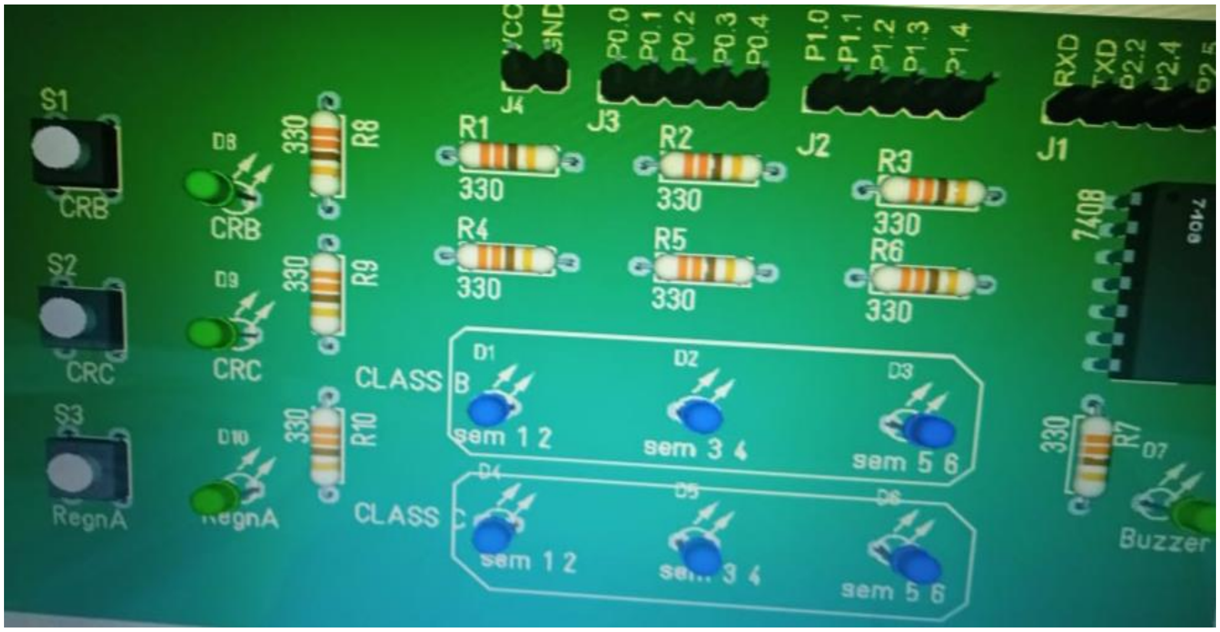
```
{         unsigned int i;
    Buzz=1;
    for (i=0;i<Length;i++)
             {
      delay(msec);
      Buzz=!Buzz;
     }
}

//ISR for serial-port  interrupt
interrupt void serialisr(void)
{unsigned int i,j,Outer=50, Inner=50;

 ES=0;            //Disable serial-port interrupt
 RI=0;            // Clear intpt flag
 ACC=SBUF; // Reading from the port buffer

 if(ACC5==1&&ACC4==0)         //to C?
    if(ACC7==0&&ACC6==0)      // from A?
      {if(ACC3==0&&ACC2==0)   //Room status data?
         {LEDC0Cls=!LEDC0Cls;
          ACC7=1;ACC6=0;     //from C
          ACC5=0;ACC4=0;     // to A
          ACC3=1;ACC2=0;      //Setting Ack
          SBUF=ACC;                      //Preparing to send Ack to A
          while(TI==0);       //Sending Ack to A
          TI=0;
```

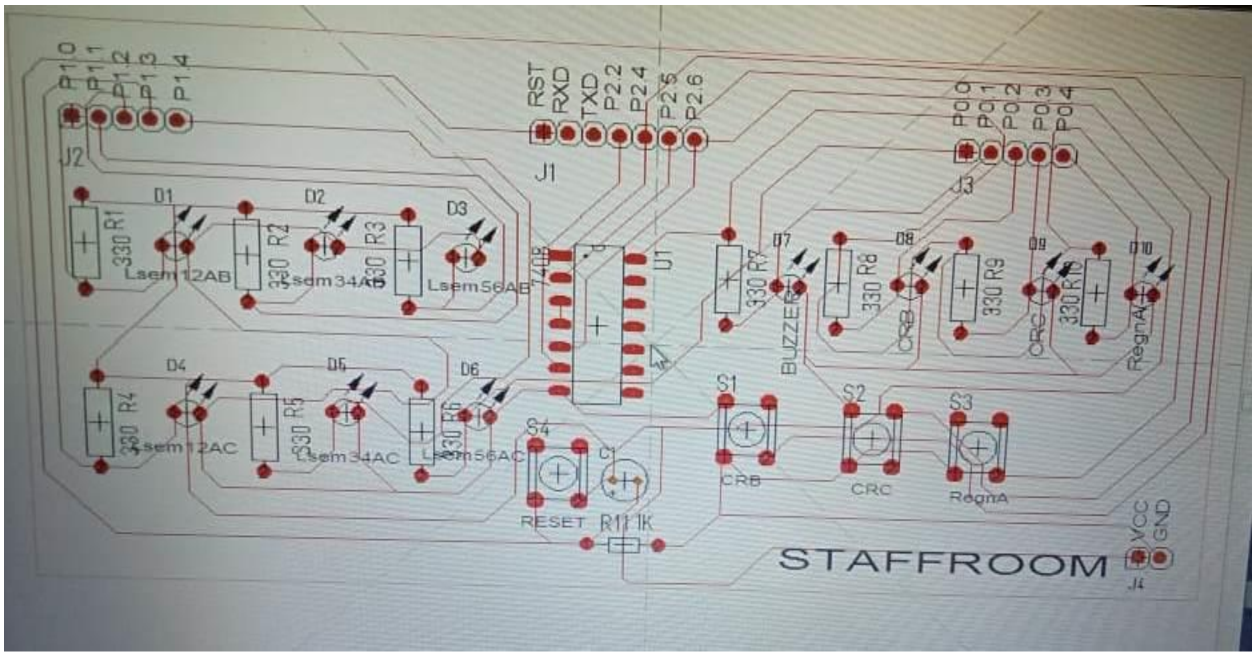
```
        }                   //Sent ack to A
       if(ACC3==0&&ACC2==1)   //CR status data?
        {LEDC1Crc=!LEDC1Crc;
         ACC7=1;ACC6=0;      // from C
         ACC5=0;ACC4=0;      // to A
         ACC3=1;ACC2=0;      //Setting Ack
         SBUF=ACC;                      //Preparing to send Ack to A
         while(TI==0);       //Sending Ack to A
         TI=0;
        }

        if(ACC3==1&&ACC2==0)   //Ack data?
        {LEDC2Ack=1;
         delay(5);
         LEDC2Ack=0;
         buzzer(3, 2);
        }
      }
  ES=1;                                                           //Enable serial-port interrupt
 }

//ISR for Ext. Interrupt-0
interrupt void ext0isr(void)
{          EX0=0; //Disable ext intpt-0
           LEDC0Cls=!LEDC0Cls; //Complementing class-room status LED
           ACC=0x00;
           ACC7=1;ACC6=0;      //from C
```

```
            ACC5=0;ACC4=0;   //to A
            ACC3=0;ACC2=0;   //Room status data
            ACC0=LEDC0Cls;   //Current room-status data in room C
            SBUF=ACC;
            while(TI==0);
            TI=0;
            EX0=1;                          //Enable external interrupt0
}

//ISR for Ext. Interrupt-1
interrupt void ext1isr(void)
{EX1=0; //Disable ext intpt-0
 LEDC1Crc=!LEDC1Crc; //Complementing class-room status LED
 ACC=0x00;
 ACC7=1;ACC6=0;      //from C
 ACC5=0;ACC4=0;   //to A
 ACC3=0;ACC2=1;   //CR status data
 ACC0=LEDC1Crc;   //Current room-status data in room C
 SBUF=ACC;
 while(TI==0);
 TI=0;
 EX1=1;                          //Enable external interrupt0
}

// Main program
void main()
{  unsigned int i;
```

```
   LEDC0Cls=1;
   LEDC2Ack=1;
   LEDC1Crc=1;
   for (i=1;i<4;i++)                  // Indicators blinking
   {LEDC0Cls=!LEDC0Cls;
    LEDC2Ack=!LEDC2Ack;
    LEDC1Crc=!LEDC1Crc;
    delay(20);
   }
   LEDC0Cls=0;  // Indicator initialising
   LEDC2Ack=0;
   LEDC1Crc=0;
   EA=1;   //interrupt settings
   ES=1;
   EX0=1;IT0=1;
   EX1=1;
   TMOD=0x20; //timer setting
   TH1=0xFD;
   SCON=0x90;
   TR1=1;
   while(1);
}
```


Class Room B
Class Room C
LSem12AB
LSem34AB
LSem56AB
LSem12AC
LSem34AC
LSem56AC
BuzzA
LCrAB
LCrAC
LAckA
U5
VCC
R1
R2
R3
U1
J7
C2
R5
CrB
CrC
RegnA
U4C
U6A
U6B
U7D
Sem12B
Sem34B
Sem56B
U7A
U7B
BSem12
BSem34
BSem56
LAckB
LCrB
LRsB
U2
RegnB
CrBb
RsB
U7C
U9B
J10
C3
R6
CSem12
CSem34
CSem56
LAckC
LCrC
LRsC
Sem12C
Sem34C
Sem56C
U8A
U8B
U3
RegnC
U8C
CrCc
RsC
U8D
J6
C1
R4


S1
CRB
S2
CRC
S3
RegnA
D8
D9
D10
R8
R9
R10
330
R1
R2
R3
R4
R5
R6
R7
J4
J3
J2
J1
VCC
GND
P0.0
P0.1
P0.2
P0.3
P0.4
P1.0
P1.1
P1.2
P1.3
P1.4
RXD
TXD
P2.2
P2.4
7408
CLASS B
CLASS C
D1
D2
D3
D4
D5
D6
D7
sem 1 2
sem 3 4
sem 5 6
Buzzer


P1.0
P1.1
P1.2
P1.3
P1.4
J2
RST
RXD
TXD
P2.2
P2.4
P2.5
P2.6
J1
P0.0
P0.1
P0.2
P0.3
P0.4
J3
Lsem12AB
Lsem34AB
Lsem56AB
Lsem12AC
Lsem34AC
Lsem56AC
7408
U1
BUZZER
CRB
CRC
RegnA
S1
S2
S3
S4
C1
RESET
R11 1K
STAFFROOM
VCC
GND
J4

CHAPTER 5 [OPERATING INSTRUCTION]

5.1 OPERATING INSTRUCTION

1. The teacher is supposed to press the key.

2. The teacher should select the sem.

3. Green led turns on, both in staffroom and classroom.

4. The timer turns on.

5. The teacher should press the key while leaving the class, it turns red led and off green led.

6. If there is any call to CR, it can also be notified by teacher pressing a pushbutton in staffroom

5.2 CONCLUSION:

Finally we have successfully completed our project classroom management system .We learnt to design a microcontroller based –system design.

5.3 REFRENCES

- www.electronicwings.com
- www.gadgetronicx.com
- www.googlescholor.com