SRI SHANMUGHA COLLEGE OF ENGINEERING AND TECHNOLOGY

SANKARI, SALEM

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
Regulation – 2017

EC8711 - EMBEDDED LABORATORY

Academic Year 2021-2022

IV YEAR VII SEMESTER

Prepared by

Mr.P.Surendar,

Assistant Professor

SYLLABUS

EC8711-Embedded Lab **L T P C**
0 0 4 2

Course Objective:

The student should be made to:

- Learn the working of ARM processor
- Understand the Building Blocks of Embedded Systems
- Learn the concept of memory map and memory interface
- Write programs to interface memory, I/Os with processor
- Study the interrupt performance

List of Experiments:

1. Study of ARM evaluation system
2. Interfacing ADC and DAC.
3. Interfacing LED and PWM.
4. Interfacing real time clock and serial port.
5. Interfacing keyboard and LCD.
6. Interfacing EPROM and interrupt.
7. Mailbox.
8. Interrupt performance characteristics of ARM and FPGA.
9. Flashing of LEDS.
10. Interfacing stepper motor and temperature sensor.
11. Implementing zigbee protocol with ARM.

Course Outcome:

At the end of the course, the student should be able to:

- Write programs in ARM for a specific Application
- Interface memory, A/D and D/A convertors with ARM system
- Analyze the performance of interrupt
- Write program for interfacing keyboard, display, motor and sensor.
- Formulate a mini project using embedded system

TOTAL: 60 PERIODS

LIST OF EXPERIMENTS

Expt. No.	Title of the Experiment
1	Study of ARM evaluation system
2	Interfacing ADC and DAC.
3	Interfacing LED and PWM.
4	Interfacing real time clock and serial port.
5	Interfacing keyboard and LCD.
6	Interfacing EPROM and interrupt.
7	Mailbox.
8	Interrupt performance characteristics of ARM and FPGA.
9	Flashing of LEDS.
10	Interfacing stepper motor and temperature sensor.
11	Implementing zigbee protocol with ARM.
12	Simulation using Proteus Software.- An Introduction
13	Simulation of calculator using 8051 microcontroller in Proteus software

1. Study of ARM Evaluation System

Aim

To learn about the evolution, core features, general characteristics and applications of ARM processors.

Pre Lab Questions

1. What is an embedded system?
2. Mention the difference between microprocessor and microcontroller.
3. Enumerate the terms object oriented and object based language
4. Define Pipelining.
5. List the basic units of Microprocessor

Theory

The LPC2148 microcontrollers are based on a 32/16 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16- bit Thumb mode reduces code by more than 30 % with minimal performance penalty.

Due to their tiny size and low power consumption, LPC2148 are ideal for applications where miniaturization is a key requirement, such as access control and point- of-sale. A blend of serial communications interfaces ranging from a USB 2.0 Full Speed device, multiple UARTS, SPI, SSP to I2C s and on-chip SRAM of 8 kb up to 40 kb, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical system**s.**

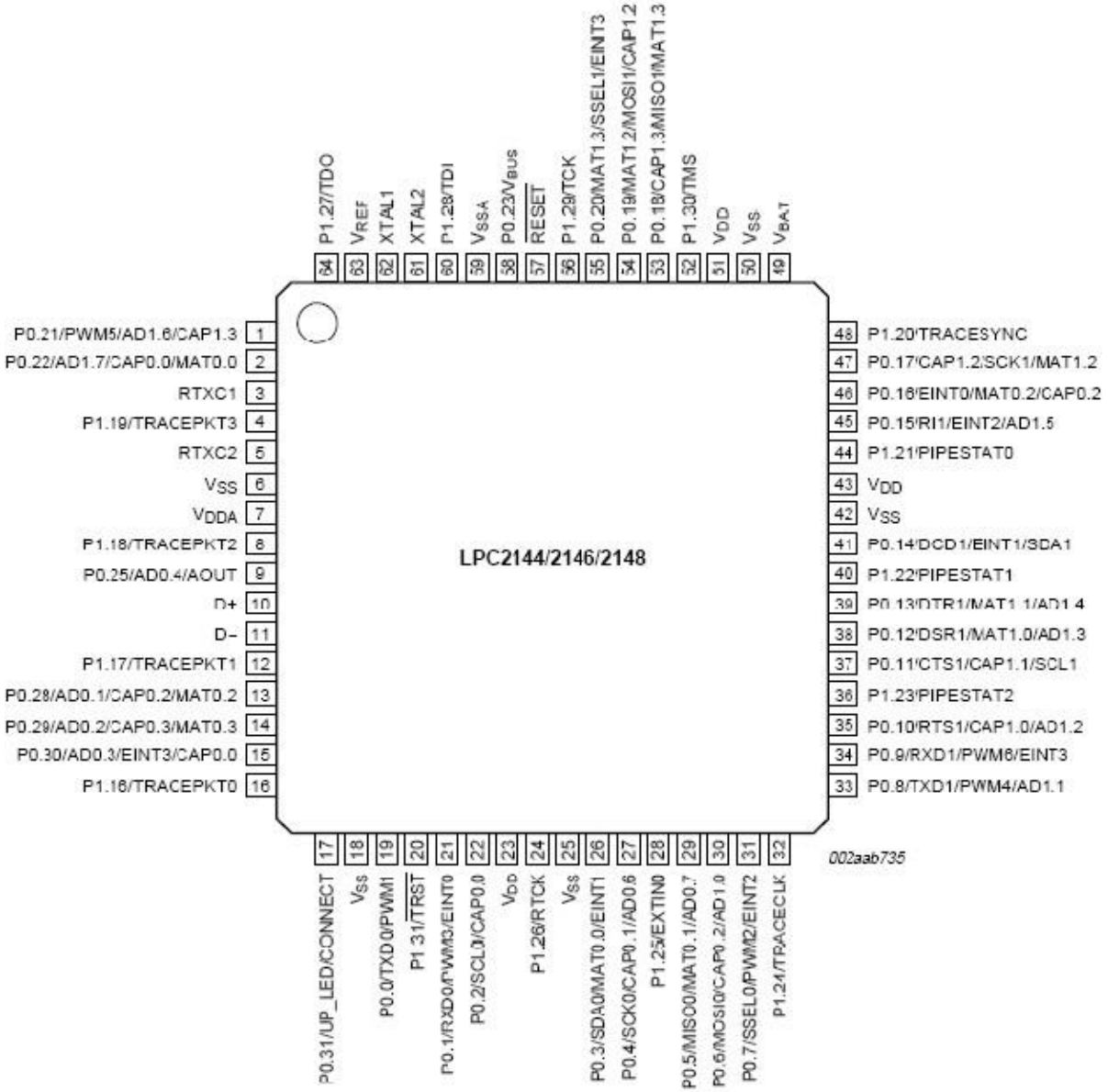
LPC2148 specification

Features:

- 16/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64package.
- 8 to 40 KB of on-chip static RAM and 32 to 512kB of on-chip flash Program memory.
- 128 bit wide interface/accelerator enables high speed 60 MHz operation.
- In-System/In-Application Programming(ISP/IAP)via on-chip boot- loader software. Single flash sector or full chip erase in 400ms and programming of 256 bytes in 1 ms.
- Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high speed tracing of instruction execution.
- USB 2.0 Full Speed compliant Device Controller with 2 kb of endpoint RAM. In addition, the LPC2146/8 provide 8 kb of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a
    - total of 6/14 analog inputs, with conversion times as low as 2.44µsper channel.
- Single 10-bit D/A converter provide variable analog output.
- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power real-time clock with independent power and dedicated 32kHz Clock input.
- Multiple serial interfaces including two UARTs (16C550), two Fast I2C -bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller with configurable priorities and vector addresses.

- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tinyLQFP64 package.
- Up to nine edge or level sensitive external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100µs.
- On-chip integrated oscillator operates with an external crystal in range from 1MHz to 30 MHz and with an external oscillator up to 50MHz.
- Power saving modes include Idle and Power-down.
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt, USB, Brown-Out Detect (BOD) or Real-Time Clock(RTC).
- Single power supply chip with Power-On Reset (POR) and BOD circuits:
    - CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %)with 5V tolerant I/O pads.

Pin Configuration:

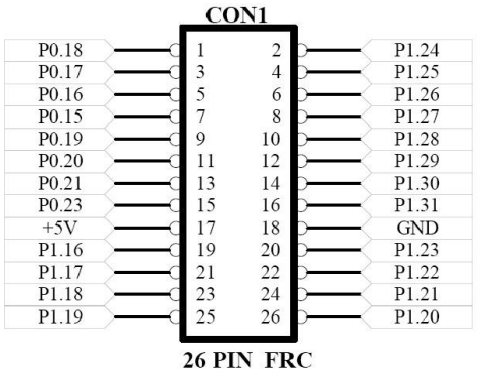


For further studies about LPC2148 specification refer NXP's website to download LPC2148 user manual.

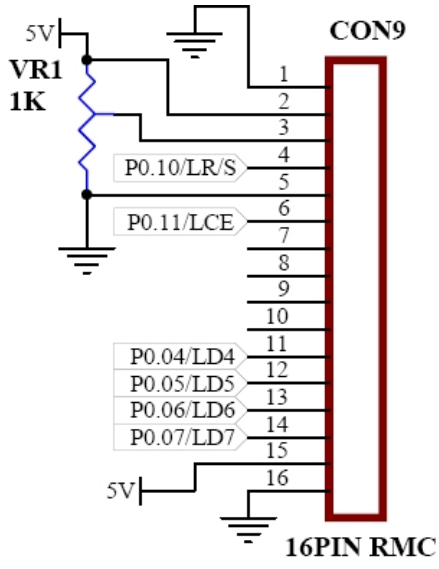
APPENDIX-II: CONNECTORS & CONNECTION DETAILS

Type	Label	Description
26 Pin FRC	CON1	For GPIO
20 Pin FRC	CON2	For JTAG
DB9	CON3	UART0
DB9	CON4	UART1
3 Pin RMC	CON5	PWM output
6 Pin RMC	CON6	ADC input
3 Pin RMC	CON7	DAC output from OPAMP
3 Pin RMC	CON8	Temperature sensor input
16 Pin RMC	CON9	LCD Display
Power jack	C10	Power input
PS2	CON11	Keyboard interface

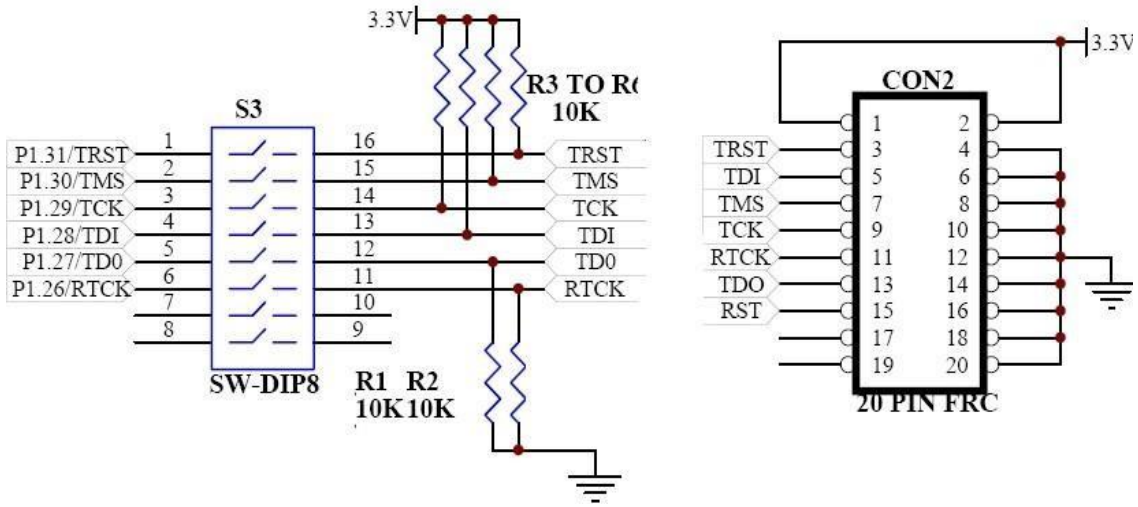
Connections Details for 26 pin FRC:



Connection Details for LCD(16 pin RMC):



Connection Details for JTAG(20 pin FRC):



Post Lab Questions

1. What is the difference between embedded systems and the system in which rtos is running?
2. Discuss about semaphore.
3. What are the instructions used to access the memory in ARM?
4. Mention the characteristics of RISK Instruction.
5. Define Interrupt.

Result

The evolution, core features, general characteristics and the applications of ARM processors has been studied and is evaluated.

2. INTERFACING ADC & DAC

Aim

To develop a C-Language program for reading an on-chip ADC, convert into decimal and to display it in PC and to generate a square wave depending on this ADC reading. The ADC input is connected to any analog sensor/ on board potentiometer.

Pre Lab Questions

1. List the types of ADC and DAC
2. Define resolution.
3. Summarize the features of Conversion time in ADC.
4, What is the function of Sample-and-hold circuits in analog-to digital converters?
5. Why are internal ADCs preferred over external ADCs?

Apparatus & Software Required

1. LPC2148 Development board.
2. KeilµVision5software.
3. Flash Magic.
4. USB cable.
5. CRO.

Theory

The LPC 2148 has 10-bit successive approximation analog to digital converter. Basic clocking for the A/D converters is provided by the VPB clock. A programmable divider is included in each converter, to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 11 of these clocks. The ADC cell can measure the voltage on any of the ADC input signals.

ARM Board has one potentiometer for working with A/D Converter. Potentiometer outputs are in the range of 0V to 3.3V. Switch select in right position for reading the Potentiometer value by ADC.

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

MAIN ADC TEST

```
/*****************************************************************************/
        /* This is a test program to ADC in theARMLPC2148 developmentboard*/
/*****************************************************************************/

#include<LPC214x.H>                     /* LPC214x definitions*/
#include"ADC_Driver.c"                          /* contains prototypes of driverfunctions*/ #include"lcd.c"
#include <stdio.h>

int main (void)
  {
    unsigned int adc_val;
    unsigned int temp;
    unsigned char buf[4] ={0,0,0,0}; ADCInit();
    lcdinit();
         //wait();
              clrscr(10);
         printstr("ADC Test",0,0); wait();
    while(1)                                /* Loop forever*/
      {
              adc_val = ADC_ReadChannel();
              temp = (unsigned int)((3*adc_val*100)/1024);
              sprintf(buf,"%d",temp);
              printstr(buf,0,1);

      }

  }

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

LCD.C

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

#include <LPC214x.h>

#defineRS               0x00000400 /* P0.10 */
#defineCE               0x00001800 /* P1.11*/
void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1 void
printstr(unsignedchar*,char,char);                      //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);

#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones;
```

```
voidwait(void)
{       /* wait function */
int     d;
   for (d = 0; d <100000;d++);                          /* only to delay for LED flashes*/
}


void lcdinit()
{
       IODIR0 |= 0xFFFFFFFF;
            IOCLR0 |= 0X00000FFF;
            lcdcmd(0x28);lcd
       cmd(0x28);
       lcdcmd(0x0c);
       lcdcmd(0x06);
       lcdcmd(0x01);
       lcdcmd(0x0f);
       wait();
}

void gotoxy(char x, char y)
{
       if(y == 0)
              lcdcmd(0x80+x);
       else
              lcdcmd(0xc0+x);

}
void printstr(unsigned char *str, char x, char y)
{
       char i; gotoxy(x,y);
       wait();//(500);
       for(i=0;str[i]!='\0';i++)lcddat(str[i
              ]);

}
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
              LCDDAT = (cmd&0xf0);              //higher nibble
              IOSET0 =LCDDAT;
         IOCLR0 = RS;
              IOSET0 = CE;
         wait();//(100);                                        //enablelcd
              IOCLR0 = CE;
              IOCLR0 = 0X00000FFF;


         LCDDAT = ((cmd<<0x04)&0xf0);                   //lower nibble
              IOSET0 =LCDDAT;
         IOCLR0 = RS;
              IOSET0 = CE;
         wait();//(100);                                        //enablelcd
              IOCLR0 = CE;
              IOCLR0 = 0X00000FFF;

}

void lcddat(char cmd)
```

```
{
    unsigned charLCDDAT;
          LCDDAT = (cmd&0xf0);              //higher nibble
          IOSET0 =LCDDAT;
      IOSET0 = RS;
          IOSET0 = CE;
      wait();//(100);                                   //enablelcd
          IOCLR0 = CE;
          IOCLR0 = 0X00000FFF;

      LCDDAT = ((cmd<<0x04)&0xf0);                      //lower nibble
          IOSET0 =LCDDAT;
      IOSET0 = RS;
          IOSET0 =CE;
      wait();//(100);                                    //enable lcd
          IOCLR0 =CE;
          IOCLR0 = 0X00000FFF;
 }

void clrscr(char ch)
{
      if(ch==0)
      {
            printstr("                              ",0,0);
            gotoxy(0,0);
      }
        else if(ch == 1)
      {
            printstr("                              ",0,1);
            gotoxy(0,1);
      }
      else
      {
                lcdcmd(0x01);
                  //delay(100);
      }
}


void split_numbers(unsigned int number)
{
   thousands = (number /1000);
   number %= 1000;
   hundreds = (number / 100);
   number %= 100;
   tens = (number / 10);
   number %= 10;
   ones = number ;
}

void Wait_Msg(void)
{
 lcdcmd(0x01);
 printstr("          Please Wait ", 0,0);
}
void Welcome_Msg(void)
{
 lcdcmd(0x01);
 printstr("           Welcometo          ", 0,0);
 printstr("             SMMICRRO     ", 0,1);
}
```

```
/*****************************************************************/
                        ADC   DRIVER.C
/*****************************************************************/


#include<LPC214x.H>                     /* LPC214x definitions */

Void ADCInit(void)
  {
    PINSEL1|=0x04000000;          /*For Channel AD0.2 is P0.29*/
    IODIR0 |=~(0x04000000);
    AD0CR    |=0x00200204;           /*0x04 selects AD0.2 to mux output, 0x20 makes ADCin operational*/
    AD0GDR;                         /*A read on AD0GDR clears the DONEbit*/
  }

void ADC_StartConversion(void)
  {
     AD0CR |= (1<<24);
  }
void ADC_StopConversion(void)
  {
     AD0CR &= (~ (1<<24));
  }


unsigned int ADC_ReadChannel(void)
  {
//       unsigned int i; unsigned long
      ADC_Val, t;
      ADC_StartConversion();
      while((AD0DR2&0x80000000)==0); /*wait until ADC conversion completes*/
      if(AD0STAT & 0x00000400)
                {
                  //printstr("OVR",0,1);return(
                  0);
                }
      t = AD0DR2;
      ADC_Val = ((t>>6) & 0x000003FF);//(AD0DR2 & 0x000003FF); //((AD0CR>>6) & 0x000003FF);
      //ADC_StopConversion();retur
      n(ADC_Val);
```

ADC PROGRAM PORT DETAILS

ARM	DETAILS
P0.29	ADC0.2
PO.10	RS LCD PIN
P1.11	CE LCD PIN

DAC PROGRAM

```
/*****************************************************************/ DAC.C
/*****************************************************************/
        This is a test program to DAC in the ARM LPC2148 Development board
/*****************************************************************/

#include<LPC214X.H>


void wait_long (void)
{                                   /* wait function*/
   int      d;
   for (d = 0; d <1000000;d++);                          /* only to delay*/
}

int main()
  {
          wait_long();wai
          t_long();
          IODIR0   =   0X00000FFF;
          IODIR1   =   0XFFFF0000;
          IOSET0  =  0XFFFFFFFF;
          IOCLR1 = 0XFFFF0000;
      PINSEL1 |= 0x00080000; //Enablepin0.25                asDAC
          DACR=0X00017FC0;              // 000 = 0V (min),7FC = 1.6V,7FF =3.3V(max)
          While (1);
}
```

DAC PROGRAM PORT DETAILS

ARM	DETAILS
P0.25	DAC ENABLE PIN

Post Lab Questions

1. What are the ADC operating modes in LPC2148?
2. What is the function of A/D Status Register
3. Which pin provides a voltage reference level for the D/A converter?
4. What is Burst conversion mode?
5. What is settling time?

Result :

The C-Language program for reading an on-chip ADC, convert into decimal and to display it in PC was written & output is verified with the ADC input is connected to on board potentiometer

The DAC, convert digital data into analog signal& output is verified with the DAC input and the square wave has been generated to display it in CRO.

3. INTERFACING LED & PWM

Aim

To write a C program for Switch & L ED to activate LED's and generate a PWM and to vary the duty cycle .

Pre Lab Questions

1. What happens if the junction temperature of LED is increased?
2. Mention the principle of PWM.
3. What are the materials used to make LED?
4. What are the types of seven segment display.
5. Differentiate LED from LCD.

Apparatus & Software Required

1. LPC2148 Development board.
2. Keil µVision 5 software.
3. Flash Magic.
4. USB cable.
5. CRO .

Theory

The PWM is based on the standard timer block and inherits all of its features, although only the PWM function is pinned out on the LPC2148. The timer is designed to count cycles of the peripheral clock (PCLK) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. The PWM function is also based on match register events.

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

```
/*****************************************************************************/
                        SWITCH AND LED PROGRAM
/*****************************************************************************/
/* Description: This program gets DIP switch inputs and switches ON corresponding LED
                                                                        */
/*                  P1.16 to P1.31 are output switch                   */

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

#include<LPC214x.H>

int main()
{

        IO1DIR=0xFFFF0000;                          // P1.16 TO P1.31 OUTPUTPIN

         while(1)
           {
                        IOCLR1 = 0xFFFF0000; // output pin cleared for enable the led
                }

         }
```

SWITCH AND LED PORT DETAILS

ARM	DETAILS
P1.16	S&L ENABLE PIN
P1.17	S&L ENABLE PIN
P1.18	S&L ENABLE PIN
P1.19	S&L ENABLE PIN
P1.20	S&L ENABLE PIN
P1.21	S&L ENABLE PIN
P1.22	S&L ENABLE PIN
P1.23	S&L ENABLE PIN
P1.24	S&L ENABLE PIN
P1.25	S&L ENABLE PIN
P1.26	S&L ENABLE PIN
P1.27	S&L ENABLE PIN
P1.28	S&L ENABLE PIN
P1.29	S&L ENABLE PIN
P1.30	S&L ENABLE PIN
P1.31	S&L ENABLE PIN

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

PWM.C

```
/*****************************************************************************/
/* Place lcd.c file into following directories C:\Keil\ARM\INC\Philips.*/
/* This program is used to Generate the PWM, Frequency and Duty cycle can be changed*/
*****************************************************************************/

#include<LPC214x.H>

int main(void
{

PINSEL1 |= 0x00000400;//Enablepin0.7          asPWM2
PWMPR           =0x00000100;      //Load prescaler (to vary the frequency can modify here)

PWMPCR=0x00002000;     //PWM channel single edge control, output enabled
PWMMCR = 0x00000003; //On match with timer reset the counter

/* PWMR0 AND PWMR5 Both Value can change the duty cyle ex : PWMR0 = 10 AND PWMR5 = 2*/
PWMMR0 = 0x00000010; //set cycle rate to sixteen ticks
PWMMR5 = 0x00000008; //set rising edge of PWM2 to 2 ticks

PWMLER = 0x00000021; //enable shadow latch for match 0 - 2
PWMTCR = 0x00000002; //Reset counter and prescaler
PWMTCR = 0x00000009; //enable counter and PWM, release counter fromreset
while(1)      // mainloop
 {
 }
}
```

PWM PROGRAM PORT DETAIL

ARM	DETAILS
P0.7	PWM2

Post Lab Questions

1. How do the variations in an average value get affected by PWM period?
2. Name the common formats available for LED display
3. Why are the pulse width modulated outputs required in most of the applications?
4. How do you determine the duty cycle of the waveform ?
5. What is the function of GPIO?

Result

The C code is generated for Switch & LED and output is verified in LED's by Switches
The C code is generated for PWM and to vary the duty cycle and verified in CRO output.

4. INTERFACING REAL TIME CLOCK PROGRAM AND SERIAL PORT

Aim

To develop a C-Language program for reading the RTC, convert into decimal and to display it.

Pre Lab Questions

1. How would you define real time clock?
2. List the applications of real time clock.
3. What is the baud rate of serial port ?
4. Compare serial communication and parallel communication.
5. Enumerate the different modes of communication.

Apparatus & Software Required

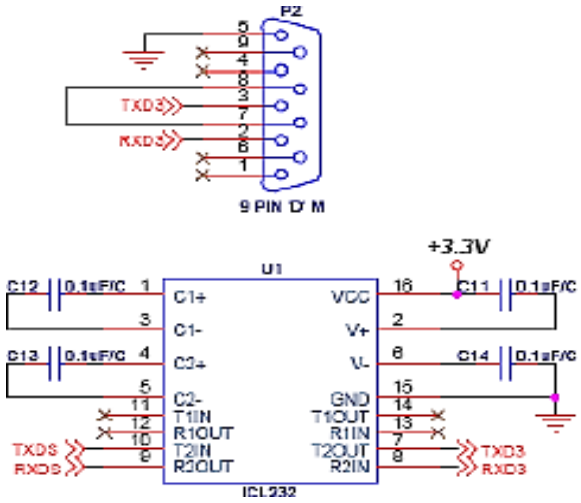
1. LPC2148 Development board.
2. KeilµVision 5 software.
3. Flash Magic.
4. USB cable.
5. RS232 Serial cable.

Theory

The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off. It uses little power in Power-down mode. On the LPC2141/2/4/6/8, the RTC can be clocked by a separate 32.768 KHz oscillator, or by a programmable prescale divider based on the VPB clock. Also, the RTC is powered by its own power supply pin,VBAT,which can be connected to a battery or to the same 3.3 V supply used by the rest of the device.

Serial Communication

Serial communication takes a byte of data and transmits the 8 bits in the byte one at a time. The advantage is that a serial port needs only one wire to transmit the 8 bits (while a parallel port needs 8). The disadvantage is that it takes 8 times longer to transmit the data than it would if there were 8 wires. Serial ports lower cable costs and make cables smaller.



Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

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

RTC.C

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

        /* Place lcd.c file into following directories C:\Keil\ARM\INC\Philips.******/
      /* This program is used to interface the RTC.You can change the date and time*/
     /* If you want. This Program can both Read and write data into RTC.RTC has a*/
   /* Battery backup for continuous Running. ************************************/
                                                    /*
pclk = 30,000,000 Hz
PREINT = (int)(pclk/32768)-1
PREFRAC = pclk - ((PREINT+1) x 32768)
*/
#include<LPC214X.H>
#include<lcd.c>
int main()
{
unsigned int hrs,min,sec;
            wait();
             wait();
             wait();
             wait();
             lcdinit();clrs
             cr(2);
             printstr("SM MICRROSYSTEM",0,0);
             printstr("         ARMDEVKIT  ",0,1);

             VPBDIV = 0x00000002; // VPB bus clock is one half of the processor clock(cclk)

             PREINT = 0x00000392; // Set RTC prescaler for 30MHz Pclk
                                                    // PREINT = (int) (30,000,000/32768(RTC
crystal))-1 = 914
              PREFRAC = 0x00004380;
              CIIR=0x00000001;          // Enable seconds counterinterrupt
              CCR=0x00000001;                  // Start theRTC
            YEAR=2009;           // Year
       MONTH=11;          //Month
       DOM=25;           // Day of month
       DOY=0;           // Day ofyear
           DOW=0;          // Day of week
           HOUR=18;          //Hours
           MIN=30;           // Minutes
           SEC =30;
           printstr("                                  ",0,1);
           while(1)
           {
              gotoxy(0,1);hrs
              = HOUR; min
              = MIN; sec =
              SEC;
              split_numbers(hrs);lcdd
              at(tens+0x30);
              lcddat(ones+0x30);
              lcddat(':');
              split_numbers(min);lcd
              dat(tens+0x30);
```

```
            lcddat(ones+0x30);
            lcddat(':');
            split_numbers(sec);lcdd
            at(tens+0x30);
            lcddat(ones+0x30);
            //lcddat(':');
        }
}
```

RTC PROGRAM PORT DETAILS

PORT	INBUILT

SERIAL PORT PROGRAM

```
/*****************************************************************/
                    /* Uart0 Initialization */

   /* This is a test program to send and receive data via uart0 in theARMLPC2148 */
                          Development board itself
    /*****************************************************************************
#include<LPC214x.H>                     /* LPC214x definitions*/
   #include"uart0.h"                    /* contains prototypes of driverfunctions*/


   int main (void)
    {
      unsigned char *s1;
      initserial();                     /* uart0 initialization */
      send_string("***********************************************");
      send_string("                  SMMicrroSystem               ");
      send_string("                     Tambaram                  ");
      send_string("                      Chennai                  ");
      send_string("***********************************************");
      send_string("");send_string("");
      send_string("This program Echos the string entered byuser."); send_string("So,type
      some strings and press ENTERkey");

      while(1)                          /* Loop forever*/
        {
              s1 = receive_string();
              send_string(s1);
        }


                                                  }
          /*****************************************************************************/
                                    SERIAL PORT PROGRAM

            /*****************************************************************************/
                                    /* Uart1 Initialization */
            /*****************************************************************************/
   #include<lpc214x.h>#inc
   lude<stdio.h>
   #include<stdlib.h>

   #include "uart1_driver.c"



   int main()
   {
   unsigned char *a;
   //unsigned char *w;
   a=malloc(sizeof(100));

   inituart1();
   sendstring1("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
   sendstring1("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
```

```
sendstring1("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
sendstring1("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
sendstring1("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
   /*sendstring1("          *         ");
   sendstring1("          **  ");
   sendstring1("         *** "); s
   endstring1("        * *** "); sendstring1(" *
   * * * * ");
   sendstring1(" SM MICRRO ");
   sendstring1(" * * * * * ");
   sendstring1("       * *** "); sendstring1("
                       ***   ");
   sendstring1("         **   ");
    sendstring1("          *         ");          */

   while(1)
   {
   receivestring1(a);send
   string1(a);

   }
      }
```

SERIAL PROGRAM

PORTDETAILSUART0

ARM	DETAILS
P0.0	TXDO
P0.1	RXDO

UART1

ARM	DETAILS
P0.8	TXD1
P0.9	RXD1

Post Lab Questions

1. What is I2C and how does it work?
2. Summarize the features of I2C in LPC2148 ARM7 microcontroller.
3. Through which port the date and time is displayed in RTC?
4. What is a serial port?
5. **List the registers used to transfer data in serial port.**

Result

The C-Language program for reading RTC and displaying it in LCD was written & output is verified with running the RTC from a default/specified time.

5.INTERFACING KEYBOARD ANDLCD

MATRIX KEYBOARD PROGRAM

Aim

To develop a C-Language program for displaying the Key pressed in the Keypad in the LCD module. The display should come in the desired line and column.

Pre Lab Questions

1. Mention the function of pull up resistor?
2. Outline the keyboard matrix.
3. Summarize the working principal of LCD.
4. What kind of interrupt is generated if a key has to be operated in an interrupt mode?
5. How many rows and columns are present in a 16 x 2 alphanumeric LCD?

Apparatus & Software Required

1. LPC2148 Development board.

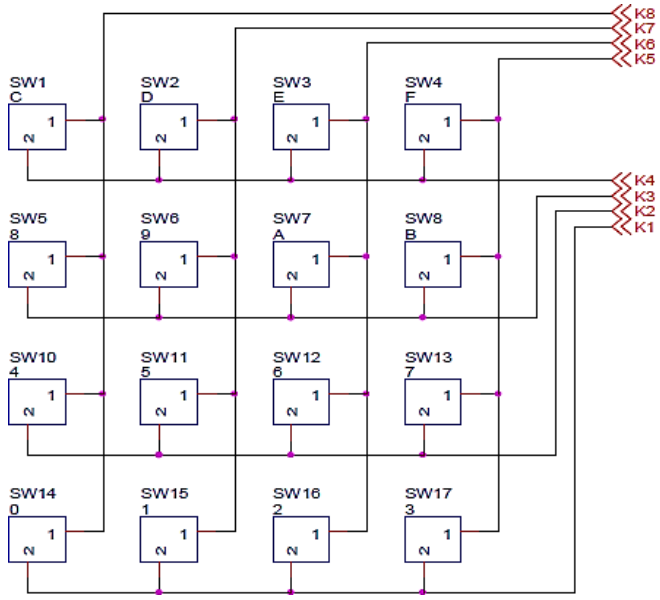
2. KeilµVision5software.

3. Flash Magic.

4. USB cable.

Theory

The Matrix keyboard is used to minimize the number of I/O lines. Normally it is possible to connect only one key or switch with an I/O line. If the number of keys in the system exceeds the more I/O lines are required. To reduce the number of I/O lines the keys are connected in the matrix circuit. Keyboards use a matrix with the rows and columns made up of wires. Each key acts like a switch. When a key is pressed a column wire makes contact with row wire and completes a circuit. For example 16 keys arranged in a matrix circuit uses only 8 I/O lines.



Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

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

MAIN.C

```
/*****************************************************************************/
/* Description: This program gets input from Matrix key board and displays
corresponding */
/*                  Key value in 7segment display. Hence this program demonstratesboth
*/                     7 segment display as well as Matrix keyboard.*/
/*                  P1.16 to P1.23 are inputs from matrix keyboard,*/
/*                  P1.24 to P1.31 are outputs to 7 segmentdisplay
*/
/*****************************************************************************
****/
/*                   ------- matrix key boarddescription----------
*/
/*                  --          --          --          --
*/
/*   row1     --| c |----| d |-- --| e |-- --|F|--                  (SW1,SW2,SW3,SW4)
*/
/*                  --          --          --          --
*/
/*                  --          --          --          --
*/
/*   row2     --| 8 |-- --| 9 |-- --| A |---- |b|--                  (SW5,SW6,SW7,SW8)
*/
/*                  --          --          --          --
*/
/*                  --          --          --          --
*/
/*   row3     --| 4 |-- --| 5 |-- --| 6 |-- --|7|--                   (SW9,SW10,SW11,SW12)
*/
/*                  --          --          --          --
*/
/*                  --          --          --          --
*/
/*   row4     --| 0 |-- --| 1 |-- --| 2 |-- --|3|--                  (SW13,SW14,SW15,SW16)
*/
/*                  --          --          --          --*/
/*****************************************************************************/
#include <LPC214x.h>
#include "mat_7seg.h"
int main()
{
 unsigned int key, last_key, Disp_key;
 init_Matrix_7seg();          // Initialize matrix keyboard and 7segment dispaly
 clearall_7seg();             // clear 7 segmentdisplay
 last_key=0;                        // Initialize this variable to zero
 while(1)
     {
          key=catch_key();          // scan for a valid keypress
          if(key!=0)                        // zero means no key ispressed
          {
           if(key!=last_key)        // check whether the same key is pressed again(assume this as STEP1)
             {
```

```
                    Disp_key=key;           // valid new key is stored in anothervariable
                    last_key=key;          // this variable's value is used forSTEP1
                  }
               }
               //Display_Number(Disp_key);          /*this function is used to display number in
decimalformat*/
               Alpha_Dispay(4,Disp_key);                  /*this function is used to display number in hex format
(single digitonly)*/
            }

}
```

/**/

MATRIX SEVEN SEGMENT DRIVER.C

/**/

```
#include <LPC214x.h>
#include "defs.h"
                              /*********************************Global
                          variables*****************************************/
                                 unsigned int thousands,hundreds,tens,ones;
     /*****************************************************************************************
                                                 ******/

     void init_Matrix_7seg(void)
      {
          IODIR1 |= 0xff0f0000; // set 7seg LEDs as output ports and matrix's MSB as
inputs and LSB as outputs
               IODIR0|=S7SEG_ENB;         // set P0.19 to P0.22 as outputs to drive 7segenable
pins
               IOPIN0|=S7SEG_ENB;                 // since we are using active low 7 seg display,the
                                                 enable signals
                                               // should be initially set to HIGH.
      }


/*****************************************************************************
*******/
     unsigned long scan_row(unsigned int row_num)
            {
               //unsigned int row,i;
               unsigned long val;

               IOSET1=ROW_MASK;     //clear the previous scan row output ie make all row opshigh
               switch(row_num)
                 {
                              case 1: IOCLR1 = ROW1;break; // make P1.16 low
                              case 2: IOCLR1 = ROW2;break; // make P1.17 low
                              case 3: IOCLR1 = ROW3;break; // make P1.18 low
                              case 4: IOCLR1 = ROW4;break; // make P1.19 low
                                          //default: row = ERR;
                 }
//             for(i=0;i<=65000;i++);
               val=IOPIN1;                    // read the matrixinputs
               val = ((val >> 20) & 0x0000000F)^0x0000000F; // shift the colum value so that it comes to LSB
                                                                  // XORing is done to take 1's
complement of shifted value.
                                                                                        //
               return(val);
            }
```

```
unsigned int catch_key(void)
  {
        unsigned long v; v =
        scan_row(1);
        switch(v)
          {
            case 1:return(13);
            case 2:return(14);
            case 4:return(15);
            case 8:return(16);
          }
    v = scan_row(2);
        switch(v)
          {
            case  1: return(9);
            case 2:return(10);
            case 4:return(11);
            case 8:return(12);
          }
    v = scan_row(3);
        switch(v)
          {
            case 1:return(5);
            case 2:return(6);
            case 4:return(7);
            case 8:return(8);
          }
    v = scan_row(4);
        switch(v)
          {
            case 1:return(1);
            case 2:return(2);
            case 4:return(3);
            case 8: return(4); default:
            return(0);
          }
      }
/*****************************************************************************************
*****/
  void clearall_7seg(void)
    {
          IOPIN1 &= ~S7SEG_LED; // make all the 7seg led pins toLOW
          IOPIN0|=S7SEG_ENB              // Disable all the 7 segdisplay
      }
/*****************************************************************************************
****/
  void clearDigit_7seg(int digit_num)
    {
          IOPIN0 |= S7SEG_ENB; // clear enables first
          switch(digit_num)
            {
              case 1: {
                        IOPIN0 =  ~DIGI1_ENB;          // now enable only the digit1
                    break;
                          }
               case  2:  {
                          IOPIN0 =  ~DIGI2_ENB;          // now enable only the digit2
                    break;
                          }
               case  3:  {
                          IOPIN0 =  ~DIGI3_ENB;          // now enable only the digit3
```

```
                                    break;
                                        }
                    case 4: {
                                IOPIN0=~DIGI4_ENB;              // now enable only the digit4 break;
                                        }
                     }

                    IOPIN1&=~S7SEG_LED;         // make all the 7seg LED pinsLOW
                }
/*****************************************************************************************
*****/
    void Digit_Dispay(int digit_num, unsigned int value)
       {
            clearDigit_7seg(digit_num);switch(
            value)
             {
                    case  0:     IOPIN1  |=   ZERO;break;
                    case  1:     IOPIN1  |=   ONE; break;
                    case  2:     IOPIN1  |=   TWO; break;
                    case  3:     IOPIN1  |=   THREE; break;
                    case  4:     IOPIN1  |=   FOUR; break;
                    case  5:     IOPIN1  |=   FIVE; break;
                    case  6:     IOPIN1  |=   SIX; break;
                    case  7:     IOPIN1  |=   SEVEN; break;
                    case  8:     IOPIN1  |=   EIGHT; break;
                    case  9:     IOPIN1  |=   NINE; break;
                }
           }
/*****************************************************************************************
*****/
    void Alpha_Dispay(int digit_num, unsigned int value)
       {
            clearDigit_7seg(digit_num);switch(
            value)
             {
                    case  1:     IOPIN1  |=   ZERO;break;
                    case  2:     IOPIN1  |=   ONE; break;
                    case  3:     IOPIN1  |=   TWO; break;
                    case  4:     IOPIN1  |=   THREE; break;
                    case  5:     IOPIN1  |=   FOUR; break;
                    case  6:     IOPIN1  |=   FIVE; break;
                    case  7:     IOPIN1  |=   SIX; break;
                    case  8:     IOPIN1  |=   SEVEN; break;
                    case  9:     IOPIN1  |=   EIGHT; break;
                   case10:        IOPIN1 |= NINE; break; case
                   11: IOPIN1 |= AAA; break; case 12:
                   IOPIN1 |= bbb; break; case 13: IOPIN1 |=
                   ccc; break; case 14: IOPIN1 |= ddd; break;
                   case 15: IOPIN1 |= eee; break; case 16:
                   IOPIN1 |= fff;break;
                }
           }
/*****************************************************************************************
******/
    void split_numbers(unsigned int number)
      {
        thousands = (number /1000);
        number %= 1000;
        hundreds = (number / 100);
        number %= 100;
        tens = (number / 10);
        number %= 10;
```

```
        ones = number ;
      }
/*****************************************************************************************
******/
    void Display_Number(unsigned int num)
      {
            unsigned int i;
            if(num <= 9999)
            {
              clearall_7seg();
              split_numbers((unsignedint)num);
              Digit_Dispay(4, ones);
              for(i=0;i<10000;i++);
              Digit_Dispay(3, tens);
              for(i=0;i<10000;i++);
              Digit_Dispay(2,hundreds);
              for(i=0;i<10000;i++);
              Digit_Dispay(1,thousands);
              for(i=0;i<10000;i++);
            }

          }
```

MATRIX SEVEN SEGMENT PROGRAM PORT DETAIL

ARM	DETAILS
P0.19	SEGMENT ENABLE PIN
P0.21	SEGMENT ENABLE PIN
P0.22	SEGMENT ENABLE PIN
P1.16	KEY BOARD INPUT
P1.17	KEY BOARD INPUT
P1.18	KEY BOARD INPUT
P1.19	KEY BOARD INPUT
P1.20	KEY BOARD INPUT
P1.21	KEY BOARD INPUT
P1.22	KEY BOARD INPUT
P1.23	KEY BOARD INPUT
P1.24	OUTPUT SEGMENT
P1.25	OUTPUT SEGMENT
P1.26	OUTPUT SEGMENT
P1.27	OUTPUT SEGMENT
P1.28	OUTPUT SEGMENT
P1.29	OUTPUT SEGMENT
P1.30	OUTPUT SEGMENT

LCD PROGRAM

/***/

LCD.h

/***/

```
void clrscr(char ch);
void lcdinit(void);
void lcdcmd(char);
void lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
voidprintstr(char*,char,char);                    //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);
void Wait_Msg(void);
void Welcome_Msg(void);
```

/***/

LCD.c

/***/

```
#include <LPC214x.h>

#defineRS              0x00000400     /* P0.10 */
#defineCE              0x00001800     /* P1.11 */

void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
voidprintstr(char*,char,char);                    //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);

#define SET1
#define OFF0

unsigned int thousands,hundreds,tens,ones;

voidwait(void)            {                                  /* wait function */ int
          d;
   for (d = 0; d <100000;d++);                              /* only to delay for LED flashes*/
}

void lcdinit()
{
      IODIR0 |= 0x0000FFFF;
          IOCLR0 |= 0X00000FFF;
          lcdcmd(0x28);lcd
      cmd(0x28);
      lcdcmd(0x0c);
      lcdcmd(0x06);
      lcdcmd(0x01);
```

```
            lcdcmd(0x0f);wait();
}

void gotoxy(char x, char y)
{
      if(y == 0)
            lcdcmd(0x80+x);
      else
            lcdcmd(0xc0+x);


}


void printstr(char *str, char x, char y)
{
      char i; gotoxy(x,y);
      wait();//(500);
      for(i=0;str[i]!='\0';i++)lcddat(str[i
            ]);

}
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
            LCDDAT = (cmd&0xf0);              //higher nibble
            IOSET0 =LCDDAT;
        IOCLR0 = RS;
            IOSET0 = CE;
        wait();//(100);                                   //enablelcd
            IOCLR0 = CE;
            IOCLR0 = 0X00000FFF;


        LCDDAT = ((cmd<<0x04)&0xf0);                      //lower nibble
            IOSET0 =LCDDAT;
        IOCLR0 = RS;
            IOSET0 = CE;
        wait();//(100);                                   //enablelcd
            IOCLR0 = CE;
            IOCLR0 = 0X00000FFF;

}

void lcddat(char cmd)
{
    unsigned charLCDDAT;
            LCDDAT = (cmd&0xf0);              //higher nibble
            IOSET0 =LCDDAT;
        IOSET0 = RS;
            IOSET0 = CE;
        wait();//(100);                                   //enablelcd
            IOCLR0 = CE;
            IOCLR0 = 0X00000FFF;

        LCDDAT = ((cmd<<0x04)&0xf0);                      //lower nibble
            IOSET0 =LCDDAT;
        IOSET0 = RS;
            IOSET0 = CE;
        wait();//(100);                                   //enablelcd
            IOCLR0 = CE;
```

```
  IOCLR0 = 0X00000FFF;
  }

void clrscr(char ch)
{
        if(ch==0)
        {
                printstr("                                ",0,0);
                gotoxy(0,0);
        }
        else if(ch ==1)
        {
                printstr("                                ",0,1);
                gotoxy(0,1);
        }
        else
        {
                    lcdcmd(0x01);
                      //delay(100);
        }
}


void split_numbers(unsigned int number)
{
    thousands = (number /1000);
    number %= 1000;
    hundreds = (number / 100);
    number %= 100;
    tens = (number / 10);
    number %= 10;
    ones = number ;
}

void Wait_Msg(void)
{
  lcdcmd(0x01);
  printstr("          PLEASEWAIT     ", 0,0);
}
void Welcome_Msg(void)
{
  lcdcmd(0x01);
  printstr("            WELCOMETO ", 0,0);
  printstr("SM MICRRO SYSTEM", 0,1);
}
```

```
/*****************************************************************************/
                                LCDmain.c
/*****************************************************************************/
/* This is a test program to display strings in LCD module in theARMLPC2148Development board itself*/
/*****************************************************************************/
#include<LPC214x.H>              /* LPC214x definitions*/
#include"lcd.h"                  /* includes lcd driverfuntions*/

int main (void)
  {
    lcdinit();                  /*Initializelcd*/
    Wait_Msg();          /*Display message - "Please Wait"*/ Welcome_Msg();
                         /*Display message - "Welcome to SMMICRRO"*/

    while(1)                    /*LoopForever*/
       {

           }
```

LCD PROGRAM PORT DETAILS

ARM	Details
PO.10	RS LCD PIN
P1.11	CE LCD PIN

Post Lab Questions

1. Outline the operations involved when the key in a 4 x 4 keyboard matrix is being pressed.
2. List the registers used to store the keyboard, display modes and other operations programmed by CPU.
3. What is switch bouncing ? How to prevent it using de-bounce circuit?
4. How to adjust the contrast of the LCD?
5. Which command of an LCD is used to shift the entire display to the right?

Result

The C-Language program for displaying the Key pressed in the Keyboard is displayed in the seven segment display and LCD module and the output was verified on the LCD on the desires line and column/address.

6. INTERFACING EPROM AND INTERRUPT

Aim

To develop a C-Language program to write and read a data in EEPROM and also to analyze its performance with the interrupt

Pre Lab Questions

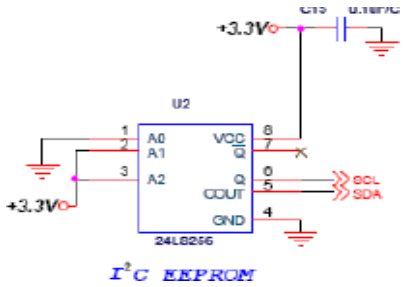
1. What is an edge triggering ?
2. Mention the advantages and disadvantages of level triggering pulse.
3. Differentiate EPROM and ROM.
4. List the different types of Memory devices.
5. Which interrupt is said to be non maskable interrupt , Why?

Apparatus & Software Required

1. LPC2148 Development board.
2. Keil µVision5 software.
3. Flash Magic.
4. USB cable.

Theory

Serial-interface EEPROM's are used in a broad spectrum of consumer, automotive, telecommunication, medical, industrial and PC related markets. Primarily used to store personal preference data and configuration/setup data, Serial EEPROM's are the most flexible type of nonvolatile memory utilized today. Compared to other NVM solutions, Serial EEPROM devices offer a lower pin count, smaller packages, lower voltages, as well as lower powerconsumption.



I²C EEPROM

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

EPROM PROGRAM

/***/

I2C .C

/***/

```
/* This Program For I2C Interface */
#include<LPC214x.H>
#include "lcd.c"

void InitI2C (void);
void SendI2C Address(unsigned char Addr_S); void
WriteI2C (unsigned char Data);
void StopI2C (void); void
StartI2C (void);

#define    STA    0x20
#define    SIC    0x08
#define     SI    0x08
#define    STO    0x10
#define   STAC   0x20
#define AA 0x04

void InitI2C (void)
{
    I2C 0CONCLR = 0xFF;
    PINSEL0   |=0x50;          //     Set pinouts as scl andsda
    I2C 0SCLL    =19;          //speed at 100Khz for a VPBClockDivider          = 4 at 12 MHz
    I2C 0SCLH    =19;
    I2C 0CONSET=0x40;          //Active Master Mode on I2C bus
}


void SendI2C Address(unsigned char Addr_S)
{
      while(I2C 0STAT!=0x08);          // Wait for start to becompleted
      I2C 0DAT     =Addr_S;            // Charge slaveAddress
      I2C 0CONCLR = SIC|STAC;          // Clear I2C interrupt bit to send the data
      while(!( I2C 0CONSET &SI));      // wait till statusavailable
}


unsigned char ReadI2C (void)
{
unsigned char r;
```

```
        I2C 0CONCLR = SIC;
        I2C 0CONSET=0x04;                       // clearSIC;
    while(!(I2C 0CONSET&0x8));                  // wait till statusavailable r=I2C
    0STAT;
        wait();                                 // check forerror
    if (r==0x50){                               // look for "Data byte has been received; ACKhas been returned"
         lcdcmd(0x01);
            printstr("Read Sucess",0,0);
    }

    return I2C 0DAT;
}

void WriteI2C (unsigned char Data)
{
    unsigned char r;
    I2C 0DAT      =Data;                        // ChargeData
    I2C 0CONCLR=0x8;                            // SIC; Clear I2C interrupt bit to send thedata while(!(I2C
    0CONSET&0x8));                              // wait till statusavailable
    r=I2C 0STAT;
    if (r == 0x28)
        {                           // look for "Data byte in S1DAT has been transmitted; ACKhas been received"
       lcdcmd(0x01);
           printstr("Write Sucess",0,0);
    }
}
void StopI2C (void)
{
    I2C 0CONCLR = SIC;
    I2C 0CONSET =
    STO;
    while((I2C 0CONSET&STO));                   // wait for Stopped busI2C
}

void StartI2C (void)
{
     I2C 0CONCLR=0xFF;                          // clear I2C - included if User forgot to "StopI2C ()"
                                                // else this function would hang.
    I2C 0CONSET=0x40;                           // Active Master Mode on I2C bus I2C
        0CONSET=0x00000020;                         // Startcondition
}
int main()
{

  unsigned char r;

            wait();
            wait();
            wait();
            wait();
            lcdinit();clrsc
            r(2);
            printstr("SM MICRRO SYSTEM",0,0);
            printstr("          ARMDEVKIT    ",0,1);
       InitI2C ();
            StartI2C ();
            SendI2C Address(0xa0);          // EEPROM device address
       WriteI2C (0);                        // Set the control portvalue
            WriteI2C ('B');
```

```
        StopI2C
        ();wait();
        wait();
        StartI2C ();
        SendI2C Address(0xa0);          // EEPROM device address
    WriteI2C (0);                       // Set the control portvalue
        StopI2C ();
        StartI2C ();
        SendI2C Address(0xa1);          // Start the read
        r=ReadI2C ();                   // read the result
        StopI2C ();
        gotoxy(0,1);
        split_numbers(r);
        lcddat(0x30+hundreds);lcdda
        t(0x30+tens);
        lcddat(0x30+ones); while(1);

  }
/*****************************************************************************************/
```

LCD.C

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

#defineRS           0x00000400    /* P0.10 */
#defineCE           0x00001800    /* P1.11 */
void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
voidprintstr(char*,char,char);                  //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);
#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones;
voidwait(void)          {                       /* wait function */
intd;
  for (d = 0; d <100000;d++);                   /* only to delay for LED flashes*/
}

void lcdinit()
{
    IODIR0 = 0xFFFFFFFF;
      IOCLR0 = 0X00000FFF;
      lcdcmd(0x28);
    lcdcmd(0x28);
    lcdcmd(0x0c);
    lcdcmd(0x06);
    lcdcmd(0x01);
    lcdcmd(0x0f);
    wait();//(1600);
}
```

```
void gotoxy(char x, char y)
{
        if(y == 0)
                lcdcmd(0x80+x);
        else
                lcdcmd(0xc0+x);


}
void printstr(char *str, char x, char y)
{
        char i; gotoxy(x,y);
        wait();//(500);
        for(i=0;str[i]!='\0';i++)lcddat(str[i
                ]);

}
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
                LCDDAT = (cmd&0xf0);                    //higher nibble
                IOSET0 =LCDDAT;
          IOCLR0 = RS;
                IOSET0 = CE;
          wait();//(100);                                       //enablelcd
                IOCLR0 = CE;
                IOCLR0 = 0X00000FFF;


          LCDDAT = ((cmd<<0x04)&0xf0);                  //lower nibble
                IOSET0 =LCDDAT;
          IOCLR0 = RS;
                IOSET0 = CE;
          wait();//(100);                                       //enablelcd
                IOCLR0 = CE;
                IOCLR0 = 0X00000FFF;

}


void lcddat(char cmd)
{
     unsigned charLCDDAT;
                LCDDAT = (cmd&0xf0);                    //higher nibble
                IOSET0 =LCDDAT;
          IOSET0 = RS;
                IOSET0 = CE;
          wait();//(100);                                       //enablelcd
                IOCLR0 = CE;
                IOCLR0 = 0X00000FFF;
          LCDDAT = ((cmd<<0x04)&0xf0);                  //lower nibble
                IOSET0 =LCDDAT;
          IOSET0 = RS;
                IOSET0 = CE;
          wait();//(100);                                       //enablelcd
                IOCLR0 = CE;
                IOCLR0 = 0X00000FFF;
}
void clrscr(char ch)
{
```

```
        if(ch==0)
        {
                printstr("                              ",0,0);
                gotoxy(0,0);
        }
        else if(ch ==1)
        {
                printstr("                              ",0,1);
                gotoxy(0,1);
        }
        else
        {
                lcdcmd(0x01);
                  //delay(100);
        }
}


void split_numbers(unsigned int number)
{
   thousands = (number /1000);
   number %= 1000;
   hundreds = (number / 100);
   number %= 100;
   tens = (number / 10);
   number %= 10;
   ones = number ;
}
```

EPROM (I2C) PROGRAM PORT SETAILS

ARM	DETAILS
PO.10	RS LCD PIN
P1.11	CE LCD PIN
P0.11	SCL
P0.14	SDA

INTERRUPT BUZZER PROGRAM

/***/

ExtDriver.C

/***/

```
#include <LPC214x.h>
  void init_VIC(void)
    {
                /* initialize VIC*/
        VICIntEnClr       =0xffffffff;
        VICVectAddr     =0;
        VICIntSelect =0;
    }


  void ExtInt_ISR(void)irq
  {
    //EXTINT=(1<<2);             /* clear EINT2 flag by writing HIGH to corespondingbit*/
    //IOCLR0 = 0x40000000; /* Trigger the relay*/
              IOCLR1 = 0x400f0000; /* P1.18 Trigger the relay*/
            //IOPIN1 = 0x00000000;
    EXTINT = (1<<2);
    VICVectAddr=0;                      /* Acknowledge Interrupt*/
  }
  void init_Interrupt(void)
        {
                PINSEL0=0x80000000;               // select P0.15 for EINT2
                VICIntEnable = (1<<16);                 // External interrupt 2(EINT2)
                VICVectCntl0=(1<<5)|(16);               // set the VIC control reg for EINT2
                VICVectAddr0 = (unsignedlong)ExtInt_ISR;
                EXTMODE&=~(1<<2);       // set VIC for egdse sensitive forEINT2
//              EXTPOLAR = ~(1<<2); // set VIC for falling edge sensitive forEINT2
            }
  void init_ports(void)
        {
                IODIR0   =   0x40000000;
                IODIR1   =   0x400f0000;
                IOPIN1   =    0xff010000;
                IOSET0 = 0x40000000;
                        IOSET1 = 0x400f0000;

            }
  /*void wait_for_turnoffRelay(void)
        {
                int val;
                val=IOPIN1;                                     // read the ports for key board input
                while((~(val>>20))!=0);                         // wait until 1st key in the matrixkeyboard
is pressed
                IOCLR0=0x00010000;                      // switch off therelay
            }*/
```

```
/*******************************************************************************************/
                                        XINTR_RELAY.C
/*******************************************************************************************/
#include <LPC214x.h>
#include "ext.h"
int main()
 {
    init_VIC();
    init_Interrupt();init_po
    rts();
        while(1)
                {
                  //wait_for_turnoffRelay();
                }
 }
```

INTERRUPT BUZZERPROGRAM

ARM	DETAILS
P1.18	TRIGGER THE RELAY
P0.15	EINT2

Post Lab Questions

1. What will be the initial values in all the cells of an EPROM ?
2. What are the contents of the IE register, when the interrupt of the memory location 0x00 is caused?
3. Why normally LJMP instructions are the topmost lines of the ISR?
4. Enumerate the features of nested interrupt.
5. Illustrate the Master Slave mode.

Result

The C-Language program to write and read a data in EEPROM and also to analyze its performance with the interrupt is developed and is verified.

7. MAILBOX

Aim

To develop a 'C'code to create a mailbox and to understand the RTOS functions.

Pre Lab Questions

1. How does mailbox works in RTOS?
2. What is Semaphore?
3. Differentiate mailbox and queue.
4. List the synchronous and asynchronous modes are there in serial port?
5. Interpret the inter process communication

Apparatus & Software Required

1. LPC2148 Development board.
2. KeilµVision5software.
3. Flash Magic.
4. USB cable.

.

Theory

Real-time and embedded systems operate in constrained environments in which computer memory and processing power are limited. They often need to provide their services within strict time deadlines to their users and to the surrounding world. It is these memory, speed and timing constraints that dictate the use of real-time operating systems in embedded software.

The "kernel" of a real-time operating system ("RTOS") provides an "abstraction layer" that hides from application software the hardware details of the processor (or set of processors) up on which the application software will run.

In providing this "abstraction layer" the RTOS kernel supplies five main categories of basic services to application software

The most basic category of kernel services is Task Management. This set of services allows application software developers to design their software as a number of separate "chunks" of software -- each handling a distinct topic, a distinct goal, and perhaps its own real-time deadline. Each separate "chunk" of software is called a "task." The main RTOS service in this category is the scheduling of tasks as the embedded system is in operation.

The second category of kernel services is Inter task Communication and Synchronization. These services make it possible for tasks to pass information from one to another, without danger of that information ever being damaged. They also make it possible for tasks to coordinate, so that they can productively cooperate with one another. Without the help of these RTOS services, tasks might well communicate corrupted information or otherwise interfere with each other. Since many embedded systems have stringent timing requirements, most RTOS kernels also provide some basic Timer services, such as task delays and time-outs.

Many (but not all) RTOS kernels provide Dynamic Memory Allocation services. This category of services allows tasks to "borrow" chunks of RAM memory for temporary use in application software. Often these chunks of memory are then passed from task to task, as a means of quickly communicating large amounts of data between tasks. Some very small RTOS kernels that are intended for tightly memory-limited environments, do not offer Dynamic memory allocation.

Many (but not all) RTOS kernels also provide a "Device I/O Supervisor" category of services. These services, if available, provide a uniform framework for organizing and accessing the many hardware device drivers that are typical of an embedded system.

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

/***/

MAILBOX.C

/***/

```
#include <string.h>
#include <stdio.h>
#include <RTL.h>
#include <LPC214x.H>                        /* LPC214x definitions                  */
#include "config.h"
#include "uart.h" #include
"lcd.h"

OS_TID tsk1;                                /* assigned identification for task 1   */
OS_TID tsk2;                                /* assigned identification for task 2   */


typedefstruct{                              /* Messageobjectstructure               */
   charmsgBuf[MBOX_MSG_BUF_SIZE];
} T_MEAS;

os_mbx_declare(MsgBox,MAILBOX_MEMORY_POOL_CNT); /* Declare an RTXmailbox*/
_declare_box (mpool,sizeof(T_MEAS),MAILBOX_MEMORY_POOL_CNT);/* Dynamic memory pool*/

task void send_task (void);
task void rec_task (void);
void main_menu()
{
      send_string(USE_UART,"\n\r\n\n*********************************************");
      send_string(USE_UART,"\n\r          SM Micrro System,Tambaram,Chennai          ");
      send_string(USE_UART,"\n\r               MailBoxMessageSimulation             ");
      send_string(USE_UART,"\n\r                        MAIN MENU");
      send_string(USE_UART,"\n\r*********************************************");
      send_string(USE_UART,"\n\n\r");
      send_string(USE_UART,"\n\rThis program simulates MailBox IPC mechanism.");
      send_string(USE_UART,"\n\rPlease Follow Below Commands"); send_string(USE_UART,"\n\r - Type any
      string and press enter to send"); send_string(USE_UART,"\n\r                                the
      stringusingmailbox.                        "); send_string(USE_UART,"\n\r-Press     SPACE Key to
      check available MailboxCount"); send_string(USE_UART,"\n\r-Press                    ESC Key to reset
      the input string"); send_string(USE_UART,"\n\n\r");
}
/* ---------------------------------------------------------------------------------------------------------
 *    Task1:       RTX Kernel starts this task with os_sys_init(send_task)
 *  ---------------------------------------------------------------------------------------------------------*/
task void send_task (void)
{
      T_MEAS *mptr;
      static unsigned char sInputBuf[MBOX_MSG_BUF_SIZE]; int
      cnt=0;
      char sSndTskBuf[30]; char
      ch;
      int MsgFree = 0;

      tsk1 =os_tsk_self();                  /* get own taskidentificationnumber     */
#ifndefDISABLE_RECV_TASK
      tsk2 = os_tsk_create (rec_task, 0); /* starttask2                             */
#endif /* DISABLE_RECV_TASK*/

      os_mbx_init (MsgBox, sizeof(MsgBox));/* initializethemailbox                 */
      os_dly_wait(5);                       /* Startup delayforMCB21xx              */
          lcdinit();
```

```
        clrscr(10);
      printstr("          MailBox       ",0,0);
      printstr("        Simulation      ",0,1);

#ifndef DISABLE_RECV_TASK
      mptr =_alloc_box(mpool);                    /* Allocate a memory forthemessage          */
      memset(mptr->msgBuf,'\0',MBOX_MSG_BUF_SIZE);
      memcpy ( mptr->msgBuf, STD_MSG1, sizeof(STD_MSG1) );
      os_mbx_send (MsgBox, mptr, 0xffff); /* Send the message to the mailbox*/
      os_dly_wait (100);

      mptr = _alloc_box (mpool);
      memset (mptr->msgBuf,'\0',MBOX_MSG_BUF_SIZE);
      memcpy ( mptr->msgBuf, STD_MSG2, sizeof(STD_MSG2) );
      os_mbx_send (MsgBox, mptr, 0xffff); /* Andsendit.                                */
      os_tsk_pass();                              /*Cooperativemultitasking                  */
      os_dly_wait(100);

      mptr = _alloc_box (mpool);
      memset (mptr->msgBuf,'\0',MBOX_MSG_BUF_SIZE);
      memcpy ( mptr->msgBuf, STD_MSG3, sizeof(STD_MSG3) );
      os_mbx_send (MsgBox, mptr, 0xffff); /* Andsendit.                                */
      os_dly_wait(100);
      #endif /* DISABLE_RECV_TASK */
      memset (sInputBuf,'\0',MBOX_MSG_BUF_SIZE);
      cnt = 0;
      main_menu();w
      hile(1)
      {
            ch = receive(USE_UART);
            if(ch == CARRIAGE_RET && cnt > 0)
            {
send_string(USE_UART,"\n\n\n\r**********SENDING MAILBOX USER MESSAGE**********");
                        MsgFree = os_mbx_check (MsgBox); if
                        (MsgFree != 0)
                        {
                              mptr = _alloc_box (mpool);
                              memset (mptr->msgBuf,'\0',MBOX_MSG_BUF_SIZE);

                                 memcpy ( mptr->msgBuf, sInputBuf, strlen((const char *)sInputBuf) );

*/                            sprintf (sSndTskBuf, "\n\ros_mbx_send byTaskID:%d ",tsk1);

                                 send_string(USE_UART,sSndTskBuf);send
                                 _string(USE_UART,"\n\n\r");

                        }
                              os_mbx_send (MsgBox, mptr, 0xffff); /* And send it. os_dly_wait
                        else
                        {
                              (100);

                        }
                              memset (sInputBuf,'\0',MBOX_MSG_BUF_SIZE); cnt
#ifdefDISABLE_RECV_TASK       = 0;



                              send_string(USE_UART,"\n\rMailbox is FULL");
                        main_menu();
```

```
#endif /* DISABLE_RECV_TASK */
            }
            else if ( KEY_SPACE == ch)
            {
                MsgFree = os_mbx_check (MsgBox); if
                (MsgFree == 0)
                {
                    send_string(USE_UART,"\n\rMailboxisFULL.....");
                }
                else
                {
                    sprintf (sSndTskBuf, "\n\rMailBox Free Count : %d ", MsgFree);
                    send_string(USE_UART,sSndTskBuf);
                                send_string(USE_UART,"\n\n\r");
                }
                    os_dly_wait (100);
                main_menu();

            }
            else if ( KEY_ESC == ch)
            {
                cnt = 0;
                memset (sInputBuf,'\0',MBOX_MSG_BUF_SIZE);
                                                send_string(USE_UART,"\n\rClearing Buffer
PleaseWait. ..... ");
                                                os_dly_wait (100);
                main_menu();
            }
            else if('\0' != ch)
            {


                        sInputBuf[cnt++] = ch;

                        cnt %= MBOX_MSG_BUF_SIZE;
                        if(ch == CARRIAGE_RET &&cnt==1)          //emptystring
                        {
                                cnt = 0;
                        }
                        else
                        {
                                putcharr (USE_UART,ch);
                        }


}
        }

//          os_tsk_delete_self();                       /* We are done here, deletethistask                    */
}

#ifndef DISABLE_RECV_TASK
/* ---------------------------------------------------------------------------------------------------------------------
 *    Task 2: RTX Kernel starts this task with os_tsk_create (rec_task,0)
 *--------------------------------------------------------------------------------------------------------------------*/
task void rec_task (void)
{
   T_MEAS *rptr;
   static char sRxTskBuf[MBOX_MSG_BUF_SIZE];

        for (;;)
        {
            os_mbx_wait (MsgBox, (void **)&rptr, 0xffff); /* wait forthemessage                                    */
            send_string(USE_UART,"\n\n\n\r************MAILBOXMESSAGE
```

```
RECEIVED**********');
```

```
sprintf (sRxTskBuf, "\n\rec_task by Task ID: %d ", tsk2); send_string(USE_UART,sRxTskBuf);

                memset (sRxTskBuf,'\0',MBOX_MSG_BUF_SIZE);
                memcpy ( sRxTskBuf, rptr->msgBuf, strlen(rptr->msgBuf) ); send_string(USE_UART,"\n\n\rReceived Mbox: ");
                send_string(USE_UART,sRxTskBuf);

    send_string(USE_UART,"\n\r*****************************************************");
                _free_box(mpool,rptr);                          /* free memory allocatedformessage                  */
                main_menu();
        }
    }
    #endif /* DISABLE_RECV_TASK */

    /* ----------------------------------------------------------------------------------------------------------------
     *                Main: Initialize and start RTXKernel
     *---------------------------------------------------------------------------------------------------------------*/
    int main (void)
    {
          initserial(USE_UART);                             /* uart0 initialization*/

          _init_box(mpool,sizeof(mpool),                    /* initialize the 'mpool'memoryfor                  */
                          sizeof(T_MEAS));                  /* the memboxdynamicallocation                      */
          os_sys_init(send_task);                           /* initialize and starttask1                        */
    }

    /* ----------------------------------------------------------------------------------------------------------------
     * end of file
     *---------------------------------------------------------------------------------------------------------------*/


                            /*************************************************************/
                                                RTX_CONFIG.C
                            /*************************************************************/
     /*----------------------------------------------------------------------------------------------------------------
     *          RL-ARM -RTX
     *----------------------------------------------------------------------------------------------------------------
     *          Name:       RTX_CONFIG.C
     *          Purpose: Configuration of RTX Kernel for NXPLPC21xx
     *          Rev.:         V4.20
     *----------------------------------------------------------------------------------------------------------------
     *          This code is part of the RealView Run-TimeLibrary.
     *          Copyright (c) 2004-2011 KEIL - An ARM Company. All rightsreserved.
     *---------------------------------------------------------------------------------------------------------------*/


    #include <RTL.h>
    #include<LPC21xx.H>                                     /*LPC21xxdefinitions                               */

    /* ----------------------------------------------------------------------------------------------------------------
     *          RTX User configuration partBEGIN
     *---------------------------------------------------------------------------------------------------------------*/


    //-------- <<< Use Configuration Wizard in Context Menu >>> -----------------
    //
    // <h>TaskConfiguration
    //======================
    //
```

```
//          <o>Number of concurrent running tasks<0-250>
    //       <i> Define max. number of tasks that will run at the sametime.
    //       <i> Default: 6
    #ifndefOS_TASKCNT
    #defineOS_TASKCNT          6
    #endif

    //       <o>Number of tasks with user-provided stack<0-250>
    //       <i> Define the number of tasks that will use a biggerstack.
    //       <i> The memory space for the stack is provided by theuser.
    //       <i> Default: 0
    #ifndefOS_PRIVCNT
    #defineOS_PRIVCNT          0
    #endif
    //       <o>Task stack size [bytes]<20-4096:8><#/4>
    //       <i> Set the stack size for tasks which is assigned by thesystem.
    //       <i> Default: 200
    #ifndefOS_STKSIZE
    #defineOS_STKSIZE          50
    #endif

    // <q>Check for the stackoverflow
    //================================
    // <i> Include the stack checking code for a stack overflow.
    // <i> Note that additional code reduces the RTX performance. #ifndef
    OS_STKCHECK
      #defineOS_STKCHECK       1
    #endif

    // </h>
    // <h>Tick Timer Configuration
    // ============================
    //       <o>Hardware timer <0=> Timer 0 <1=> Timer1
    //       <i> Define the on-chip timer used as a time-base forRTX.
    //       <i> Default: Timer 0
    #ifndefOS_TIMER
    #defineOS_TIMER            1
    #endif

    //       <o>Timer clock value [Hz]<1-1000000000>
    //       <i> Set the timer clock value for selectedtimer.
    //       <i>Default:15000000          (15MHz at 60MHz CCLK and VPBDIV = 4)
    #ifndefOS_CLOCK
      #defineOS_CLOCK          15000000
    #endif

    //       <o>Timer tick value [us]<1-1000000>
    //       <i> Set the timer tick value for selectedtimer.
    //       <i>Default:10000          (10ms)
    #ifndefOS_TICK
      #defineOS_TICK           10000
    #endif

    // </h>

    // <h>SystemConfiguration
    //=======================
    // <e>Round-Robin Taskswitching
    //=============================
    // <i> Enable Round-Robin Task switching. #ifndef
    OS_ROBIN
      #defineOS_ROBIN          1
```

```
#endif

//       <o>Round-Robin Timeout [ticks]<1-1000>
//       <i> Define how long a task will execute before a taskswitch.
//       <i> Default: 5
#ifndefOS_ROBINTOUT
#defineOS_ROBINTOUT      5
#endif

// </e>

//       <o>Number of user timers<0-250>
//       <i> Define max. number of user timers that will run at the sametime.
//       <i>Default:0           (User timersdisabled)
#ifndefOS_TIMERCNT
 #defineOS_TIMERCNT       0
#endif

//       <o>ISR FIFOQueuesize<4=>          4entries        <8=>     8entries
//                                    <12=>12entries        <16=> 16entries
//                                    <24=>24entries        <32=> 32entries
//                                    <48=>48entries        <64=> 64entries
//                                    <96=> 96entries
//       <i> ISR functions store requests to thisbuffer,
//       <i> when they are called from the IRQhandler.
//       <i> Default: 16 entries
#ifndefOS_FIFOSZ
#defineOS_FIFOSZ             16
#endif

// </h>
//------------- <<< end of configuration section >>> -----------------------

// Standard library systemmutexes
//==============================
//     Define max. number system mutexes that are used toprotect
//     the arm standard runtime library. For microlib they arenotused.
#ifndefOS_MUTEXCNT
 #defineOS_MUTEXCNT       8
#endif

/* ----------------------------------------------------------------------------------------------------------------------
 *          RTX User configuration partEND
 *---------------------------------------------------------------------------------------------------------------------*/

#if       (OS_TIMER==0)                                          /*Timer0                */
 #defineOS_TID_              4                                   /*     TimerID          */
 #defineTIMx(reg)            T0##reg
#elif (OS_TIMER==1)                                              /*Timer1                */
 #defineOS_TID_              5                                   /*     TimerID          */
 #defineTIMx(reg)            T1##reg
#else
 #error OS_TIMER invalid
#endif

#defineOS_TIM_               (1<<OS_TID_)                        /*     InterruptMask    */
#defineOS_TRV                ((U32)(((double)OS_CLOCK*(double)OS_TICK)/1E6)-1)
#defineOS_TVAL               TIMx(TC)                            /*     TimerValue       */
#defineOS_TOVF               (TIMx(IR)&1)     /*                 OverflowFlag          */
#defineOS_TFIRQ()            VICSoftInt       =OS_TIM_;          /*     Force Interrupt */
#defineOS_TIACK()            TIMx(IR)=1;                         /*     InterruptAck     */ \
```

```
#define   OS_TINIT()      VICSoftIntClr = OS_TIM_;                                  \
                          VICVectAddr      = 0;
                          TIMx(MR0) =OS_TRV;          /*   Initialization   */ \
                          TIMx(MCR) =3;                                           \
                          TIMx(TCR) =1;                                           \
                          VICDefVectAddr = (U32)os_def_interrupt;                 \
                          VICVectAddr15    =(U32)os_clock_interrupt;              \
                          VICVectCntl15    = 0x20 |OS_TID_;
#define   OS_IACK()       VICVectAddr      =0;        /* InterruptAck      */

#define#  OS_LOCK()       VICIntEnClr      =OS_TIM_;  /* Task Lock         */
define    OS_UNLOCK       VICIntEnable     =OS_TIM_;  /* Task Unlock       */
          ()

/* WARNING: Using IDLE mode might cause you troubles while debugging.*/ #define_idle_()
                          PCON =1;


/* ---------------------------------------------------------------------------------------------------------
 *           GlobalFunctions
 *--------------------------------------------------------------------------------------------------------*/

/* ---------------------------------------- os_idle_demon  ---------------------------------------------*/
task void os_idle_demon (void) {
   /* The idle demon is a system task, running when no other task is ready*/
   /* to run. The 'os_xxx' function calls are not allowed fromthistask.                */


   for (;;) {
   /* HERE: include optional user code to be executed when no task runs.*/
   }
}

/* ---------------------------------------- os_tmr_call  ----------------------------------------------*/

void os_tmr_call (U16 info) {
   /* This function is called when the user timer hasexpired.Parameter                 */
   /* 'info' holds the value, defined when the timerwascreated.                        */

   /* HERE: include optional user code to be executed on timeout. */
}


/*----------------------------------------os_error  ----------------------------------------------*/

void os_error (U32 err_code){
   /* This function is called when a runtime error is detected. Parameter*/
   /* 'err_code' holds the runtime error code (definedinRTL.H).                        */


   /* HERE: include optional code to be executed on runtime error. */ for (;;);
}
/* ---------------------------------------------------------------------------------------------------------
 *           RTX ConfigurationFunctions
 *--------------------------------------------------------------------------------------------------------*/

static void os_def_interrupt(void)irq                    {
   /* Default Interrupt Function: may be called when timer ISR is disabled */ OS_IACK();
}
```

```
#include <RTX_lib.c>

/* ------------------------------------------------------------------------------------------------------------------
 *  end offile
 *-----------------------------------------------------------------------------------------------------------------*


 /*****************************************************************************/
                                                LCD.C
                        /*************************************************************/

#include<LPC214x.H>                     /* LPC214x definitions */
#defineRS               0x00000400      /* P0.10*/
#defineCE               0x00001000      /* P1.11*/

#define SET1
#define OFF0

void lcdcmd(char cmd); void
lcddat(char cmd);
void printstr(unsigned char *str, char x, char y);

voidwait(void)              {                                   /* wait function */ int
            d;
   for (d = 0; d <100000;d++);                                  /* only to delay for LED flashes*/
}


void lcdinit(void)
{
        IODIR0 |= 0x000014f0;
        lcdcmd(0x28);
        lcdcmd(0x28);
        lcdcmd(0x0c);
        lcdcmd(0x06);
        lcdcmd(0x01);
        lcdcmd(0x0f);
        wait();//(1600);

}

void gotoxy(char x, char y)
{
        if(y == 0)
                lcdcmd(0x80+x);
        else
                lcdcmd(0xc0+x);

}


void printstr(unsigned char *str, char x, char y)
{
        char i; gotoxy(x,y);
        wait();//(500);
        for(i=0;str[i]!='\0';i++)lcddat(str[i
                ]);

}
```

```
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
            LCDDAT = (cmd&0xf0);                    //higher nibble
            IOSET0 |=LCDDAT;
        IOCLR0 |= RS;
            IOSET0 |= CE;
        wait();//(100);                                         //enablelcd
            IOCLR0 |= CE;
            IOCLR0 |= 0X00000FFF;


        LCDDAT = ((cmd<<0x04)&0xf0);                    //lower nibble
            IOSET0 |=LCDDAT;
        IOCLR0 |= RS;
            IOSET0 |= CE;
        wait();//(100);                                         //enablelcd
            IOCLR0 |= CE;
            IOCLR0 |= 0X00000FFF;

}

void lcddat(char cmd)
{
    unsigned char LCDDAT;
            LCDDAT = (cmd&0xf0);                    //higher nibble
            IOSET0 |=LCDDAT;
        IOSET0 |= RS;
            IOSET0 |= CE;
        wait();//(100);                                         //enablelcd
            IOCLR0 |= CE;
            IOCLR0 |= 0X00000FFF;


        LCDDAT = ((cmd<<0x04)&0xf0);                    //lower nibble
            IOSET0 |=LCDDAT;
        IOSET0 |= RS;
            IOSET0 |= CE;
        wait();//(100);                                         //enablelcd
            IOCLR0 |= CE;
            IOCLR0 |= 0X00000FFF;
}

void clrscr(char ch)
{
      if(ch==0)
      {
            printstr("                                ",0,0);
            gotoxy(0,0);
      }
      else if(ch ==1)
      {
            printstr("                                ",0,1);
            gotoxy(0,1);
      }
      else
      {
              lcdcmd(0x01);
                //delay(100);
      }
}
```

```
/*************************************************************/
                        UART.C
/*************************************************************/


 /* This file contains driver functions to send and receive data via uart0inthe                    */
                        /* ARM LPC2148 Development board itself */
 /*****************************************************************************/ #include
<LPC214x.H>
#include "config.h"
#define TEMT (1<<6)

void initserial(unsigned char uart)
{
        if(0 == uart)
        {
                PINSEL0=0x00000005;         /* Make pins 19 and 21 to function as TXD0and RXD0
forUART0*/
                U0LCR=0x83;                 /* 8 bits, no Parity, 1Stopbit                    */
                U0FDR=0x00000010;           /* DIVADDVAL = 0; MULVAL = 1*/
                U0DLL=98;                   /* 9600 Baud Rate @ 15MHz VPB Clock; =97.65=98

*/
                U0LCR=0x03;                 /* DLAB = 0*/
                U0IER=0x01;                 /* Enable reciever data availableinterrupt*/

        }
        else
        {
                PINSEL0=0x00050000;         /* Make pins 19 and 21 to function as TXD0and
RXD0 for UART0*/
                U1LCR=0x83;                 /* 8 bits, no Parity, 1Stopbit                    */
                U1FDR=0x00000010;               /* DIVADDVAL = 0; MULVAL = 1*/
                U1DLL=98;                   /* 9600 Baud Rate @ 15MHz VPB Clock; =97.65=98
*/
                U1LCR=0x03;                 /* DLAB = 0*/
                U1IER=0x01;                 /* Enable reciever data availableinterrupt*/


        }
}
void putcharr (unsigned char uart, unsignedcharch)                  /* Writes character to
SerialPort*/
{
        if(0 == uart)
        {
                while (!(U0LSR &TEMT)); U0THR
                =ch;
        }
        else
        {
                while (!(U1LSR &TEMT)); U1THR
                =ch;
        }
}



unsigned char getcharr (unsignedcharuart)                       /* Reads character from
SerialPort*/
{
        if(0 == uart)
        {
                while (!(U0LSR & 0x01));
```

```
return (U0RBR);
```

```
        }
        else
        {
            while (!(U1LSR & 0x01));
            return (U1RBR);
        }
}

char receive(unsigned char uart) /*function for receiving data from sensor (readsbyte by byte & returns value if
exist,else#)                                                    */
{
        if(0 == uart)
        {
                if (U0LSR&0x01)          /* If U0LSR 1st bit contains valid data, thenreturn value ofU0RBR*/
                {
                        return (U0RBR);
                }
                return'\0';                         /* If other than 0 to 9 data is recievedreturn
#*/
        }
        else
        {
                if (U1LSR&0x01)          /* If U0LSR 1st bit contains valid data, thenreturn

value of U0RBR*/
                {
                        return (U1RBR);
                }


                return'\0';                         /* If other than 0 to 9 data is recievedreturn
#*/


        }
}

void send_string(unsigned charuart,char*cpr)                    /* Writes string to serial port*/
{
    while(*cpr != '\0')
    {
        putcharr (uart,*cpr); cpr++;
    }
}

unsigned char* receive_string(unsignedcharuart)                         /* Reads string toserial
port*/
{
    static unsigned char c[30]; unsigned
    char i=0;
    c[i] = getcharr(uart); while(c[i] !=
    CARRIAGE_RET)
      {
          i++;
          c[i] = getcharr(uart);


      }
    c[i] = '\0';
    return(c);
}
```

CONFIG.H

```
/*****************************************************************/
#define          MAILBOX_MEMORY_POOL_CNT16
#define          MBOX_MSG_BUF_SIZE       100
#defineUSE_UART                                          0
/*****************************************************************/
/*              Enable below macro to disable the the recv task to check mailbox full*/
//#define            DISABLE_RECV_TASK
/*****************************************************************/
#define STD_MSG1 "MailBox Test Message 1" #define
STD_MSG2 "MailBox Test Message 2" #define STD_MSG3
"MailBox Test Message3"
#define  LINE_FEED  0x0A  #define
CARRIAGE_RET    0x0D    #define
KEY_SPACE       0x20       #define
KEY_ESC0x1B
```

MAIL PROGRAM PROGRAM

PORTDETAILS UART0

ARM	DETAILS
P0.0	TXDO
P0.1	RXDO

UART1

ARM	DETAILS
P0.8	TXD1
P0.9	RXD1

LCD PORTDETAILS

ARM	DETAILS
PO.10	RS LCD PIN
P1.11	CE LCD PIN

Post Lab Questions

1. Mention the operations that can be performed on a mailbox.
2. When does the mailbox will get deleted?
3. Illustrate the operation of reading operation from a mailbox.
4. How to configure the mailbox?
5. What is branch prediction?

Result

The C-Language program to create a mailbox and to understand the about the RTOS functions is developed and is verified.

8. Interrupt Performance Characteristics ofARM andFPGA

Aim

To study about the Interrupt performance characteristics between ARM and FPGA.

Pre Lab Questions

1. Define interrupts.
2. What is FPGA?
3. Difference between ARM and FPGA.
4. What are PROS and CONS for ARM?
5. What is interrupt pipelining?

Apparatus & Software Required

1. LPC2148 Development board.
2. KeilµVision 5software.
3. Flash Magic.
4. USB cable.
5. Xilinx FPGA Spartan6
6. Xilinx ISE Design suite
7. JTAG Cable
8. FRC Cable

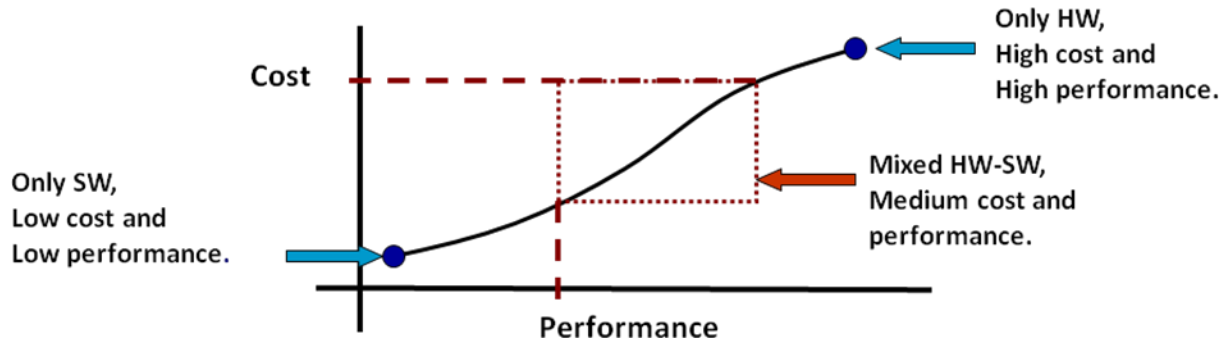
Theory

UART implementation FPGA & ARM7

Why we are doing this experiment?

An embedded system typically consists of both hardware and software. During the design phase of an embedded system the system design Engineer has to choose the components of software and hardware. In a system, implementing a specified logic or algorithm can be done exclusively using software alone or hardware alone.

Implementing a logic or algorithm using only with software involves low cost but delivers only low performance. Implementing the same logic or algorithm only with hardware involves high performance but with low cost.



Hence, from the above graphical analysis it is clear that while designing an embedded system the design engineer must choose a heterogeneous methodology which involves both hardware and software. In this heterogeneous method the engineer has to decide which part of the logic to be implemented in software and which other part to be implemented in software based on performance. So from this experiment a student can learn how to design an embedded system based on performance characteristics.

What is hardware software partitioning on performancecharacteristics?

Choosing the implementation method based on cost is called hardware software portioning on cost characteristics. The design engineer must also consider the performance characteristics like speed, power consumption and reliability while choosing the implementation methodology. Hence deciding which part of our algorithm has to be implemented in software and which other part of the algorithm has to be implemented as hardware based on performance characteristics is called Hardware software partitioning on performance characteristics

Objective of the Experiment:

The aim of this experiment is to implement a UART serial communication algorithm as an embedded system by which to learn hardware-software partitioning based on performance characteristics.

Design of the system:

It is assumed that the reader knows about

i. UART communication protocol
ii. How microprocessor/microcontroller works
iii. What is FPGA and its use
iv. What is ASIC

Software UART	Hardware UART
Advantages: i. Simple to implement ii. Can be easily modified when need arises	Advantages: i. Speed depends only on the FPGA's clocking speed ii. The same design can be manufactured several copies. Hence it is reliable
Disadvantages: i. Complexity of writing code increases when the design involves the operating system. ii. Speed depends on the microprocessor's execution speed and other application code that club with this UART algorithm iii. Since the software code depends on processor architecture, porting the same code in other type of processor needs modifying code again and hence itis not reliable	Disadvantages: i. Designing and writing code in HDL needs good understanding of digital circuit design and a HDL language. Hence complex to implement. ii. The hardware cannot be modified when there is need to upgrade the implemented protocol if it is implemented as ASIC.

Hence, in this experiment we are going to follow the heterogeneous approach to implement the UART algorithm. So it involves both software part and as well as hardware part.

Every protocol has two basic techniques,

i. Data driving logic and

ii. Data packing/ unpackinglogic

Data driving logic:

This involves feeding and obtaining the actual data to be packed to the data packing/ unpacking logic.

Data packing/unpacking logic:

This involves packing the data or unpacking the data and error checking as per the protocol specification.

In this experiment, to design UART communication

a) Data driving logic is implemented in Software. There a son for choosing this logic to be implemented in software is

i) The data is user dependent

ii) Initiating the data transfer is also user dependent

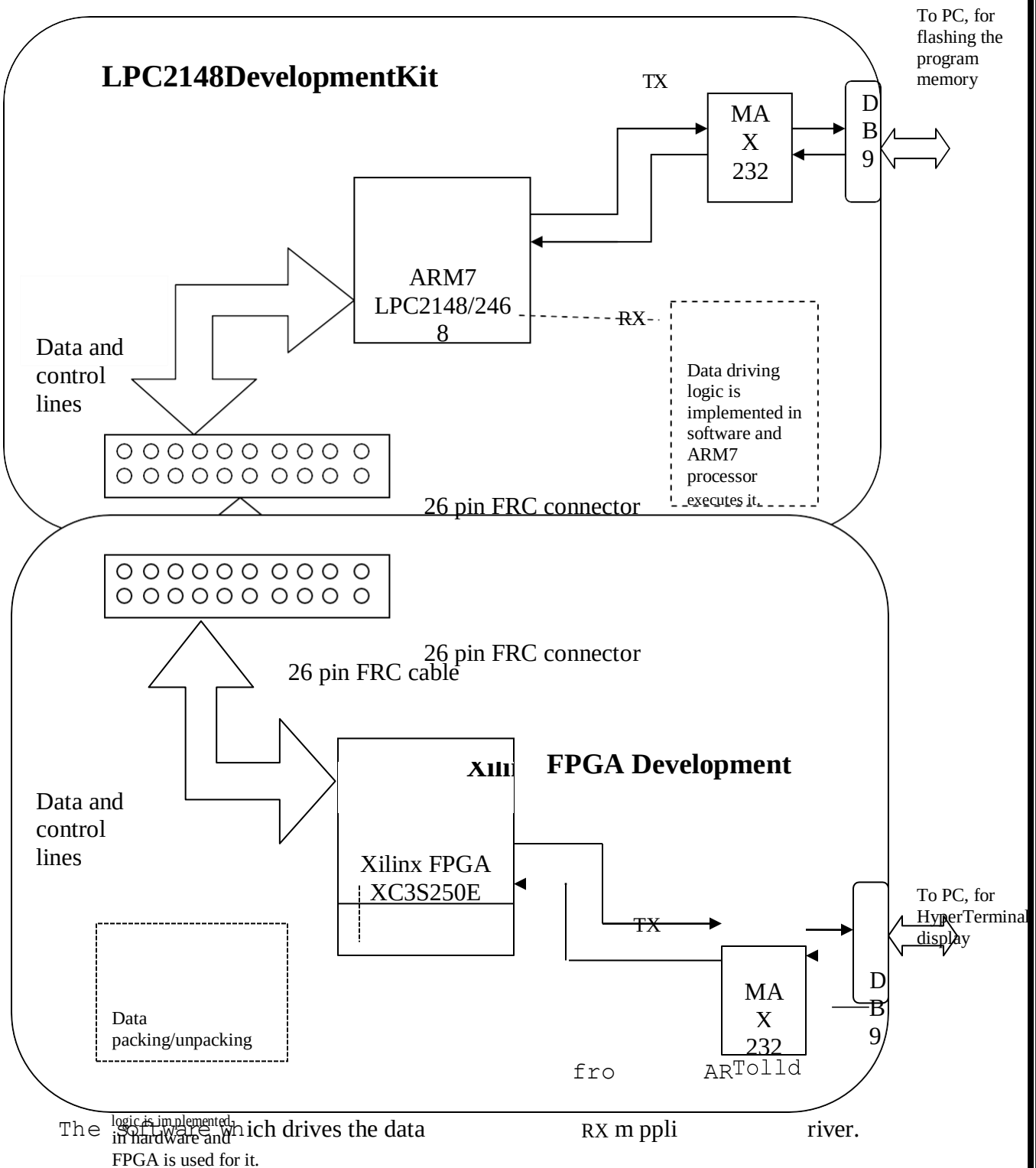
iii) Software design can easily be changed based on the user requirement

b) Data packing/unpacking logic is implemented inhardware. The reason for choosing this logic to be implemented in hardware is

i. Since the protocol specification is fixed one and the packing/ unpacking logic involves designing algorithm for the protocolalone

ii. Data packing/ unpacking speed is high

Pictorial representation:



The software which drives the data RX m ppli river.
fro ARTolld

Software Part:

1. UART driver i.e. data driving logic is software part in this experiment.
2. This software is written using Embedded Clanguage
3. Keil uVision4 IDE and keil arm compiler is used for compiling the Ccode.
4. Flash Magic tool has been used to download the program in to ARM7 microcontroller.

Hardware part:

1. UART protocol i.e. packing/unpacking logic is hardware part in this experiment.
2. This hardware is designed using Verilog HDLprogram
3. The Model sim IDE is used for simulation to test the functionality of the hardware designed using Verilog HDL
4. Xilinx ISE is used to synthesize the design and to download the synthesized RTL file in to the Xilinx FPGAXC3S250E.

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

/***/

UART implementation FPGA & ARM7

ARM UART

/***/

```
#include <LPC214x.h>

void init_ios(void)
{
   /*
            P1.16toP1.23          --IN        --      rx data
             P1.24toP1.31          --OUT      --      tx data
             P0.15                 --OUT      --      txcmd
             P0.16                 --IN       --      txdone
             P0.17                 --IN       --      rx_rdy
   */
   IODIR1=0xff000000;          // tx data made as output
   IODIR0=0x00008000;          //      tx cmd made as output
   IOCLR0=0xff008000;          // Default both the outputs toZero
}


void tx_char(unsigned char data)
{
   char tx_done;
   unsigned int i;
   tx_done = ((IOPIN0>>16) & 0x00000001);
              if(tx_done)
                {
                  IOPIN1 &= ~0xff000000; // clr data port
                  IOPIN1|=data<<24;                    // place data on port
                       for(i=0;i<1000;i++);                  //wait
                       IOSET0=1<<15;                                 // enable tx command
                       for(i=0;i<9000;i++);                  //wait
                       IOCLR0=1<<15;                                 // Disable tx command
                       while(!tx_done)
                            tx_done = ((IOPIN0>>16) & 0x00000001);
                }

}
void tx_string(unsigned char* stringg)
{
   while(*stringg != '\0')
     {
        tx_char(*stringg);string
             g++;
     }
   tx_char(0x0a);tx_c
   har(0x0d);
}


int main (void)
  {
   char i;
   init_ios();
   for(i=0;i<100;i++);
```

```
while (1)
        {
          tx_string("SM MICRRO SYSTEM");
        }
    }

/*************************************************************/
                        FPGA UART
/*************************************************************/
```

S/GName		Portinarm	Data Direction inFPGA
tx_cmd	--	P0.15	IN
tx_data[0]	--	P1.24	IN
tx_data[1]	--	P1.25	IN
tx_data[2]	--	P1.26	IN
tx_data[3]	--	P1.27	IN
tx_data[4]	--	P1.28	IN
tx_data[5]	--	P1.29	IN
tx_data[6]	--	P1.30	IN
tx_data[7]	--	P1.31	IN
tx_done	--	P0.16	OUT
rx_data[0]	--	P1.16	OUT
rx_data[1]	--	P1.17	OUT
rx_data[2]	--	P1.18	OUT
rx_data[3]	--	P1.19	OUT
rx_data[4]	--	P1.20	OUT
rx_data[5]	--	P1.21	OUT
rx_data[6]	--	P1.22	OUT
rx_data[7]	--	P1.23	OUT
rx_rdy	--	P0.17	OUT

Connector Details

P0.18	- P113	P1.24-	P116
Po.17	- P59	P1.25-	P66
P0.16	- P112	P1.26-	P91
P0.15	- P92	P1.27-	P93
P0.19	- P94	P1.28-	P98
P0.20	- P97	P1.29-	P106
P0.21	- P105	P1.30-	P103
P0.23	- P104	P1.31-	P96
+5v	-	GND -	
P1.16	- P135	P1.23-	P134
P1.17	- P132	P1.22-	P131
P1.18	- P130	P1.21-	P117
P1.19	- P139	P1.20-	P126

```
*/

module uart_top( clk,
                 rst,tx
                 ,
```

```
                    rx,
                    lcd_rs,
                    lcd_en,
                    //lcd_rw,lcd
                    _dat,
                                        tx_cmd_i,
                                        tx_data_i,tx_
                                        done_o,rx_da
                                        ta_o,rx_rdy_
                                        o
                );


/***********     I/O       Declecration**********/
    input  clk;
    input           rst;
    input           rx;
    output          tx;
    input           tx_cmd_i;
    input     [7:0]tx_data_i;
    output          lcd_rs;
    output          lcd_en;
    //output            lcd_rw;
    output [3:0]lcd_dat; output
                    tx_done_o;
    output [7:0]rx_data_o; output
                    rx_rdy_o;

    /*
        Local Wire and reg decleration for I/Os
    */

    wire u_clk; wire
    tx_done;
    reg [7:0] dat = 8'h31;
    //reg tx_cm;
    //reg [15:0]c1;

    wire [3:0]lcddata_in; wire
    rx_rdy;

    baud        b1(
                    .sys_clk(clk),
                        .sys_rst_l(rst),
                        .baud_clk(u_clk)
                );

    u_xmitt1(       .sys_clk(u_clk),
                        .sys_rst_l(rst),
                        .uart_xmitH(tx),
                        .xmitH(tx_cmd_i),
                        .xmit_dataH(tx_data_i),
                        .xmit_doneH(tx_done_o)
            );

    u_recr1(        .sys_rst_l(rst),
                        .sys_clk(u_clk),
                        .uart_dataH(rx),
                        .rec_dataH(lcddata_in),
                        .rec_readyH( rx_rdy)
            );

    lcd_dis lcd1( .clk(clk),
                    .rs(lcd_rs),
```

```
                    .en(lcd_en),
                    //.rw(lcd_rw),
                    .data(lcd_dat),
                    .wr_sig(rx_rdy),
                    .data_in(lcddata_in)
                  );
/*
always @(posedge u_clk or negedge rst) begin
                    if(~rst)
                                begin
                                    c1 <= 16'd0;
                                    end
                            else

                                begin
                                            c1 <= c1 + 16'd1; if(c1 ==16'd1000)
                                                        tx_cm <= 1'b1; if(c1
                                                ==16'd1010)
                                                  begin
                                                  tx_cm <= 1'b0; end



      end                                   end
*/



assign rx_data_o = lcddata_in; assignrx_rdy_o
                            =rx_rdy;

endmodule


/**********************************************************************/
                                        FPGA UCF FILE
/**********************************************************************/

#PACE: Start of Constraints generated by PACE #PACE:

Start of PACE I/O Pin Assignments
NET"clk"      LOC="p80"       ;
NET"lcd_dat[0]"  LOC="p2"  ;
NET"lcd_dat[1]"  LOC="p3"  ;
NET"lcd_dat[2]"  LOC="p4"  ;
NET"lcd_dat[3]"  LOC="p5"  ;
NET"lcd_en" LOC="p16"      ;
NET"lcd_rs" LOC="p15"      ; NET"rst"
                LOC="p6"       ;
NET"rx"      LOC="p120"       ;
NET "rx_data_o[0]" LOC    = "p168" ; NET
"rx_data_o[1]"    LOC   =   "p171"   ;   NET
"rx_data_o[2]"    LOC   =   "p172"   ;   NET
"rx_data_o[3]"    LOC   =   "p177"   ;   NET
"rx_data_o[4]"    LOC   =   "p178"   ;   NET
"rx_data_o[5]"    LOC   =   "p179"   ;   NET
"rx_data_o[6]"    LOC   =   "p180"   ;   NET
"rx_data_o[7]" LOC = "p181";
```

```
NET"rx_rdy_o"        LOC="p200"        ;
NET"tx"       LOC="p119"        ;
NET "tx_cmd_i" LOC = "p197";
NET "tx_data_i[0]" LOC = "p185" ; NET
"tx_data_i[1]"    LOC    =    "p186"    ;   NET
"tx_data_i[2]"    LOC    =    "p187"    ;   NET
"tx_data_i[3]"    LOC    =    "p189"    ;   NET
"tx_data_i[4]"    LOC    =    "p190"    ;   NET
"tx_data_i[5]"    LOC    =    "p192"    ;   NET
"tx_data_i[6]"    LOC    =    "p193"    ;   NET
"tx_data_i[7]"    LOC    =    "p196"    ;   NET
"tx_done_o" LOC = "p199";

#PACE: Start of PACE Area Constraints #PACE: Start of

PACE Prohibit Constraints #PACE: End of Constraints

generated by PACE
```

Post Lab Questions

1. What are basically ARM processors?
2. Compare RISC and CISC.
3. How to interface interrupts?
4. Explain the importance of using LPC2148 Development board.
5. List out the applications of ARM.

Result

The C-Language program for Interrupt performance characteristics between ARM and FPGA and its characteristics was studied.

9. Flashing of LEDS

Aim

To develop a 'C' program to make the LED blink (including delay routine). Upon change in the delay program the speed should vary.

Pre Lab Questions

1. What is seven segment displays?
2. Where LEDs are used?
3. What are the different configurations of LED?
4. What is the use of flash magic software?
5. Differentiate LED from LCD.

Apparatus & Software Required

1. LPC2148 Development board.
2. KeilµVision 5 software.
3. Flash Magic.
4. USB cable.

Theory

LEDs are based on the semiconductor diode. When the diode is forward biased (switched on), electrons are able to recombine with holes and energy is released in the form of light. This effect is called electroluminescence and the color of the light is determined by the energy gap of the semiconductor.

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

```
/* This is a test program to make the LEDs L2 and L3 Blink in theARMLPC2148 */
                                    Development board itself
/*****************************************************************************/
#include<LPC214x.H>                                 /* LPC214x definitions*/
voidwait(void)                                      /* wait function*/
{
   int     d;

   for (d = 0; d <1000000;d++);                     /* only to delay for LED flashes*/
}

int main (void)
 {

   IODIR0=0x80002000;                               /* P0.13 and P0.31 defined as Outputs*/

   while(1)                                         /* Loop forever*/
    {
          IOCLR0=0x80002000;                                  /*Active Low outputs makes the
LEDsON*/
       wait ();

          IOSET0=0x80002000;                                  /* High outputs makes the LEDs
OFF*
/
          wait();

    }

 }
```

FLASHING LED PROGRAM PORT DETAILS

ARM	DETAILS
P0.13	LED PIN
P0.31	LED PIN

Post Lab Questions

1. What is the function of GPIO?
2. What are the Pins which are used to connect LEDs?
3. How to identify 'Polarity' of LED?
4. What is a use of Jumper?
5. Which port is used in ARM 7 processor kit?

Result

The C-Language program to make the LED blink was developed and output was verified. Upon change in the delay program the speed variation was verified.

10. INTERFACING STEPPER MOTOR ANDTEMPERATURE SENSOR

Aim

To write C Programs for running stepper motor either in clock- wise or counter-clock- wise and the direction of the rotation of the stepper motor depends on the variation in the temperature sensor.

Pre Lab Questions

1. What is LM35?
2. List the devices used to sense temperature.
3. What is the purpose of a thermocouple?
4. What is signal conditioning?
5. What is the output voltage of a thermocouple?

Apparatus & Software Required

1. LPC2148 Development board.
2. KeilµVision 5 software.
3. Flash Magic.
4. USB cable.
5. Stepper Motor.

Theory

Stepper motors, effectively have multiple "toothed" electromagnets arranged around a central metal gear. To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet.

So when the next electromagnet is turned on and the first willturn off, the gear rotates slightly to align with the next one and from there the process is repeated. Each of those slight rotations is called a "step." In that way, the motor can be turned to a précised angle. There are two basic arrangements for the electromagnetic coils: bipolar and unipolar.

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

STEPPER MOTOR PROGRAM

```
/* This is a test program to stepper motor interface in theARMLPC2148                    */
/* developmentboarditself                                                                */
/*****************************************************************************/
#include<LPC214x.H>             /* LPC214x definitions*/

#definestep1            0x00010000      /* P1.16 */
#definestep2            0x00020000      /* P1.17 */


void wait (void)
{                                       /* wait function*/
   int      d;
   for (d = 0; d <10000;d++);                       /* only to delay for LED flashes*/
}

call_stepper_forw()
{
  IOCLR1 = 0X00FF0000;
  IOSET1 = 0X00040000;
//wait();
//wait();
  wait();
  wait();
  IOCLR1 = 0X00FF0000;
  IOSET1 = 0X00060000;
//wait();
//wait();
```

```
  wait();
  wait();
  IOCLR1 = 0X00FF0000;
  IOSET1 = 0X00070000;
//wait();
//wait();
  wait();
  wait();
  IOCLR1 = 0X00FF0000;
  IOSET1 = 0X00050000;
//wait();
//wait();
  wait();
  wait();
}



int main (void)
  {
       IODIR1 |= 0xFFFFFFFF; IOCLR1
            |= 0X00FF0000;
            wait();
    while(1)                        /*LoopForever*/
       {
                call_stepper_forw();
     //                wait();
     //                wait();
                      wait();
                      wait();
                IOCLR1 = 0X00FF0000;
            }
  }
```

STEPPER MOTOR PROGRAM PORT DETAILS

ARM	DETAILS
P1.16	STEP 1
P1.17	STEP 2

TEMPERATURE SENSOR PROGRAM

/***/

MAIN ADC TEST

/***/

```
/* This is a test program to temperature sensor in the ARM LPC2148 developmentboard*/
/*****************************************************************************/
#include<LPC214x.H>                    /* LPC214x definitions*/
#include"ADC_Driver.c"                        /* contains prototypes of driverfunctions*/ #include"lcd.c"
#include <stdio.h>

int main (void)
  {
    unsigned int adc_val;
    unsigned int temp;
    unsigned char buf[4] ={0,0,0,0}; ADCInit();
    lcdinit();
         //wait();
              clrscr(10);
         printstr("ADC Test",0,0); wait();
    while(1)                                /* Loop forever*/
      {
              adc_val = ADC_ReadChannel();
              temp = (unsigned int)((3*adc_val*100)/1024);
              sprintf(buf,"%d",temp);
              printstr(buf,0,1);


      }
  }

/*****************************************************************************************/
                                                    LCD.C
/*****************************************************************************/

 #include <LPC214x.h>

#defineRS              0x00000400     /* P0.10 */
#defineCE              0x00001800     /* P1.11 */


void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
void printstr(unsignedchar*,char,char);                    //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);

#define SET1
#define OFF0
```

```
unsigned int thousands,hundreds,tens,ones;

voidwait(void)          {                                   /* wait function */
intd;
   for (d = 0; d <100000;d++);                              /* only to delay for LED flashes*/
}

void lcdinit()
{
      IODIR0 |= 0xFFFFFFFF;
          IOCLR0 |= 0X00000FFF;
          lcdcmd(0x28);lcd
      cmd(0x28);
      lcdcmd(0x0c);
      lcdcmd(0x06);
      lcdcmd(0x01);
      lcdcmd(0x0f);
      wait();
}

void gotoxy(char x, char y)
{
      if(y == 0)
            lcdcmd(0x80+x);
      else
            lcdcmd(0xc0+x);


}


void printstr(unsigned char *str, char x, char y)
{
      char i; gotoxy(x,y);
      wait();//(500);
      for(i=0;str[i]!='\0';i++)lcddat(str[i
            ]);

}
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
            LCDDAT = (cmd&0xf0);              //higher nibble
            IOSET0 =LCDDAT;
        IOCLR0 = RS;
            IOSET0 = CE;
        wait();//(100);                                     //enablelcd
            IOCLR0 = CE;
            IOCLR0 = 0X00000FFF;


        LCDDAT = ((cmd<<0x04)&0xf0);                        //lower nibble
            IOSET0 =LCDDAT;
        IOCLR0 = RS;
            IOSET0 = CE;
        wait();//(100);                                     //enablelcd
            IOCLR0 = CE;
            IOCLR0 = 0X00000FFF;
}


void lcddat(char cmd)
```

```
{
        unsigned charLCDDAT;
                LCDDAT = (cmd&0xf0);                    //higher nibble
                IOSET0 =LCDDAT;
           IOSET0 = RS;
                IOSET0 = CE;
           wait();//(100);                                          //enablelcd
                IOCLR0 = CE;
                IOCLR0 = 0X00000FFF;

           LCDDAT = ((cmd<<0x04)&0xf0);                  //lower nibble
                IOSET0 =LCDDAT;
           IOSET0 = RS;
                IOSET0 = CE;
           wait();//(100);                                          //enablelcd
                IOCLR0 = CE;
                IOCLR0 = 0X00000FFF;
     }

    void clrscr(char ch)
    {
          if(ch==0)
          {
                printstr("                                ",0,0);
                gotoxy(0,0);
          }
          else if(ch ==1)
          {
                printstr("                                ",0,1);
                gotoxy(0,1);
          }
          else
          {
                    lcdcmd(0x01);
                      //delay(100);
          }
    }


    void split_numbers(unsigned int number)
    {
       thousands = (number /1000);
       number %= 1000;
       hundreds = (number / 100);
       number %= 100;
       tens = (number / 10);
       number %= 10;
       ones = number ;
    }

    void Wait_Msg(void)
    {
      lcdcmd(0x01);
      printstr("          Please Wait ", 0,0);
    }
    void Welcome_Msg(void)
    {
      lcdcmd(0x01);
      printstr("            Welcometo          ", 0,0);
      printstr("              SMMICRRO     ", 0,1);
    }
```

ADC_DRIVER.C

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

#include<LPC214x.H>                         /* LPC214x definitions */
Void ADCInit(void)
  {
    PINSEL1|=0x04000000;              /*For Channel AD0.2 is P0.29*/
    IODIR0 |=~(0x04000000);
    AD0CR     |=0x00200204;              /*0x04 selects AD0.2 to mux output, 0x20 makes ADCin operational*/
    AD0GDR;                             /*A read on AD0GDR clears the DONEbit*/
  }
void ADC_StartConversion(void)
  {
     AD0CR |= (1<<24);
  }
void ADC_StopConversion(void)
  {
     AD0CR &= (~(1<<24));
  }

unsigned int ADC_ReadChannel(void)
  {
//       unsigned int i; unsigned long
     ADC_Val, t;
     ADC_StartConversion();
     while((AD0DR2&0x80000000)==0); /*wait until ADC conversion completes*/
     if(AD0STAT & 0x00000400)
                {
                  //printstr("OVR",0,1);return(0);
                }
     t = AD0DR2;
     ADC_Val = ((t>>6) & 0x000003FF);//(AD0DR2 & 0x000003FF); //((AD0CR>>6) & 0x000003FF);
     //ADC_StopConversion();return(
     ADC_Val);
  }.
```

TEMPERATURE SENSOR PROGRAM PORT DETAILS

ARM	DETAILS
P0.29	ADC0.2
PO.10	RS LCD PIN
P1.11	CE LCD PIN

Post Lab Questions

1. Why LM35 is used to Measure Temperature?
2. Compare the difference between LM 34 and LM 35 sensors?
3. What is the operating temperature range in LM35?
4. How many pins are available in LM35?
5. What is the main function of analog pin in LPC 2148?

Result

The C-Language program for running stepper motor either in clock-wise or counter-clock-wise Depending on the temperature is developed in the sensor LM35 and the output is verified in LCD.

11. Implementing zigbee protocol with ARM

Aim

To write C Programs for Zigbee Protocol and verify the communication between Xbee Module Transmitter and Receiver.

Pre Lab Questions

1. What are the applications of zigbee protocol?
2. Why Zigbee based is preferred for wireless communication?
3. What is the function of a scheduler?
4. What is the main function of voltage convertors in UART?
5. List the advantages of using Zigbee protocol.

Apparatus & Software Required

1. LPC2148 Development board.
2. KeilµVision 5software.
3. Flash Magic.
4. USB cable.
5. Zigbee Module Tx and Rx.

Theory

The X Bee/X Bee-PRO ZNet 2.5 (formerly known as Series 2 and Series 2 PRO) RF Modules were directed to operate within the ZigBee protocol. The modules provide reliable delivery of data between remote devices. Zigbee is the communication protocol like wifi and Bluetooth. Xbee is the module using Zigbee protocol

Some of its features are:

- ZigBee is targeted at radio-frequency (RF) applications
- Low data rate, long battery life, and secure networking
- Transmission range is between 10 and 75 meters (33~246 feet)
- The addressing space allows of extreme node density—

up to 18,450,000,000,000,000,000 devices (64 bit IEEE address)

- Using local addressing, simple networks of more than 65,000 nodes can be configured, with reduced address overhead
- The radios use direct-sequence spread spectrum coding, which is managed by the digital stream into the modulator.
- To ensure reliable data transmission

- Binary phase shift keying (BPSK) in the 868/915 MHz

- Offset quadrature phase shift keying (O-QPSK) at 2.4 GHz

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

/***/

ARM TRANSMITTER
PROGRAMLCD.C

/***/

```
#include <LPC214x.h>
#include "lcd.h"
#defineRS            0x00000400     /* P0.10 */
#defineCE            0x00001800     /* P1.11 */

/*void clrscr(char ch); void
lcdinit(void);
voidlcdcmd(char);
voidlcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
voidprintstr(char*,char,char);                   //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);*/

#define SET1
#define OFF0

unsigned int thousands,hundreds,tens,ones; void wait

(void)
{                                   /* wait function*/
   int      d;
   for (d = 0; d <100000;d++);                         /* only to delay for LED flashes*/
}


void lcdinit()
{
      IODIR0 |= 0xFFFFFFFF;
           IOCLR0 |= 0X00000FFF;
           lcdcmd(0x28);lcd
      cmd(0x28);
      lcdcmd(0x0c);
      lcdcmd(0x06);
      lcdcmd(0x01);
      lcdcmd(0x0f);
      wait();
}


void gotoxy(char x, char y)
{
      if(y == 0)
            lcdcmd(0x80+x);
      else
            lcdcmd(0xc0+x);


}


void printstr(char *str, char x, char y)
{
      char i; gotoxy(x,y);
      wait();//(500);
      for(i=0;str[i]!='\0';i++)
```

```
lcddat(str[i]);

}
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
        LCDDAT = (cmd&0xf0);            //higher nibble
        IOSET0 =LCDDAT;
    IOCLR0 = RS;
        IOSET0 = CE;
    wait();                                                         //(100);
                                                                    //enable lcd
        IOCLR0 = CE;
        IOCLR0 = 0X00000FFF;


    LCDDAT = ((cmd<<0x04)&0xf0);                //lower nibble
        IOSET0 =LCDDAT;
    IOCLR0 = RS;
        IOSET0 = CE;
    wait();//(100);                                             //enablelcd
        IOCLR0 = CE;
        IOCLR0 = 0X00000FFF;


}
void lcddat(char cmd)
{
  unsigned charLCDDAT;
        LCDDAT = (cmd&0xf0);            //higher nibble
        IOSET0 =LCDDAT;
    IOSET0 = RS;
        IOSET0 = CE;
    wait();//(100);                                             //enablelcd
        IOCLR0 = CE;
        IOCLR0 = 0X00000FFF;


    LCDDAT = ((cmd<<0x04)&0xf0);                //lower nibble
        IOSET0 =LCDDAT;
    IOSET0 = RS;
        IOSET0 = CE;
    wait();//(100);                                             //enablelcd
        IOCLR0 = CE;
        IOCLR0 = 0X00000FFF;
 }
void clrscr(char ch)
{
    if(ch==0)
    {
        printstr("                          ",0,0);
        gotoxy(0,0);
    }
    else if(ch ==1)
    {
        printstr("                          ",0,1);
        gotoxy(0,1);
    }
    else
    {
            lcdcmd(0x01);
              //delay(100);
    }
```

```
}

    void split_numbers(unsigned int number)
    {
       thousands = (number /1000);
       number %= 1000;
       hundreds = (number / 100);
       number %= 100;
       tens = (number / 10);
       number %= 10;
       ones = number ;
    }

    void Wait_Msg(void)
    {
      lcdcmd(0x01);
      printstr("          Please Wait ", 0,0);
    }
    void Welcome_Msg(void)
    {
      lcdcmd(0x01);
      printstr("            Welcometo          ", 0,0);
      printstr("              SMMICRRO      ", 0,1);
    }
```

/***/

LCD.h

/***/

```
    void clrscr(char ch); void
    lcdinit(void); void
    lcdcmd(char); void
    lcddat(char);
    void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
    voidprintstr(char*,char,char);                    //string,column(x),line(y)
    void wait (void);
    void split_numbers(unsigned int number); void
    Wait_Msg(void);
    void Welcome_Msg(void);
```

/***/

UART_1.C

/***/

```
    #include <LPC214X.H>
    #include "lcd.c"

    #define TEMT0X40

    void uart_1(void);
    voiddelay(void);
    void putcharr (unsignedcharch);                    /* Writes character to Serial Port*/
    void tx_string(charstr);

    int main(void)
    {
                uart_1();
                lcdinit();dela
                y();
                delay();
```

```
        delay();
            delay();
            printstr("SM MICRRO SYSTEM",0,0);
            while(1)
            {
                        tx_string('C');

                        gotoxy(7,1);
                        lcddat('C');delay
                        ();
                        delay();
                        delay();
                        delay();
                        while(1);
            }
}

void uart_1(void)
{
            PINSEL0 = 0x00050000;
            U1LCR = 0x83;
            U1FDR = 0x00000010;
            U1DLL = 98;
            U1LCR = 0x03;
            U1IER = 0x01;
}
void delay(void)
{
   int      d;
   for (d = 0; d <100000;d++);                    /* only to delay for LED flashes*/
}

void tx_string(char str)
{
            putcharr(str);
}
void putcharr (unsignedcharch)                    /* Writes character to SerialPort*/
{
       while (!(U1LSR&TEMT));                                  /* U1LSR --> Statusregister
*/
            U1THR = ch;
}
```

/***/

ARM RECEIVERPROGRAM

/***/

```
#include <LPC214X.H>
#include "lcd.c"
void uart_1(void); void
delay(void);
unsigned chargetcharr(void);                      /* Reads character from SerialPort*/

int main(void)
{
char rx_data;
            uart_1();
            lcdinit();
            printstr("SM MICRRO SYSTEM",0,0);
            while(1)
            {
```

```
                rx_data=getcharr();                    /* Reads character fromSerial
Port*/
                gotoxy(7,1);
                lcddat(rx_data);
        }
}


voiduart_1(void) /* UART Installation*/
{
        PINSEL0 = 0x00050000;
        U1LCR = 0x83;
        U1FDR = 0x00000010;
        U1DLL = 98;
        U1LCR = 0x03;
        U1IER = 0x01;
}
void delay(void)
{
   int     d;
   for (d = 0; d <100000;d++);              /* only to delay for LED flashes*/
}
unsigned chargetcharr(void)                 /* Reads character from SerialPort*/
 {
   while (!(U1LSR & 0x01));
   return (U1RBR);
}
```

Implementing zigbee protocol with ARM PROGRAMS PORTDETAIL

TRANSMITTER PROGRAM

ARM	Details
P0.8	TXD1
P0.9	RXD1
P0.10	RS LCD PIN
P1.11	CE LCD PIN

RECEIVER PROGRAM

ARM	Details
P0.8	TXD1
P0.9	RXD1
P0.10	RS LCD PIN
P1.11	CE LCD PIN

Post Lab Questions

1. How to verify the communication between Transmitter and Receiver?
2. Which module is using Zigbee protocol?
3. How many UART ports available in LPC2148?
4. Write the two modes of communication are used in a ZigBee network.
5. Mention the transmission range for Zigbee protocol.

Result

The C-Language program for Zigbee Protocol is written and the communication between Xbee Module Transmitter and Receiver is verified.

12. SIMULATION USING PROTEUS SOFTWARE –AN INTRODUCTION

Aim

To simulate a multivibrator using Proteus Software and check its functionality by verifying its output with an simulated LED.

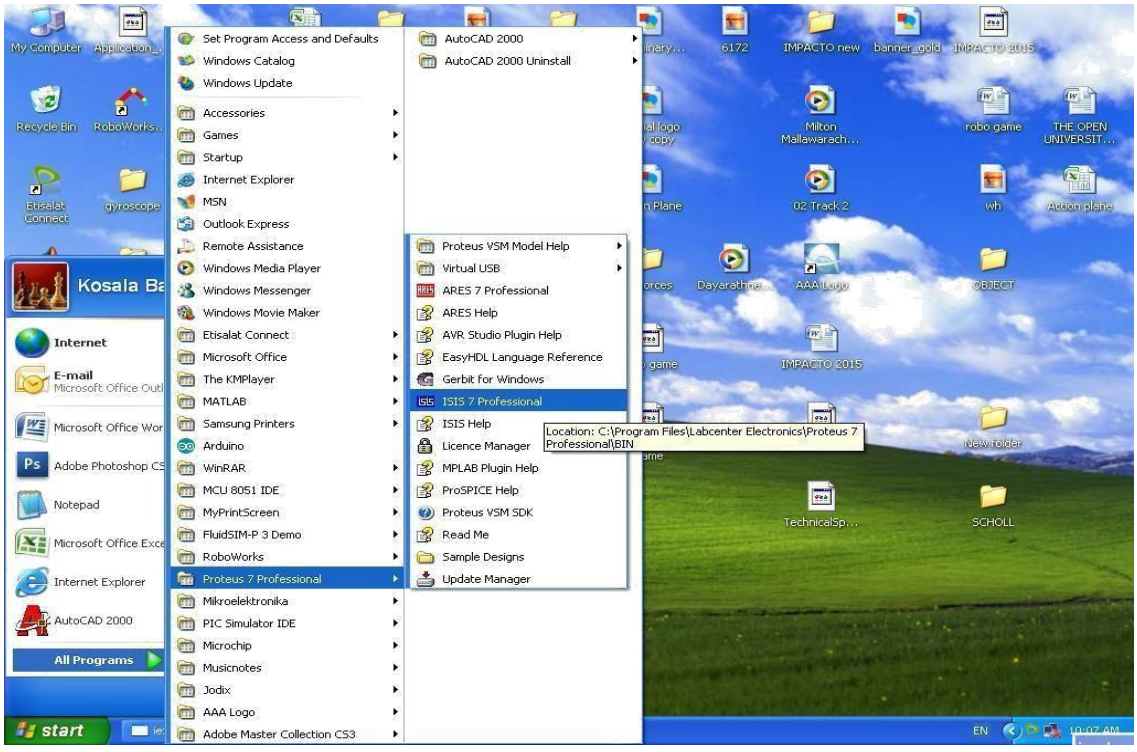
Pre Lab Questions

1. What is the need for a simulationtool?
2. What is Proteus Software?
3. List the feature of Proteus Software?
4. List some of the design tools in proteus software?
5. Give any 3 gadgets available in the proteus software?

Procedure

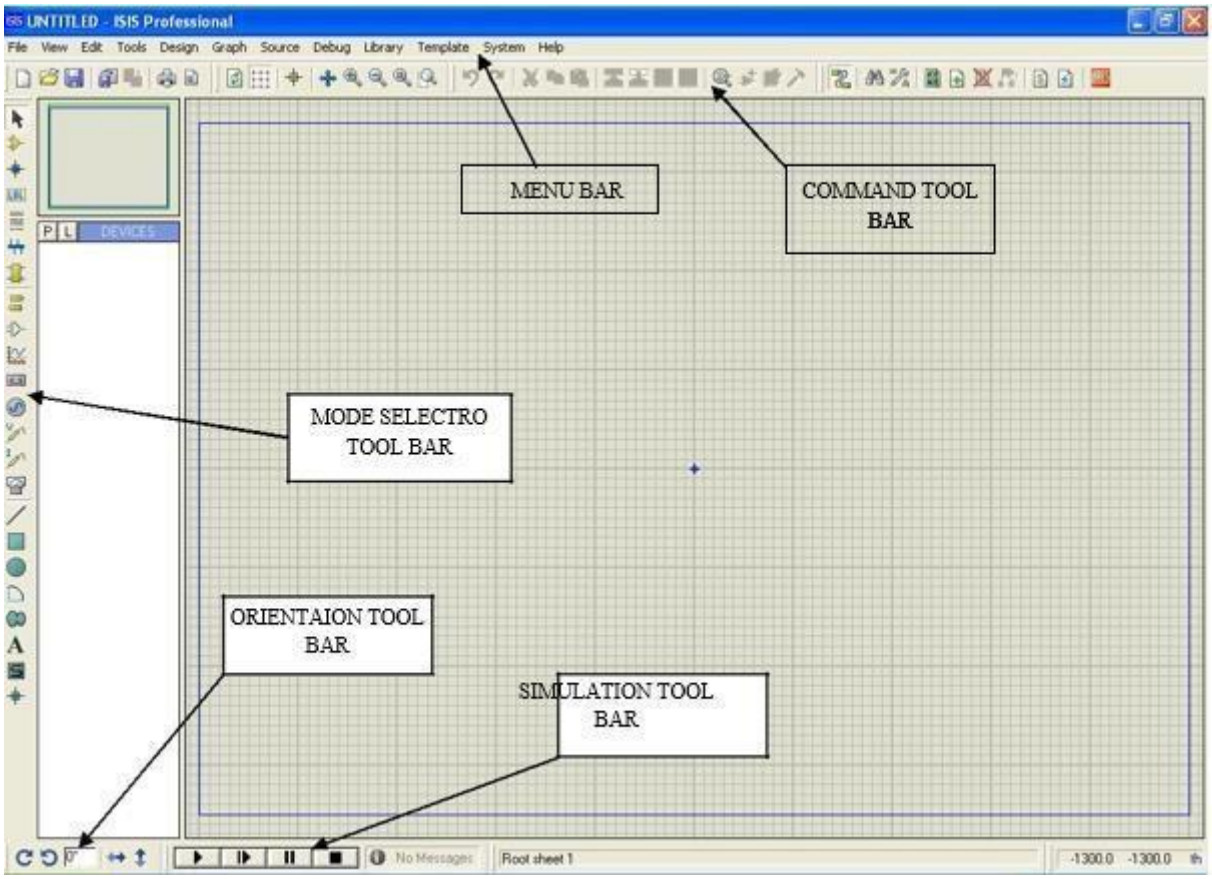
Step1:

Start → All Programs → Proteus xx Professional → ISIS xxProfessional



Note:

- **Identify the components**



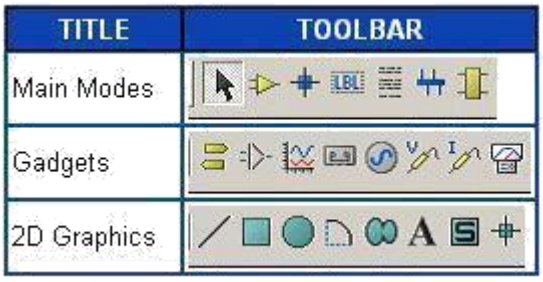
The MenuBar

TheToolbars

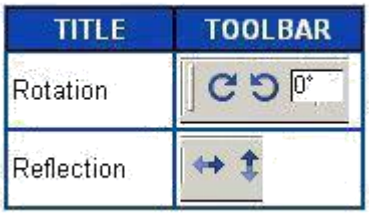
Command Toolbars

TITLE	TOOLBAR
File / Print Commands	
Display Commands	
Editing Commands	
Design Tools	

Mode Selector Toolbar



Orientation Toolbar



Simulation Toolbar

Construct the following multivibrator circuit using analog and digital components and simulate it,

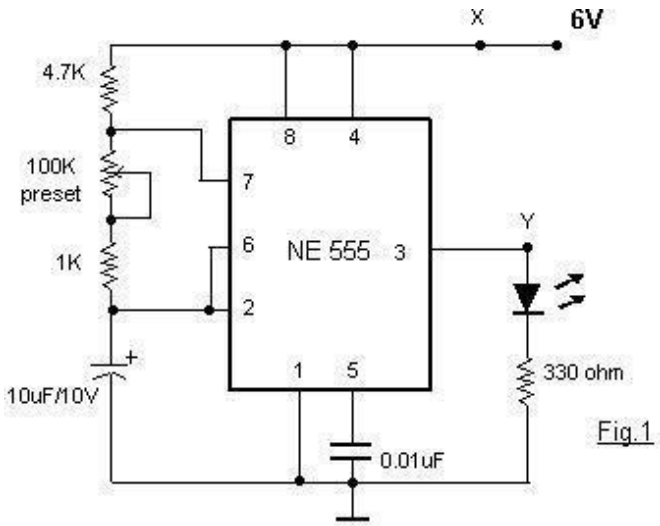


Fig.1

The Components needed are,

Resistors 4.7kΩ ,1kΩ 330Ω , 100k (variable resistor)

Capacitors 10uF ,0.01uF

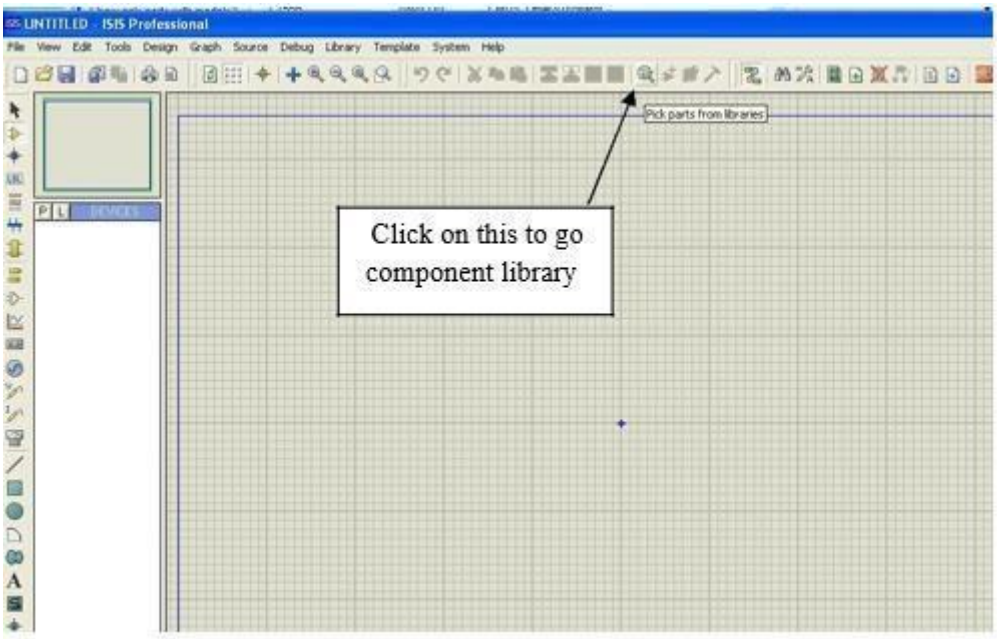
IC NE555

LED

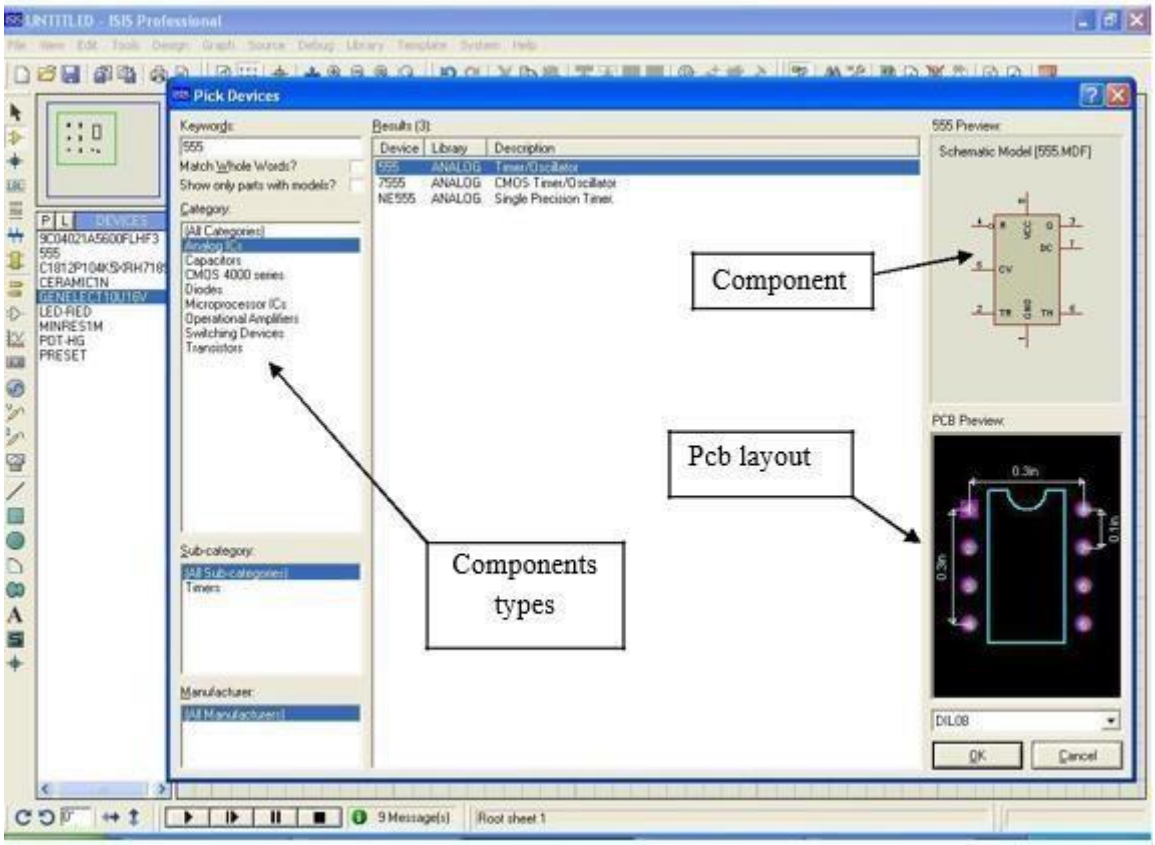
Power supply

Step 2:

Choose required components from the component library in editingcommands.

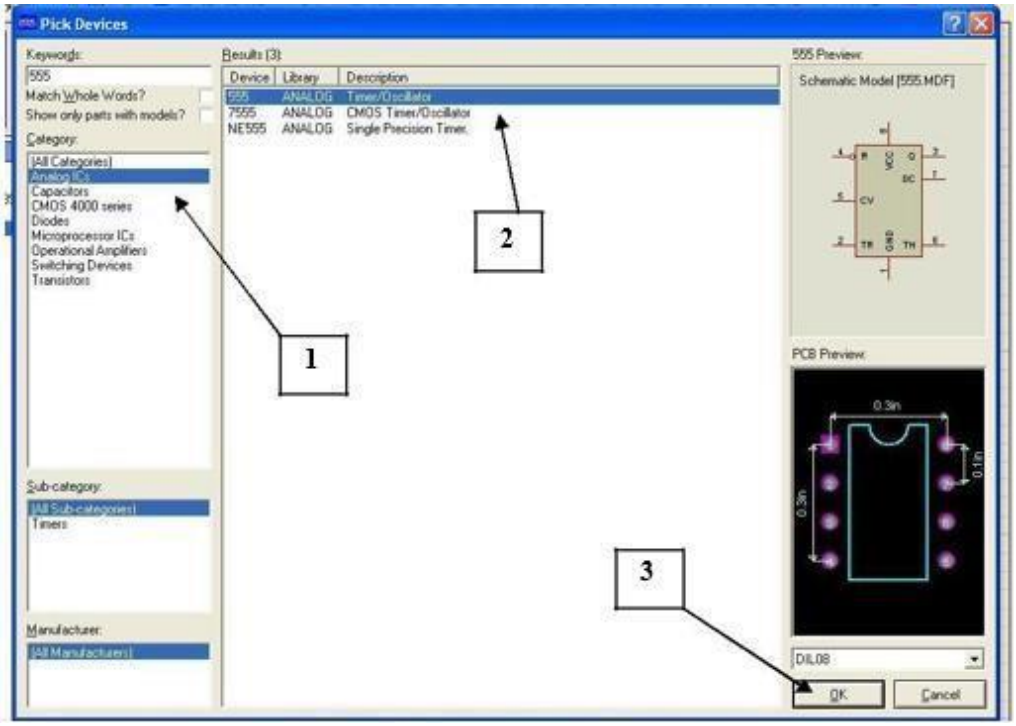


Then the component library window will appear

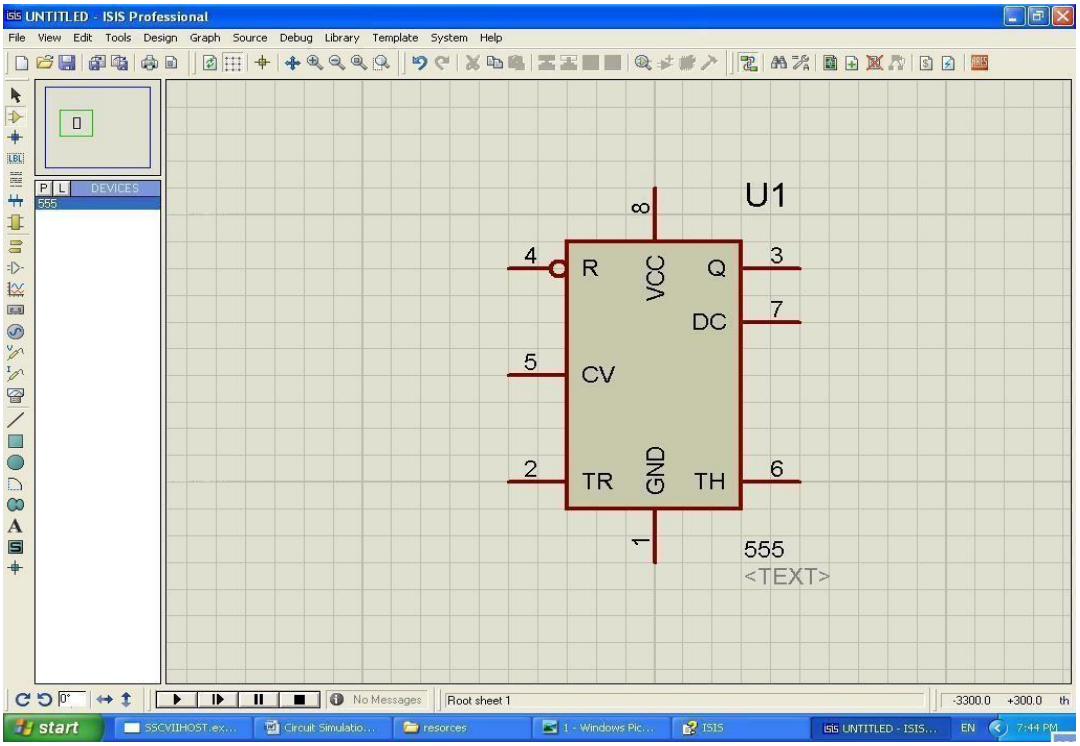


Step 3

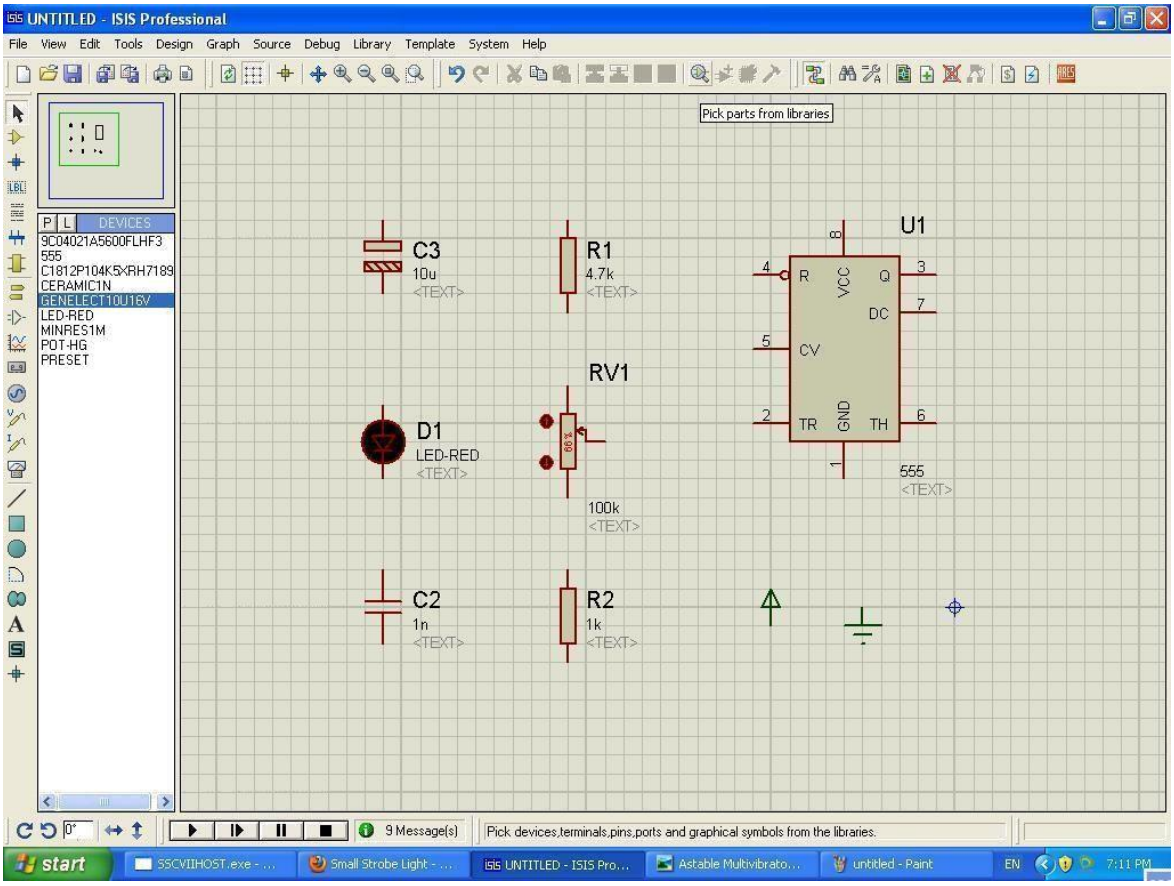
Then choose the all require components and put into the main window.



1. Select “Analog ICs” inCategory
2. Then select “555 ANALOGTIME/OSCILATOR”
3. Next click onOK
4. Finally component put into mainwindow

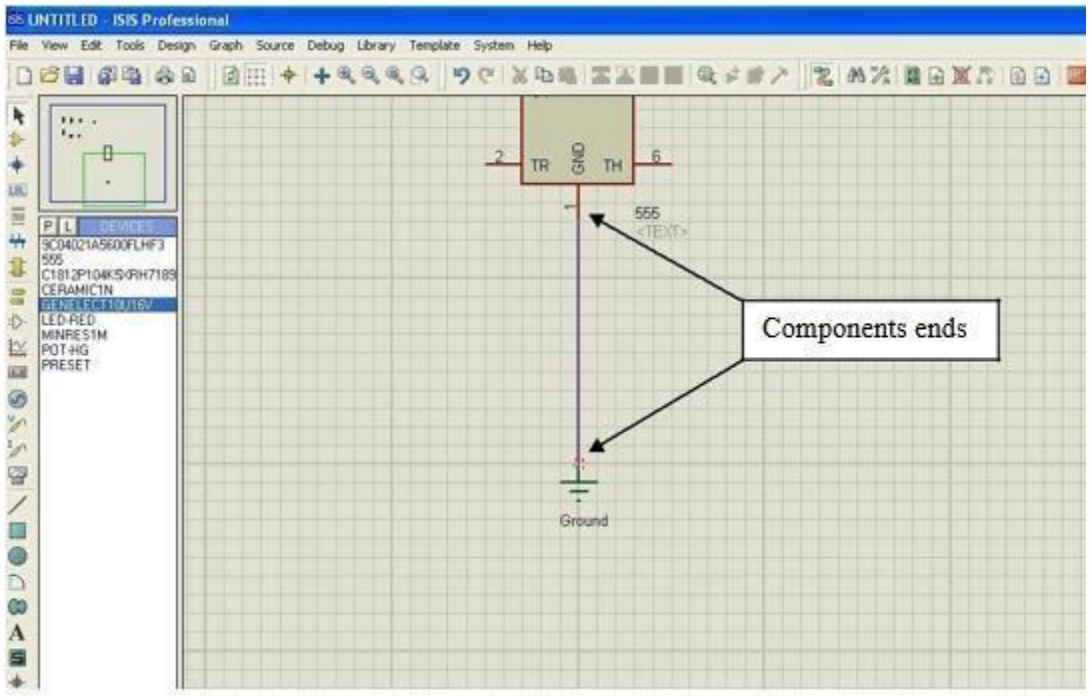


Following the same procedure, place all the required components to main window

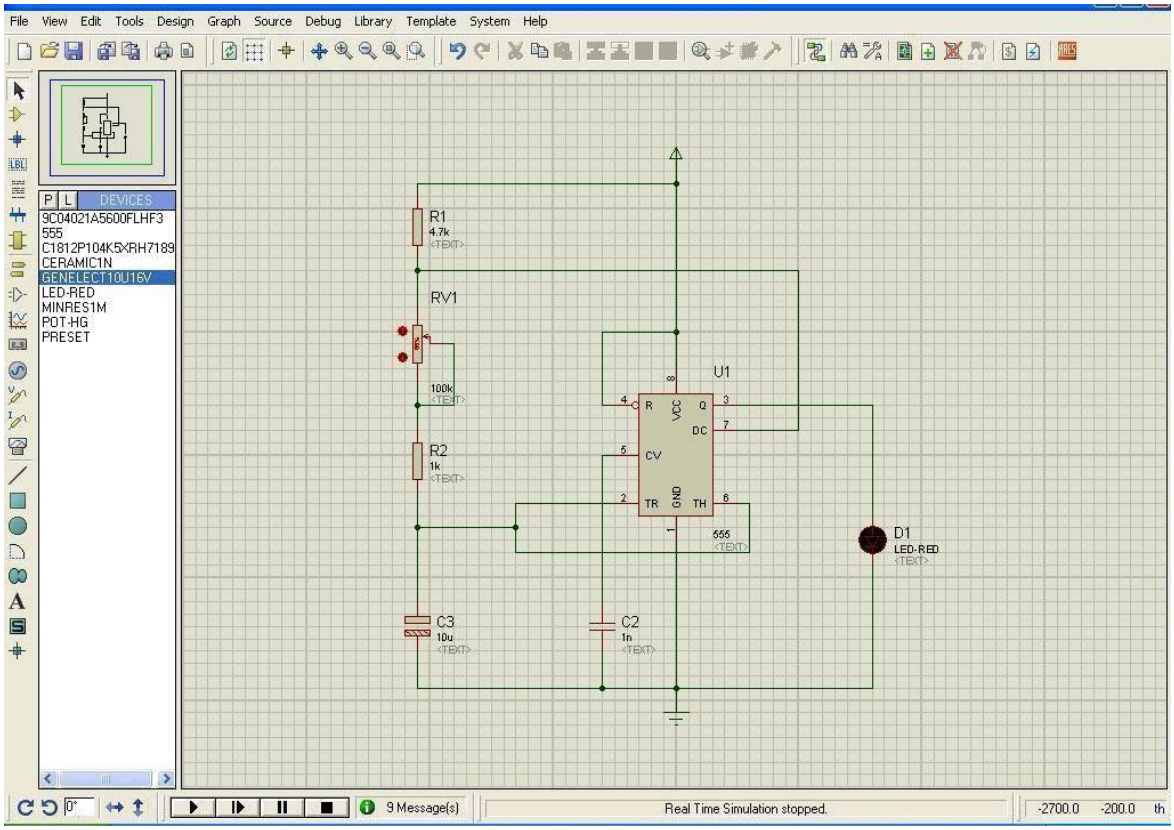


Step 4

Connect all the components as the above circuit diagram by using connecting lines. We can get connecting lines by selectingendsofthecomponentsasshowsinfollowingfigure.

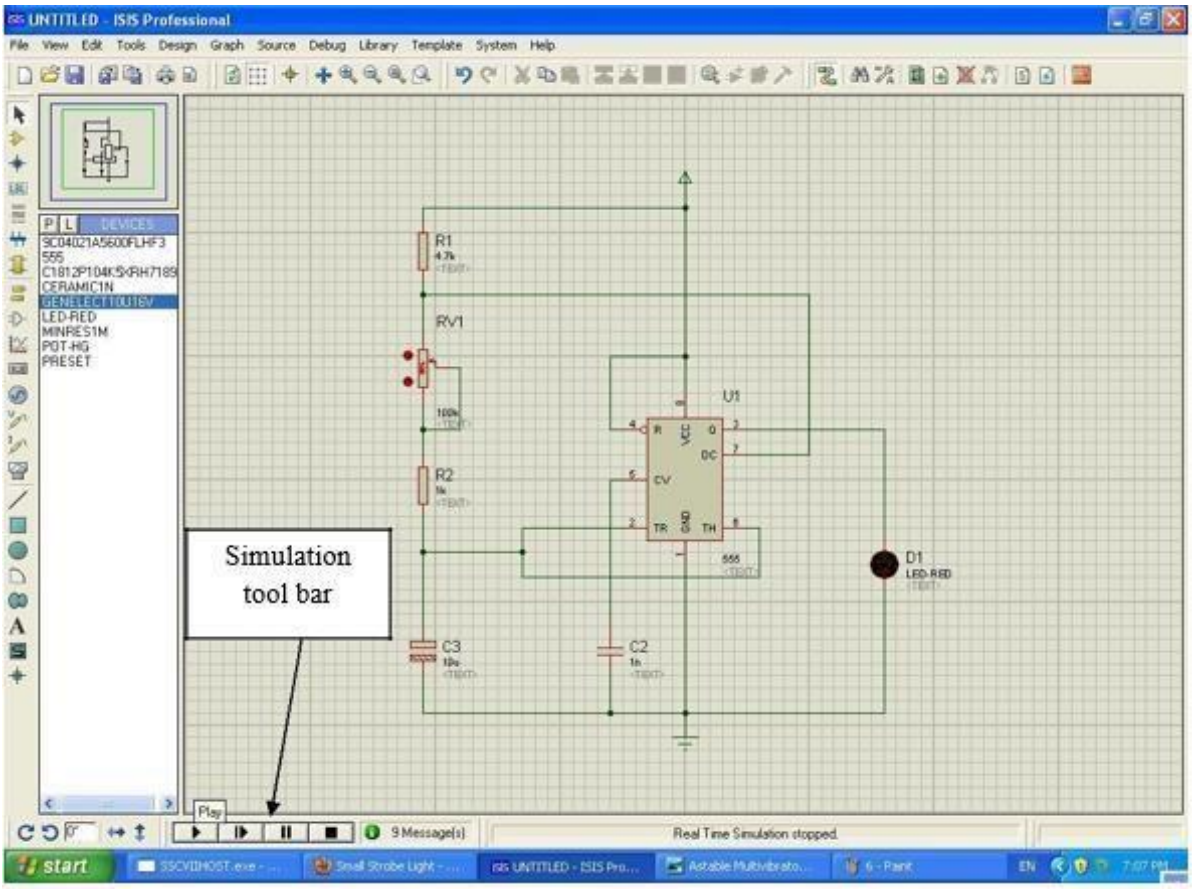


Then complete the circuit using connecting lines as following figure…..



Step 5

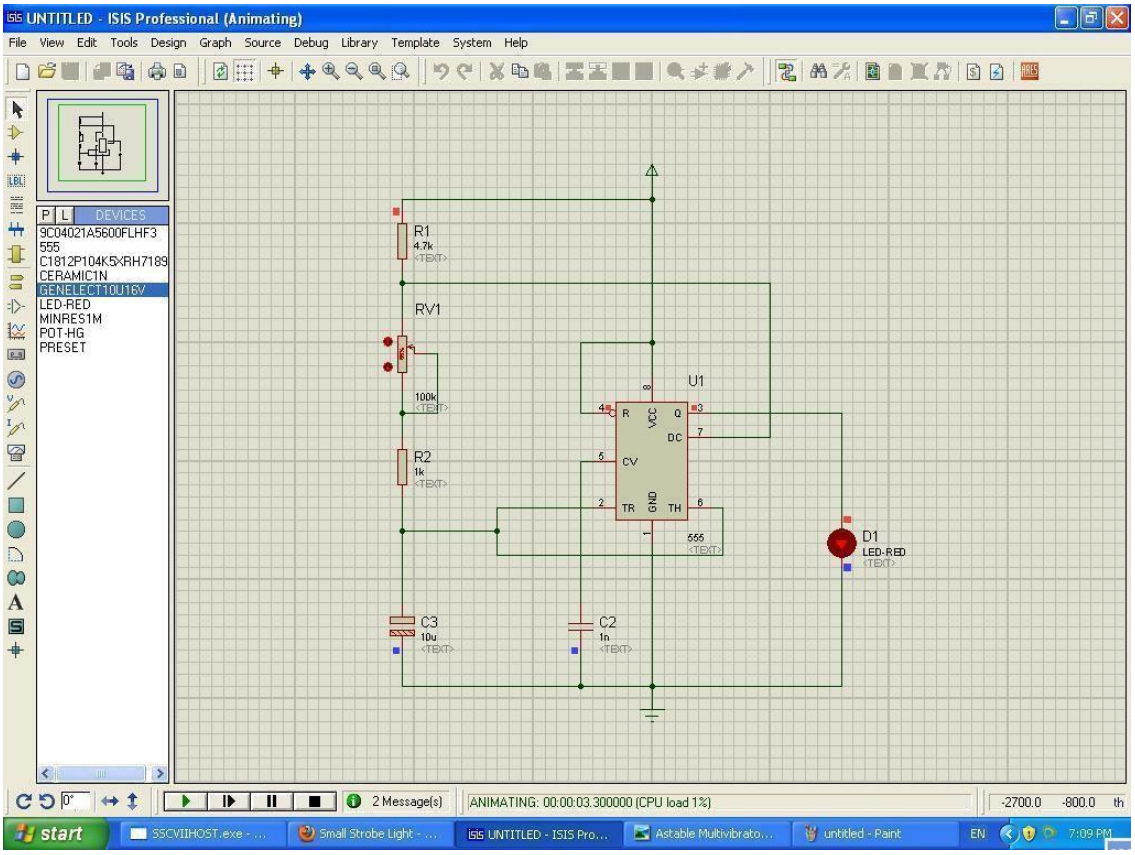
Now make sure all the components correctly connected. Then go the simulation tool bar.



To start the simulation click on play button. 

Then start the simulation if there are any errors the simulation must be fail and give error messages.

Check the output by inspecting the status of the LED.



Post Lab Questions

1. What is mode selector tool bar
2. What is orientation tool bar
3. What is the need for simulation tool bar?
4. What is the frequency of the output obtained in simulation?
5. What is the name of the library under which 555IC is located?

Result

Thus the Multivibrator is simulated using Proteus Software and the output is observed.

13. SIMULATION OF CALCULATOR USING 8051 MICROCONTROLLER IN PROTEUS SOFTWARE

Aim

To simulate a simple calculator using 8051microcontroller in Proteus Software.

Pre Lab Questions

1. List some of the microprocessor supported by proteus software
2. List the required arithmetic operations to be performed by the calculator?
3. Write down the features of 8051 microcontroller
4. What is the difference between microprocessor and microcontroller?
5. What is the size of the data bus in 8051 microcontroller?

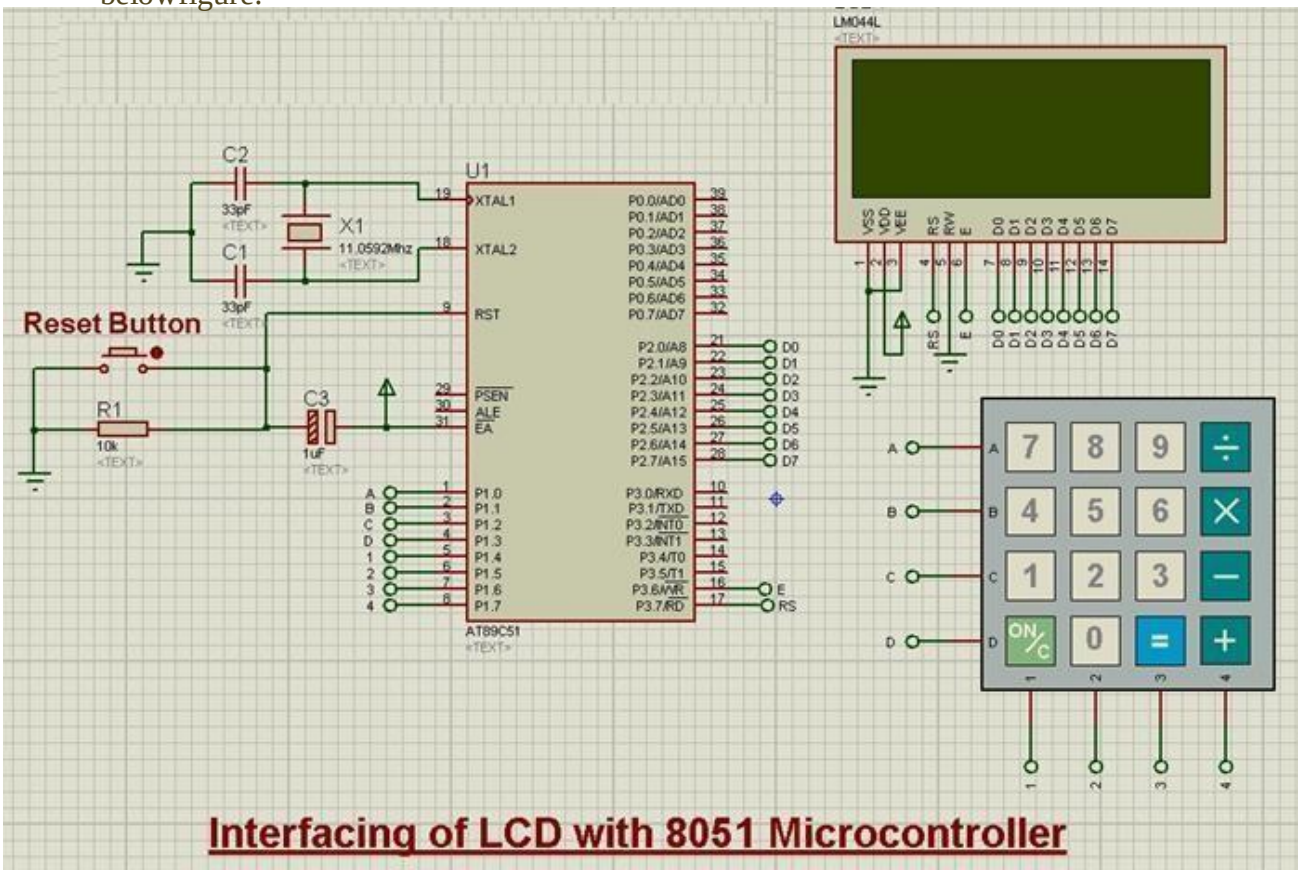
Procedure

The calculator we are going to design is quite basic calculator, it will only perform 4 tasks, which are as follows:

- When you press the **(+)** button then it will add the two digits. For example, you want to add 2 and 3 then you need to press **2 + 2 =** these four buttons in sequence and when you press the = button it will automatically will give you the sum.
- When you press **(-)** button it will subtract the two digits like **3 – 2 =** and it will give you the result.
- When you press **(x)** button it will multiply the two digits.
- When you press the **(/)** button it will simply divide the two digits.
- Whenever you press the **(=)** button, it will give you the output
- depending on the function you used before and if you press (=) in the start then it will give "Wrong Input".
- Finally,there's**(ON/C)**button on the Calculator, when you press this it will simply reset the code and will clear theLCD.
- So,that'show this calculator is goanna work. More over, it will always reset when you try to calculate new value.
- As its a simple calculator, so its only limited to 1 digit, means it will only apply the operation on single digit input like 2+3 but it won't work on more than 1 digit like 12 +13.
- After that, we will do the coding part for calculator with 8051 Microcontroller.
- So, now let's get started with Proteus Simulation.

ProteusSimulation

- TheProteusSimulationofthisCalculatorwith8051 Microcontroller and is shown in belowfigure:



ProgrammingCode

```
1       while(1)
2           {
3              //getnumb1
4              key =get_key();
5              writecmd(0x01);                  //cleardisplay
6           writedata(key);                  //Echo the key pressed toLCD
7           num1=get_num(key);               //Get int number from char value, it checksfor
8
9       wrong input as well
10          if(num1!=Error)                  //if correct input then proceed,num1==Error
11      means wrong input
12          {
13                  //getfunction
14                  key =get_key();
15                  writedata(key);                          //Echo the key pressed toLCD
16                  func=get_func(key);                      //it checks for wrong func 17
18                  if(func!='e')                            //if correct input thenproceed,
19      func=='e' means wrong input 20
        {
21                          //getnumb2
22                          key =get_key();
23                          writedata(key);                      //Echo the key pressed toLCD
24                          num2=get_num(key);                   //Get int number fromchar
25      value, it checks for wrong input as well 26
27                          if(num2!=Error)                      //if correct inputthen
28      proceed, num2==Error means wronginput
```

```
                                {
29                                      //get equal sign key =
30                                      get_key();
31                                      writedata(key);                 //Echo the key pressed to LCD if(key=='=')
32
33                                                                      //if = is pressedthen
34
35      proceed                         {
36                                              switch(func)                    //switch onfunction
37                                              {
38                                              case '+': disp_num(num1+num2); break; case '-':
39                                              disp_num(num1-num2);    break;  case    'x':
40                                              disp_num(num1*num2);    break;  case    '/':
41                                              disp_num(num1/num2); break;
42                                              }
43                                      }
44                                      else                                    //key other then=
45
46      here means error wronginput 47  {
48
                                                disp_num(num1+num2);    break; case '-':
                                                disp_num(num1-num2);    break;  case    'x':
                                                disp_num(num1*num2);    break;  case    '/':
                                                disp_num(num1/num2); break;
                                                }
                                        }
                                        else                                    //key other then=
                                        {
                                                if(key=='C')                    //if clear screenis
49      pressed then clear screen andreset
                                                        writecmd(0x01);         //Clear Screen DispError(0);
                                                  else
                                                                                //Displaywrong
        input error

                                        }
                                }
                        }
                }
        }
```

- As you can see in the above function, first check for thefirst keypress.
- When you pressed the first key on keypad then I get this key and converter it tointeger.
- AfterthatIwaitedforthenextkeywhichmustbesomeoperation keylike+– Xor/otherwiseitwillgeneratetheerrormessage.
- After that code is waiting for the third key which should be somenumericaldigitandthenItconvertsittointegeragainand if you entered some invalid key then it will generate theerror.
- Finally waiting for the = sign. When you press the = sign it will automatically perform the required operation which Iplaced in the switch caseloop.
- It will calculate the value and then print out the result and onnextkeypressitwillfirstclearthescreenandthengetthe value and willcontinue.
- Below is the detailed code for the project withcomments

```
1       #include<reg51.h>
2       #include<string.h>3
4       //DefineMacros
5       #defineError        13      // Any value other than 0 to 9 is good here 6
7       //Functiondeclarations
8       voidcct_init(void);
9       voiddelay(int);
10      voidlcdinit(void);
```

```
11      voidwritecmd(int);
12      voidwritedata(char);
13      voidwriteline(char[]);
14      voidReturnHome(void);
15      charREAD_SWITCHES(void);
16      charget_key(void);
17      intget_num(char);
18      charget_func(char);
19      voidDispError(int);
20      voiddisp_num(int);
21      void WebsiteLogo();
22      22
23      /
24      //Pindescription
25      /*
26      P2 is databus
27      P3.7 isRS
28      P3.6 isE
29      P1.0 to P1.3 are keypad rowoutputs
30      P1.4 to P1.7 are keypad column inputs 31
        */
32      //*********************
33      // DefinePins
34      //*********************
35         sbit RowA = P1^0;        //RowA
36         sbit RowB = P1^1;        //RowB
37         sbit RowC = P1^2;        //RowC
38         sbit RowD = P1^3;        //RowD
39
40         sbit C1   = P1^4;        //Column1
41         sbit C2   = P1^5;        //Column2
42         sbit C3   = P1^6;        //Column3
43         sbit C4   = P1^7;        //Column4
44
45         sbit E    = P3^6;        //E pin for LCD
46         sbit RS   = P3^7;        //RS pin for LCD
47
48      //*************************************************************
49      // Mainprogram
50      //
51      intmain(void)
52      {
53          charkey;                          //key char for keeping record ofpressed
54      key
55          int num1=0;                       //Firstnumber
56          char func='+';                    //Function to be performed amongtwo
57      numbers
58          int num2=0;                       //Second number 59
60          cct_init();                       //Make input and output pins asrequired
61          lcdinit();                        //InitilizeLCD
62          WebsiteLogo();
63          while(1)
64          {
65             WebsiteLogo();
66             //getnumb1
67             key =get_key();
68             writecmd(0x01);                //cleardisplay
69          WebsiteLogo();
70          writedata(key);                   //Echo the key pressed toLCD
71          num1=get_num(key);                //Get int number from char value, itchecks
72      for wrong input as well 73
74          if(num1!=Error)                   //if correct input then proceed,num1==Error
75      means wrong input 76
            {
77                  //getfunction
78                  key =get_key();
79                  writedata(key);                        //Echo the key pressed toLCD
80                  func=get_func(key);                    //it checks for wrong func 81
```

```
82                  if(func!='e')                       //if correct input thenproceed,
83          func=='e' means wrong input
84                  {
85                          //get numb2
86                          key = get_key();
87                          writedata(key);             //Echo the key pressed toLCD
88                          num2=get_num(key);          //Get int number fromchar
89          value, it checks for wrong input as well
90
91                          if(num2!=Error)             //if correct inputthen
92          proceed, num2==Error means wrong input
93                          {
94                                  //get equal sign
95                                  key = get_key();
96                                  writedata(key);         //Echo the key pressedto
97          LCD
98
99                                  if(key=='=')            //if = is pressedthen
100         proceed
101                                 {
102                                         switch(func)        //switch onfunction
103                                         {
104                                         case '+': disp_num(num1+num2); break;
105                                         case '-': disp_num(num1-num2); break;
106                                         case 'x': disp_num(num1*num2); break;
107                                         case '/': disp_num(num1/num2); break;
108                                         }
109                                 }
110                                 else                            //key other then=
111         here means error wrong input
112                                 {
113                                         if(key=='C')        //if clear screenis
114         pressed then clear screen and reset
115                                         {
116                                             writecmd(0x01);     //Clear Screen
117                                                 WebsiteLogo();
118                                         }
119                                         else
120                                         {
121                                                 DispError(0);           //Display     wrong
122           input     error
123                                                     WebsiteLogo();
124                                         }
125                                 }
126                         }
127                 }
128             }
129         }
130         }
131
132         void WebsiteLogo()
133         {
134             writecmd(0x95);
135             writedata('w');                                 //write
136             writedata('w');                                 //write
137             writedata('w');                                 //write
138             writedata('.');                                 //write
139             writedata('T');                                 //write
140             writedata('h');                                 //write
141             writedata('e');                                 //write
142             writedata('E');                                 //write
143             writedata('n');                                 //write
144             writedata('g');                                 //write
145             writedata('i');                                 //write
146             writedata('n');                                 //write
147             writedata('e');                                 //write
148             writedata('e');                                 //write
149             writedata('r');                                 //write
150             writedata('i');                                 //write
151             writedata('n');                                 //write
152             writedata('g');                                 //write
```

```
153
154             writecmd(0xd8);
155
156             writedata('P');                                         //write
157             writedata('r');                                         //write
158             writedata('o');                                         //write
159             writedata('j');                                         //write
160             writedata('e');                                         //write
161             writedata('c');                                         //write
162             writedata('t');                                         //write
163             writedata('s');                                         //write
164             writedata('.');                                         //write
165             writedata('c');                                         //write
166             writedata('o');                                         //write
167             writedata('m');                                         //write
168             writecmd(0x80);
169         }
170
171         voidcct_init(void)
172         {
173           P0=0x00;          //notused
174           P1=0xf0;          //used for generating outputs and taking inputs fromKeypad
175           P2=0x00;          //used as data port forLCD
176           P3=0x00;          //used for RS andE
177         }
178
179         void delay(inta)
180         {
181             inti;
182             for(i=0;i<a;i++);               //nullstatement
183         }
184
185         void writedata(chart)
186         {
187             RS=1;                           // This isdata
188             P2=t;                           //Datatransfer
189             E   =1;                           //  => E  = 1
190             delay(150);
191             E   =0;                           //  => E  = 0
192             delay(150);
193         }
194
195
196         void writecmd(int         z)
197         {
198             RS = 0;                         //     This   is command
199             P2=z;                           //Datatransfer
200             E   =1;                         // => E =1
201             delay(150);
202             E   =0;                         // => E =0
203             delay(150);
204         }
205
206         void lcdinit(void)
207         {
208            ////////////// Reset process from datasheet /////////
209                 delay(15000);
210              writecmd(0x30);
211                 delay(4500);
212              writecmd(0x30);
213                delay(300);
214              writecmd(0x30);
215                delay(650);
216            ////////////////////////////////////////////////////
217              writecmd(0x38);            //functionset
218              writecmd(0x0c);            //display on,cursor off,blinkoff
219              writecmd(0x01);            //cleardisplay
220              writecmd(0x06);            //entry mode, setincrement
221         }
222         voidReturnHome(void)                /* Return to 0 cursor location*/
```

```
223
224     {
225         writecmd(0x02);
226         delay(1500);
227         WebsiteLogo();
228     }
229
230     void writeline(char Line[])
231     {
232         int i;
233         for(i=0;i<strlen(Line);i++)
234         {
235                 writedata(Line[i]);                 /* Write Character*/
236         }
237
238         ReturnHome();                   /* Return to 0 cursor position*/
239     }
240
241     char READ_SWITCHES(void)
242     {
243       RowA =   0;  RowB    = 1; RowC = 1; RowD    = 1;                  //Test Row A
244
245       if (C1   ==  0) {    delay(10000); while    (C1==0);   return '7'; }
246       if (C2   ==  0) {    delay(10000); while    (C2==0);   return '8'; }
247       if (C3   ==  0) {    delay(10000); while    (C3==0);   return '9'; }
248       if (C4   ==  0) {    delay(10000); while    (C4==0);   return '/'; }
249
250       RowA =   1;  RowB    = 0; RowC = 1; RowD    = 1;                  //Test Row B
251       if (C1   ==  0) {    delay(10000); while    (C1==0);   return '4'; }
252       if (C2   ==  0) {    delay(10000); while    (C2==0);   return '5'; }
253       if (C3   ==  0) {    delay(10000); while    (C3==0);   return '6'; }
254       if (C4   ==  0) {    delay(10000); while    (C4==0);   return 'x'; }
255
256
257       RowA =   1;  RowB    = 1; RowC = 0; RowD    = 1;                  //Test Row C
258
259       if (C1   ==  0) {    delay(10000); while    (C1==0);   return '1'; }
260       if (C2   ==  0) {    delay(10000); while    (C2==0);   return '2'; }
261       if (C3   ==  0) {    delay(10000); while    (C3==0);   return '3'; }
262       if (C4   ==  0) {    delay(10000); while    (C4==0);   return '-'; }
263
264       RowA =   1;  RowB    = 1; RowC = 1; RowD    = 0;                  //Test Row D
265
266       if (C1   ==  0) {    delay(10000); while    (C1==0);   return 'C'; }
267       if (C2   ==  0) {    delay(10000); while    (C2==0);   return '0'; }
268       if (C3   ==  0) {    delay(10000); while    (C3==0);   return '='; }
269       if (C4   ==  0) {    delay(10000); while    (C4==0);   return '+'; }
270
271       return'n';                              // Means no key has beenpressed
272     }
273
274     charget_key(void)                         //get key fromuser
275     {
276       char key='n';                           //assume no key pressed 277
278       while(key=='n')                         //wait untill a key ispressed
279                 key=READ_SWITCHES();          //scan the keys again and again 280
281       returnkey;                              //when key pressed then return itsvalue
282     }
283
284     int get_num(char ch)                      //convert char into int
285     {
286       switch(ch)
287       {
288                 case '0': return 0; break;
289                 case '1': return 1; break;
290                 case '2': return 2; break;
291                 case '3': return 3; break;
292                 case '4': return 4; break;
293                 case '5': return 5; break;
294                 case '6': return 6; break;
```

Post Lab Questions

1. What is the need for LCD in the project?
2. What is the need for ADC in the Project?
3. What are the arithmetic operations done in the simulation?
4. What is the need for RESET button in the project?
5. What is the operating frequency of the microcontroller?

Result

Thus the calculator was simulated using8051microcontroller in Proteus Software.