Experiment--05-4X4-keypad-interface-with-LPC2148

Name :SOUVIK KUNDU

Roll no : 212221230105

Date of experiment :01-11-2022

Interfacing a 4X4 keypad LPC2148 ARM 7 Microcontroller

Aim: To Interface 4x4 keypad interface LPC2148 ARM 7 and write a code for displaying the inputs on a 16x2 lcd

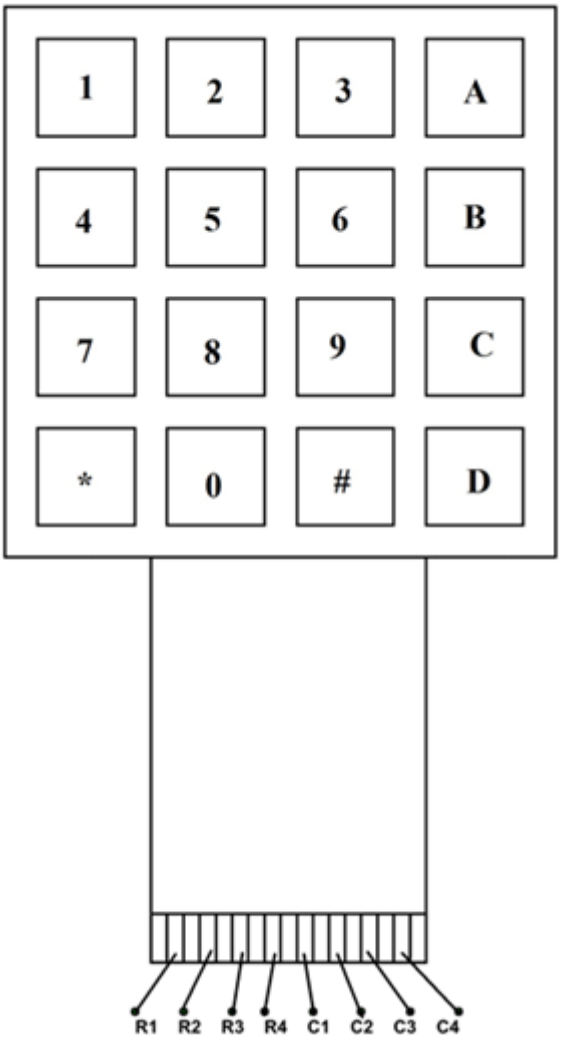
Components required: Proteus ISIS professional suite, Kiel µ vision 5 Development environment

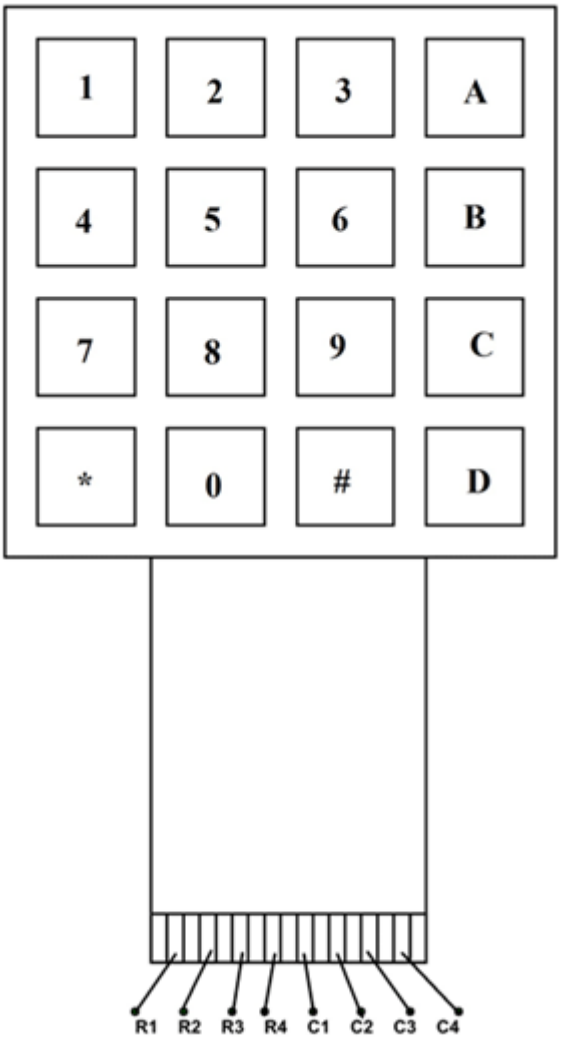
Theory

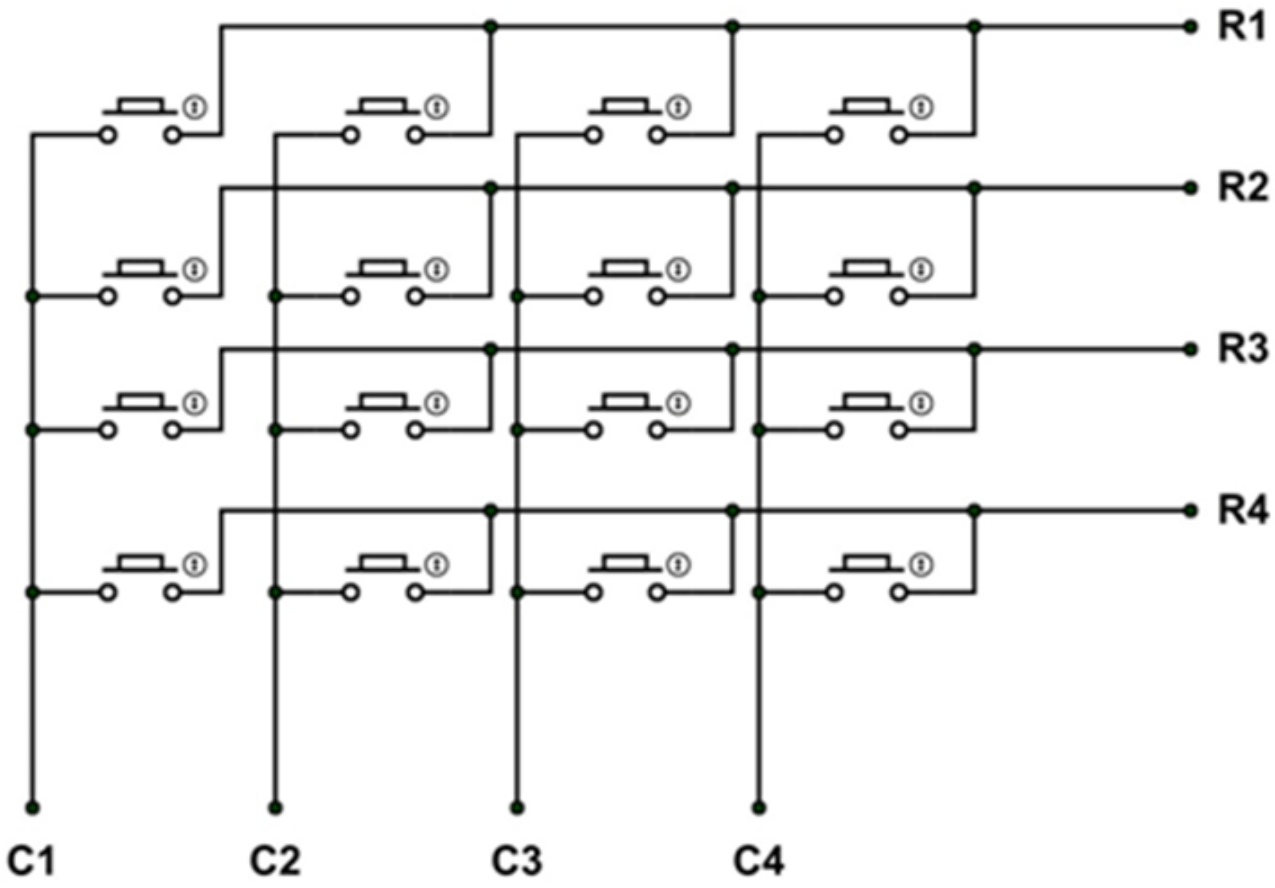
The full form of an ARM is an advanced reduced instruction set computer (RISC) machine, and it is a 32-bit processor architecture expanded by ARM holdings. The applications of an ARM processor include several microcontrollers as well as processors. The architecture of an ARM processor was licensed by many corporations for designing ARM processor-based SoC products and CPUs. This allows the corporations to manufacture their products using ARM architecture. Likewise, all main semiconductor companies will make ARM-based SOCs such as Samsung, Atmel, TI etc.

What is an ARM7 Processor? ARM7 processor is commonly used in embedded system applications. Also, it is a balance among classic as well as new-Cortex sequence. This processor is tremendous in finding the resources existing on the internet with excellence documentation offered by NXP Semiconductors. It suits completely for an apprentice to obtain in detail hardware & software design implementation. LPC2148 Microcontroller The LPC2148 microcontroller is designed by Philips (NXP Semiconductor) with several in-built features & peripherals. Due to these reasons, it will make more reliable as well as the efficient option for an application developer. LPC2148 is a 16-bit or 32-bit microcontroller based on ARM7 family. Features of LPC2148 The main features of LPC2148 include the following. • The LPC2148 is a 16 bit or 32 bit ARM7 family based microcontroller and available in a small LQFP64 package. • ISP (in system programming) or IAP (in application programming) using on-chip boot loader software. • On-chip static RAM is 8 kB-40 kB, on-chip flash memory is 32 kB-512 kB, the wide interface is 128 bit, or accelerator allows 60 MHz high-speed operation. • It takes 400 milliseconds time for erasing the data in full chip and 1 millisecond time for 256 bytes of programming. • Embedded Trace interfaces and Embedded ICE RT offers real-time debugging with high-speed tracing of instruction execution and on-chip Real Monitor software. • It has 2 kB of endpoint RAM and USB 2.0 full speed device controller. Furthermore, this microcontroller offers 8kB on-chip RAM nearby to USB with DMA. • One or two 10-bit ADCs offer 6 or 14 analogs i/ps with low conversion time as 2.44 µs/ channel. • Only 10 bit DAC offers changeable analog o/p. • External event counter/32 bit timers-2, PWM unit, & watchdog. • Low power RTC (real time clock) & 32 kHz clock input. • Several serial interfaces like two 16C550 UARTs, two I2C-buses with 400 kbit/s speed. • 5 volts tolerant quick general purpose Input/output pins in a small LQFP64 package. • Outside interrupt pins-21. • 60 MHz of utmost CPU CLK-clock obtainable from the programmable-on-chip phase locked loop by resolving time is 100 µs. • The incorporated oscillator on the chip will work by an exterior crystal that ranges from 1 MHz-25 MHz • The modes for power-conserving mainly comprise idle & power down. • For extra power optimization, there are individual enable or disable of peripheral functions and peripheral CLK scaling.

4x4 keypad


1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
R1
R2
R3
R4
C1
C2
C3
C4

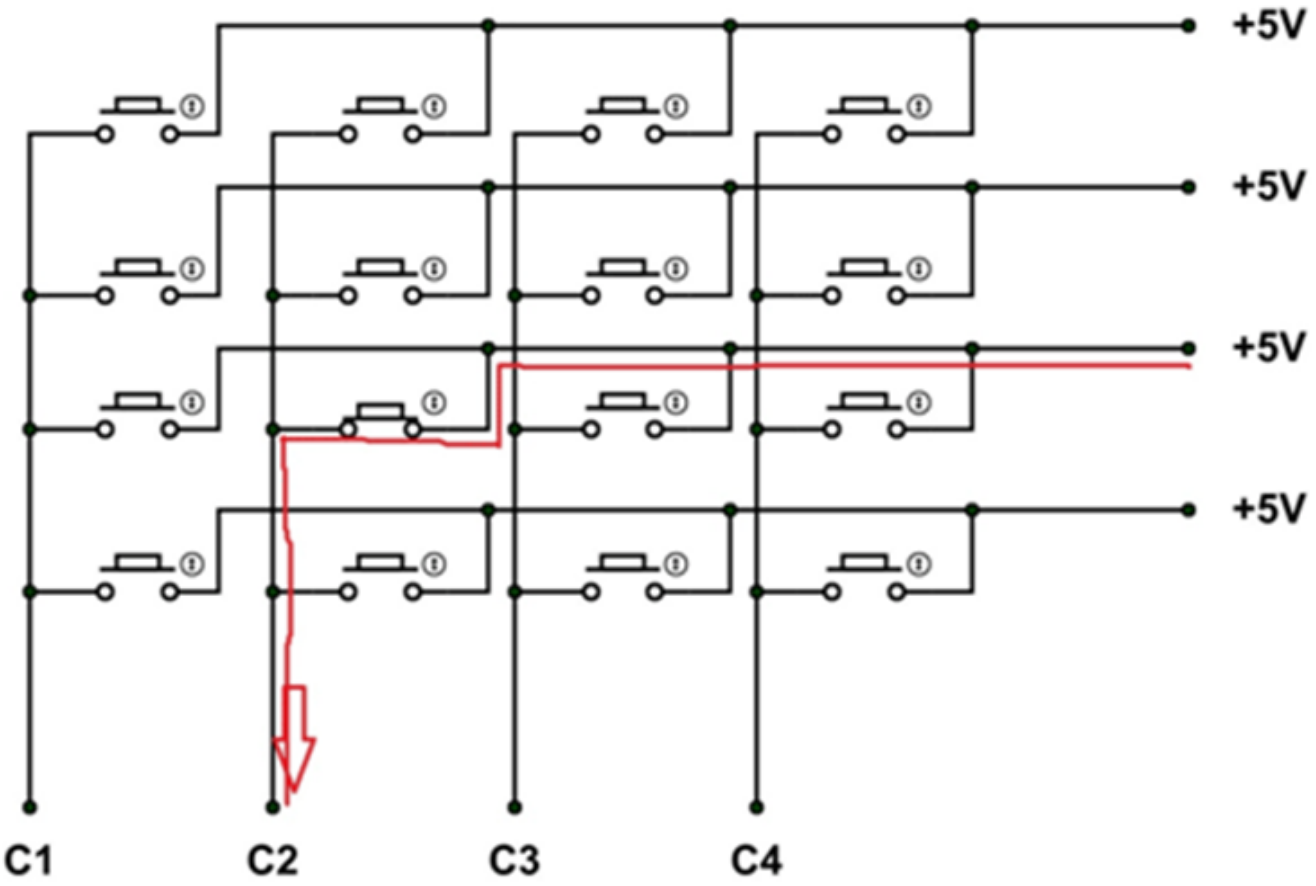

1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
R1
R2
R3
R4
C1
C2
C3
C4



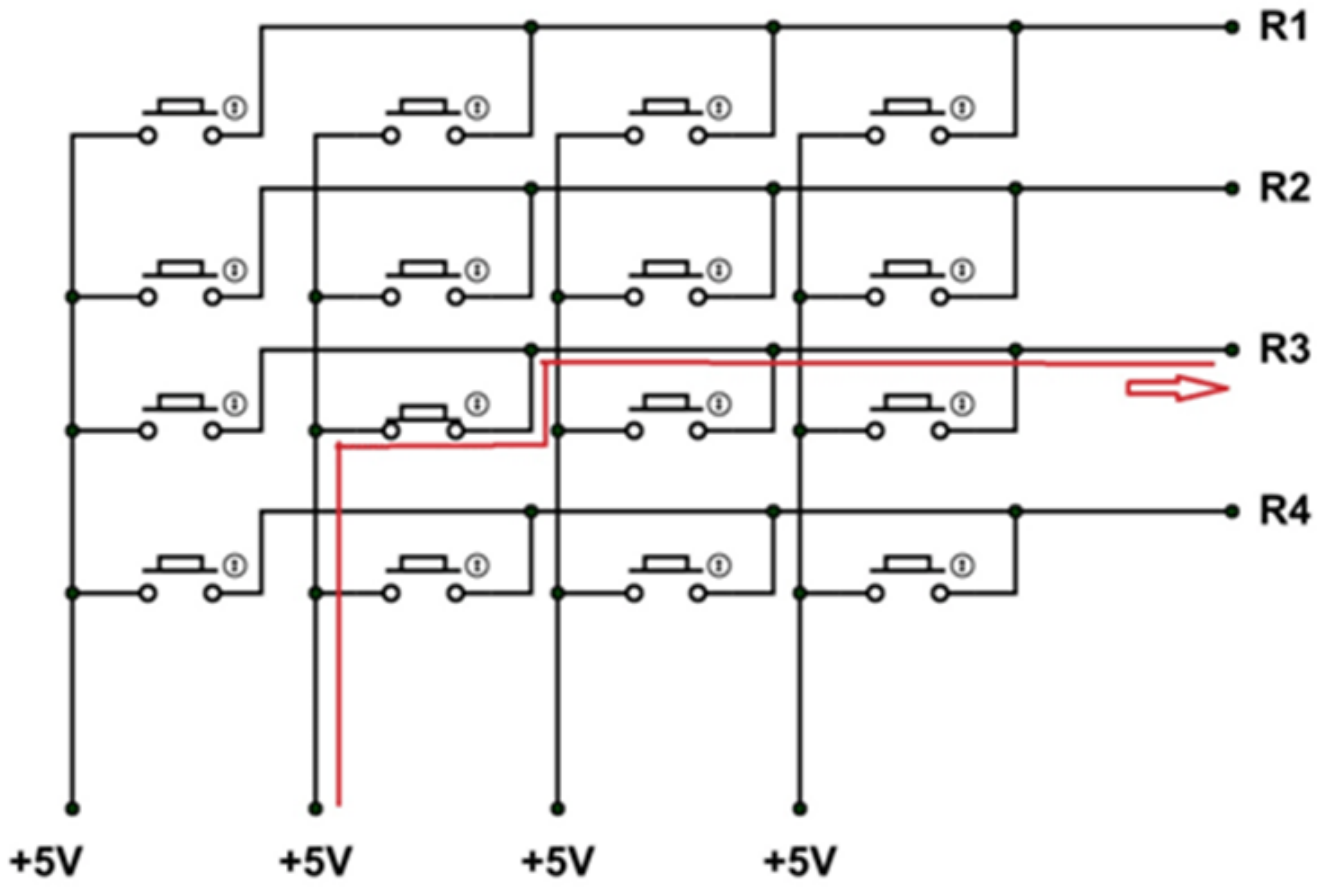
• STEP1: First set all ROWS to OUTPUT and set them at +5V. Next set all COLUMNS as INPUT to sense the HIGH logic. Now consider a button is pressed on keypad. And that key is at 2ND COLUMN and 3rd ROW.

• With the button being pressed the current flows as shown in figure. With that a voltage of +5V appears at terminal C2. Since the COLUMN pins are set as INPUTS, the controller can sense C2 going high. The controller can be programmed to remember that C2 going high and the button pressed is in C2 COLUMN.

• STEP2: Next set all COLUMNS to OUTPUT and set them at +5V. Next set all ROWS as INPUT to sense the HIGH logic. Since the key pressed is at 2ND COLUMN and 3rd ROW. The current flows as shown below.


+5V
+5V
+5V
+5V
C1
C2
C3
C4

With that current flow a positive voltage of +5V appears at R3 pin. Since all ROWS are set as INPUTS, the controller can sense +5V at R3 pin. The controller can be programmed to remember the key being pressed at third ROW of KEYPAD MATRIX.



• From previous step, we have known the COLUMN number of key pressed and now we know ROW number. With that we can match the key being pressed. We can take the key INPUT provided by this way for 4X4 •

Procedure: For creation of project on Kiel µ vision 5 Development environment (LPC21 XX/48/38)

1. Click on the menu Project — New µVision Project creates a new project. Select an empty folder and enter the project name, for example Project1. It is good practice to use a separate folder for each project.
2. Next, the dialog Select Device for Target opens.

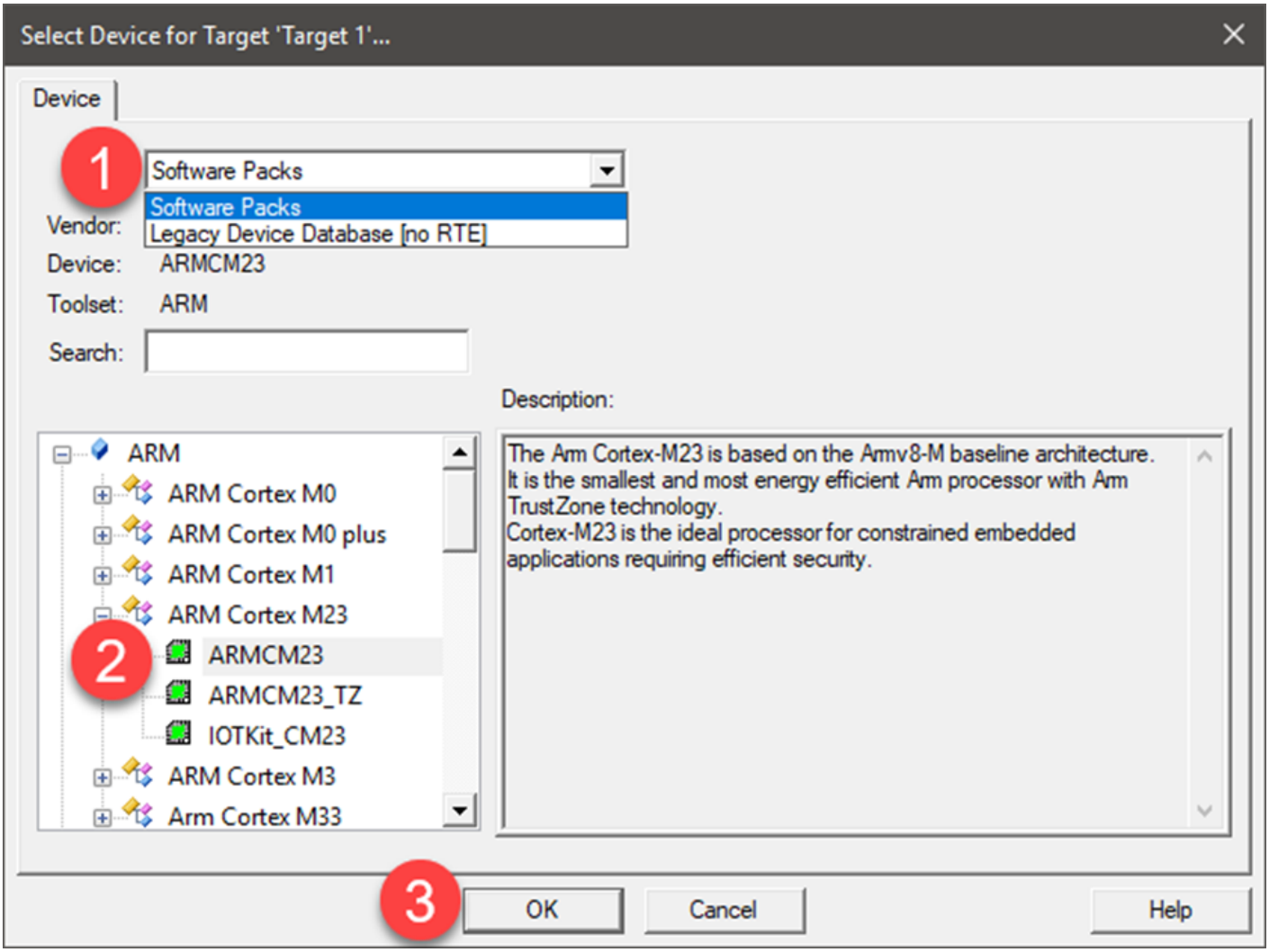


Figure -01 Target selection Select the device database. Default is Software Packs. You can have various local device databases, which show up in the drop-down box. For legacy devices (Arm7, Arm9), use the Legacy Device Database [no RTE] 3. Select the device for your application. This selection defines essential tool settings such as compiler controls, the memory layout for the linker, and the Flash programming algorithms. 4. Click OK. 5. Click on the new file option and save the file using save option with .C extension 6. Build all for the hex code in the output of the target

For creating the simulation environment in Proteus suite Starting New Design Step 1: Open ISIS software and select New design in File menu

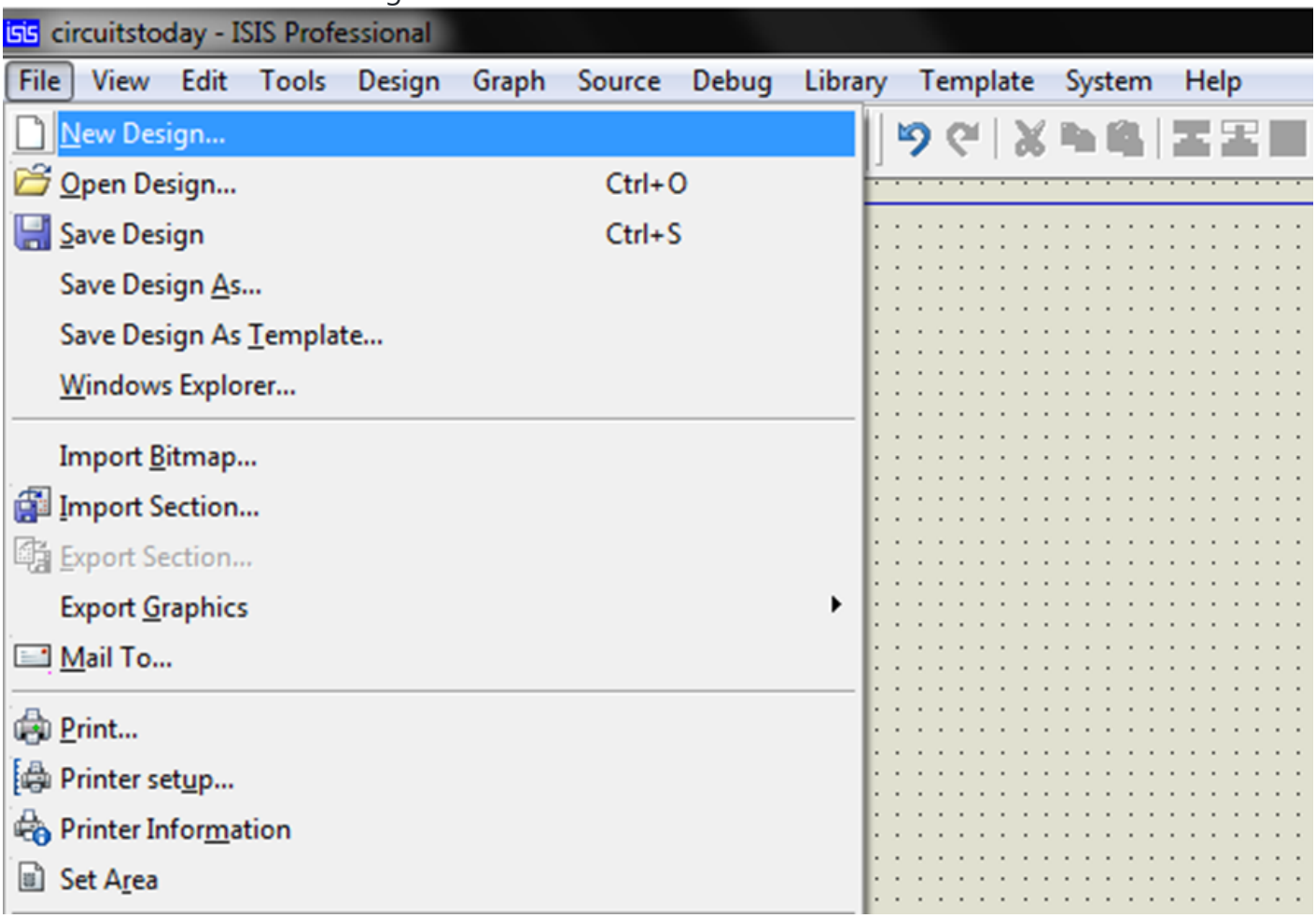


Figure -02 Proteus File Menu

Step 2: A dialogue box appears to save the current design. However, we are creating a new design file so you can click Yes or No depending on the content of the present file. Then a Pop-Up appears asking to select the template. It is similar to selecting the paper size while printing. For now select default or according to the layout size of the circuit..

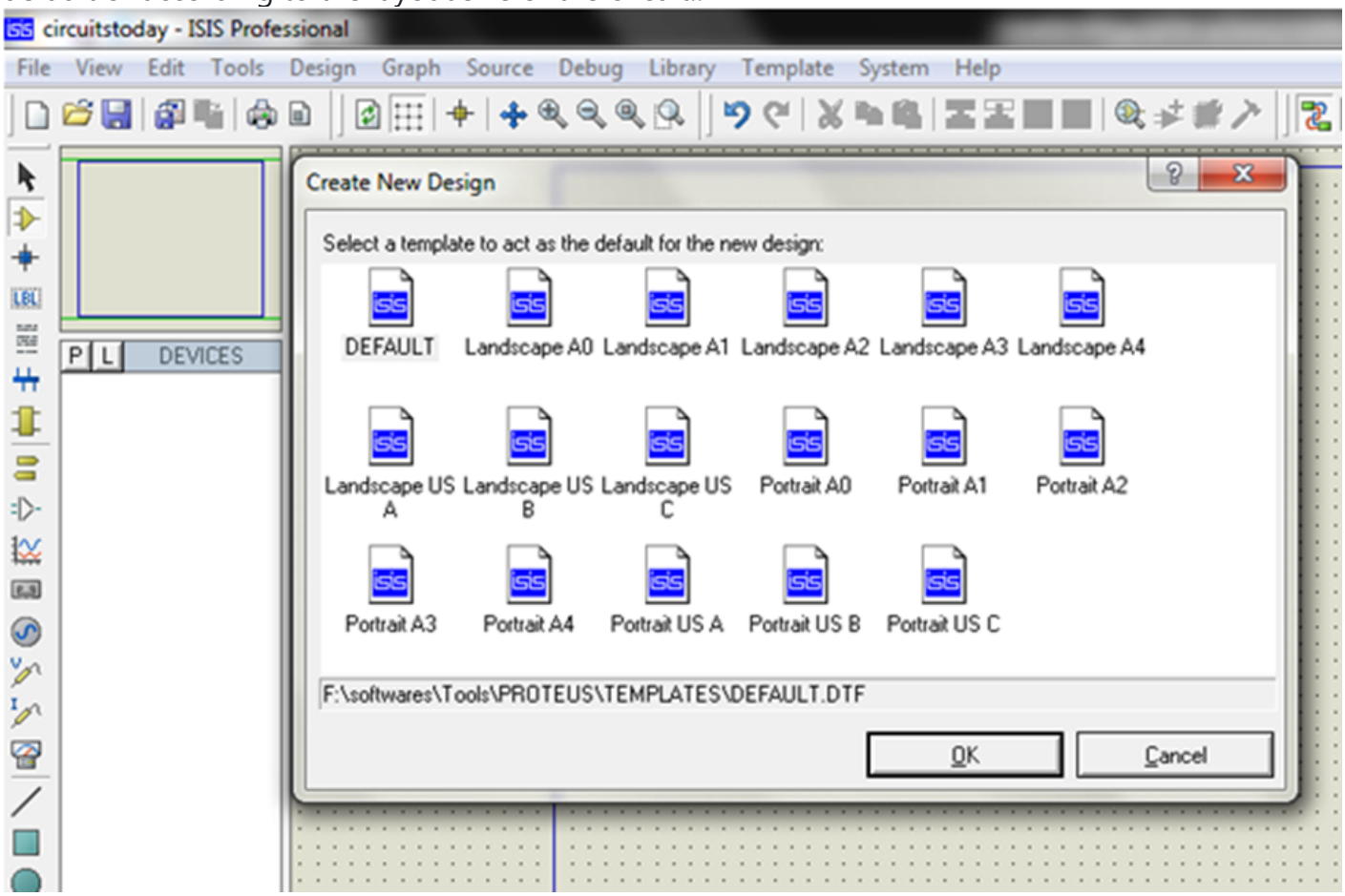
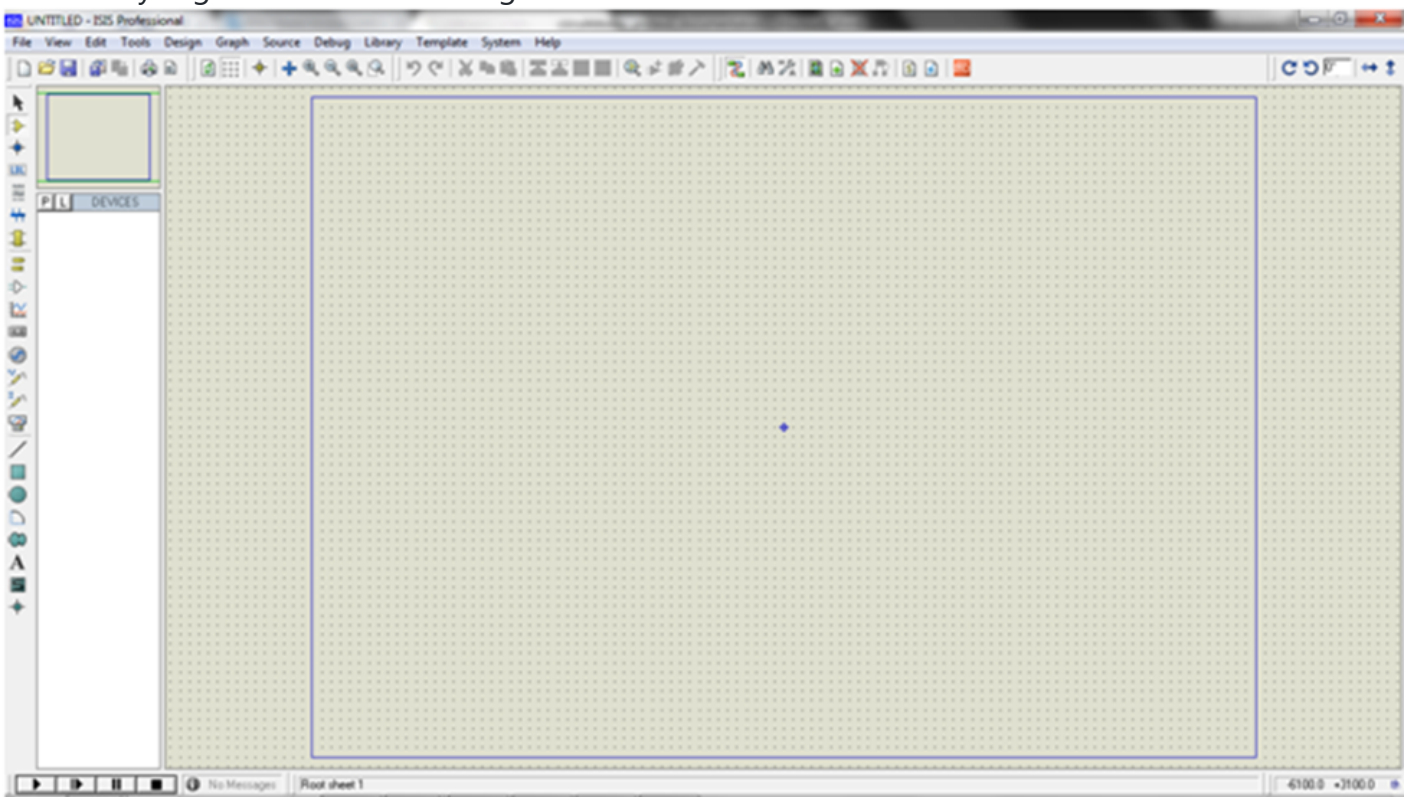


Figure -03 Proteus Default Template Select

Step 3:An untitled design sheet will be opened, save it according to your wish,it is better to create a new folder for every layout as it generates other files supporting your design. However,it is not mandatory. Figure -04 Proteus Design Sheet



Step 4:To Select components, Click on the component mode button.

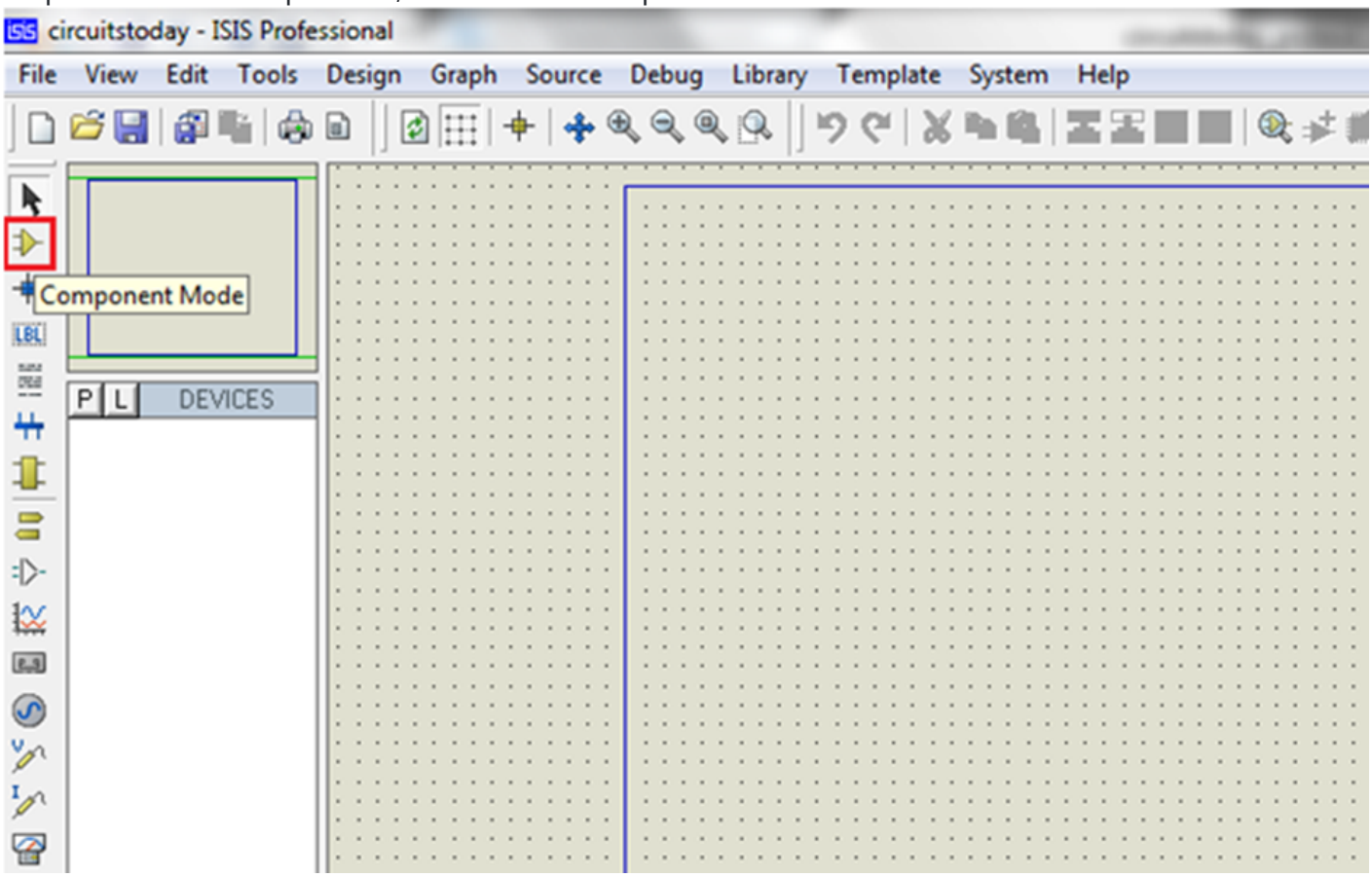


Figure -05 Component Mode Step 5:Click On Pick from Libraries. It shows the categories of components available and a search option to enter the part name.

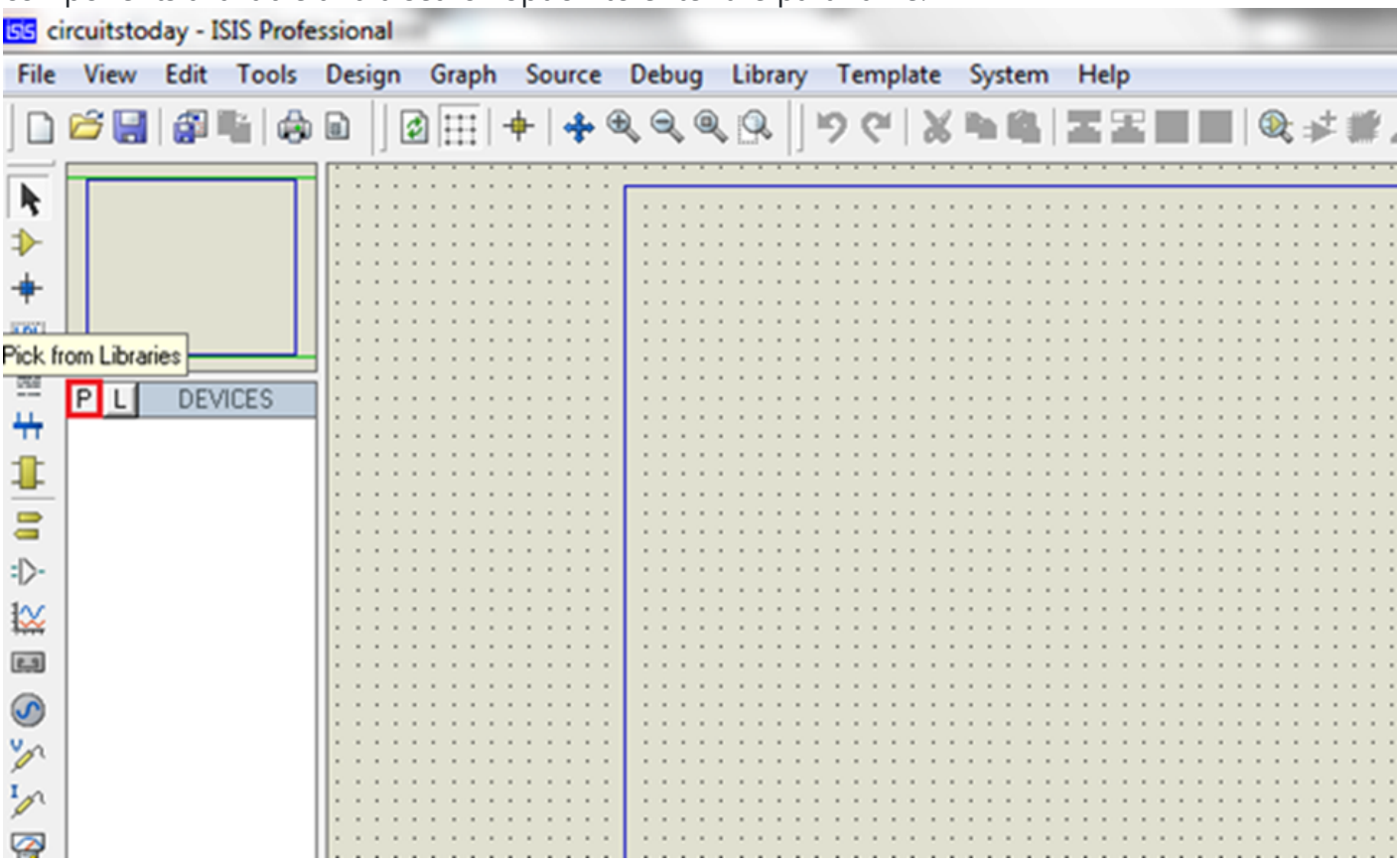


Figure -06 Pick from Libraries

Step 6: Select the components from categories or type the part name in Keywords text box. Place all the required components and route the wires i.e, make connections. Either selection mode above the component mode or component mode allows to connect through wires. Left click from one terminal to other to make connection. Double right-click on the connected wire or the component to remove connection or the component respectively.

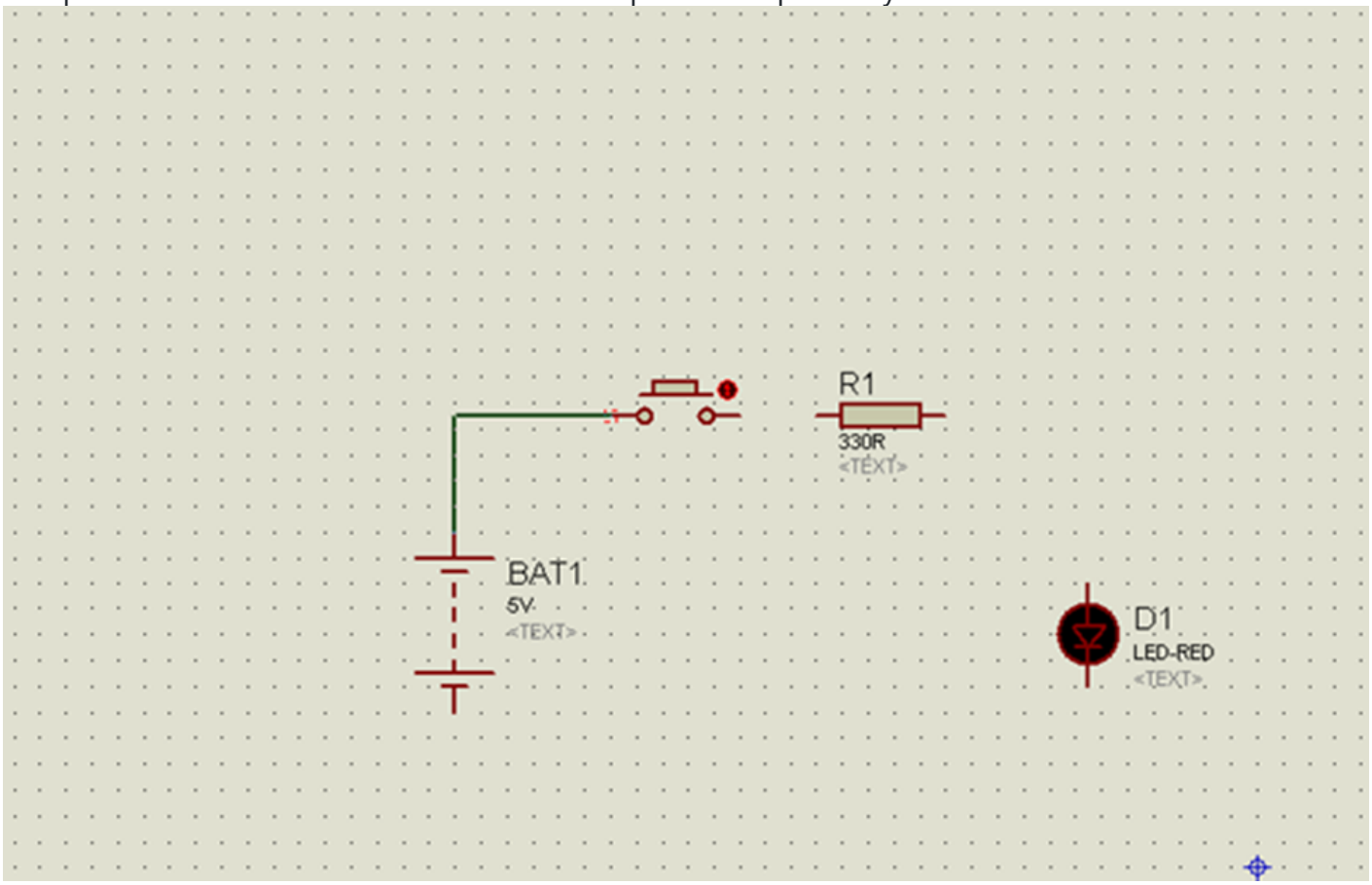


Figure -07 Component Properties Selection Double click on the component to edit the properties of the components and click on Ok. Step 8: Select ARM microcontroller form the library – pick part

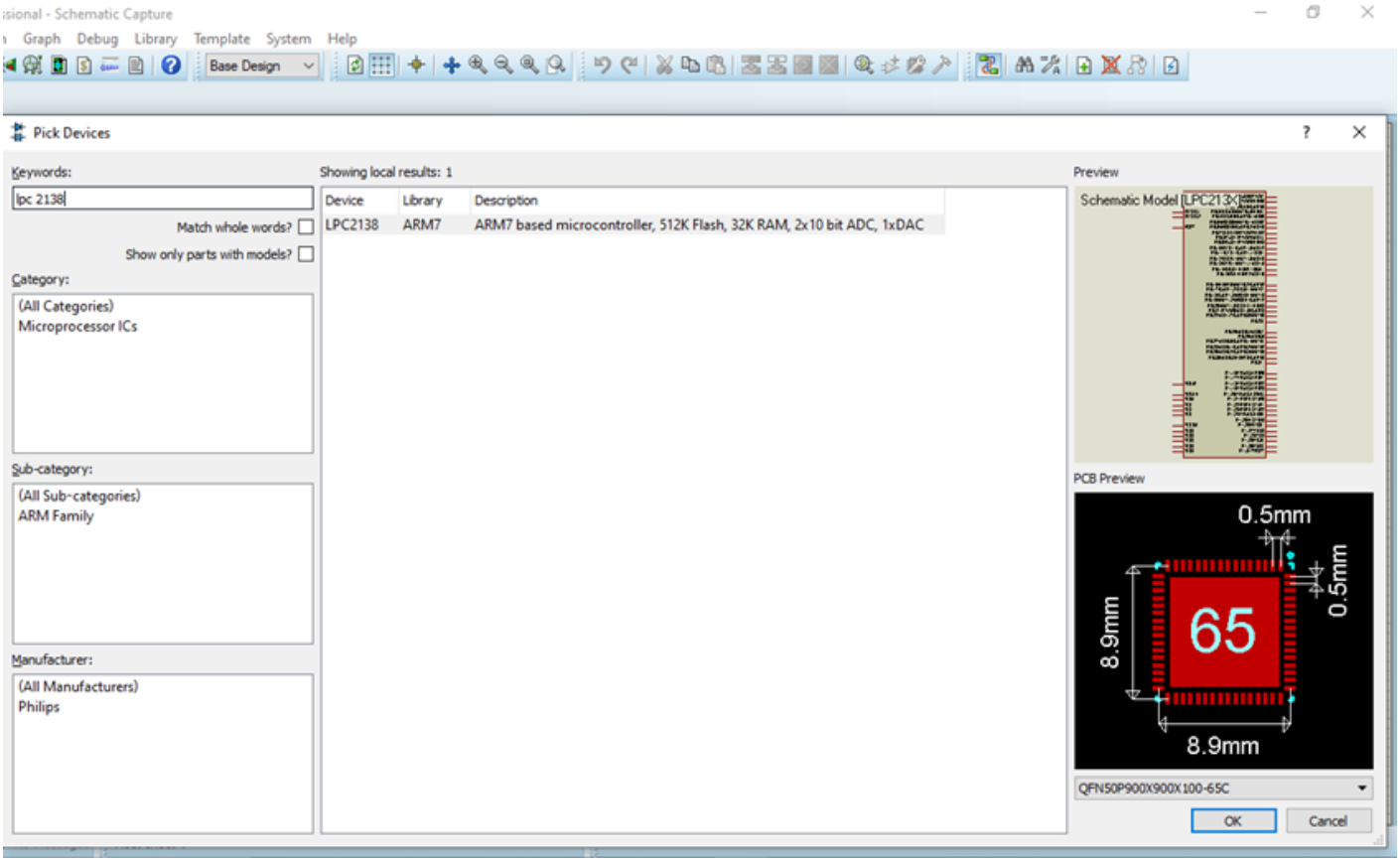


Figure -08 LPC2138/48 selection Step 7:

After making necessary connections click on debug from Figure -09 Keywords Textbox Example shows selection of push button. Select the components accordingly.

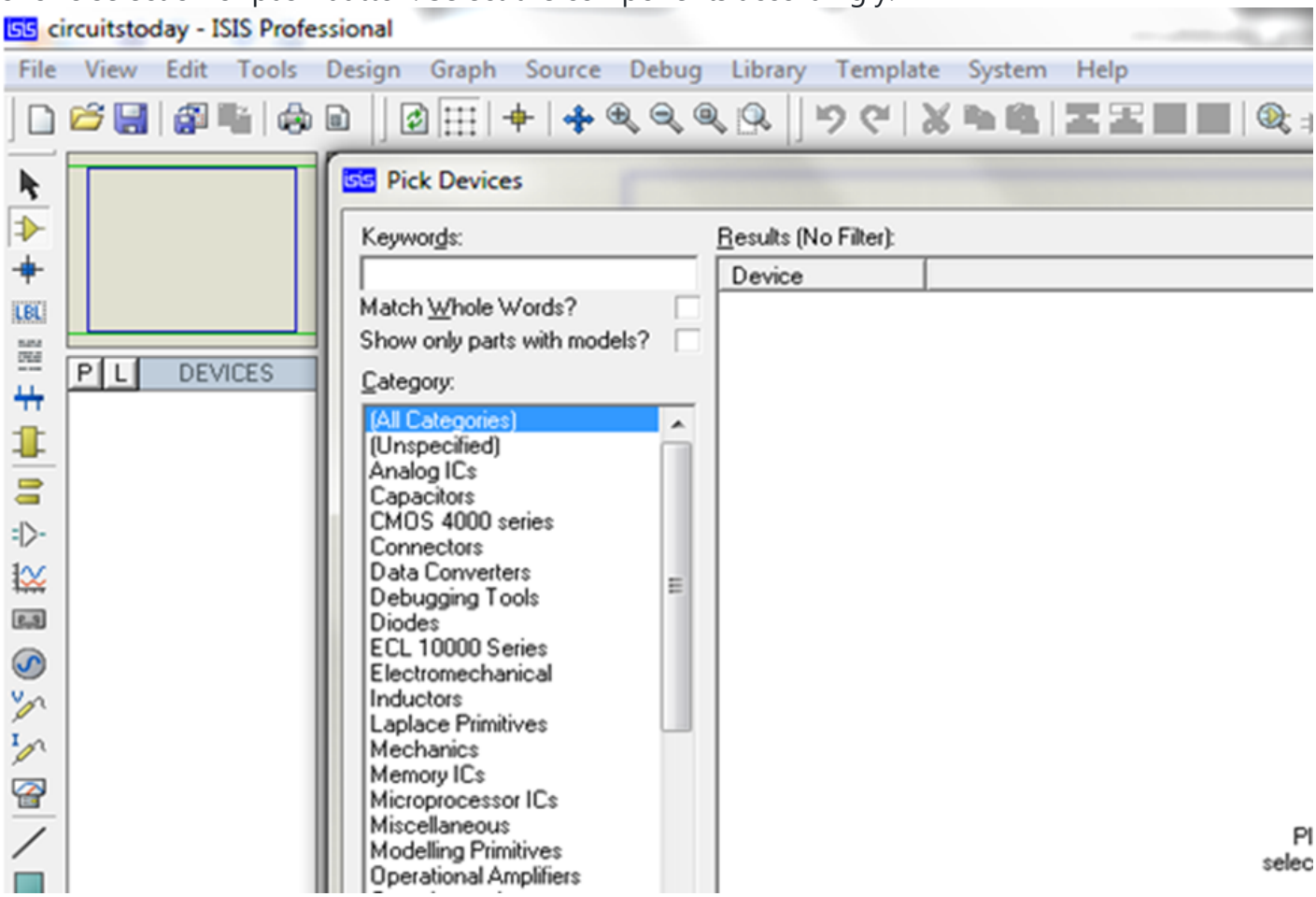


Figure -09 Push Button Selection Step 8: The selected components will appear in the devices list. Select the component and place it in the design sheet by left-click., post which select all the associated components as shown in the circuit diagram below.

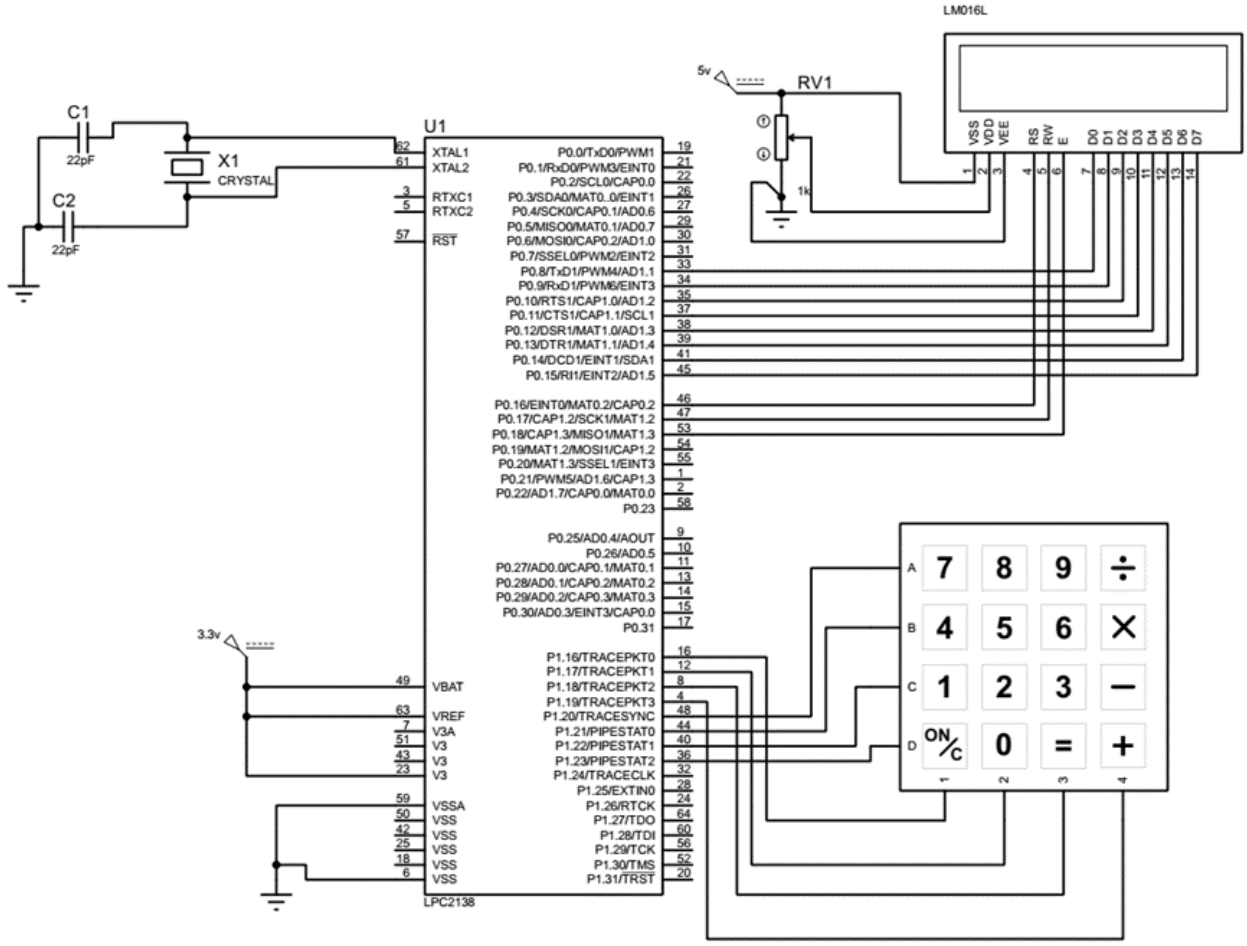

LM016L
5v
RV1
1k
C1
22pF
C2
22pF
X1
CRYSTAL
U1
XTAL1
XTAL2
RTXC1
RTXC2
RST
VBAT
VREF
V3A
V3
V3
V3
VSSA
VSS
VSS
VSS
VSS
VSS
LPC2138
3.3v
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
A 7 8 9 ÷
B 4 5 6 ×
C 1 2 3 −
D ON/C 0 = +

Figure -10 Circuit diagram of4X4 keypad and 16x2 LCD interface with LPC2148/38

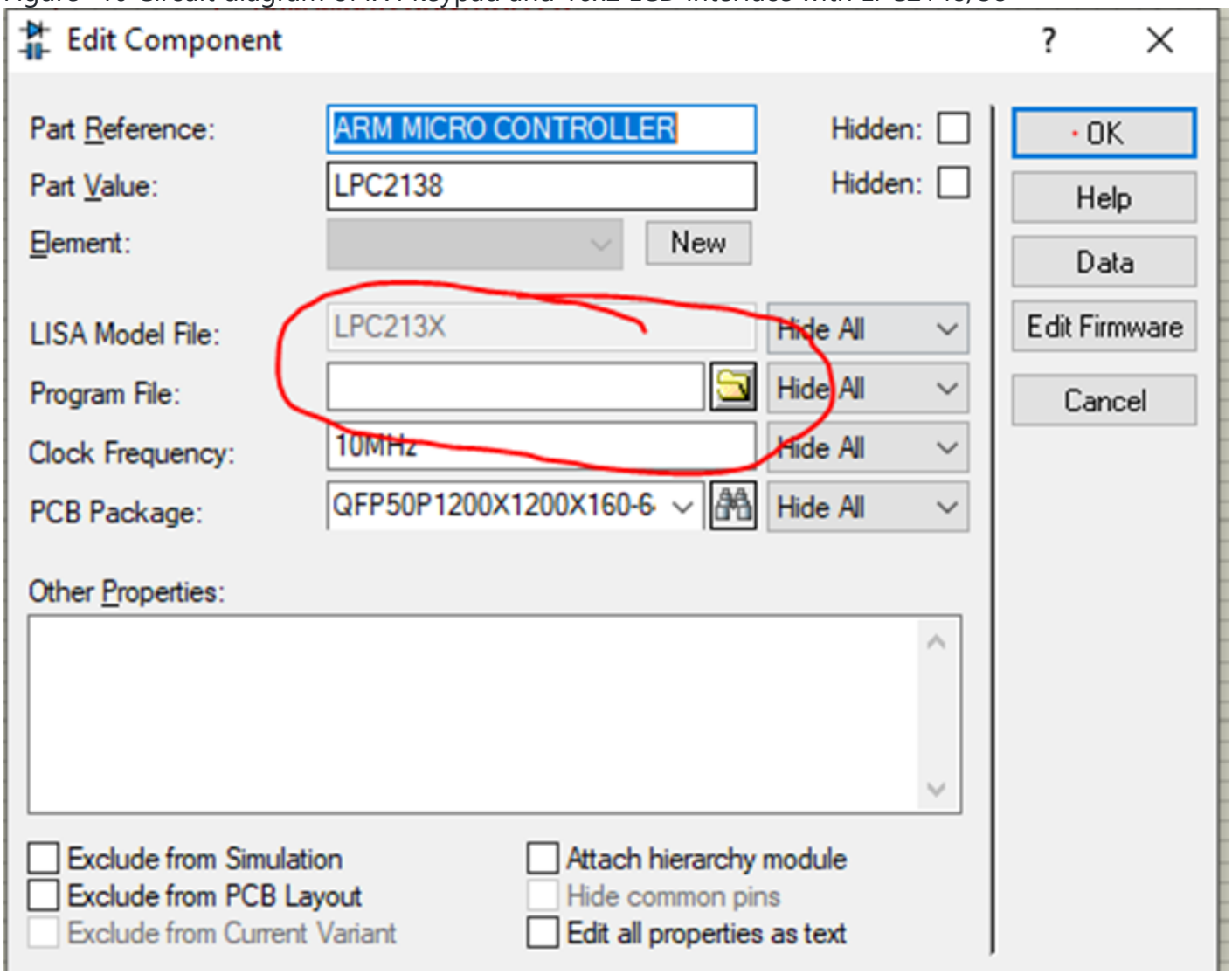


Figure -11 Hex file for simulation

Step 9: Select the hex file from the Kiel program folder and import the program in to the microcontroller as shown in figure 11 , debug and if no errors in connections are found, run the VSM simulation to view the output.

Kiel - Program:

```
#include <lpc21xx.h>
#define RS (1<<0)
#define RW (1<<1)
#define E (1<<2)

void LCD_command(unsigned char command);
void  delay_ms(unsigned char time);
void LCD_data(unsigned char data);
void LCD_init() ;


int main(void)
```

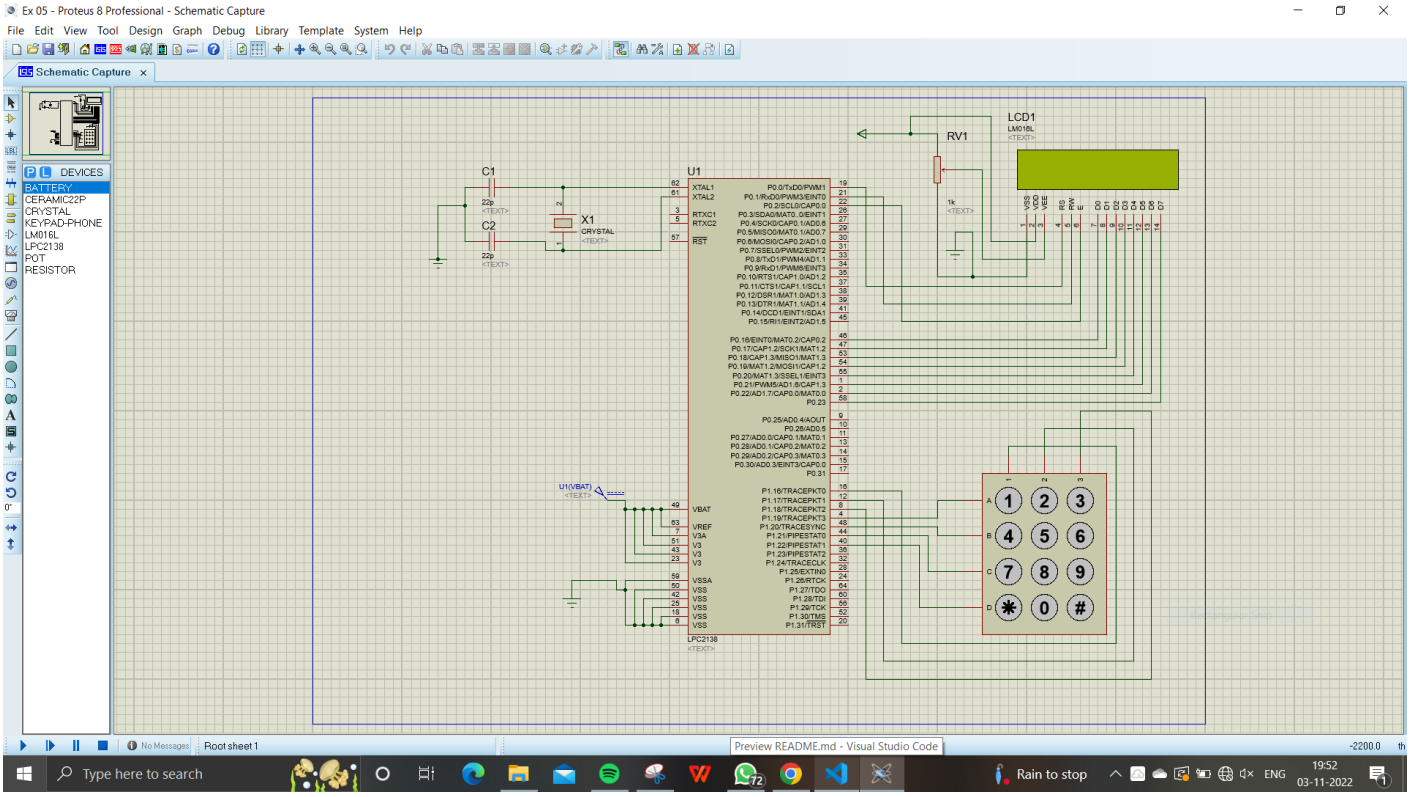
```
{
 //PINSEL1 = 0x00000000;  //Configure PORT0 as GPIO
 //PINSEL2 = 0X00000000;  //Configure PORT1 as GPIO
 IODIR1= 0x00780000; //Configure P1.18, P1.17, P1.16 as output(for rows and column)
 IODIR0= 0x00FF0007;  //Configure P0.23 - P0.16 as output for lcd data & P0.0,P0.1,P0.2
for lcd control lines.
 LCD_init();    //Initialize LCD 16x2
 LCD_command(0x01);
 while(1)
   {
      IOCLR1|=(1<<19);               //Making row1 LOW
      IOSET1|=(1<<20)|(1<<21)|(1<<22); //Making rest of the rows '1'
      if(!(IOPIN1&(1<<16)))             //Scan for key press
       {
        while(!(IOPIN1&(1<<16)));
        LCD_data('1');
       }
      if(!(IOPIN1&(1<<17)))
       {
         while(!(IOPIN1&(1<<17)));
          LCD_data('2');
       }
      if(!(IOPIN1&(1<<18)))
       {
         while(!(IOPIN1&(1<<18)));
          LCD_data('3');
       }
      IOCLR1|=(1<<20);
      IOSET1|=(1<<21)|(1<<22)|(1<<19);
      if(!(IOPIN1&(1<<16)))
{
        while(!(IOPIN1&(1<<16)));
         LCD_data('4');
      }
      if(!(IOPIN1&(1<<17)))
{
        while(!(IOPIN1&(1<<17)));
         LCD_data('5');
     }
      if(!(IOPIN1&(1<<18)))
{
        while(!(IOPIN1&(1<<18)));
         LCD_data('6');
     }
      IOCLR1|=(1<<21);
      IOSET1|=(1<<22)|(1<<20)|(1<<19);
      if(!(IOPIN1&(1<<16)))
{
        while(!(IOPIN1&(1<<16)));
         LCD_data('7');
     }
      if(!(IOPIN1&(1<<17)))
{
```

```
       while(!(IOPIN1&(1<<17)));
        LCD_data('8');
    }
      if(!(IOPIN1&(1<<18)))
{
        while(!(IOPIN1&(1<<18)));
         LCD_data('9');
}
      IOCLR1|=(1<<22);
      IOSET1|=(1<<19)|(1<<20)|(1<<21);
      if(!(IOPIN1&(1<<16)))
{
        while(!(IOPIN1&(1<<16)));
         LCD_data('*');
}
      if(!(IOPIN1&(1<<17)))
{
        while(!(IOPIN1&(1<<17)));
         LCD_data('0');
}
      if(!(IOPIN1&(1<<18)))
{
        while(!(IOPIN1&(1<<18)));
         LCD_data('#');
}
   }
}


//Function to generate software delay
//Calibrated to 1ms
void  delay_ms(unsigned char time)
{
 unsigned int  i, j;
 for (j=0; j<time; j++)
 {
  for(i=0; i<8002; i++)
  {
  }
}
}

void LCD_command(unsigned char command)
{
 IOCLR0 = 0xFF<<16; // Clear LCD Data lines
 IOCLR0=RS;     // RS=0 for command
 IOCLR0=RW;     // RW=0 for write
 IOSET0=command<<16; // put command on data line
 IOSET0=E;   // en=1
 delay_ms(10) ;   // delay
 IOCLR0=E;    // en=0
}
```

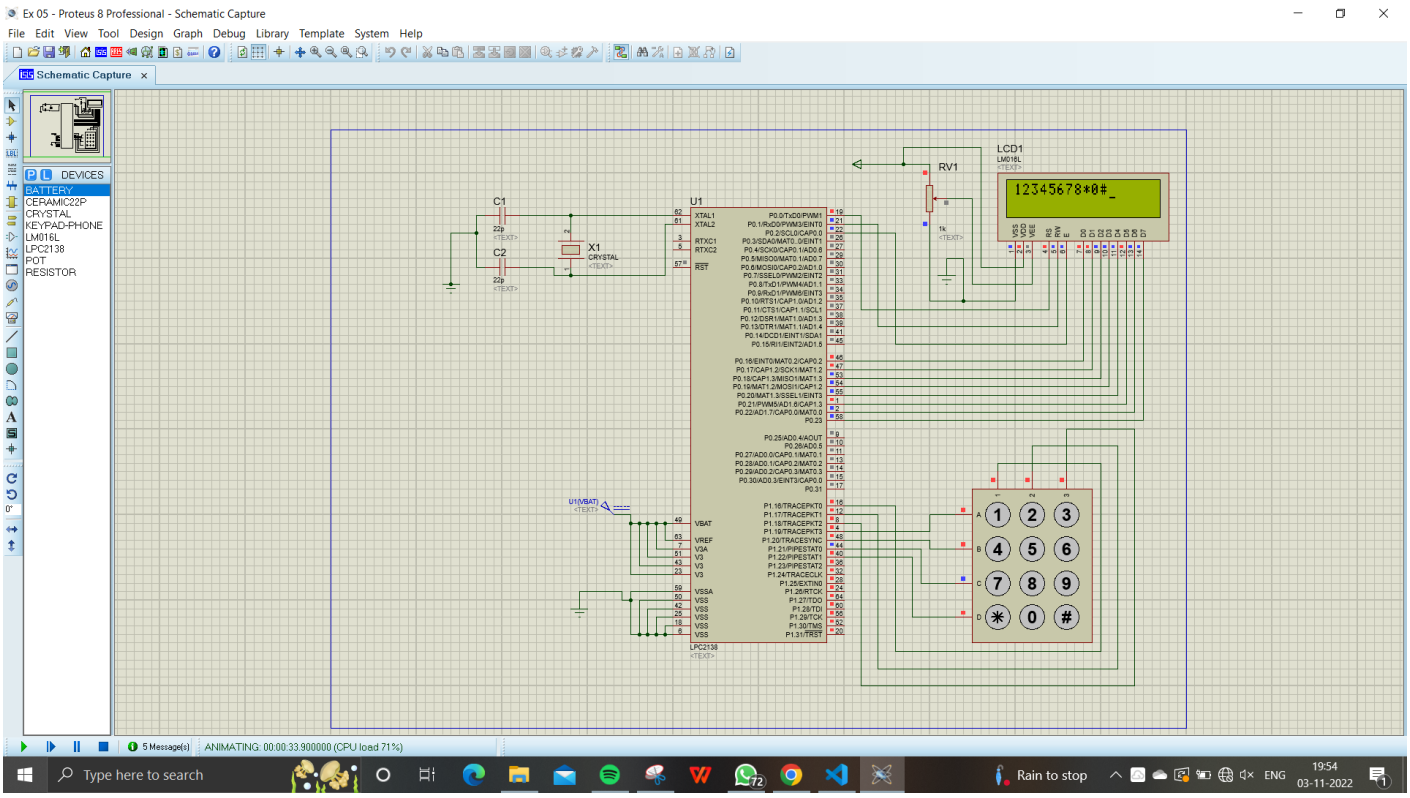
```
void LCD_data(unsigned char data)
{
 IOCLR0 = 0xFF<<16; // Clear LCD Data lines
 IOSET0=RS;      // RS=1 for data
 IOCLR0=RW;      // RW=0 for write
 IOSET0= data<<16;  // put command on data line
 IOSET0=E;   //en=1
 delay_ms(10) ;    //delay
 IOCLR0=E;   //en=0
 }

void LCD_init()
{
 LCD_command(0x38); //8bit mode and 5x8 dotes (function set)
 delay_ms(10) ;   // delay
 LCD_command(0x0c); //display on, cursor off, cursor char blinking off(display on/off)
 delay_ms(10) ;   // delay
    LCD_command(0x0e);  //cursor increment and display shift(entry mode set)
    delay_ms(10) ;   // delay
 LCD_command(0x01);  //clear lcd (clear command)
 delay_ms(10) ;   // delay
 LCD_command(0x80);
  delay_ms(10) ;//set cursor to 0th location1st lne
}
```

Output screen shots :

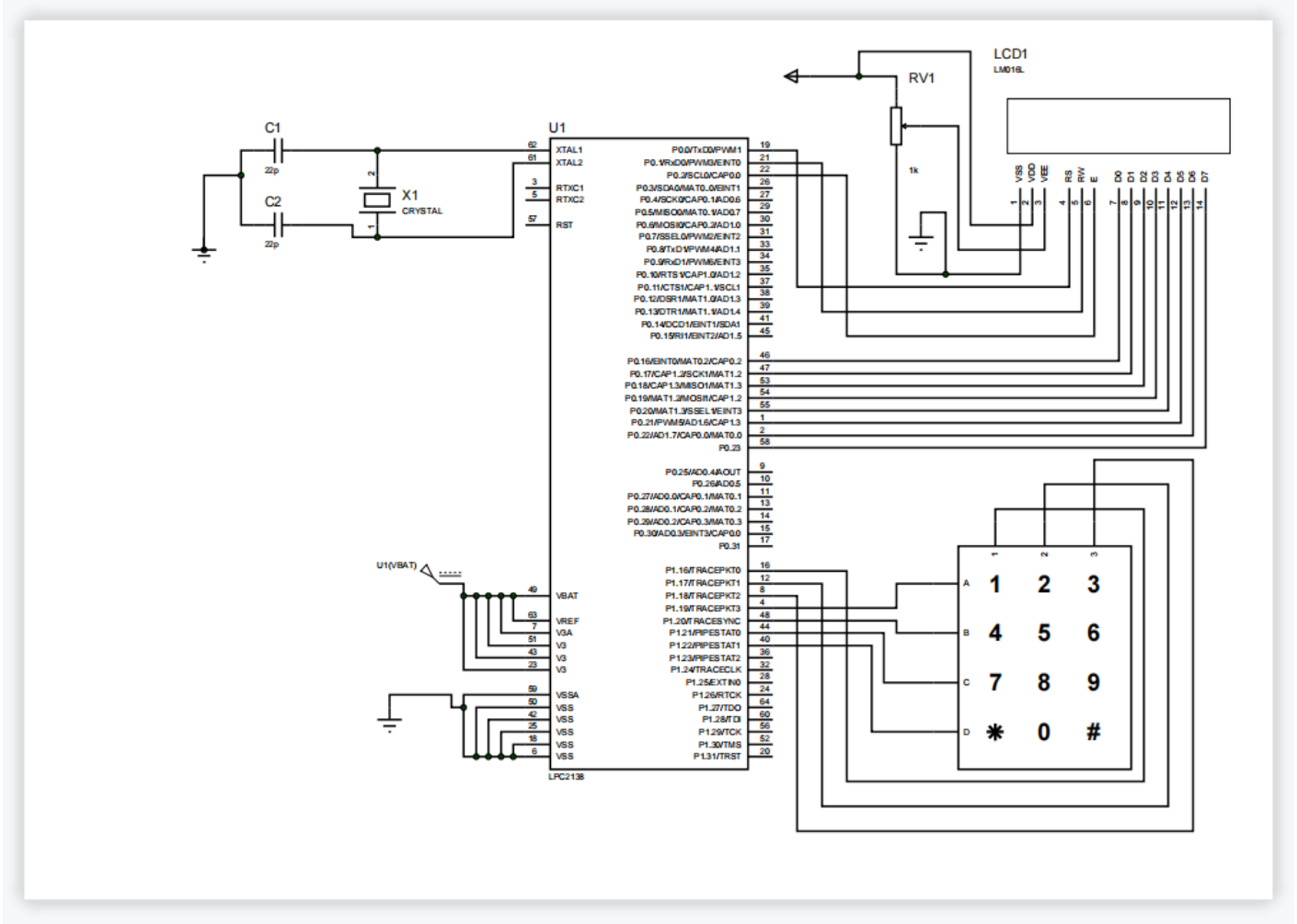
Before simulation:



After simulation:



Layout diagram:



Result :

Interfacing a keypad 4x4 is interfaced with LPC2148