ST.ANNE'S

COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN ISO 9001:2015 CERTIFIED INSTITUTION)

ANGUCHETTYPALAYAM, PANRUTI – 607 106

EC6711 EMBEDDED LABORATORY

OBSERVATION NOTE

(FOR IV B.E ELECTRONICS AND COMMUNICATION ENGINEERING STUDENTS)

NAME : _____

REGISTER NO : _____

SEMESTER : _____

JULY 2018 - NOV 2018

AS PER ANNA UNIVERSITY (CHENNAI) SYLLABUS

2013 REGULATION

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PREPARED BY: Mr. S. BALABASKER(ASP/ECE)

ABOUT OBSERVATION NOTE & PREPARATION OF RECORD

- This Observation contains the basic diagrams of the circuits enlisted in the syllabus of EC6711 EMBEDDED LABORATORY course, along with the design the design of various components of the circuit and controller.
- Aim of the experiment is also given at the beginning of each experiment. Once the student is able to design the circuit as per the circuit diagram, he/she is supposed to go through the instructions carefully and do the experiments step by step.
- They should note down the readings (observations) and tabulate them as specified.
- It is also expected that the students prepare the theory relevant to the experiment referring to prescribed reference book/journals in advance, and carry out the experiment after understanding thoroughly the concept and procedure of the experiment.
- They should get their observations verified and signed by the staff within two days and prepare & submit the record of the experiment while they come for the laboratory in subsequent week.
- The record should contain experiment No., Date ,Aim, Apparatus required, Theory, Procedure and result on one side(i.e., Right hand side, where rulings are provided) and Circuit diagram, Design, Model Graphs, Tabulations and Calculations on the other side (i.e., Left hand side, where no rulings is provided)
- All the diagrams and table lines should be drawn in pencil
- The students are directed to discuss & clarify their doubts with the staff members as and when required. They are also directed to follow strictly the guidelines specified.

EC6711 EMBEDDED LABORATORY

SYLLABUS

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.

OUTCOMES:

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

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

LIST OF EXPERIMENTS

S.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO	DATE OF SUBMISSION	MARK (10)	SIGNATURE

LIST OF EXPERIMENTS

S.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO	DATE OF SUBMISSION	MARK (10)	SIGNATURE

INTRODUCTION TO KEIL µ VISION 4 SOFTWARE

The µVision4 IDE is a Windows-based software development platform that combines a robust editor, project manager, and makes facility. µVision4 integrates all tools including the C compiler, macro assembler, linker/locator, and HEX file generator. µVision4 helps expedite the development process of your embedded applications by providing the following:

- ✓ Full-featured source code editor
- ✓ Device database for configuring the development tool setting
- ✓ Project manager for creating and maintaining your projects
- ✓ Integrated make facility for assembling, compiling, and linking your embedded applications
- ✓ Dialogs for all development tool settings
- ✓ True integrated source-level Debugger with high-speed CPU and peripheral simulator
- ✓ Advanced GDI interface for software debugging in the target hardware and for connection to Keil ULINK
- ✓ Flash programming utility for downloading the application program into Flash ROM
- ✓ Links to development tools manuals, device datasheets & user's guides

The µVision4 IDE offers numerous features and advantages that help you quickly and successfully develop embedded applications. They are easy to use and are guaranteed to help you achieve your design goals.

The installation steps for keil software are given below:

1. Double click on Keil µvision4 exe file.
2. Then click on **Next**.
3. Tick the check box towards to license agreements and click **Next**.
4. Select Destination folder and click **Next**.
5. Fill the necessary text boxes and click **Next**.
6. Finally click on **Finish**.

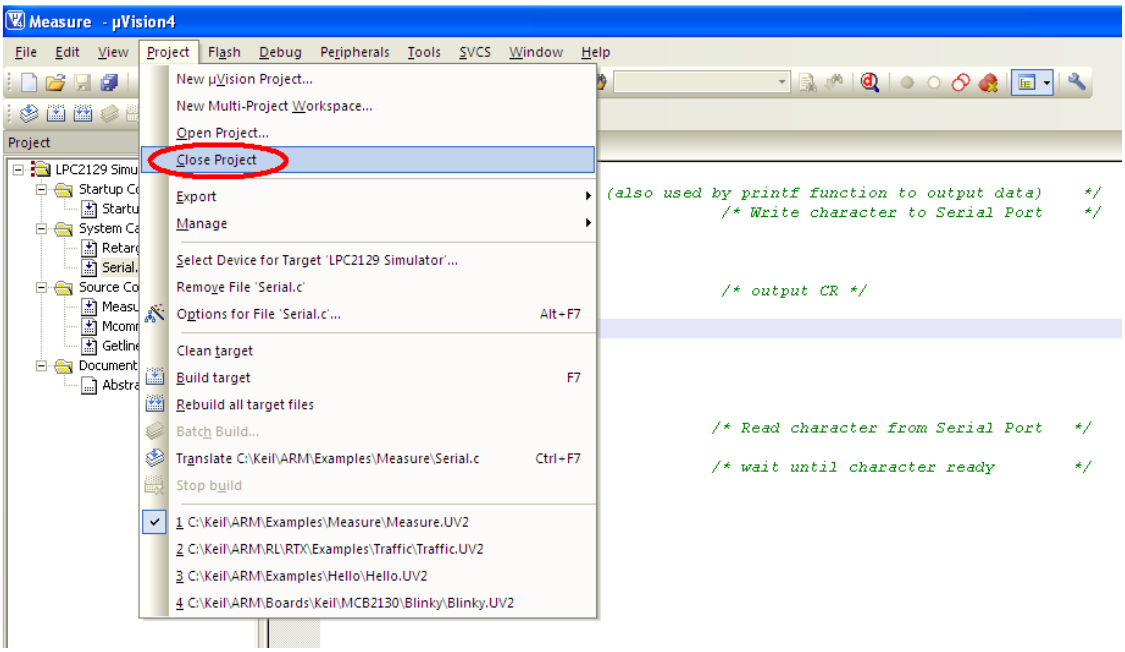
Software Flow

First open the icon keil µvision4 and the follow the steps are given below. The menu bar provides you with menus for editor operations, project maintenance, development tool option settings, program debugging, external tool control, window selection and manipulation, and on-line help. The toolbar buttons allow you to rapidly execute µVision4 commands. A Status Bar provides editor and debugger information. The various toolbars and the status bar can be enabled or disabled from the View Menu commands.

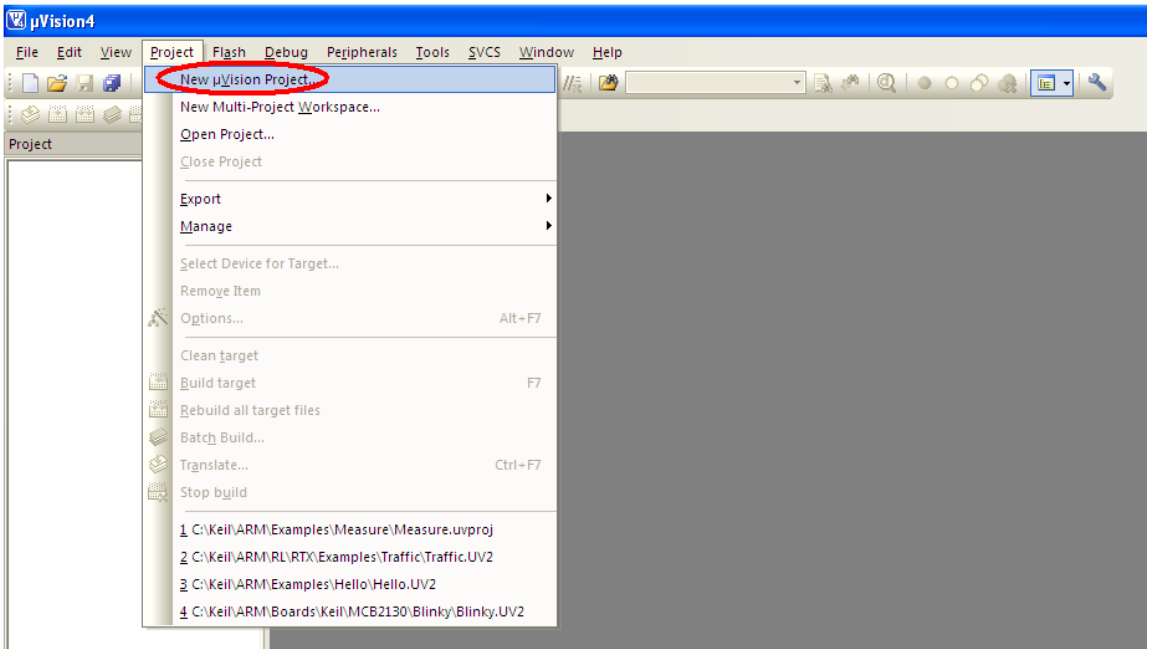
Creating a New Project

Below mentioned procedures will explains the steps required to setup a simple application and to generate HEX output.

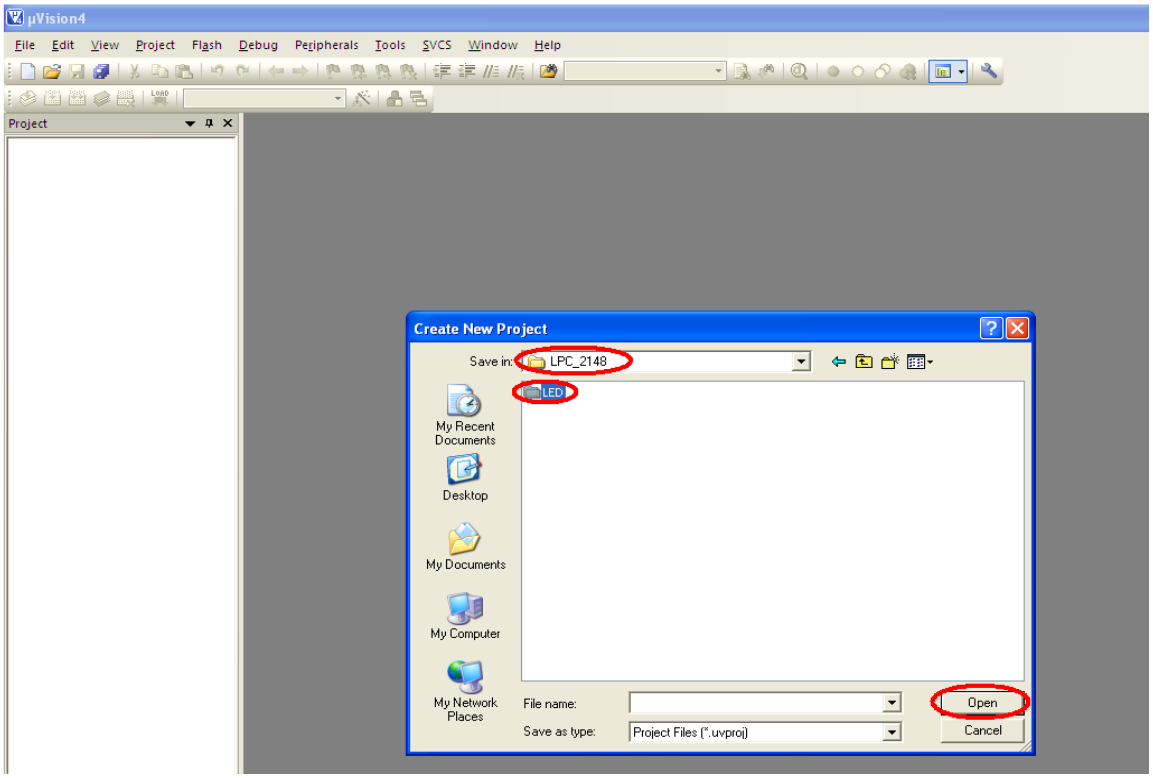
STEP 1: Go to "**Project**" and close the current project "**Close Project**".



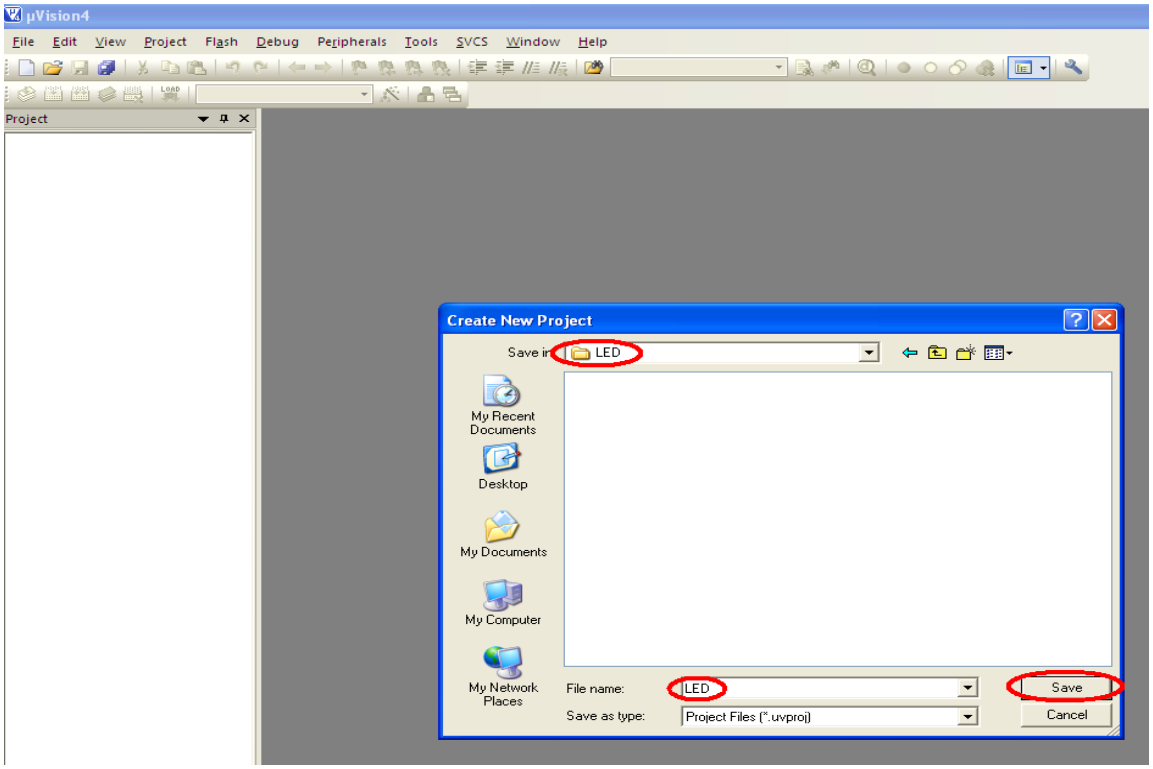
STEP 2: Go to the "**Project**" and click on "**New µvision Project**"



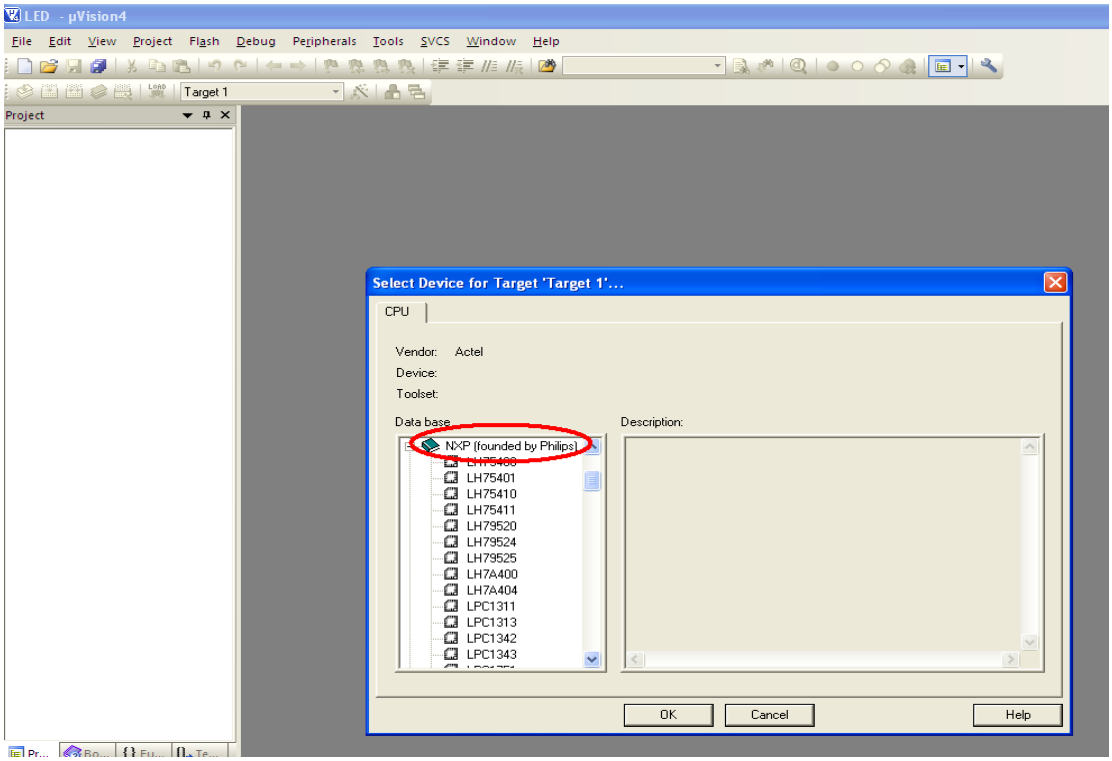
STEP 3: A small window will pop up with name "**Create New Project**" and can be create and select destination path.



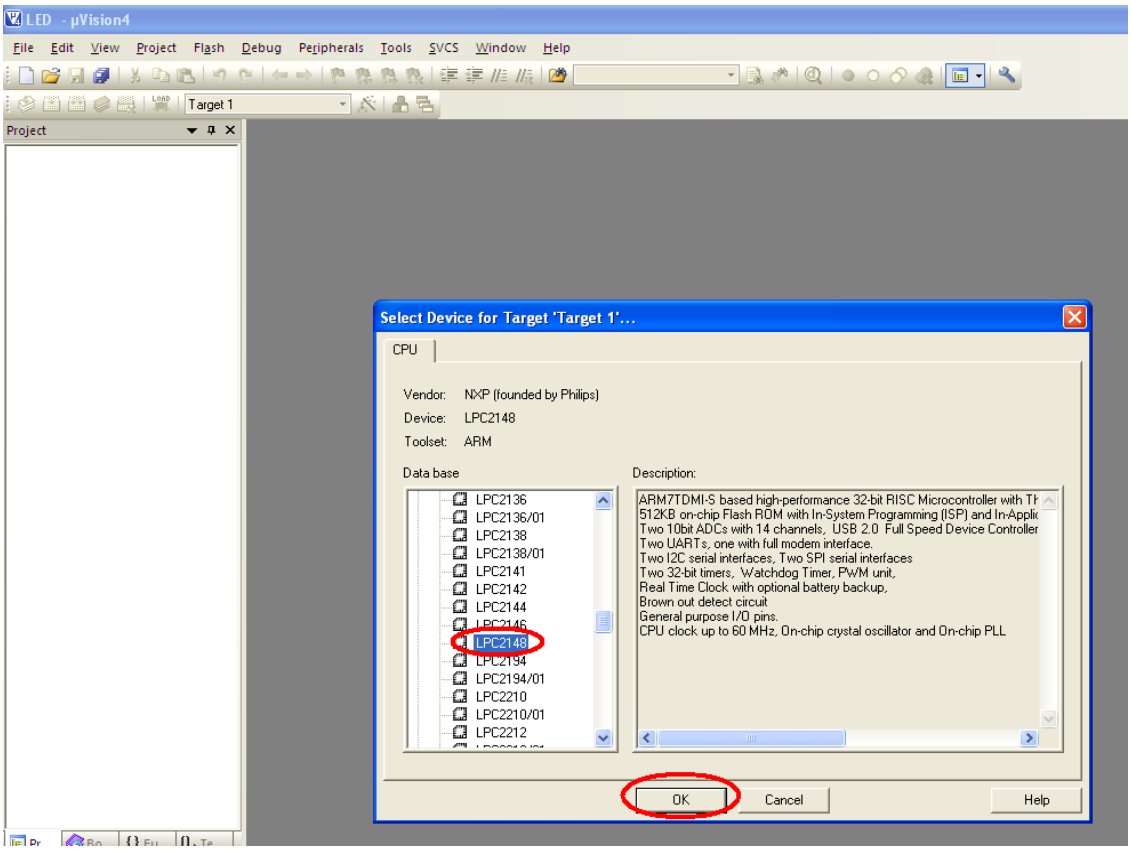
STEP 4: Create a folder and give a proper name that can be related to the Project.



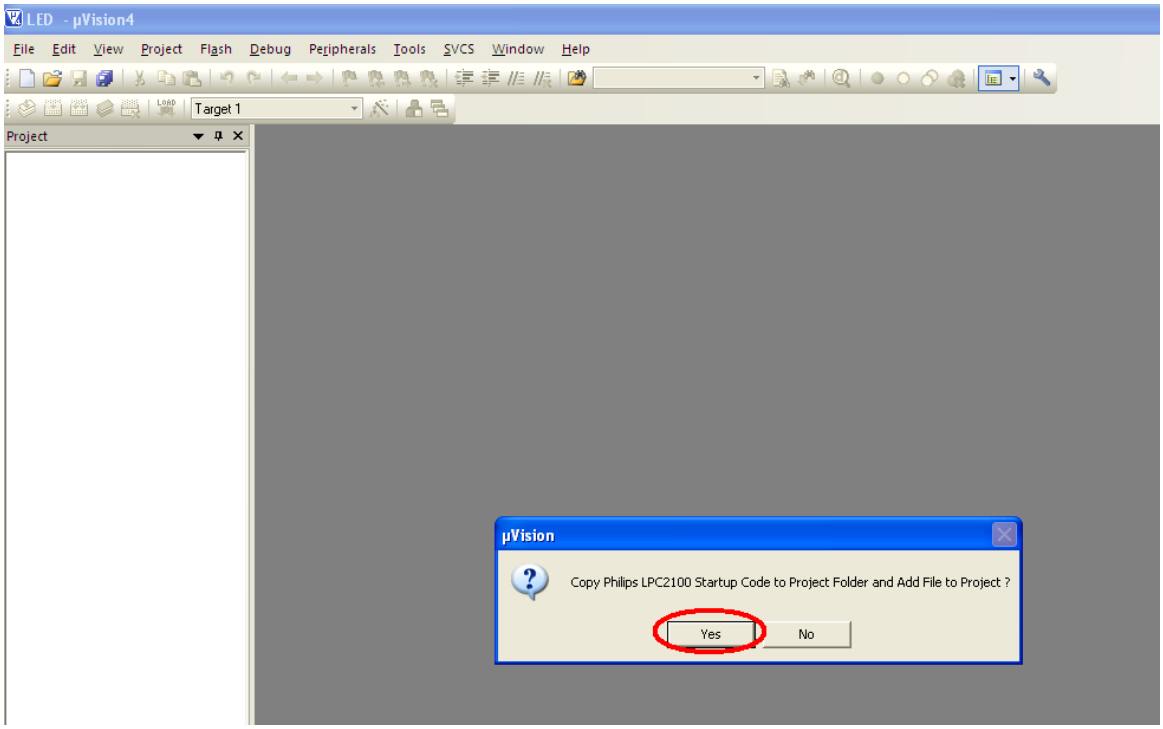
STEP 5: A small window will pop up with name "**Select Device for Target 'Target 1'**", and select the data base NXP founded by Philips.



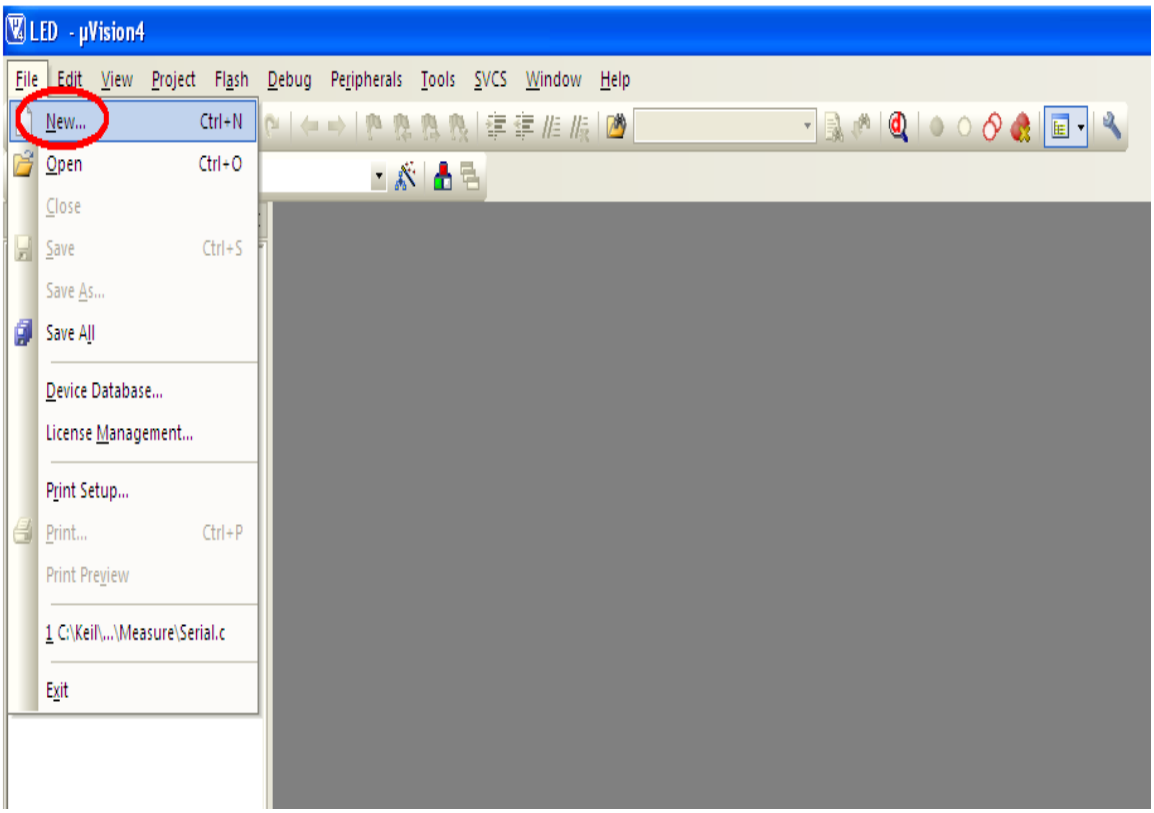
STEP 6: Within the NXP founded by Philips select **LPC2148.**



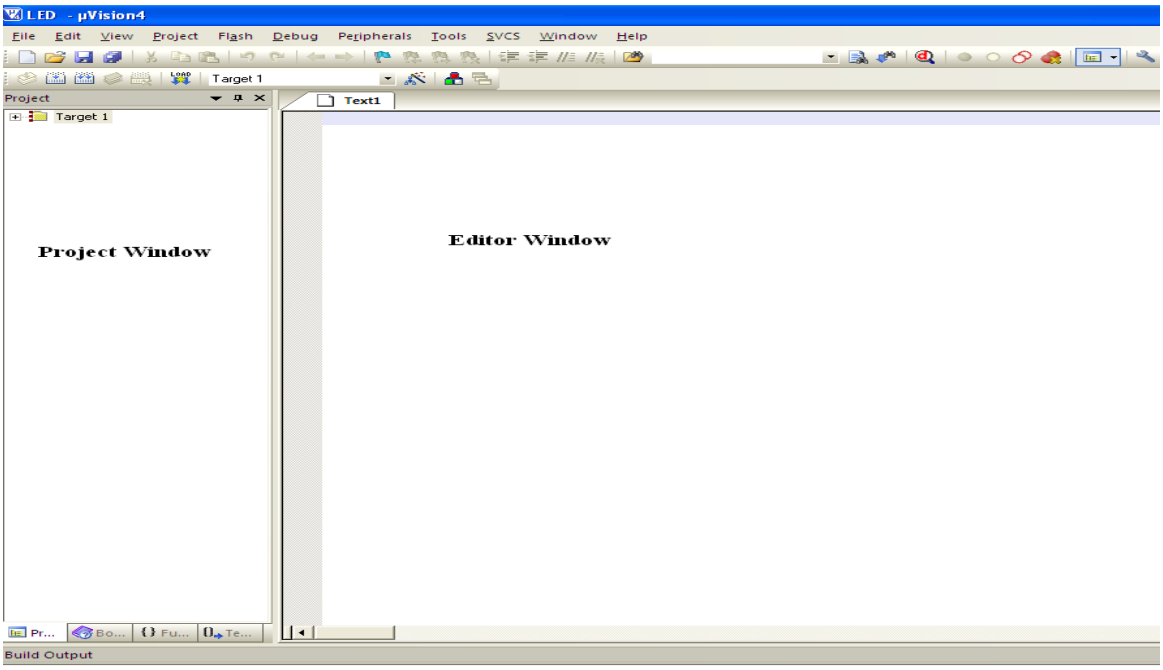
STEP 7: Add Startup file to the project by clicking "**Yes**".



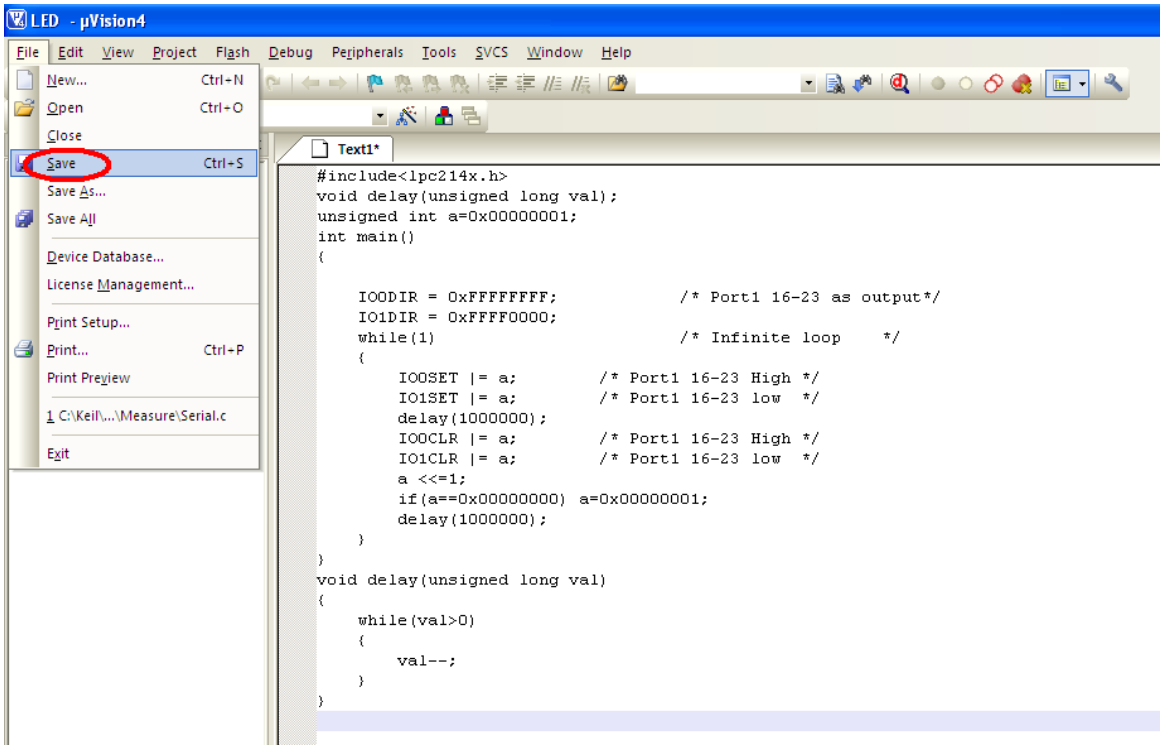
STEP 8: Next go to "**File**" and click "**New**".



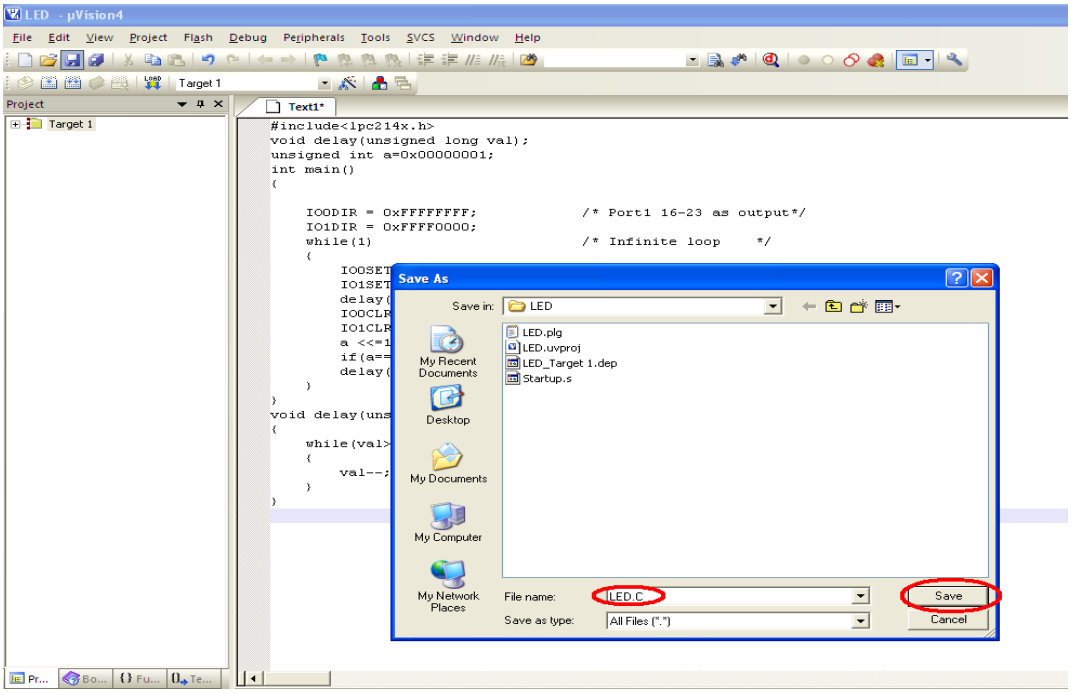
STEP 9: There are the main three windows are available in the keil IDE. First one is Project Workspace, second one is Editor Window and third one is Output Window.



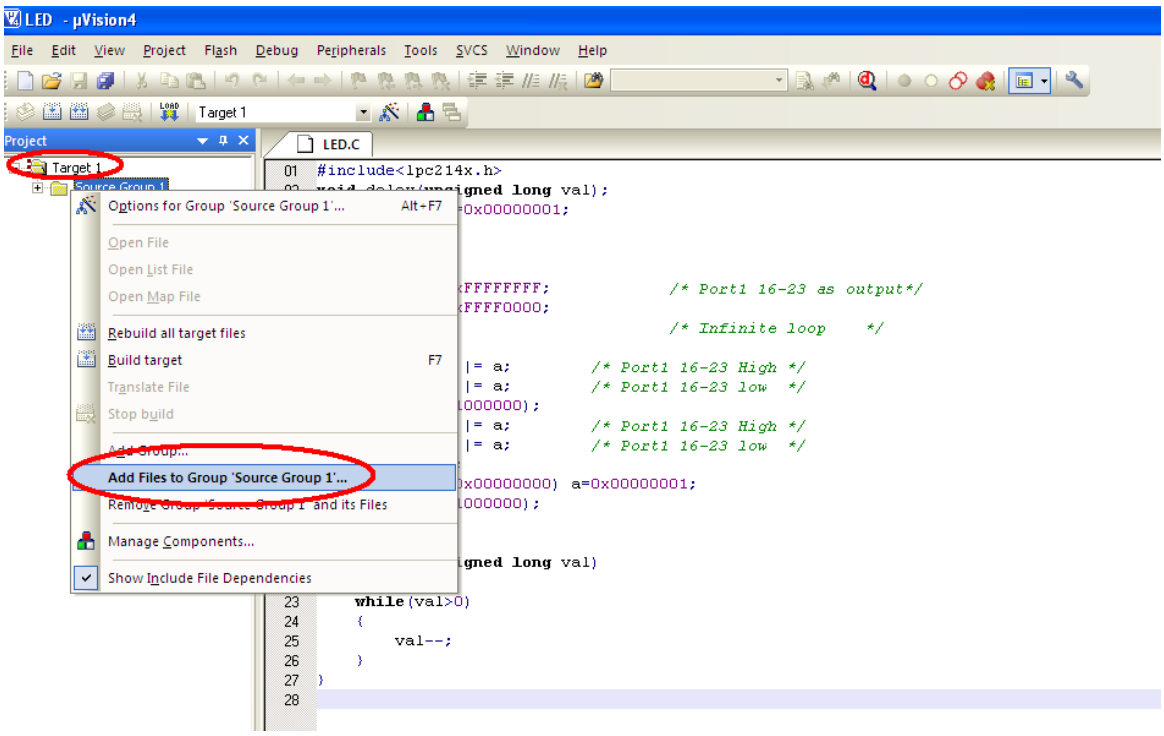
STEP 10: Can be start to write asm/c code on the editor window.



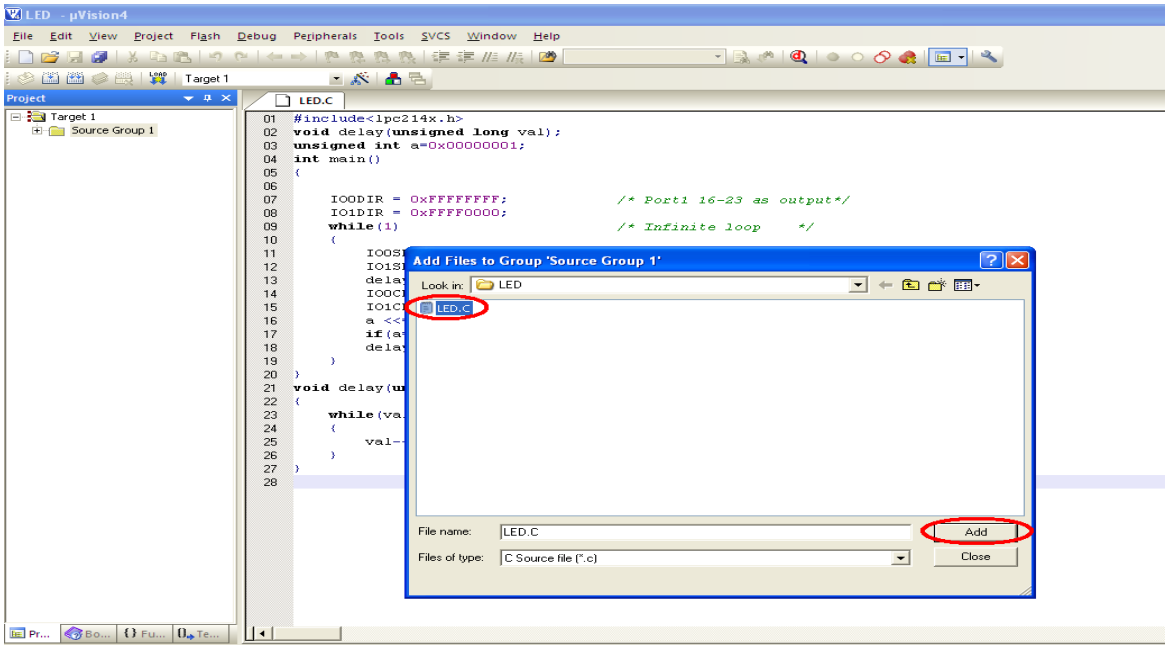
STEP 11: Can be save the file, if the program is in "**C**" save as "**filename.C**" or else save as "**filename.ASM**".



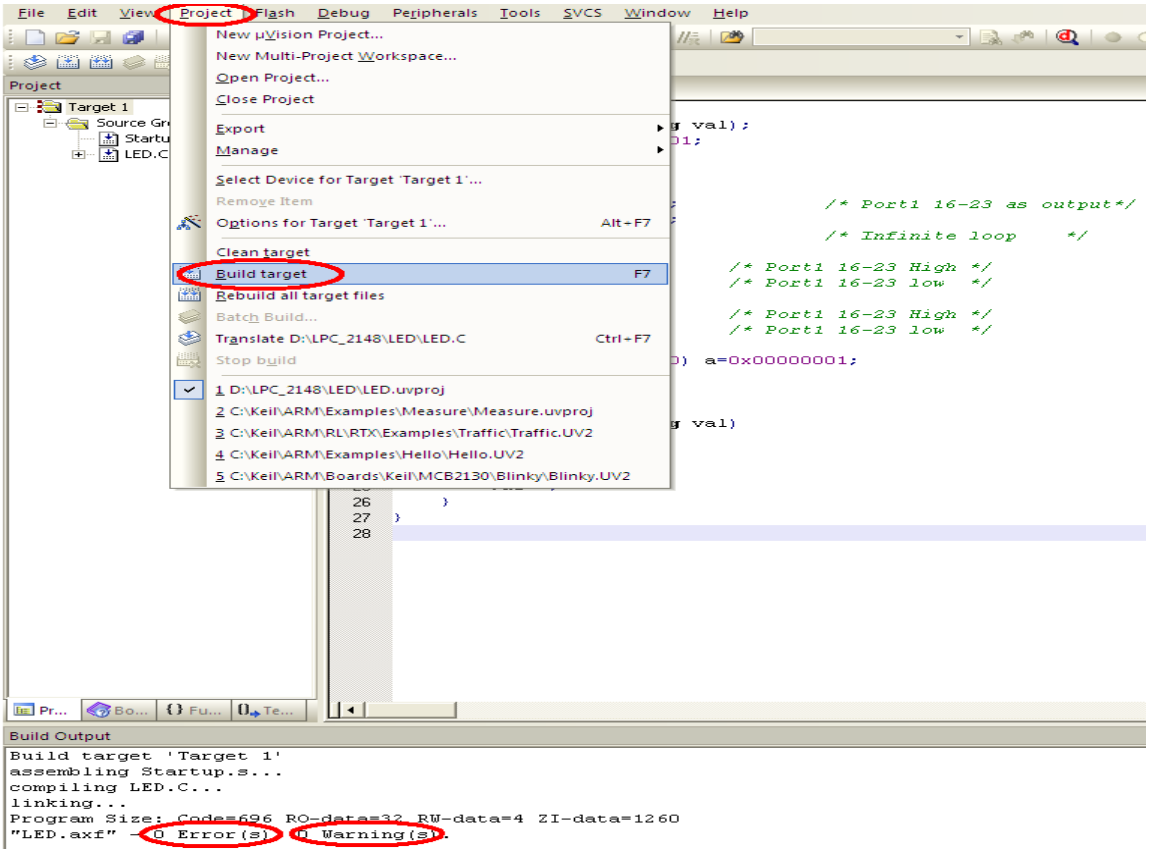
STEP 12: Add this source file to Group1, Go to the "**Project Workspace**" drag the +mark "**Target 1**" in that right click on "**Source Group1**" and click on "**Add Files to Group "Source Group1"**"".



STEP 13: A small window will pop up with name "**Add Files to Group Source Group1",** by default, the Files of type will be in **C source Files (*.C).** If the program is in C, have to select **C source Files (*.C**) or select **ASM Source file (*.s,*.src,*.a*).**

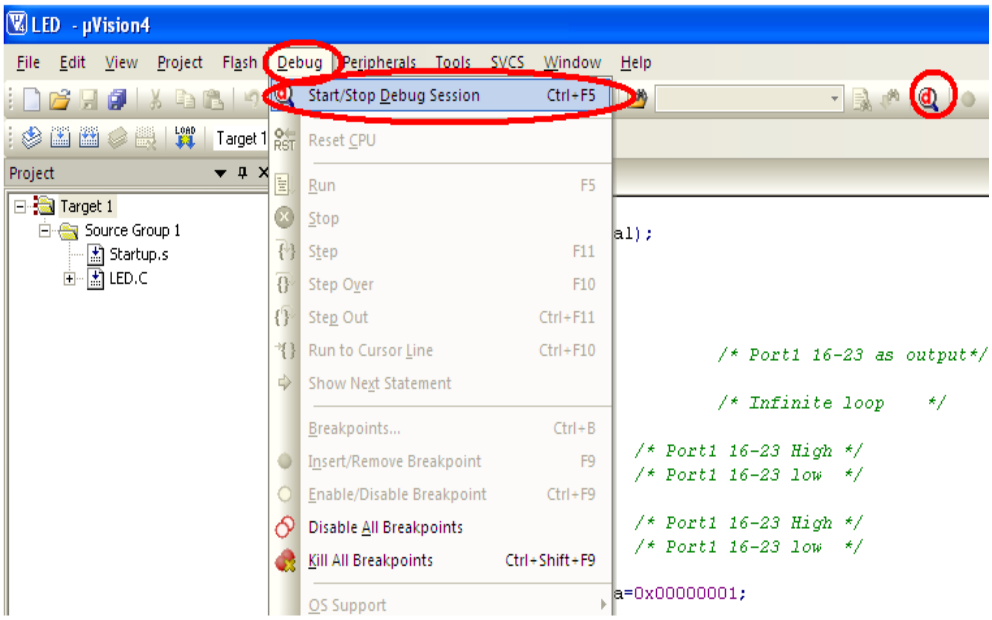


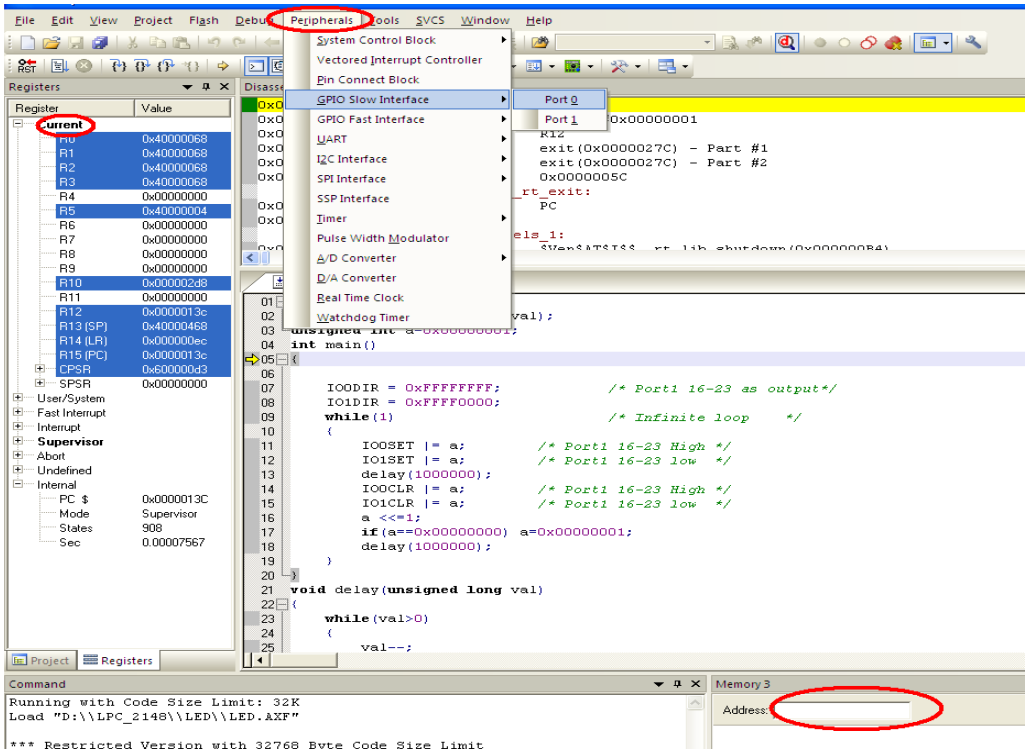
STEP 14: Then go to "**Project**" click on "**Build Target**" or **F7**. Output window will display related error and warning messages.



Simulation
Part:

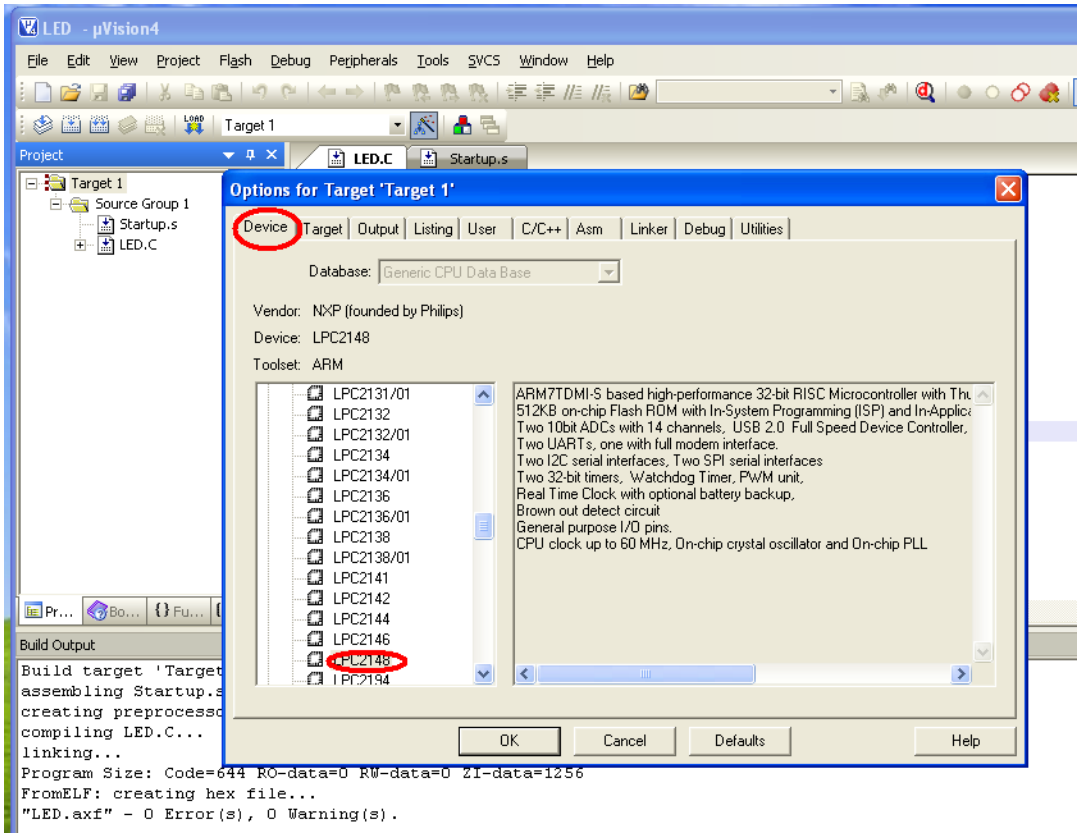
STEP 15: Go to "**Project**" menu, click on "**Rebuild all target Files**" and start **Debug**. From **View** menu can get **Memory Window** and from **Peripherals** can get I/O ports, Serial etc. For access internal memory type **i:0x_memory** location example: **i:0x30** and for external and program memory **x:0x_memory location**, **c:0x_memory location** respectively. From **Register** window we can edit and access the values also.



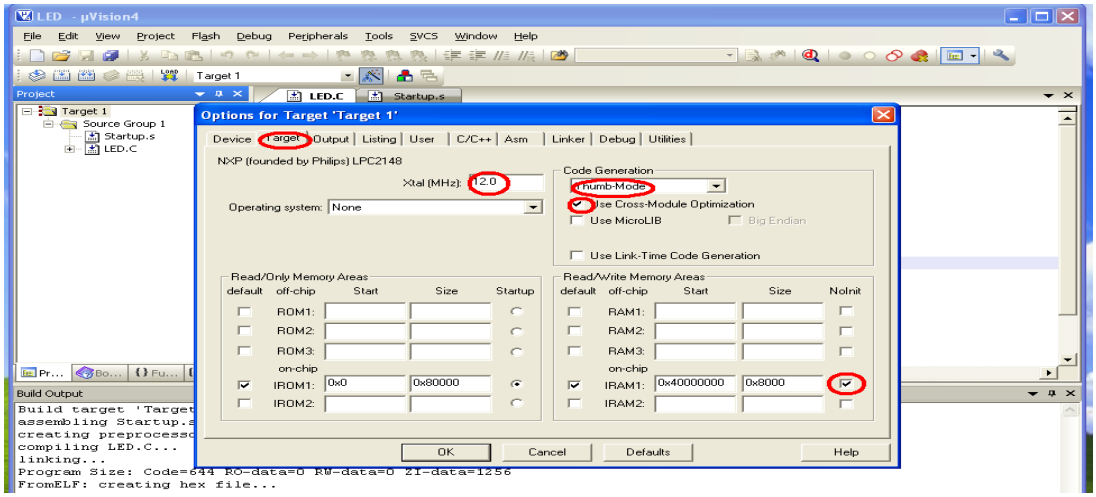


HEX file creation:

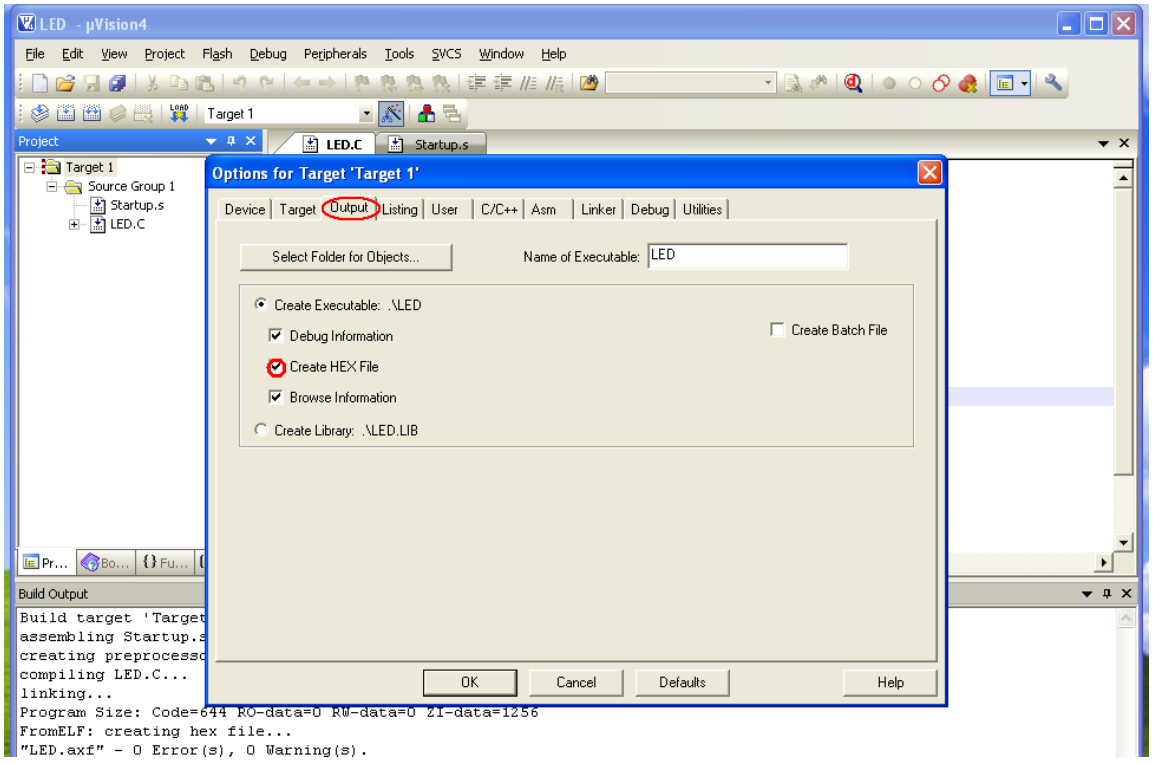
STEP 16: Follow the **STEP** up to **14**, then go to "**Project**" and click on "**Option for Group 'Source Group1**'". There a small window will open with name "**Option for Target 'Target1**'". In that window, go to first menu "Device", can be select LPC2148.



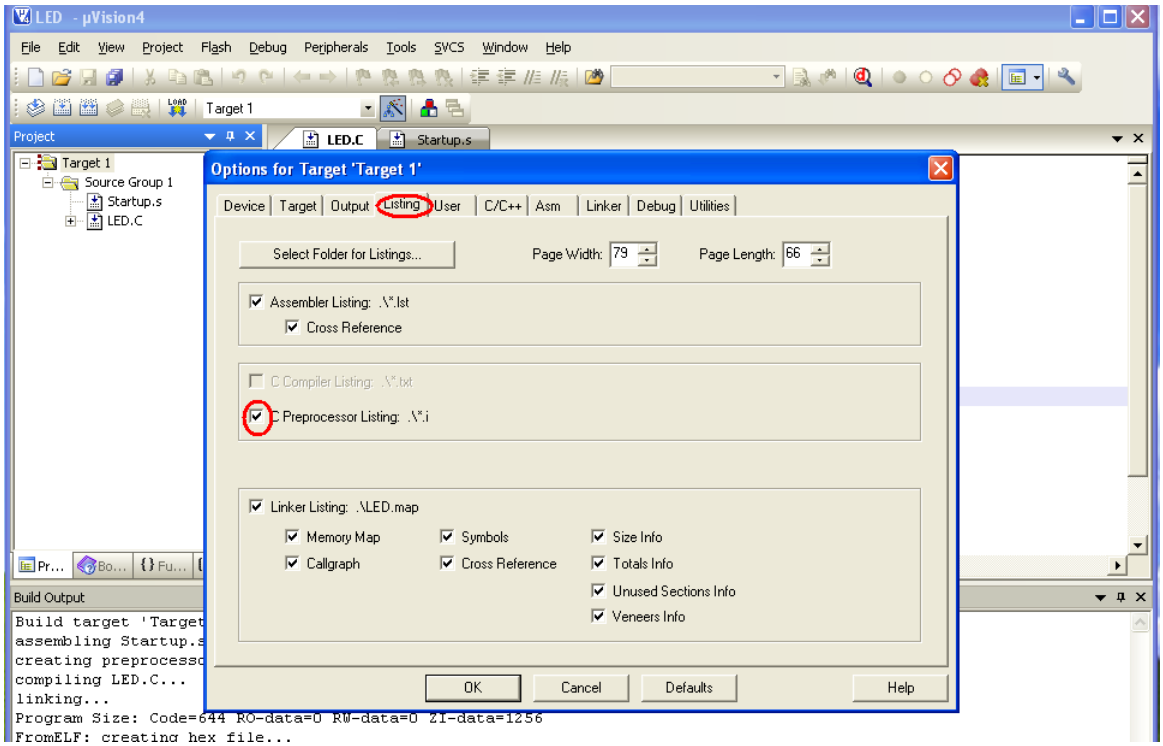
STEP 17: Next go to **Target** menu, set clock frequency as 12.0 MHz and select Thumb mode in the code generation selection box.



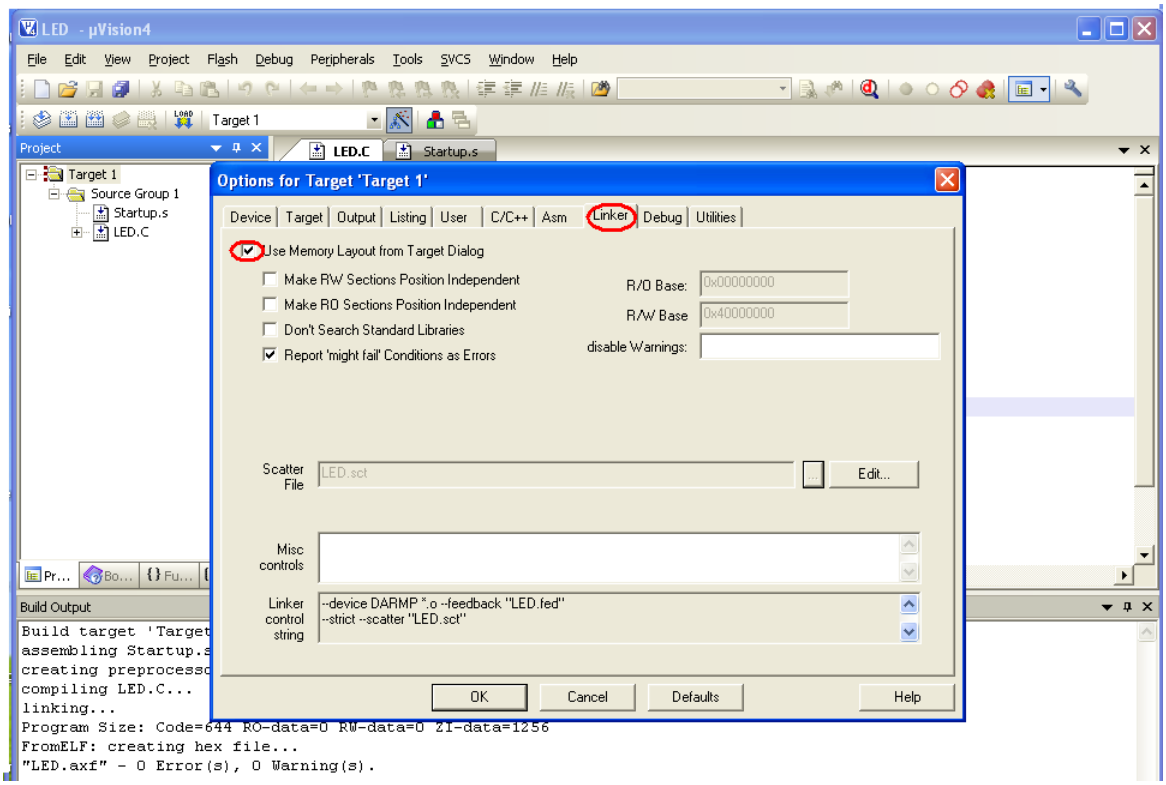
STEP 18: Then go to Output menu and click on create HEX file.



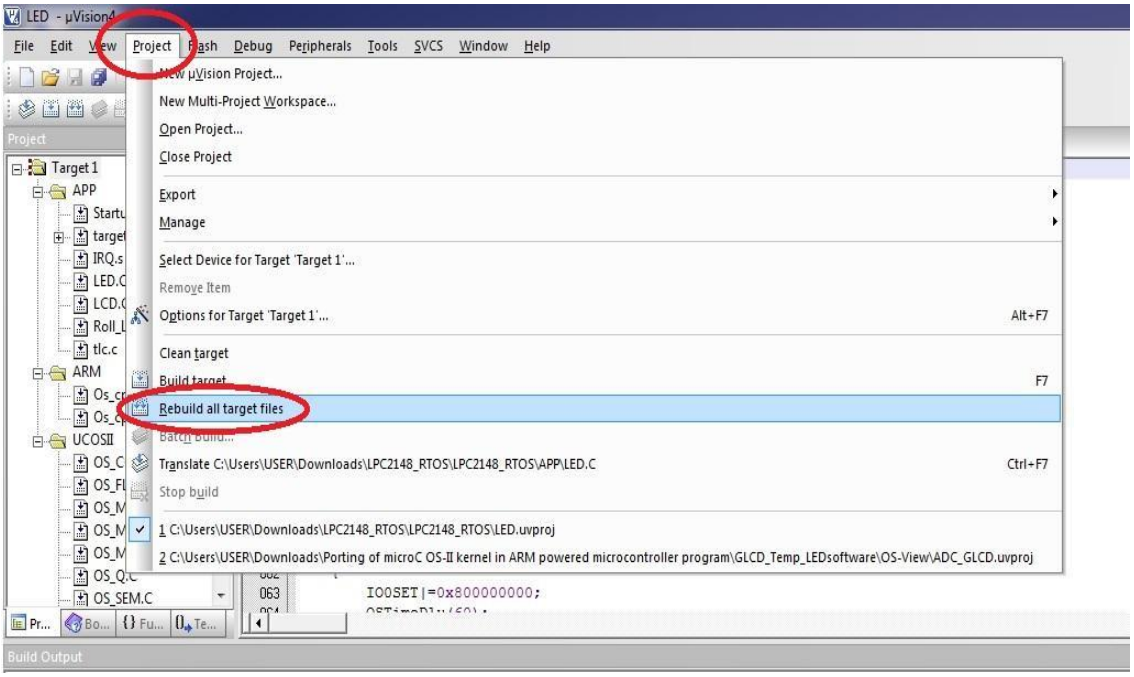
STEP 19: Then go to **Listing** menu and select C preprocessor Listing.



STEP 20: Finally in the **Linker menu,** click on use memory layout from target dialog and click ok.



STEP 21: For creating Hex file go to "**Project**" menu and click on "**Rebuild all target Files**"



FLASH MAGIC

Introduction:

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology. Flash Magic is Windows software from the Embedded Systems Academy that allows easy access to all the ISP features provided by the devices.

These features include:

- ✓ Erasing the Flash memory (individual blocks or the whole device)
- ✓ Programming the Flash memory
- ✓ Modifying the Boot Vector and Status Byte
- ✓ Reading Flash memory
- ✓ Performing a blank check on a section of Flash memory
- ✓ Reading the signature bytes
- ✓ Reading and writing the security bits
- ✓ Direct load of a new baud rate (high speed communications)
- ✓ Sending commands to place device in Bootloader mode

Flash Magic provides a clear and simple user interface to these features and more as described in the following sections. Under Windows, only one application may have access the COM Port at any one time, preventing other applications from using the COM Port. Flash Magic only obtains access to the selected COM Port when ISP operations are being performed. This means that other applications that need to use the COM Port, such as debugging tools, may be used while Flash Magic is loaded. Note that in this manual third party Compilers are listed alphabetically. No preferences are indicated or implied.

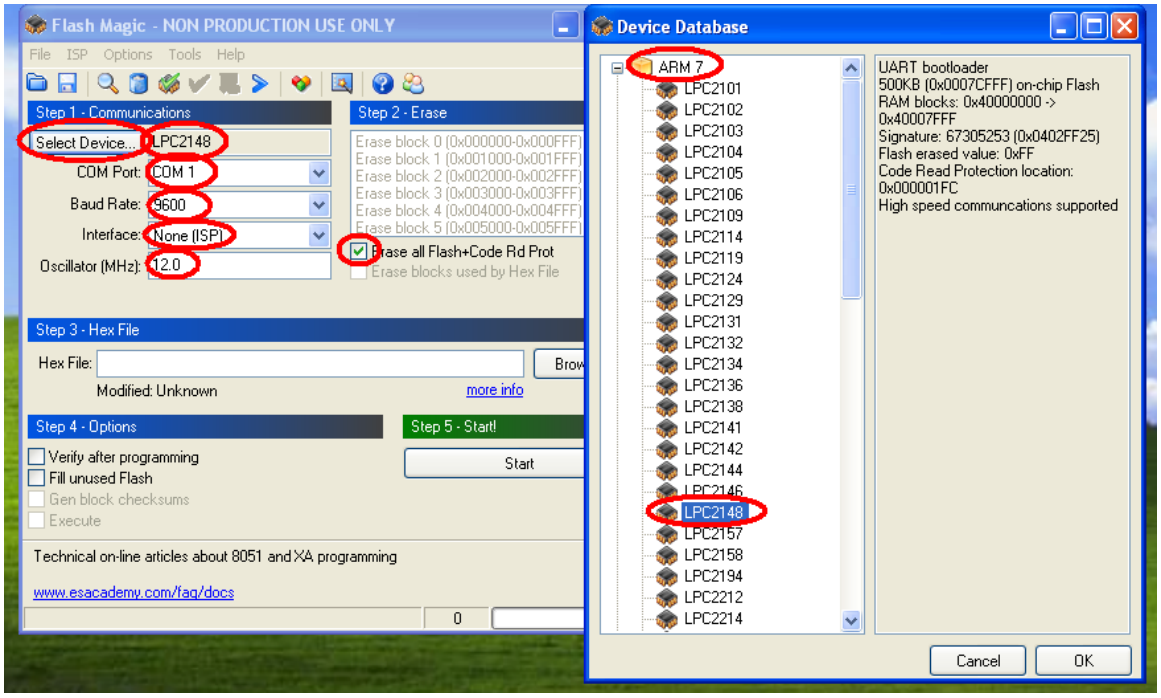
Installation Procedure for Flash Magic software:

Installation steps are given below:

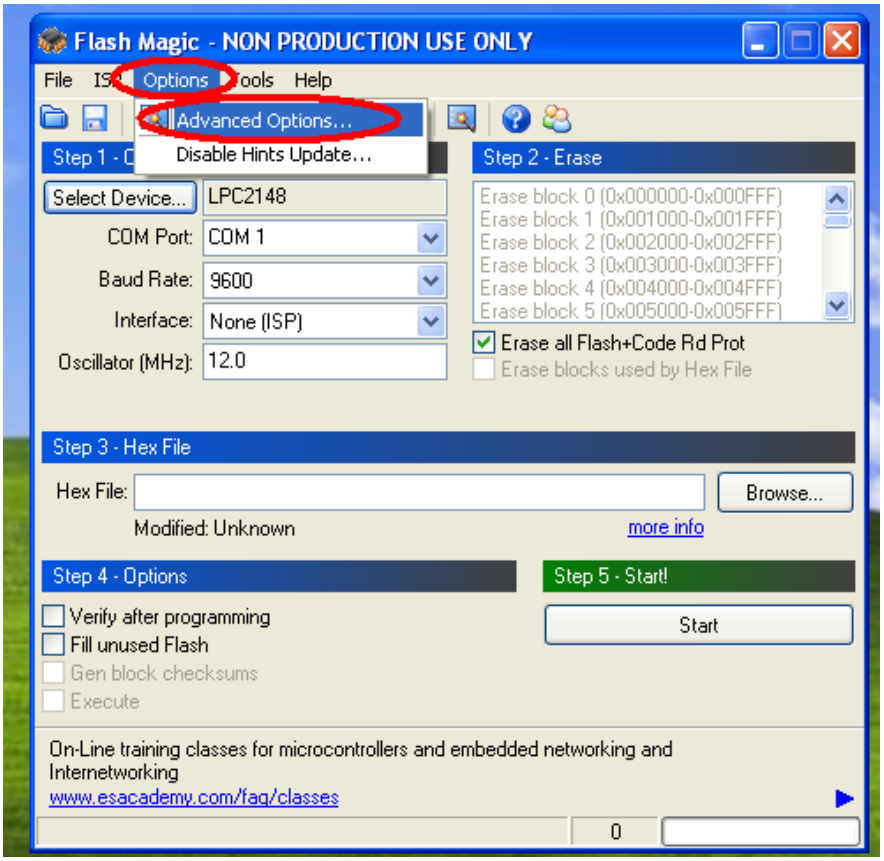
1. Double click on Flash Magic.exe.
2. Then click on Next.
3. Next accept the agreement and click on Next.
4. Select the destination folders and click on Next then Install.
5. Finally click on Finish.

Programming with communication port (COM1):

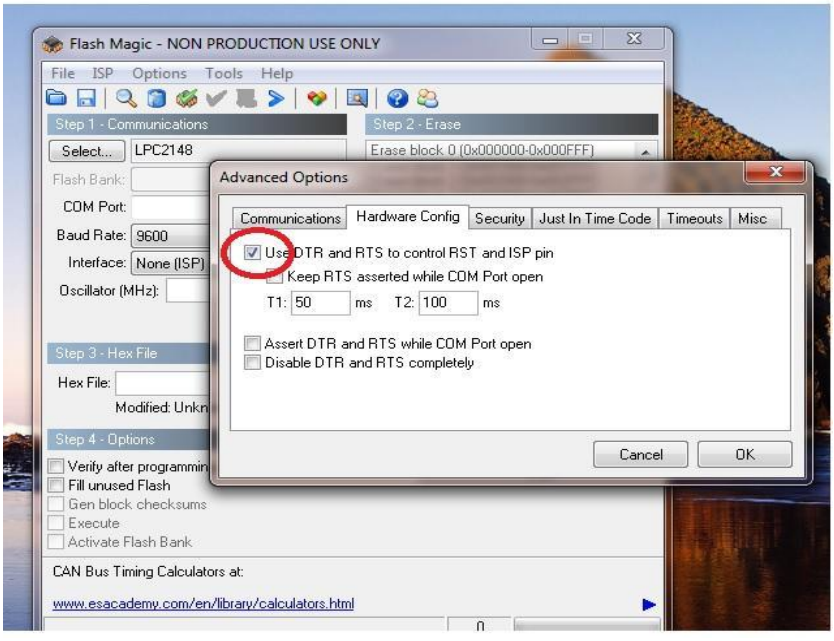
STEP 22: For programming with communication port, first select the device LPC2148 in ARM 7 category, COM port will be COM 1, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.



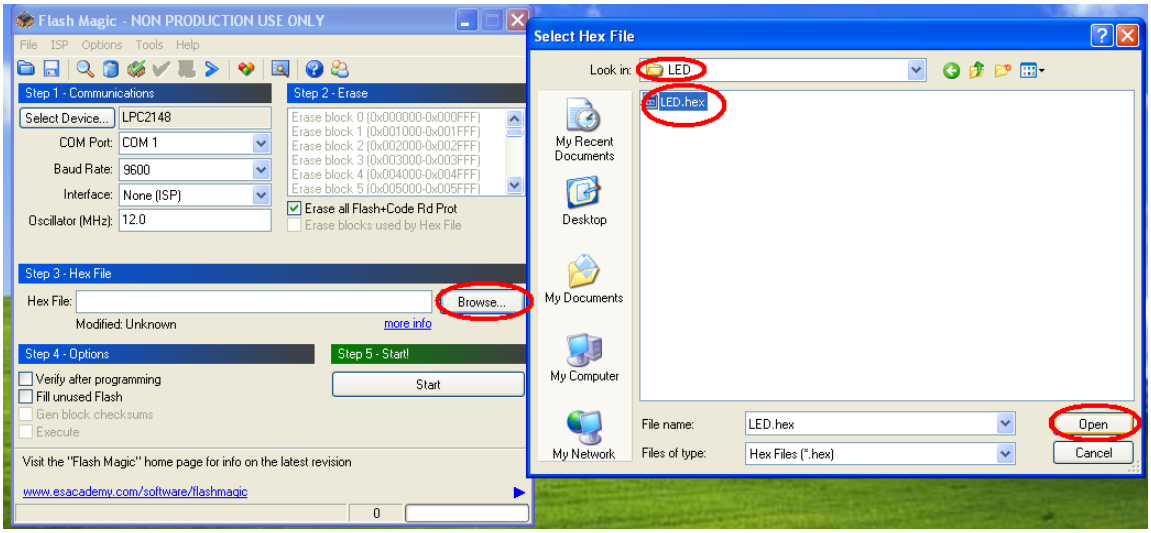
STEP 23: Under the menu **Options,** go to Advanced options.



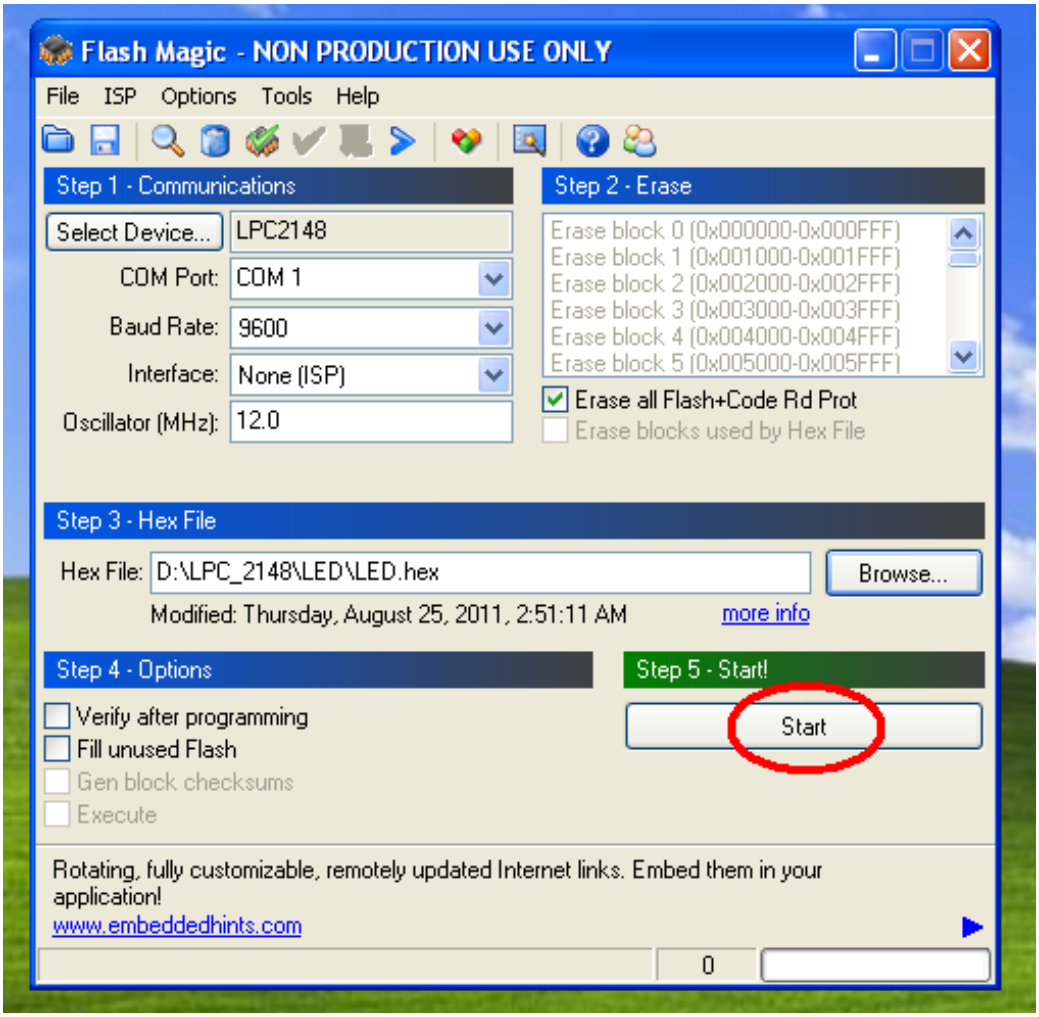
STEP 24: Under the menu Advanced options, go to Hardware configuration, click on the Use DTR and RTS to control RST and ISP pin.



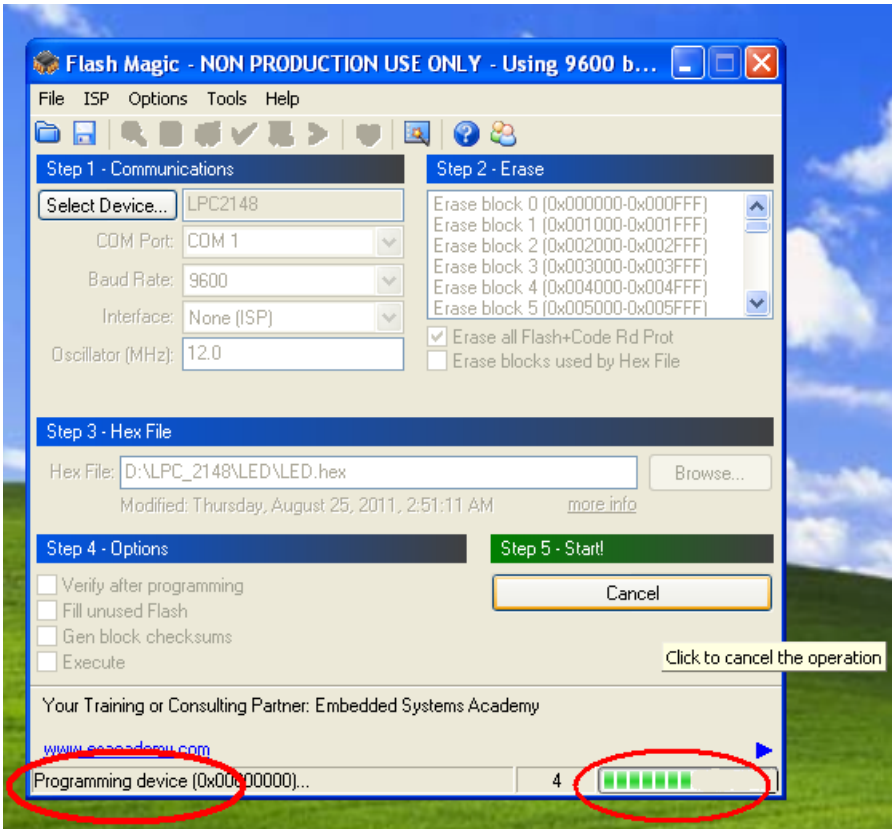
STEP 25: Next browse the path of hex file and select the file**.**



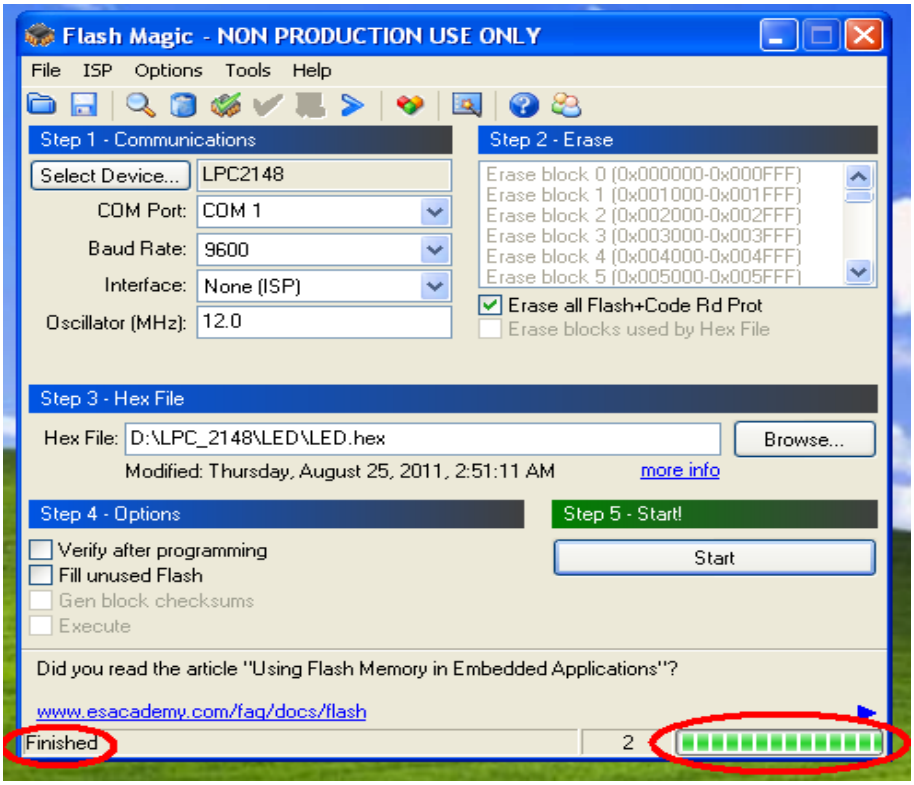
STEP 26: After selecting ISP mode on the Hardware Kit and click on start.



STEP 27: After the above steps device will start to program.



STEP 28: Finally can be see the finished indication and Reset the device into running mode.



Programming through USB:

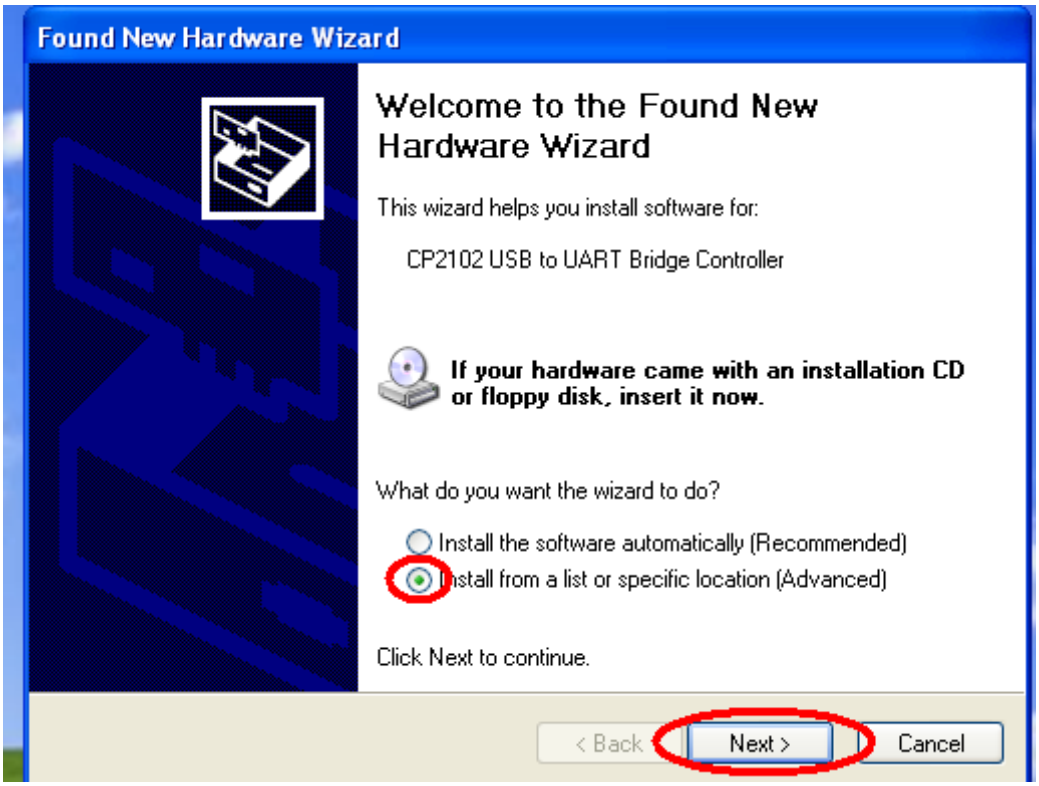
Installation Procedure for USB Cable

Driver: The installation steps are given

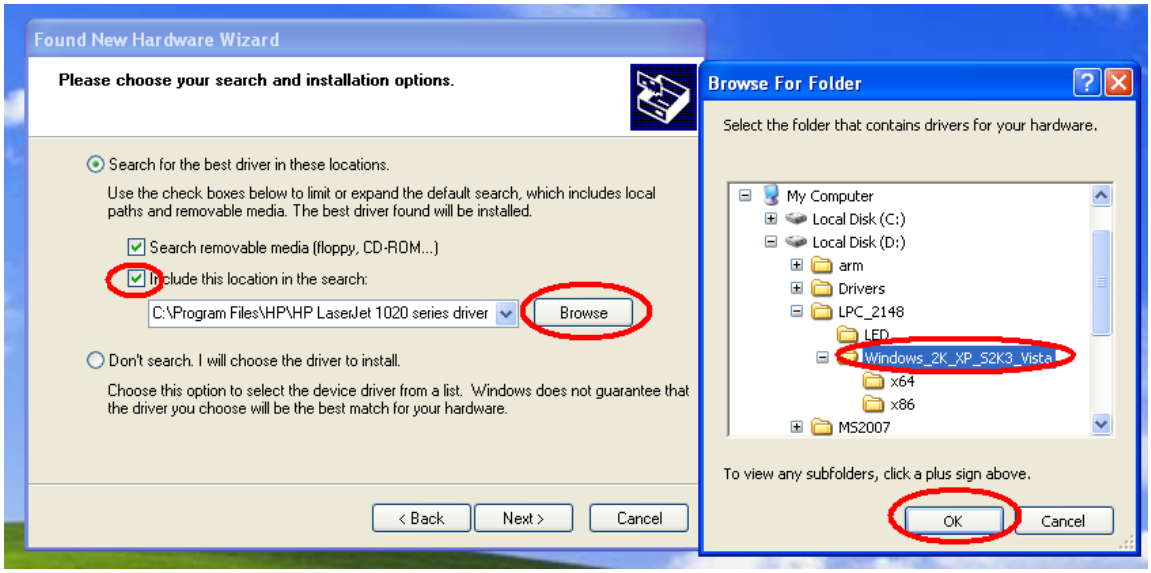
below:

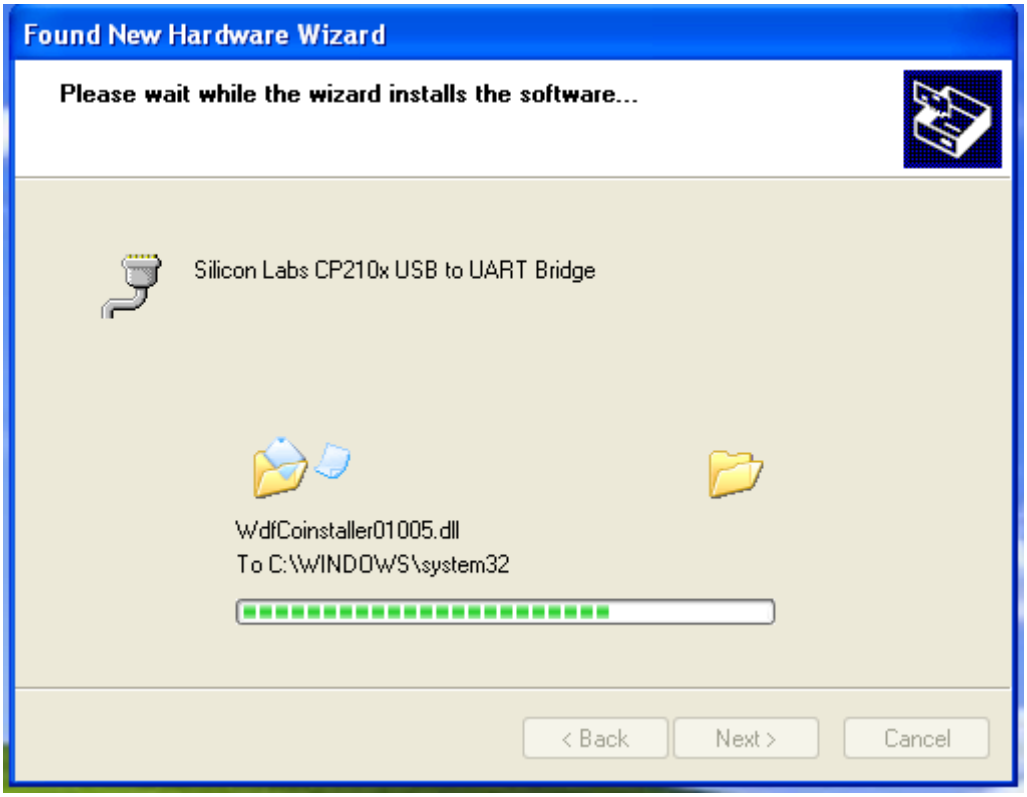
STEP 1: Connect the USB cable between ARM-7 LPC2148 Trainer Kit and PC. Connect the power supply to the trainer kit and power up. After can see a popup window with name "**Found New Hardware CP2102 USB to UART Bride Controller**".

STEP 2: Select on **Install from a list or specific location (Advanced)** and click "**Next**".

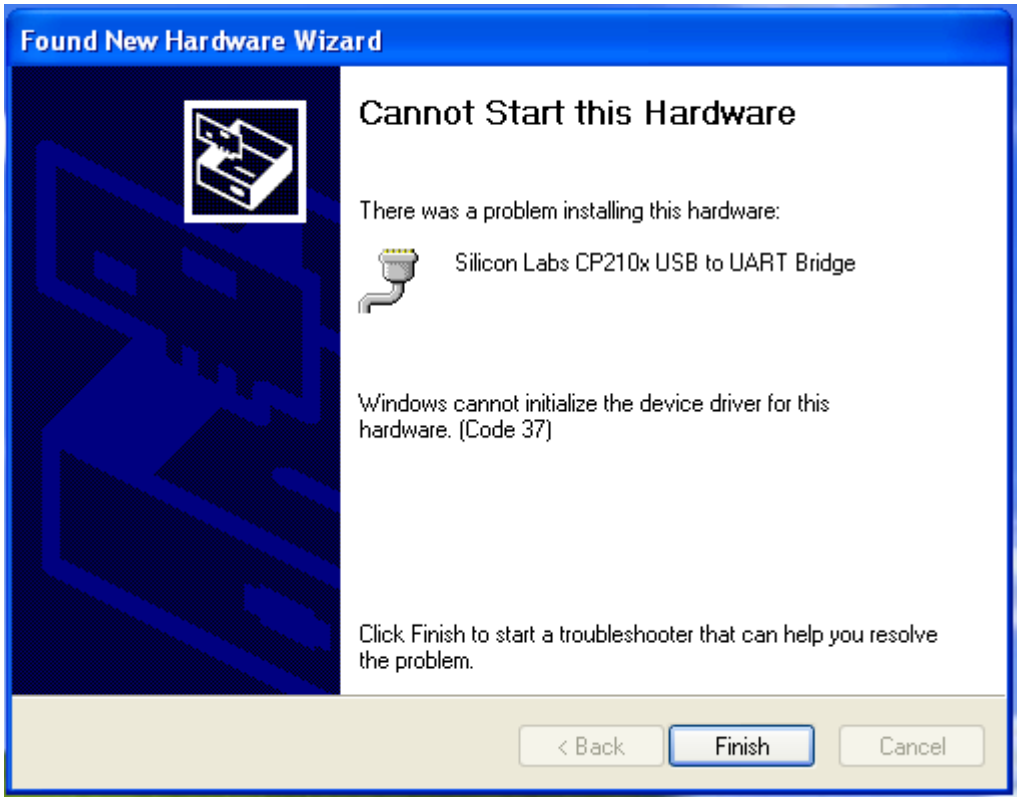


STEP 3: Browse for the driver file location and select the folder, click next.

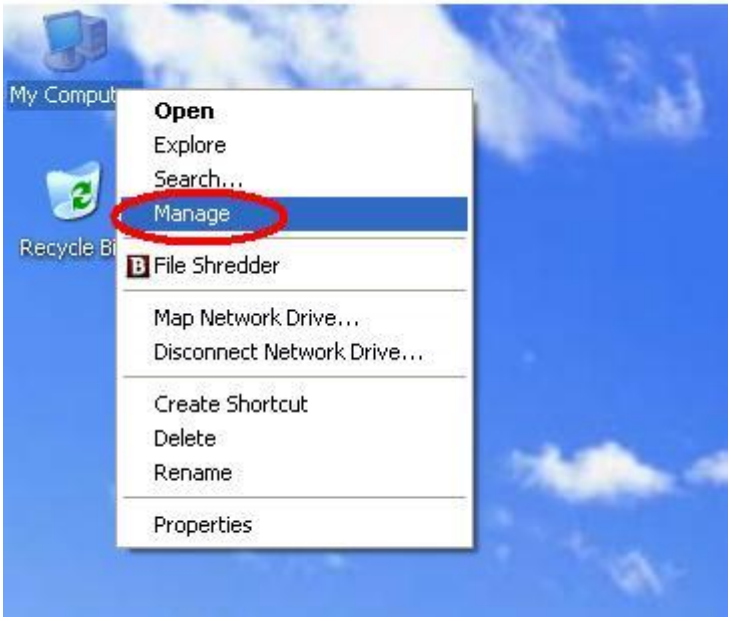




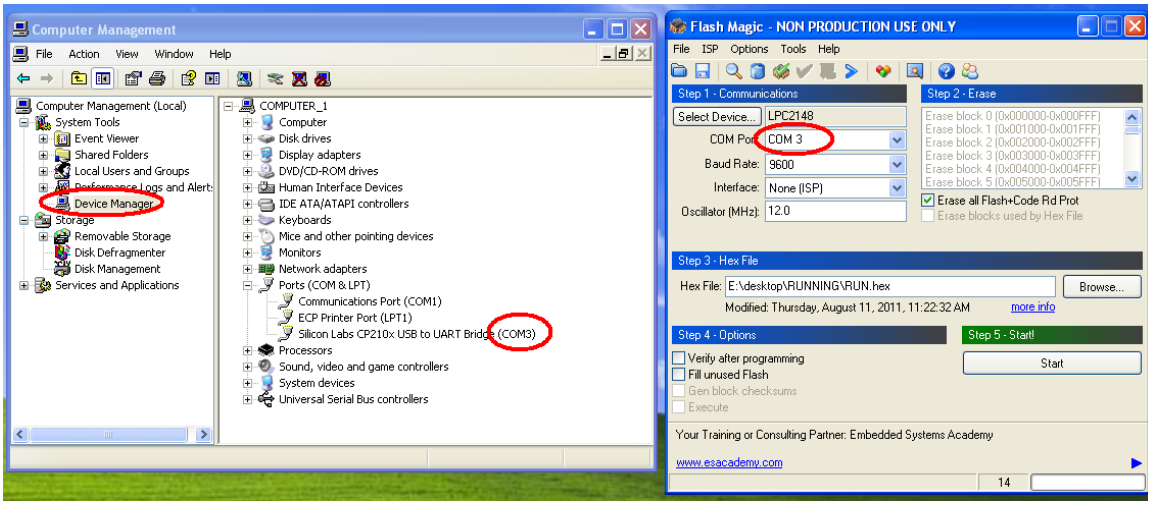
STEP 4: After that can see New Hardware wizard, and then click on finish.



STEP 5: Then right click on My Computer and select manage.



STEP 6: Then go to device manager, in that we have to select ports (COM and LPT) . There we have to find the COM port which has been selected for USB cable. The same COM has to be select for Flash Magic.

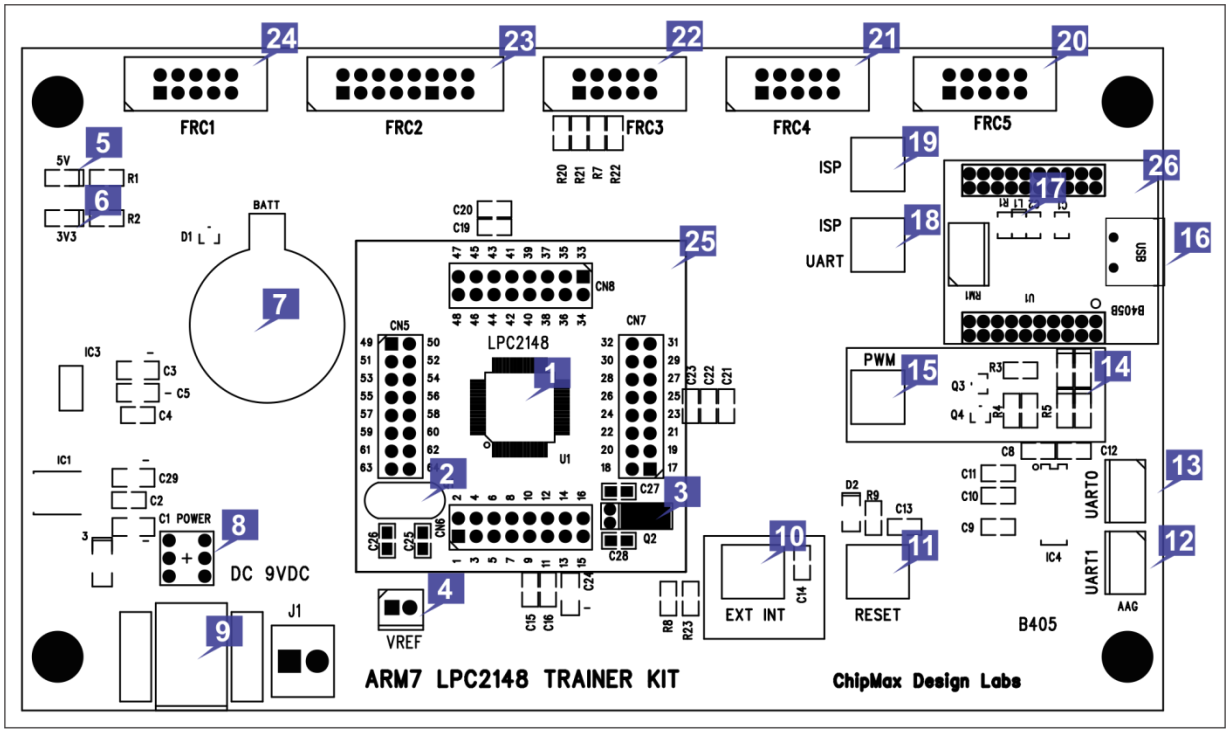


STEP 7: Repeat the **STEP** from 22 to 28

ARM 7 LPC2148 TRAINER KIT

Introduction:

Thank you for using ARM 7 LPC2148 Trainer Kit designed for educational training purpose and the embedded device from NXP. This document is a User's guide which describes the complete hardware design of the ARM 7 LPC2148 Trainer Kit.



1. The device LPC2148 from NXP.
2. 12.00 MHz Crystal Oscillator Device speed.
3. 32.768 KHz crystal oscillator for RTC clock.
4. VREF jumper setting for ADC external Voltage reference.
5. 5V Red LED indication.
6. 3V3 Green LED indication.
7. Battery for RTC backup running.
8. Power ON/OFF switch.
9. 9VDC input Power Jack.
10. Input switch for external interrupt.

    Optional: can be use for ISP manual Programming Mode.
11. Input switch for Reset the device.
12. ART1 connectivity.

13. UART0 connectivity.

    Optional: Can be use for ISP programming mode.

14. LEDs for PWM output.
15. PWM enable switch.
16. USB connector for ISP programming and Serial port connectivity.
17. LED indication for to confirm the USB connection has been established.
18. Switch selection for ISP/UART. The selection will be for ISP if the switch has been pressed else for UART.
19. Switch for to enable the ISP.
20. I/O port 10 pin FRC connector (FRC5).
21. I/O port 10 pin FRC connector (FRC4).
22. I/O port 10 pin FRC connector (FRC3).
23. I/O port 16 pin FRC connector (FRC2).
24. I/O port 10 pin FRC connector (FRC1).
25. LPC2148 Device Daughter card.
26. USB ISP programmer daughter card.

Note: For In System Programming (ISP), the switch which has been mentioned in the description line 18 and 19 has to be pressed.

Features:

1. Device daughter card, easy and flexible to upgrade the device.
2. Four 10 pin individual digital or analog I/O ports are available.
3. One16 pin digital I/O port is available.
4. Inbuilt LEDs are available for PWM output.
5. Inbuilt push to on switch for Reset.
6. Inbuilt push to on switch for External Interrupt.
7. USB ISP programmer inbuilt.
8. On board Serial to USB bridge is available

Device Daughter Board Details:

CN6	
PIN NO:	**DESCRIPTION**
1	P0.21/PWM5/AD1.6/CAP1.3
2	P0.22/AD1.7/CAP0.0/MAT0.0
3	RTCX1
4	P1.19/TRACEPKT3
5	RTCX2
6	VSS
7	VDDA
8	P1.18/TRACEPKT2
9	P0.25/AD0.4/AOUT
10	D+
11	D-
12	P1.17/TRACEPKT1
13	P0.28/AD0.1/CAP0.2/MAT0.2
14	P0.29/AD0.2/CAP0.3/MAT0.3
15	P0.30/AD0.3/CAP0.0/EINT3
16	P1.16/TRACEPKT0

CN8	
PIN NO:	**DESCRIPTION**
33	P0.8/TXD1/PWM4/AD1.1
34	P0.9/RXD1/PWM6/EINT3
35	P0.10/RTS1/CAP1.0/AD1.2
36	P1.23/PIPESTAT2
37	P0.11/CTS1/CAP1.1/SCL1
38	P0.12/DSR1/MAT1.0/AD1.3
39	P0.13/DTR1/MAT1.1/AD1.4
40	P1.22/PIPESTAT1
41	P0.14/DCD1/EINT1/SDA1
42	VSS
43	VDD
44	P1.21/PIPESTAT0
45	P0.15/RI1/EINT2/AD1.5
46	P0.16/EINT0/MAT0.2/CAP0.2
47	P0.17/CAP1.2/SCK1/MAT1.2
48	P1.20/TRACESYNC

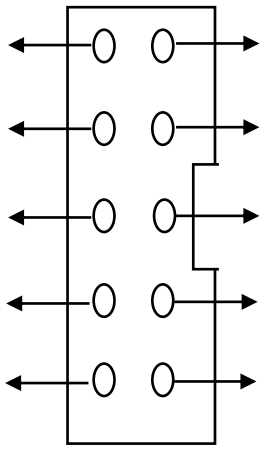
CN7	
PIN NO:	**DESCRIPTION**
17	P0.31/UP_LED/CONNECT
18	VSS
19	P0.0/TXD0/PWM1
20	P1.31/TRST
21	P0.1/RXD0/PWM3/EINT0
22	P0.2/SCL0/CAP0.0
23	VDD
24	P1.26/RTCK
25	VSS
26	P0.3/SDA0/MAT0.0/EINT1
27	P0.4/SCK0/CAP0.1/AD0.6
28	P1.25/EXTIN0
29	P0.5/MISO0/MAT0.1/AD0.7
30	P0.6/MOSI0/CAP0.2/AD1.0
31	P0.7/SSEL0/PWM2/EINT2
32	P1.24/TRACECLK

CN5	
PIN NO:	**DESCRIPTION**
49	VBAT
50	VSS
51	VDD
52	P1.30/TMS
53	P0.18/CAP1.3/MISO1/MAT1.3
54	P0.19/CAP1.2/MOSI1/MAT1.2
55	P0.20/EINT3/SSEL1/MAT1.3
56	P1.29/TCK
57	RESET
58	P0.23/VBUS
59	VSSA
60	P1.28/TDI
61	XTAL2
62	XTAL1
63	VREF
64	P1.27/TDO

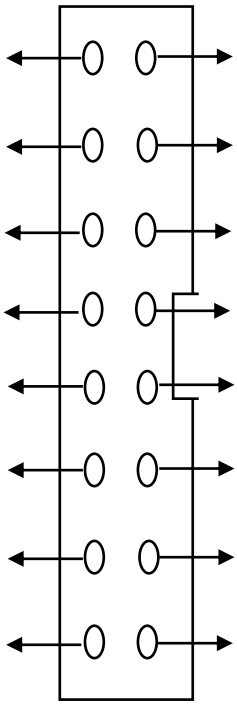
FRC PIN DETAILS:

FRC 1	
PIN NO:	**DESCRIPTION**
1	P0.16
2	P0.17
3	P0.18
4	P0.19
5	P0.20
6	P0.21
7	P0.22
8	P0.23
9	+5V
10	GND

FRC 5	
PIN NO:	**DESCRIPTION**
1	P0.25
2	P0.28
3	P0.29
4	P0.30
5	P0.31
6	NA
7	RS232 TX1
8	RS232 RX1
9	+5V
10	GND

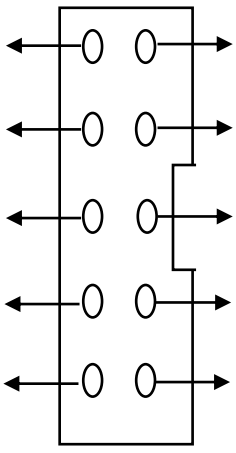


FRC 2	
PIN NO:	**DESCRIPTION**
1	P0.8
2	P0.9
3	P0.10
4	P0.11
5	P1.16
6	P1.17
7	P1.18
8	P1.19
9	P1.20
10	P1.21
11	P1.22
12	P1.23
13	+5V
14	NA
15	+3V3
16	GND



FRC 3	
PIN NO:	**DESCRIPTION**
1	P0.3(SDA0)
2	P0.2(SCL0)
3	P0.14(SDA1)
4	P0.11(SCL1)
5	P0.16 ENT1
6	P0.15 ENT2
7	RS232 TX0
8	RS232 RX0
9	+5V
10	GND

FRC 4	
PIN NO:	**DESCRIPTION**
1	P0.0
2	P0.1
3	P0.2
4	P0.3
5	P0.4
6	P0.5
7	P0.6
8	P0.7
9	+5V
10	GND



INTERFACE CARD DETAILS:

LED MODULE	
PIN NO	DESCRIPTION
1	Led1
2	Led2
3	Led3
4	Led4
5	Led5
6	Led6
7	Led7
8	Led8
9	+5V
10	GND

MATRIX KEYPAD	
PIN NO	DESCRIPTION
1	Row1
2	Row2
3	Row3
4	Row4
5	Column1
6	Column2
7	Column3
8	Column4
9	+5V
10	GND

STEPPER MOTOR	
PIN NO	DESCRIPTION
1	Input1
2	Input2
3	Input3
4	Input4
5	NA
6	NA
7	NA
8	NA
9	+5V
10	GND

EEPROM	
PIN NO	DESCRIPTION
1	SDA
2	SCL
3	NA
4	NA
5	NA
6	NA
7	NA
8	NA
9	+5V
10	GND

ADC / TEMPERATURE	
PIN NO	DESCRIPTION
1	NA
2	AD01
3	NA
4	NA
5	NA
6	NA
7	NA
8	NA
9	+5V
10	GND

ZIGBEE	
PIN NO	DESCRIPTION
1	NA
2	NA
3	NA
4	NA
5	NA
6	NA
7	RXD
8	TXD
9	+5V
10	GND

LCD	
PIN NO	DESCRIPTION
1	NA
2	RS
3	R/W
4	EN
5	D0
6	D1
7	D2
8	D3
9	D4
10	D5
11	D6
12	D7
13	+5V
14	-5V
15	+3V3
16	GND

DAC	
PIN NO	DESCRIPTION
1	NA
2	NA
3	NA
4	NA
5	A8
6	A7
7	A6
8	A5
9	A4
10	A3
11	A2
12	A1
13	+5V
14	-5V
15	NA
16	GND

TOGGLE SWITCH	
PIN NO	DESCRIPTION
1	SW1
2	SW2
3	SW3
4	SW4
5	SW5
6	SW6
7	SW7
8	SW8
9	+5V
10	GND

DC MOTOR	
PIN NO	DESCRIPTION
1	INPUT1
2	ENABLE
3	NA
4	INPUT2
5	NA
6	NA
7	NA
8	NA
9	+5V
10	GND

SEVEN SEGMENT	
PIN NO	DESCRIPTION
1	SHIFT CLK
2	DATA IN
3	LATCH CLK
4	NA
5	NA
6	NA
7	NA
8	NA
9	+5V
10	GND

FRC CONNECTION DETAILS:

S.No	Experiment	FRC1	FRC2	FRC3	FRC4	FRC5
1	ADC		LCD			ADC (Change mode)
2	DAC		DAC			
3	LED				LED	
4	PWM				LED	ADC (Change mode)
5	RTC		LCD			
6	EPROM	Keypad	LCD	EPROM		
7	Interrupt	LED				
8	Stepper Motor	Stepper Motor				
9	Temperature Sensor		LCD			ADC (Change mode)
10	Zigbee Transmitter	Keypad	LCD	Zigbee		
11	Zigbee Receiver	LED		Zigbee		
12	Seven Segment Display	Seven Segment				
13	Switch with LED	Toogle Switch			LED	
14	DC Motor	Toogle Switch			DC Motor	
15	LCD		LCD			
16	Keypad	Keypad	LCD			

EXP NO:	LED & FLASHING OF LED'S
DATE	

AIM:

To write and execute the program for LED & Flashing Led's with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

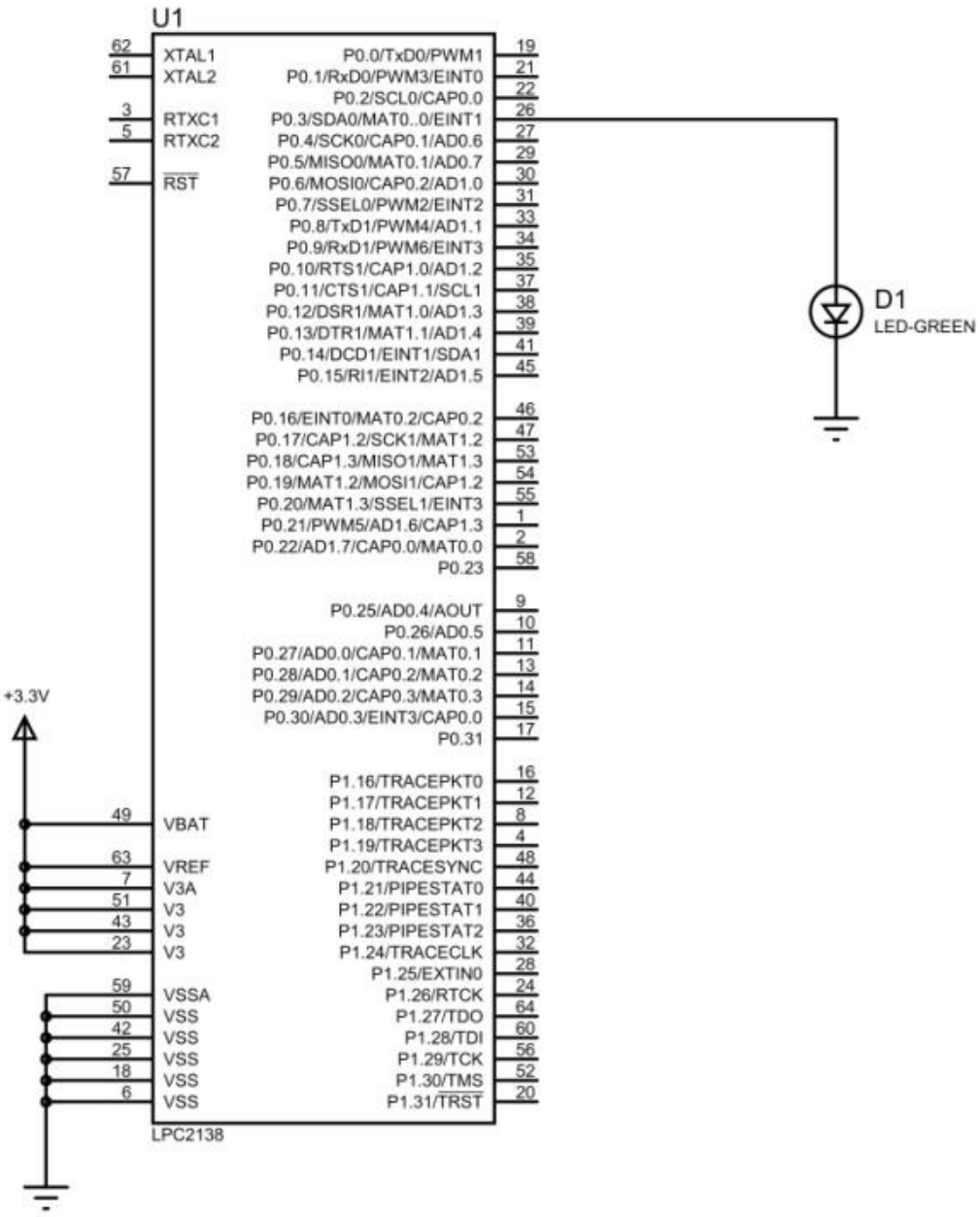
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LED Module	1
4	Power Supply (5V, DC)	1
5	Proteus ISIS Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

LED INTERFACING:

CIRCUIT DIAGRAM:

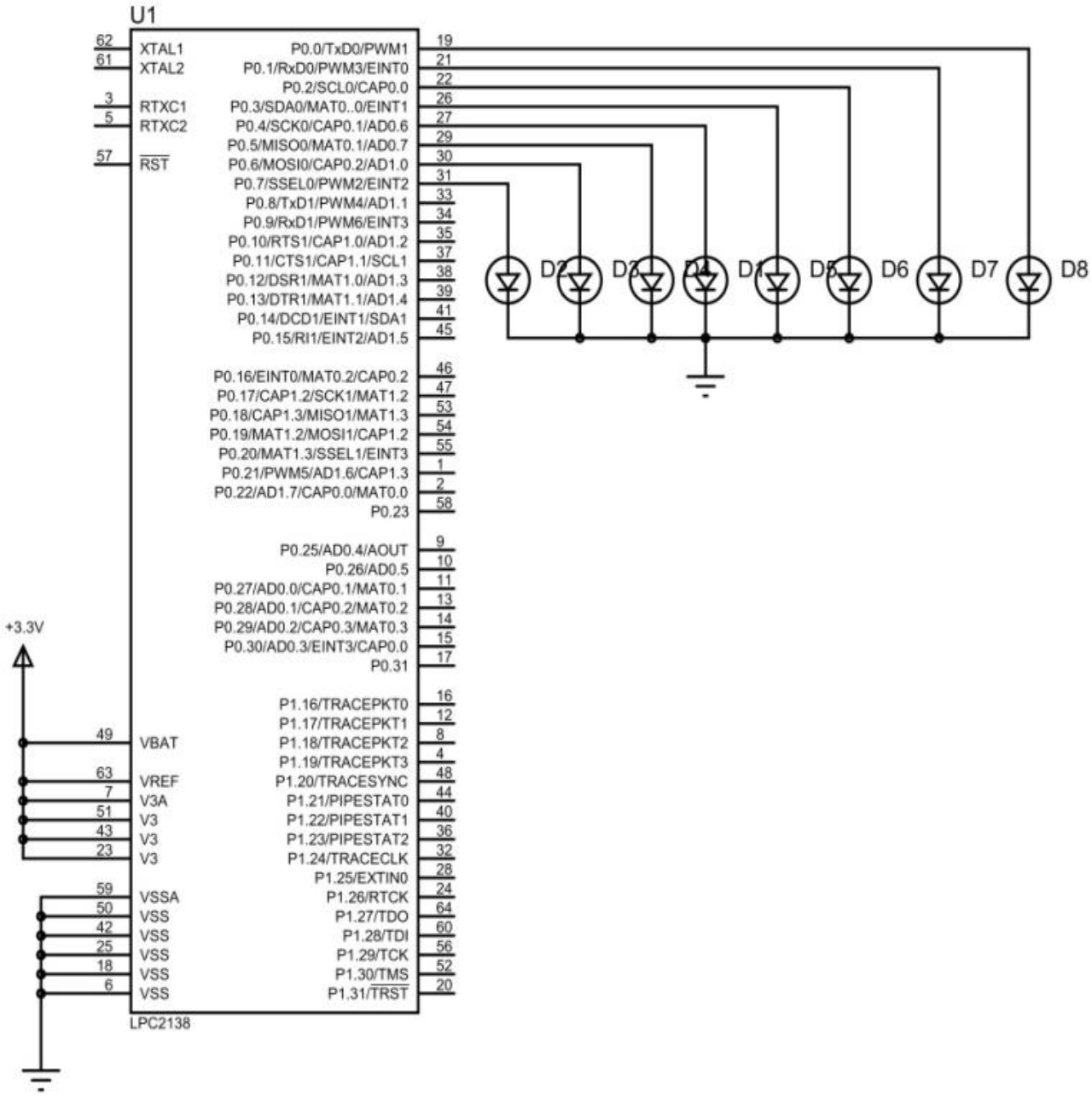


PROGRAM:

```
#include <lpc214x.h>
int i;
int main()
{
IODIR0=(1<<3);
        while(1)
        {
         IOSET0=(1<<3);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<3);
         for(i=0;i<120000;i++);
        }
}
```

FLASHING OF LED:

CIRCUIT DIAGRAM:



PROGRAM:

TYPE-I:

```
#include <lpc214x.h>
int i;
int main()
{
IODIR0=0x000000FF;
        while(1)
        {
         IOSET0=0x000000AA;
         for(i=0;i<120000;i++);
         IOCLR0=0x000000AA;
         for(i=0;i<120000;i++);
        }
}
```

TYPE-II:

```
#include <lpc214x.h>
int i,b;
int main()
{
IODIR0=0x000000FF;
while(1)
 {
 for(b=0;b<8;b++)
    {
          IOSET0=(1<<b);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<b);
         for(i=0;i<120000;i++);
    }
  }
}
```

TYPE-III

```
#include <lpc214x.h>
int i;

int main()
{
IODIR0=(1<<0)|(1<<1)|(1<<2)|(1<<3)|(1<<4)|(1<<5)|(1<<6)|(1<<7);
        while(1)
        {
         IOSET0=(1<<0);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<0);
         for(i=0;i<120000;i++);
         IOSET0=(1<<1);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<1);
         for(i=0;i<120000;i++);
         IOSET0=(1<<2);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<2);
         for(i=0;i<120000;i++);
         IOSET0=(1<<3);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<3);
         for(i=0;i<120000;i++);
         IOSET0=(1<<4);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<4);
         for(i=0;i<120000;i++);
         IOSET0=(1<<5);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<5);
         for(i=0;i<120000;i++);
         IOSET0=(1<<6);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<6);
         for(i=0;i<120000;i++);
         IOSET0=(1<<7);
         for(i=0;i<120000;i++);
         IOCLR0=(1<<7);
         for(i=0;i<120000;i++);
        }
}
```

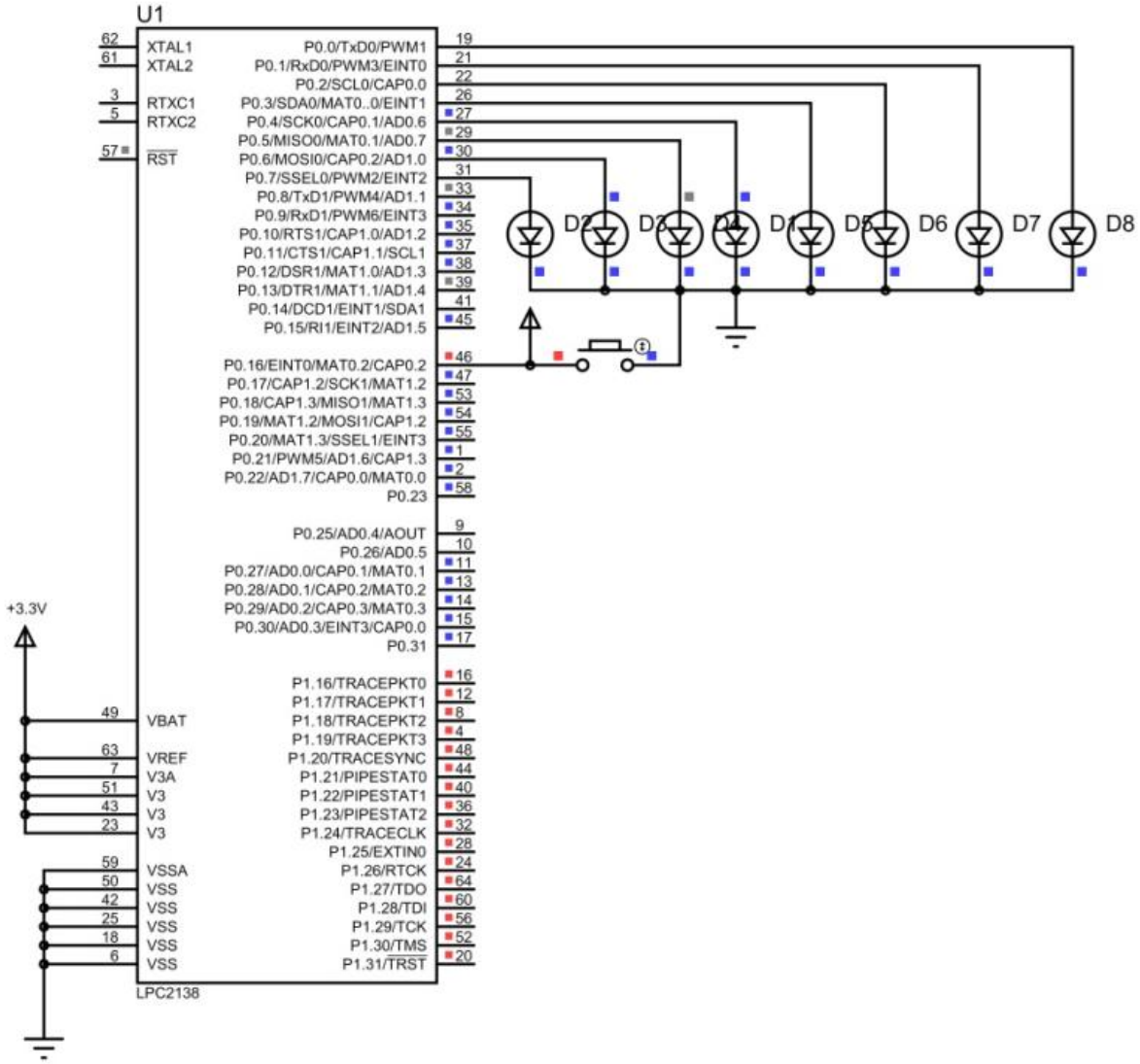
FRONT AND BACK LED:

```
#include <lpc214x.h>
int b,i;

int main()
{
IODIR0=0x000000FF;
while(1)
{
for(b=0;b<8;b++)
  {
          IOSET0=(1<<b);
          for(i=0;i<120000;i++);
          IOCLR0=(1<<b);
          for(i=0;i<120000;i++);
   }
 for(b=7;b>=0;b--)
  {
           IOSET0=(1<<b);
          for(i=0;i<120000;i++);
          IOCLR0=(1<<b);
          for(i=0;i<120000;i++);
   }
}
}
```

SWITCH CONTROLLED LED (1-SWITCH):

CIRCUIT DIAGRAM:



PROGRAM:

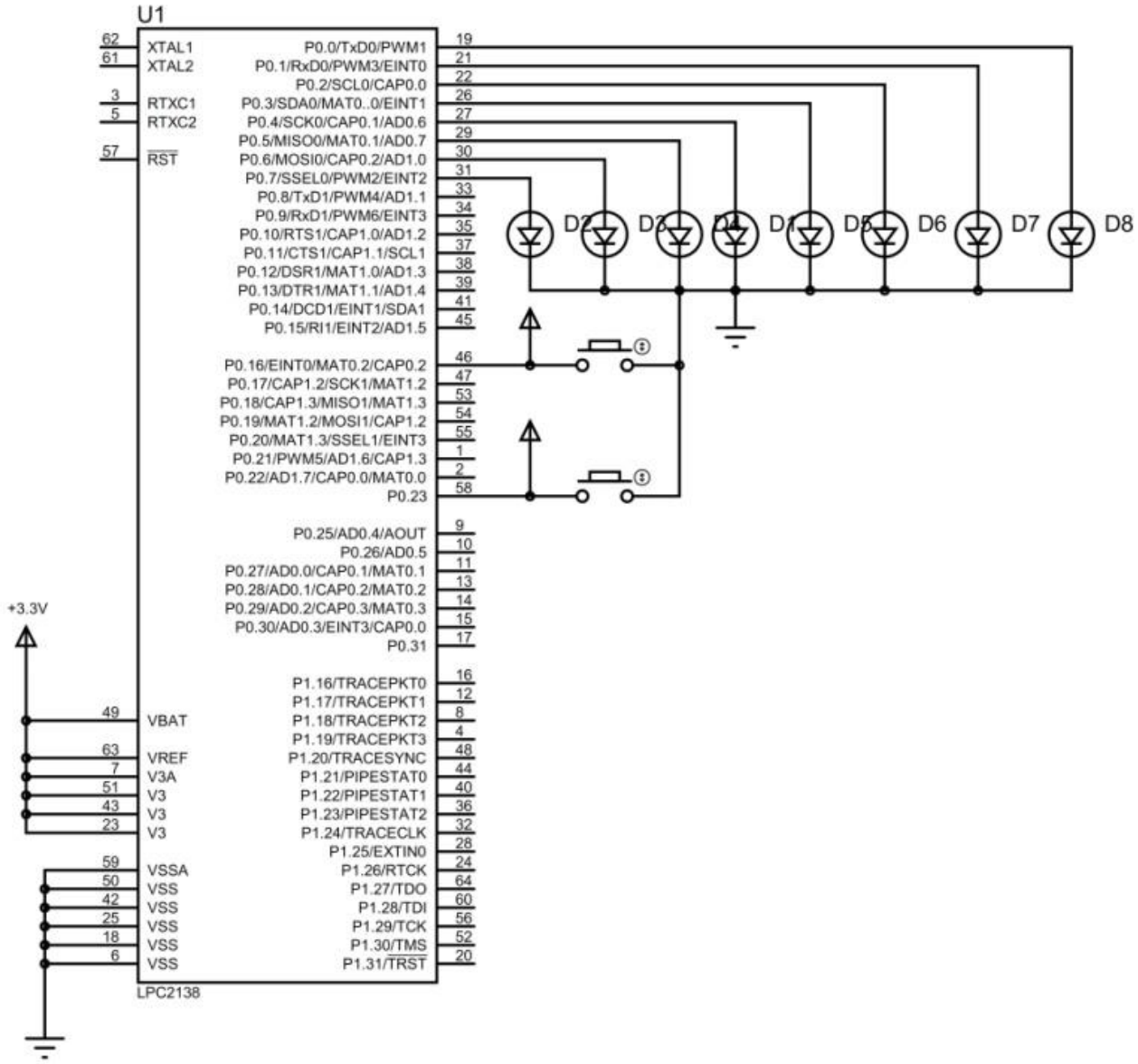
```
#include <lpc214x.h>
int i,b;

int main()
{
IODIR0=0x000000FF;
IODIR0=~(1<<16);

while(1)
{
if((IOPIN0&(1<<16))==0)
{
for(b=0;b<8;b++)
      {
          IOSET0=(1<<b);
          for(i=0;i<120000;i++);
          IOCLR0=(1<<b);
          for(i=0;i<120000;i++);
     }
}
else
          {
              IOCLR0=0x000000FF;
          }
}
}
```

SWITCH CONTROLLED LED (2-SWITCH):

CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
int i,b;

int main()
{
IODIR0=0x000000FF;
IODIR0=~(1<<16)&~(1<<23);

while(1)
{
if((IOPIN0&(1<<16))==0)
{
for(b=0;b<8;b++)
  {
          IOSET0=(1<<b);
           for(i=0;i<120000;i++);
           IOCLR0=(1<<b);
           for(i=0;i<120000;i++);
   }

}
else if((IOPIN0&(1<<23))==0)
{
  for(b=7;b>=0;b--)
  {
          IOSET0=(1<<b);
           for(i=0;i<120000;i++);
           IOCLR0=(1<<b);
           for(i=0;i<120000;i++);
   }

}
else
{
 IOCLR0=0x000000FF;
}
}
}
```

RESULT:

EXP NO:	INTERFACING OF LCD
DATE	

AIM:

To write and execute the program for LCD with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

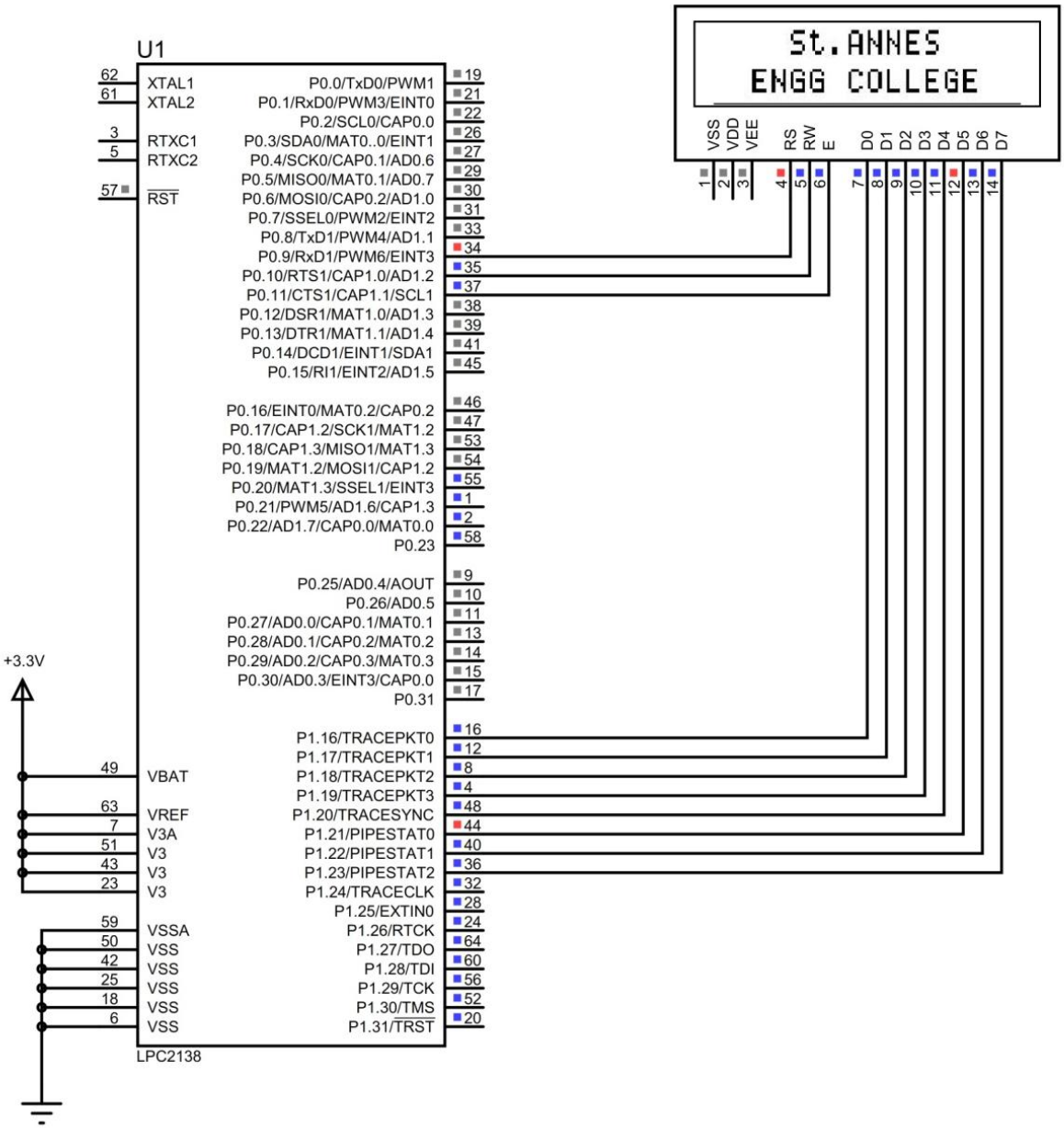
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LCD Module	1
4	Power Supply (5V, DC)	1
5	Proteus ISIS Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

INTERFACING LCD:

CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
#include <lcd.h>

int main()
 {
        LCD_INIT();
        LCDSTR(0x00000084,"St.ANNES");
        LCDSTR(0x000000C2,"ENGG COLLEGE");
        while(1)
        {
        }
 }
```

LCD LAYOUT:

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

RESULT:

EXP NO:	INTERFACING OF MATRIX KEYBOARD
DATE	

AIM:

To write and execute the program for Matrix Keyboard with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

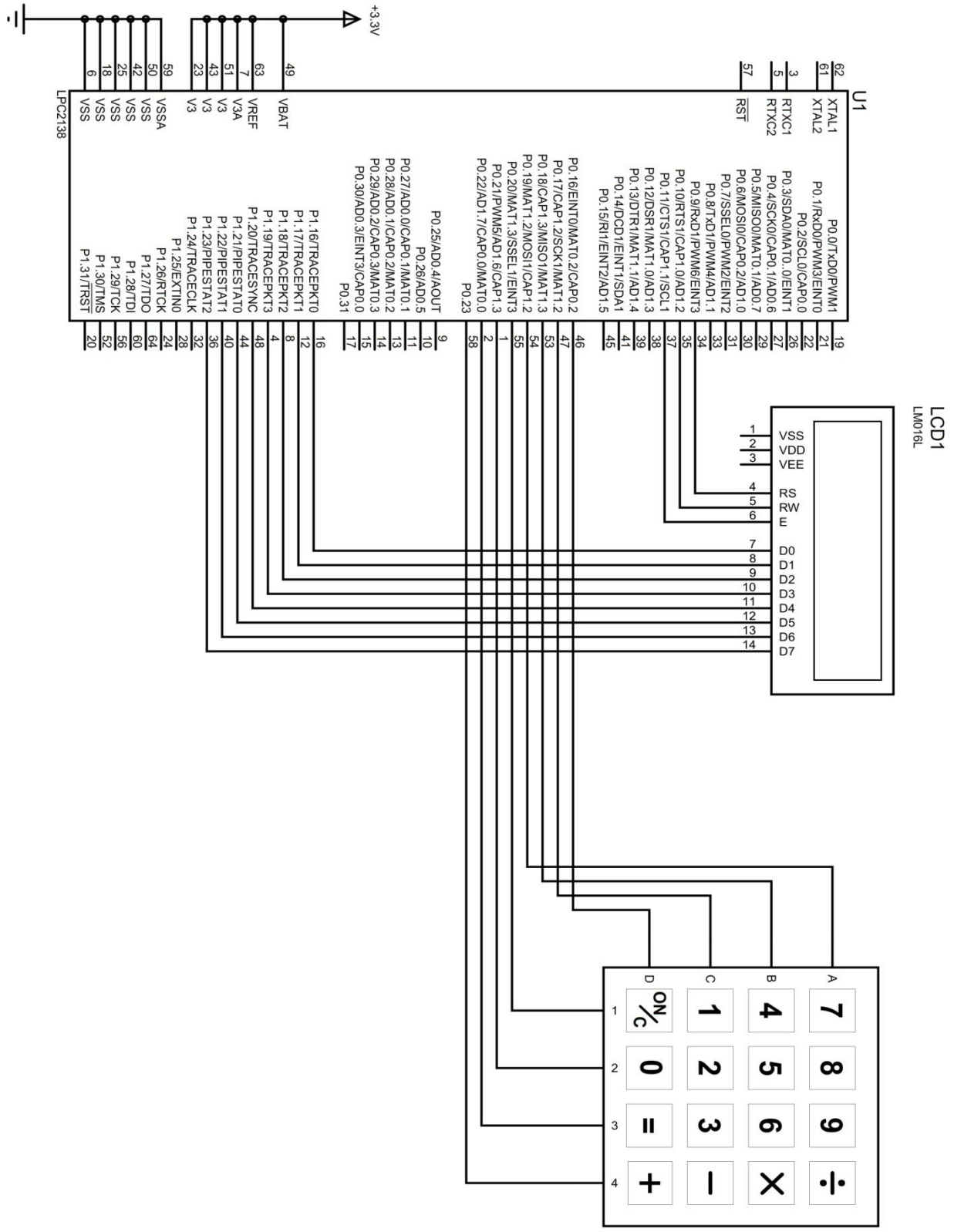
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	Matrix Keyboard Module	1
4	LCD Module	1
5	Power Supply (5V, DC)	1
6	Proteus ISIS Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

KEYBOARD INTERFACING:

CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
#include <lcd.h>
#include <keyboard.h>

int main()
 {
      LCD_INIT();
      LCDSTR(0x00000080,"Matrix Keypad");
      LCDSTR(0x000000C0,"Key Pressed:    ");
      while(1)
          {
                IO0CLR = CLR;
                IO0SET = O1;
                delay(10);
                if(scan(I1))LCDSTR(0x000000CC,"0");  //K1
                if(scan(I2))LCDSTR(0x000000CC,"4");  //K5
                if(scan(I3))LCDSTR(0x000000CC,"8");  //K9
                if(scan(I4))LCDSTR(0x000000CC,"C");  //K13
                IO0CLR = CLR;
                IO0SET = O2;
                if(scan(I1))LCDSTR(0x000000CC,"1");  //K2
                if(scan(I2))LCDSTR(0x000000CC,"5");  //K6
                if(scan(I3))LCDSTR(0x000000CC,"9");  //K10
                if(scan(I4))LCDSTR(0x000000CC,"D");  //K14
                IO0CLR = CLR;
                IO0SET = O3;
                if(scan(I1))LCDSTR(0x000000CC,"2");  //K3
                if(scan(I2))LCDSTR(0x000000CC,"6");  //K7
                if(scan(I3))LCDSTR(0x000000CC,"A");  //K11
                if(scan(I4))LCDSTR(0x000000CC,"E");  //K15
                IO0CLR = CLR;
                IO0SET = O4;
                if(scan(I1))LCDSTR(0x000000CC,"3");  //K4
                if(scan(I2))LCDSTR(0x000000CC,"7");  //K8
                if(scan(I3))LCDSTR(0x000000CC,"B");  //K12
                if(scan(I4))LCDSTR(0x000000CC,"F");  //K16
          }
 }
```

RESULT:

EXP NO:	INTERFACING OF SERIAL PORT
DATE	

AIM:

To write and execute the program for Serial Port with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

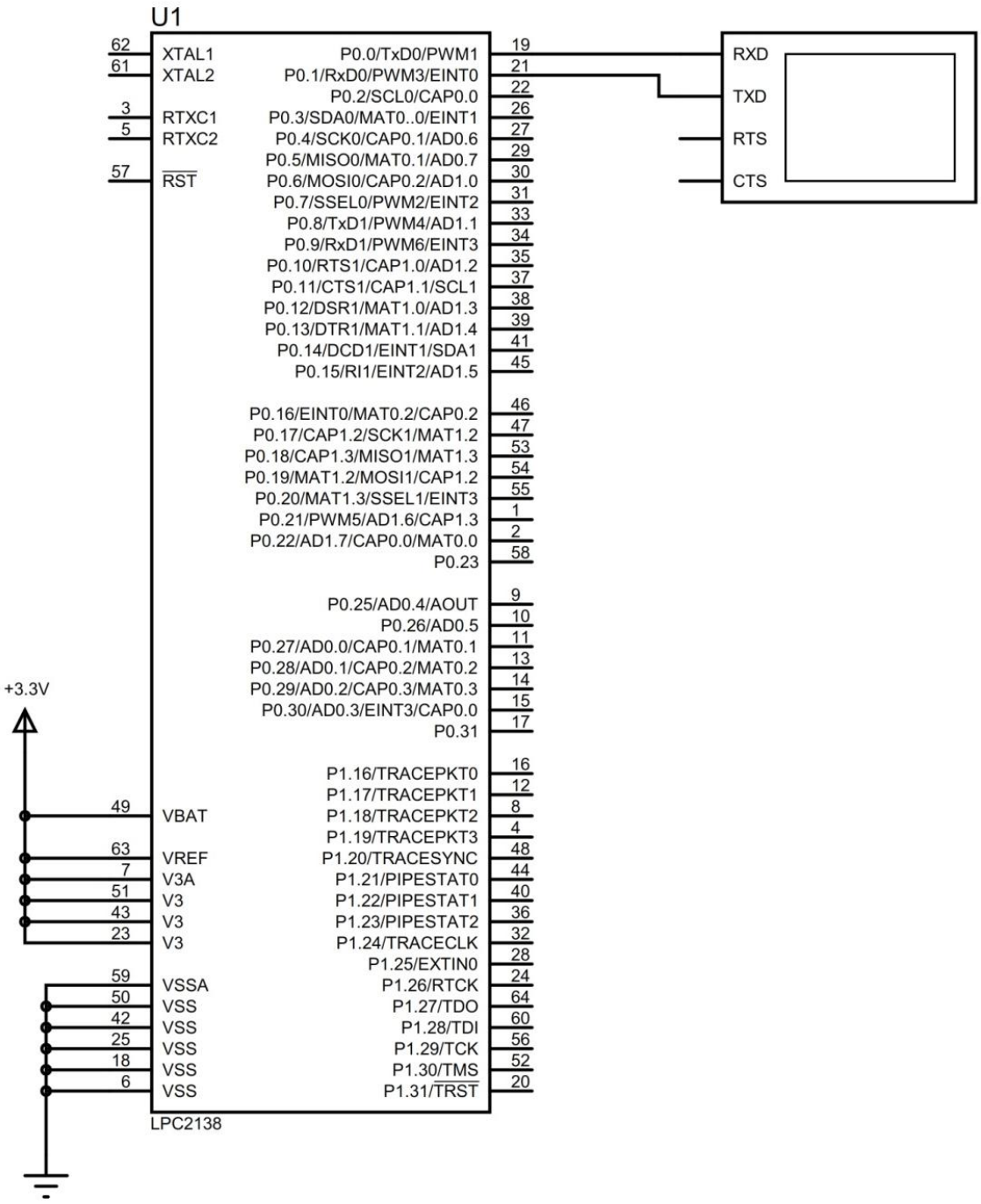
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LED Module	1
4	Power Supply (5V, DC)	1
5	Proteus ISIS Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

SERIAL PORT:

CIRCUIT DIAGRAM:



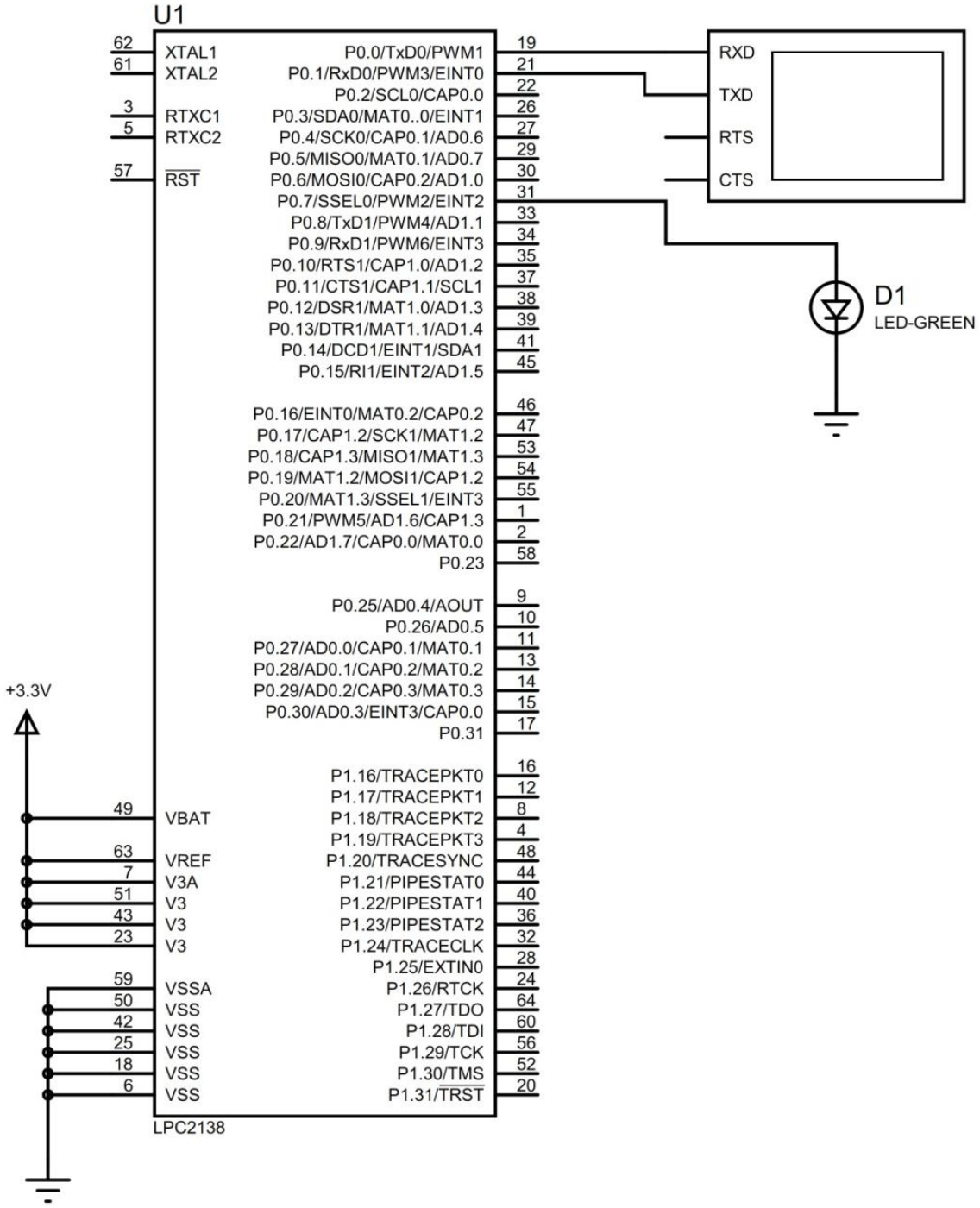
PROGRAM:

```
#include <lpc214x.h>
#include <delay.h>
#include <usart.h>

int main()
{
usart0_init();
        while(1)
        {
        getchar();
        delay_ms(1000);
        }
}
```

SERIAL PORT: (LED CONTROL)

CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
#include <delay.h>
#include <usart.h>

int key_pressed;
int main()
{
usart0_init();
IODIR0=(1<<7);
IOCLR0=(1<<7);
printf("\n Serial Communication \n");
while(1)
{
initial_state:
        printf("\n Key Pressed Is: ");
        key_pressed=getchar();
        if(key_pressed=='A')
        {
        IOSET0=(1<<7);
        printf("\n LED ON \n");
        delay_ms(100);
        goto initial_state;
        }
        else if(key_pressed=='S')
        {
        IOCLR0=(1<<7);
        printf("\n LED OFF \n");
        delay_ms(100);
        goto initial_state;
        }
        else
        {
        printf("\n INVALID KEY \n");
        delay_ms(100);
        goto initial_state;
        }
}
}
```

RESULT:

EXP NO:	INTERFACING OF STEPPER MOTOR
DATE	

AIM:

To write and execute the program for Stepper Motor with ARM7 (LPC2148) processor.

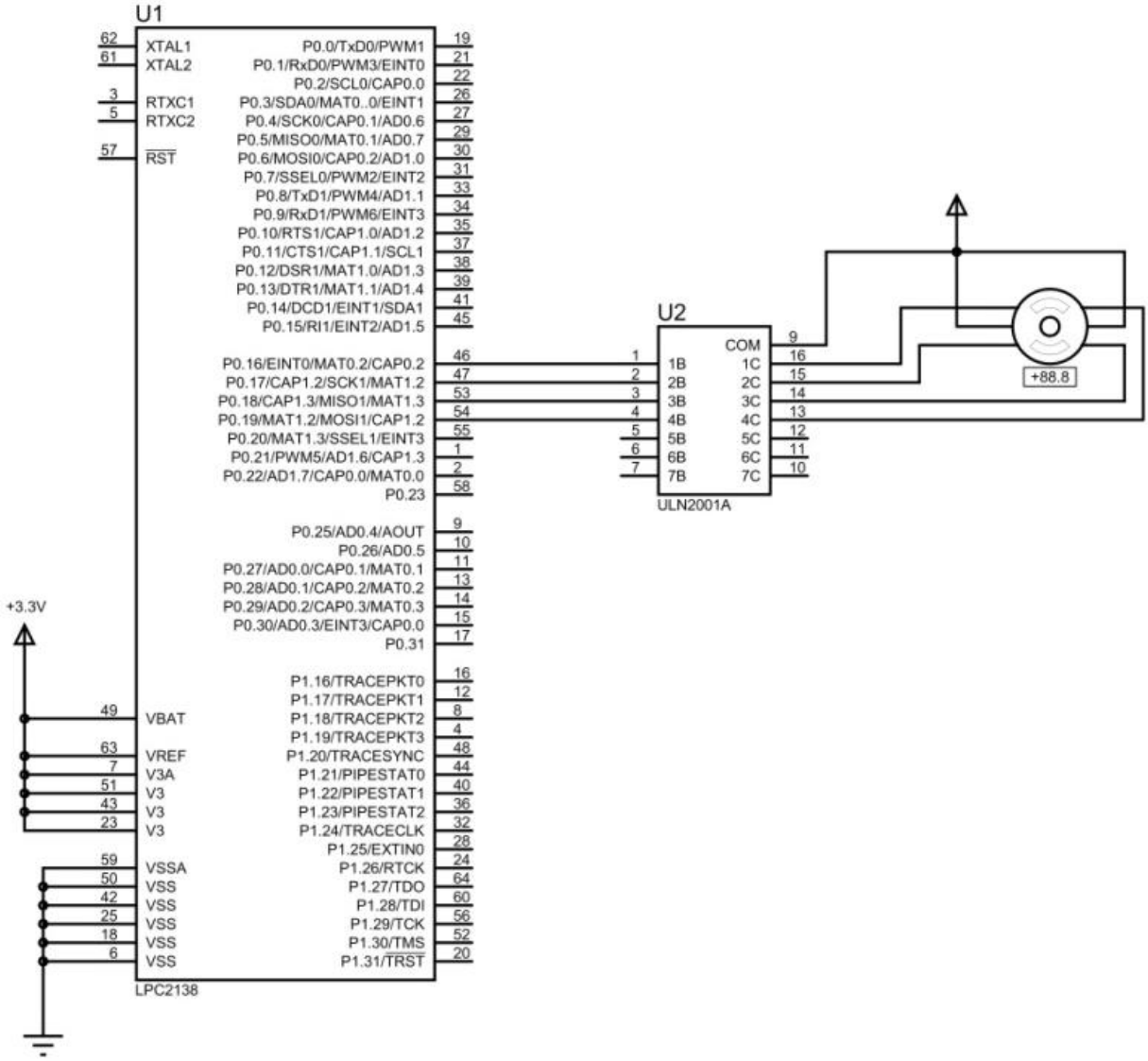
HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	Stepper Motor Module	1
4	Power Supply (5V, DC)	1
	Proteus ISIS Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

STEPPER MOTOR: (FORWARD ROTATION)
CIRCUIT DIAGRAM:

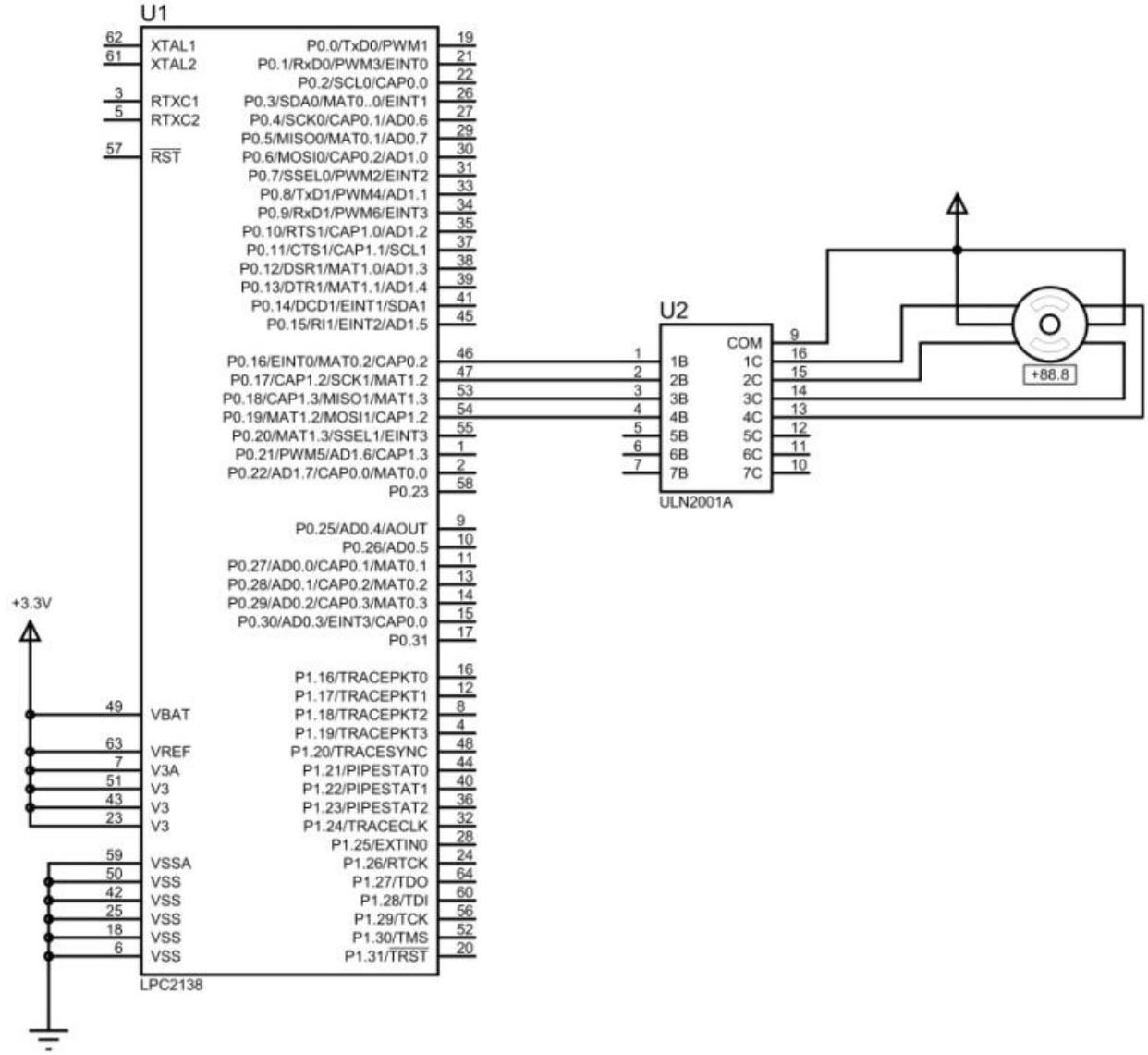


PROGRAM:

```
#include <lpc214x.h>
#include <delay.h>

int main()
{
IODIR0=(1<<16)|(1<<17)|(1<<18)|(1<<19);
        while(1)
        {
         //FORWARD DIRECTION
         IOCLR0=(1<<16);
         IOCLR0=(1<<17);
         IOSET0=(1<<18);
         IOSET0=(1<<19);
         delay_ms(100);
         IOCLR0=(1<<16);
         IOSET0=(1<<17);
         IOSET0=(1<<18);
         IOCLR0=(1<<19);
         delay_ms(100);
         IOSET0=(1<<16);
         IOSET0=(1<<17);
         IOCLR0=(1<<18);
         IOCLR0=(1<<19);
         delay_ms(100);
         IOSET0=(1<<16);
         IOCLR0=(1<<17);
         IOCLR0=(1<<18);
         IOSET0=(1<<19);
         delay_ms(100);
        }
}
```

STEPPER MOTOR: (REVERSE ROTATION)
CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
#include <delay.h>

int main()
{
IODIR0=(1<<16)|(1<<17)|(1<<18)|(1<<19);
        while(1)
        {
         //REVERSE DIRECTION
         IOSET0=(1<<16);
         IOCLR0=(1<<17);
         IOCLR0=(1<<18);
         IOSET0=(1<<19);
         delay_ms(1000);
         IOSET0=(1<<16);
         IOSET0=(1<<17);
         IOCLR0=(1<<18);
         IOCLR0=(1<<19);
         delay_ms(1000);
         IOCLR0=(1<<16);
         IOSET0=(1<<17);
         IOSET0=(1<<18);
         IOCLR0=(1<<19);
         delay_ms(1000);
         IOCLR0=(1<<16);
         IOCLR0=(1<<17);
         IOSET0=(1<<18);
         IOSET0=(1<<19);
         delay_ms(1000);
        }
}
```

RESULT:

EXP NO:	INTERFACING OF REAL TIME CLOCK
DATE	

AIM:

To write and execute the program for Real Time Clock with ARM7 (LPC2148) processor.

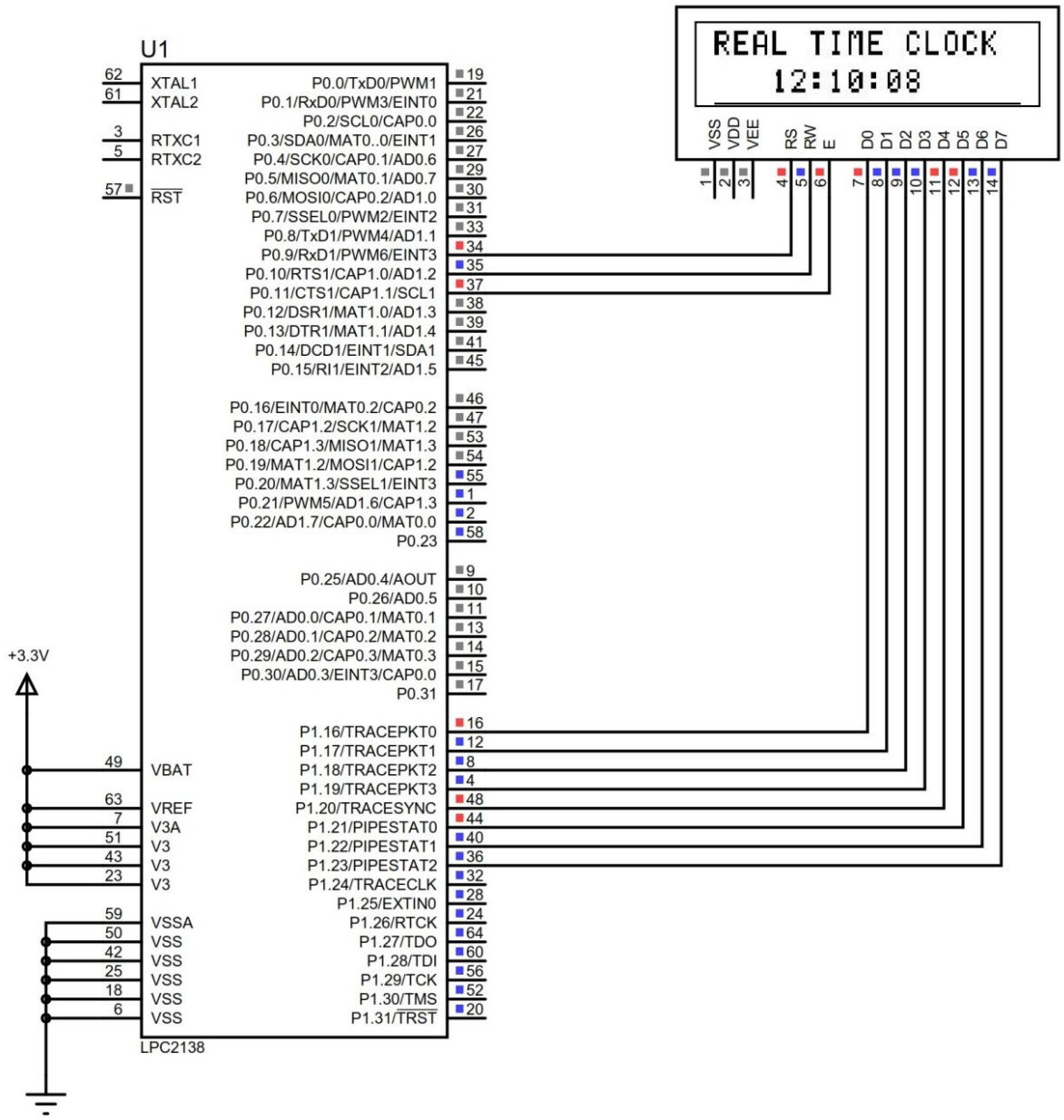
HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LCD Module	1
4	Power Supply (5V, DC)	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

REAL TIME CLOCK:
CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc21xx.h>
#include <rtc.h>
#include <lcd.h>
int main()
 {
        SETTIME();
        LCD_INIT();
        LCDSTR(0x00000080,"REAL TIME CLOCK ");
        LCDSTR(0x000000C0,"              ");
        while(1)
        {
              LCD_CMD (0x000000C3);
              LCD_DATA(HOUR/10 + '0');
              LCD_DATA(HOUR%10 + '0');
              LCD_DATA(':') ;
              LCD_DATA(MIN/10 + '0');
              LCD_DATA(MIN%10 + '0');
              LCD_DATA(':') ;
              LCD_DATA(SEC/10 + '0');
              LCD_DATA(SEC%10 + '0');
        }
 }
```

RESULT:

EXP NO:	INTERFACING OF ANALOG TO DIGITAL CONVERTOR
DATE	

AIM:

To write and execute the program for ADC with ARM7 (LPC2148) processor.

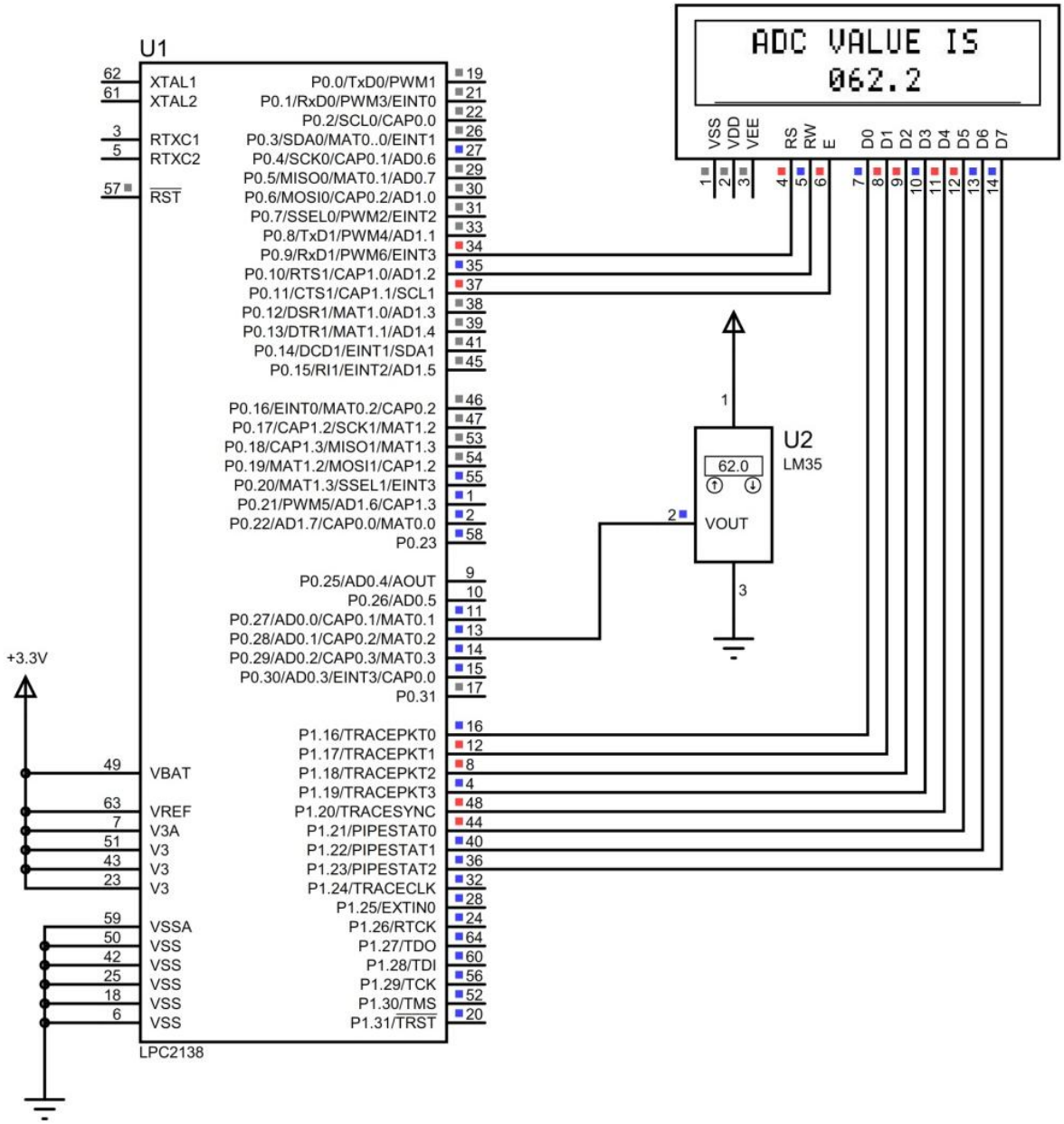
HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LCD Module	1
4	Power Supply (5V, DC)	1
5	ADC Module	1
6	Proteus ISIS software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

TEMPERATURE SENSOR:
CIRCUIT DIAGRAM:

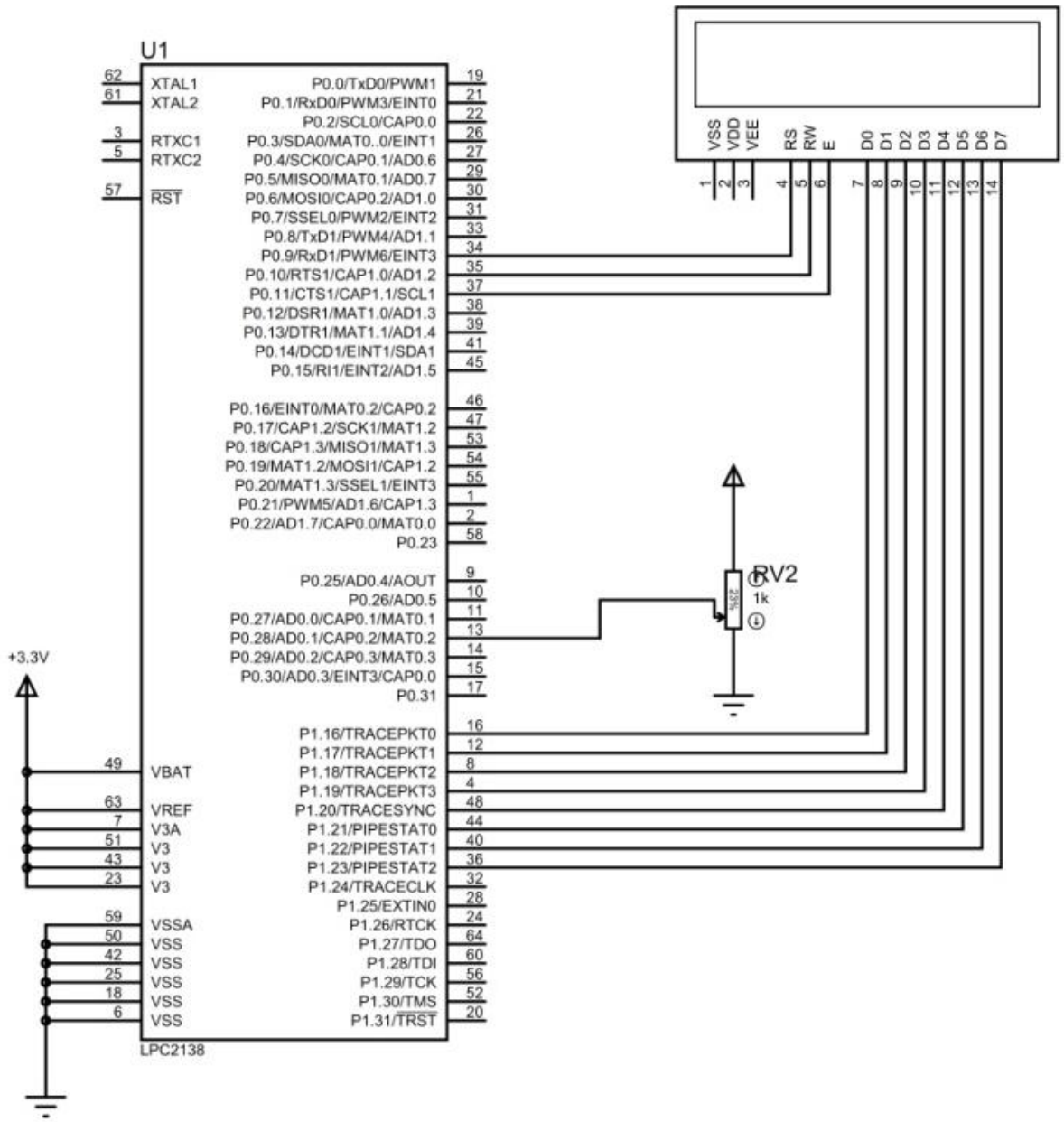


PROGRAM:

```
#include <lpc214x.h>
#include <string.h>
#include <delay.h>
#include <lcd.h>
#include <adc.h>

int main()
  {
    LCD_INIT();
    INIT_ADC();
    LCDSTR(0x00000082,"TEMPERATURE");
    LCDSTR(0x000000C0,"DEGREE C=");
    while(1)
      {
          TEMP = READ_ADC();
          ADC  = (1000*(TEMP*(3.901/1024)))-100;
          LCD_CMD(0x000000CA);
          delay_ms(1000);
          CONVERT_ADC1(ADC);
      }
  }
```

ANALOG TO DIGITAL CONVERTOR:
CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
#include <delay.h>
#include <lcd.h>
#include <adc.h>

int main()
 {
  LCD_INIT();
  INIT_ADC();
  LCDSTR(0x00000086,"ADC");
  LCDSTR(0x000000C0,"VOLTAGE =");
  while(1)
      {
         for(count=0;count<10000;count++)
          {
            temp_sum = temp_sum+READ_ADC();
          }
            temp_adv=temp_sum/10000;
            TEMP=temp_adv;
            temp_sum=0;
            ADC  = 1000*(TEMP*(3.92/1024));
            LCD_CMD(0x000000CA);
            CONVERT_ADC(ADC);
       }
 }
```

RESULT:

EXP NO:	INTERFACING OF DIGITAL TO ANALOG CONVERTOR
DATE	

AIM:

To write and execute the program for DAC with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

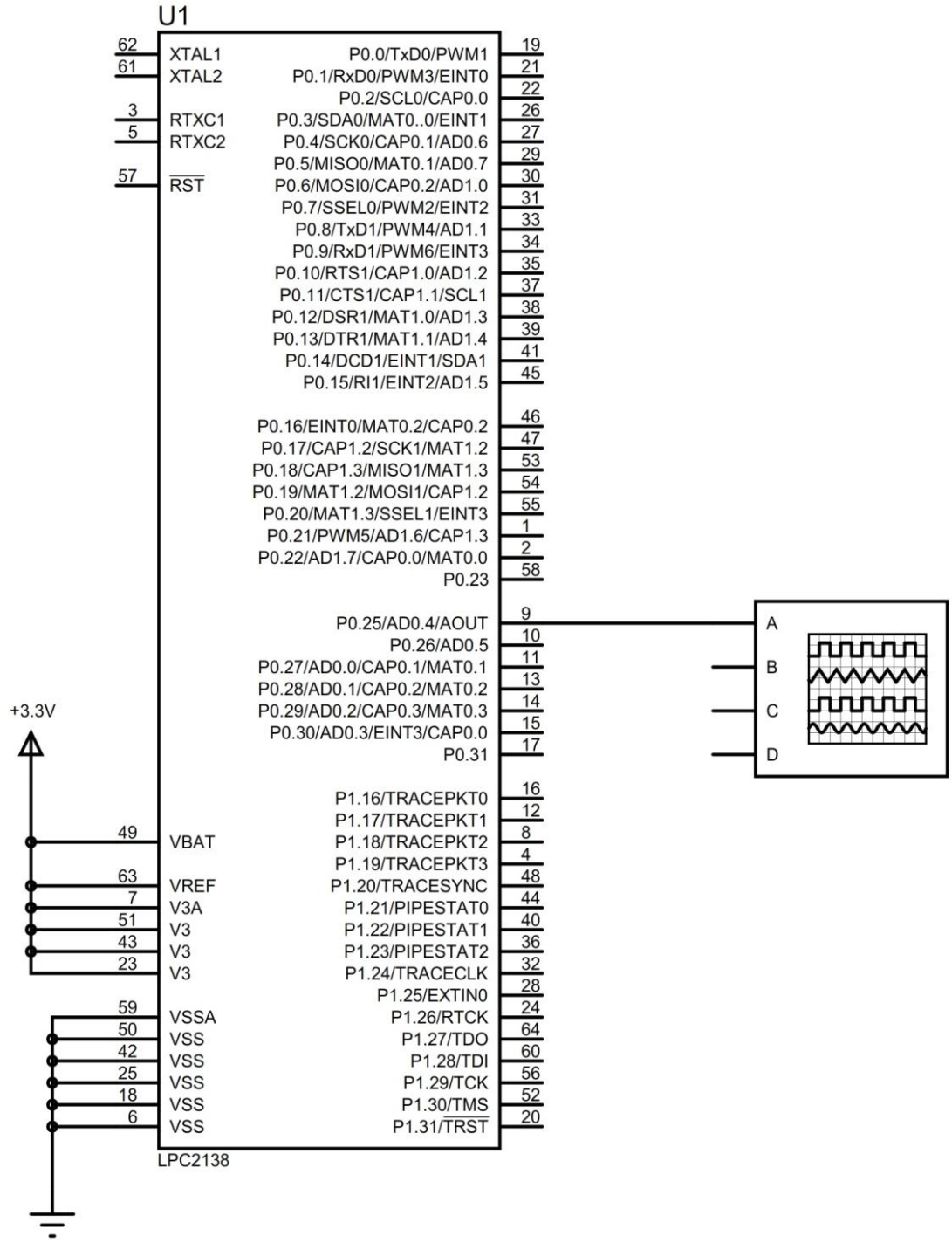
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	Power Supply (5V, DC)	1
4	DAC Module	1
5	Proteus ISIS software	1
6	CRO (Or) DSO	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

ANALOG TO DIGITAL CONVERTOR:

CIRCUIT DIAGRAM:



PROGRAM:

SIN WAVE:

```
#include<lpc214x.h>
#include<math.h>
unsigned int i;
int value;

int main()
  {
   PINSEL1 = 0X00080000;
   while(1)
            {
                     for(i=0;i<=360;i++)
                     {
                      value = 512+400*sin(i*0.01744);              //sin(2*pi*t/T)
                      DACR = value<<6;
                     }
            }
 }
```

SQUARE WAVE:

```
#include<LPC214x.H>
int main()
{
unsigned int a,b;
unsigned int delay;
PINSEL1 = 0X00080000;
while(1)
{
a=0x00;
DACR=a<<6;
for(delay=0; delay<120; delay++);
b=0x3FF;
DACR=b<<6;
for(delay=0; delay<120; delay++);
}
}
```

TRIANGULAR WAVE:

```
#include<lpc214x.h>
unsigned int i;

int main()
{
 PINSEL1 = 0X00080000;
 while(1)
{
  for(i=0;i<=1023;i++)
            {
                    DACR = i<<6;
            }
  for(i=1023;i>0;i--)
            {
                    DACR = i<<6;
            }
 }
 }
```

RAMP WAVE:

```
#include<lpc214x.h>
unsigned int i;
int main(void)
{
PINSEL1 = 0x00080000;
while (1)
   {
        for(i = 0 ; i <= 1023; i++)
        {
                DACR = i << 6 ;
        }
    }
}
```

DIGITAL TO ANALOG VALUE:

```
#include<lpc214x.h>
unsigned int i = 0;
void wait(void)
{
unsigned int delay;
for(delay=0;delay<120;delay++);
}
unsigned int
datatable[25]={0x00,0x0F,0x1F,0x2F,0x3F,0x4F,0x5F,0x6F,0x7F,0x8F,0x9F,0xAF,0xBF,0
xCF,0xDF,0xEF,0xFF,0x100,0x150,0x200,0x250,0x300,0x350,0x399};
int main (void)
{
PINSEL1=0x00200000;
while (1)
{
for(i=0;i<25;i++)
{
DACR=(datatable[i]<<6);
wait();
}
}
}
```

RESULT:

EXP NO:	INTERFACING OF PULSE WIDTH MODULATION
DATE	

AIM:

To write and execute the program for PWM with ARM7 (LPC2148) processor.

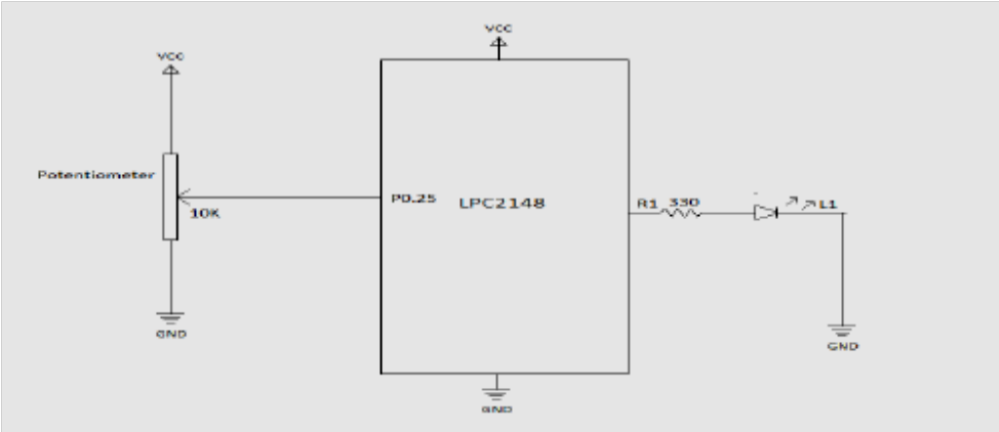
HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LED Module	1
4	Power Supply (5V, DC)	1
5	ADC Module	1
6	CRO (Or) DSO	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

PULSE WIDTH MODULATION:
CIRCUIT DIAGRAM:



PROGRAM:

```
#include<LPC214x.h>
#include<pwm.h>

int main()
 {
        PWM_INIT();
        INIT_ADC();
                while(1)
                {
                  PWM = READ_ADC();
                  if(PWM>0x000003FF) PWM=0x000003FF;
                  PWMMR2 = PWM;
                  PWMTCR   = 0X00000000;
                  PWMTCR   = 0X00000009;
                }
 }
```

RESULT:

EXP NO:	INTERFACING INTERRUPT
DATE	

AIM:

To write and execute the program for Interrupt with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

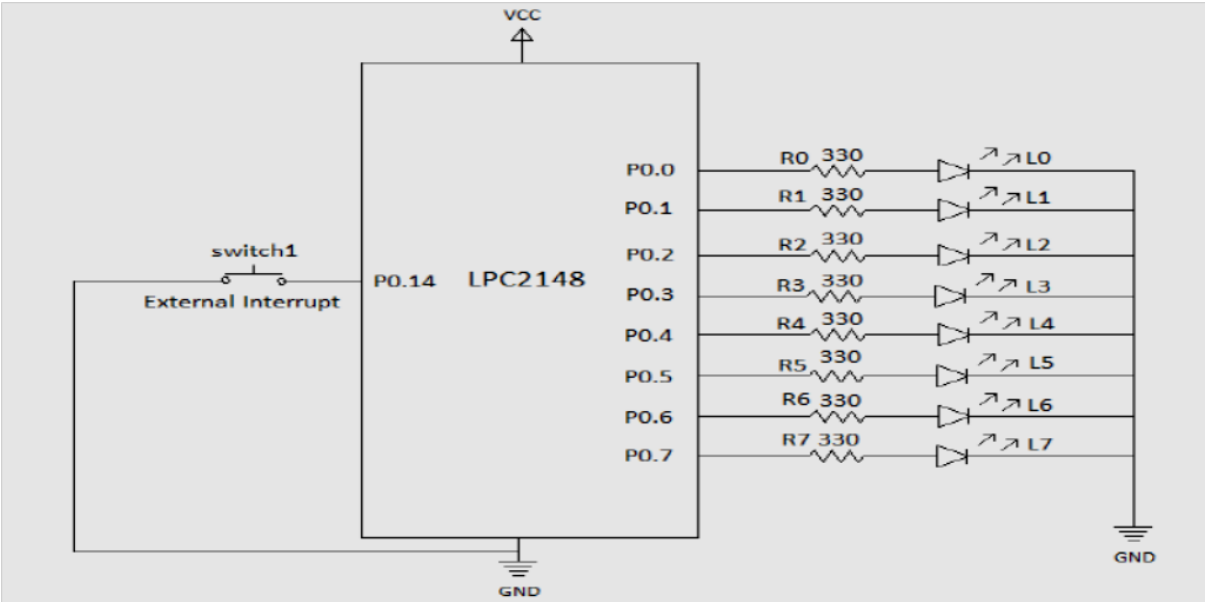
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LED Module	1
4	Power Supply (5V, DC)	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

INTERRUPT:

CIRCUIT DIAGRAM:



PROGRAM:

```
#include<lpc21xx.h>
#include<interrupt.h>
#include<delay.h>
unsigned int b;

int main(void)
{
   init_ext();
   IODIR0=0x000000FF;
 while(1)
{
for(b=0;b<8;b++)
      {
        IOSET0=(1<<b);
        delay_ms(100);
        IOCLR0=(1<<b);
        delay_ms(100);
      }
}
 }
```

RESULT:

EXP NO:	INTERFACING EEPROM
DATE	

AIM:

To write and execute the program for EEPROM with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

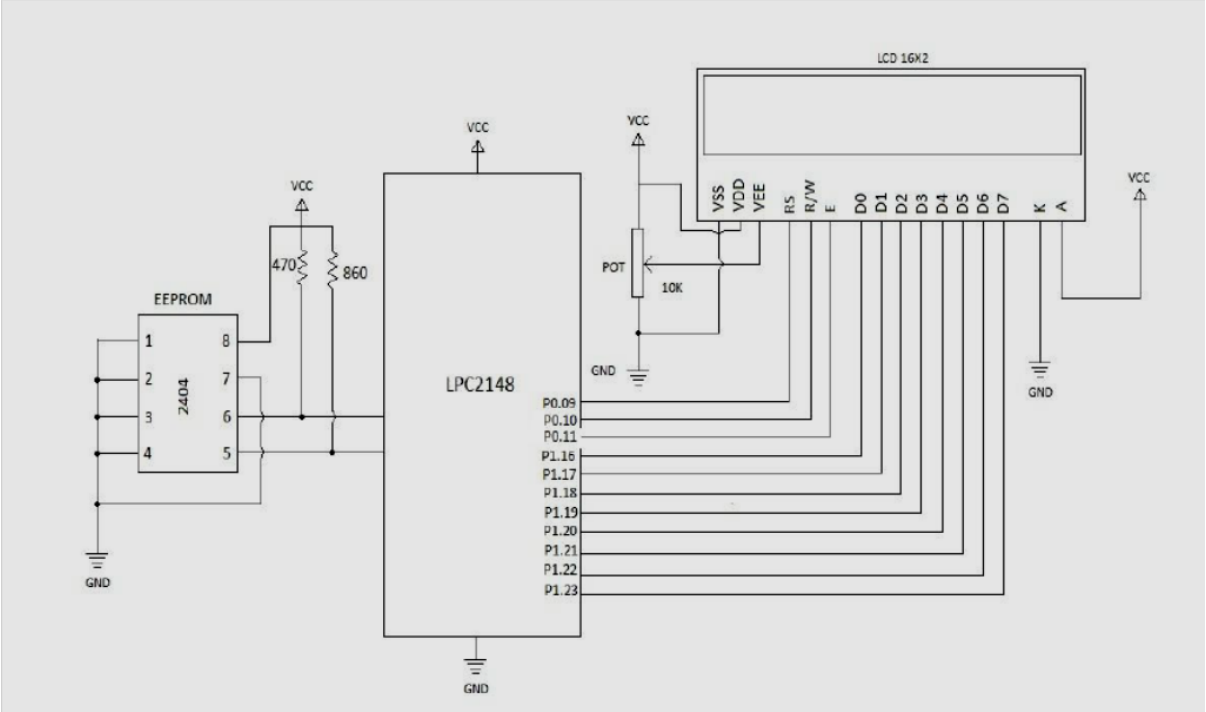
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	LCD Module	1
4	EEPROM Module	1
5	Power Supply (5V, DC)	1
6	Matrix Keypad Module	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

EEPROM:

CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc21xx.h>
#include <string.h>
#include <lcd.h>
#include <eeprom.h>
#include <keyboard.h>

char buf[16];
unsigned char cnt=0;
int main()
 {
        LCD_INIT();
        LCDSTR(0x00000082,"EEPROM TEST ");
        LCDSTR(0x000000C0,"COMMAND ENTER: 1");
        cnt=1;

        while(1)
        {
            IO0CLR = CLR;
            IO0SET = O1;
            delay(10);
         if(scan(I1))
            {
             LCDSTR(0x000000C0,"COMMAND ENTER: 1");
             cnt=1;
            }
         if(scan(I2))
            {
              if(cnt==1)
                {
                  eeprom_write(0,"HI PANRUTI      ");
                   LCDSTR(0x000000C0,"Writting Message");
                     delay(120000);
                 }
              if(cnt==2)
                {
                   eeprom_write(0,"HI CUDDALORE    ");
                   LCDSTR(0x000000C0,"Writting Message");
                     delay(120000);
                 }
              if(cnt==3)
                {
```

```
                eeprom_write(0,"HI NEYVELI      ");
                LCDSTR(0x000000C0,"Writting Message");
                 delay(120000);
               }
             if(cnt==4)
               {
                eeprom_write(0,"HI PONDICHERRY  ");
                LCDSTR(0x000000C0,"Writting Message");
                delay(120000);
               }
 }
 IO0CLR = CLR;
 IO0SET = O2;
 if(scan(I1))
       {
         LCDSTR(0x000000C0,"COMMAND ENTER: 2");
         cnt=2;
       }
 if(scan(I2))
       {
        eeprom_read(0,16, buf);
        LCDSTR(0x000000C0,"Reading Message ");
        DELAY(5000);
        LCDSTR(0xC0,buf);
        }
 IO0CLR = CLR;
 IO0SET = O3;
 if(scan(I1))
       {
        LCDSTR(0x000000C0,"COMMAND ENTER: 3");
        cnt=3;
       }
if(scan(I2))
       {
          eeprom_write(0 , "Nothing Here    ");
          LCDSTR(0x000000C0,"Erasing Message ");
          delay(120000);
          cnt=5;
       }
 IO0CLR = CLR;
 IO0SET = O4;
 if(scan(I1))
       {
         LCDSTR(0x000000C0,"COMMAND ENTER: 4");
```

```
                    cnt=4;
                  }
}
}
```

RESULT:

EXP NO:	INTERFACING ZIGBEE PROTOCOL WITH ARM
DATE	

AIM:

To write and execute the program for Zigbee Protocol with ARM7 (LPC2148) processor.

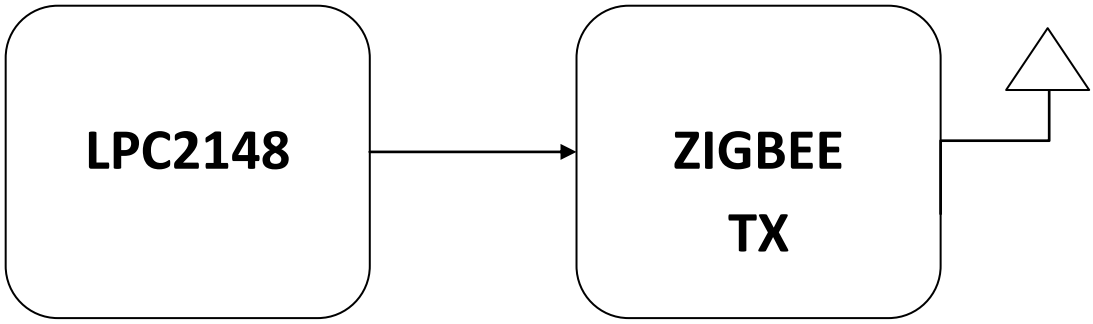
HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	Zigbee Module(TX, RX)	2
4	Power Supply (5V, DC)	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

ZIGBEE TRANSMITTER:
CIRCUIT DIAGRAM:

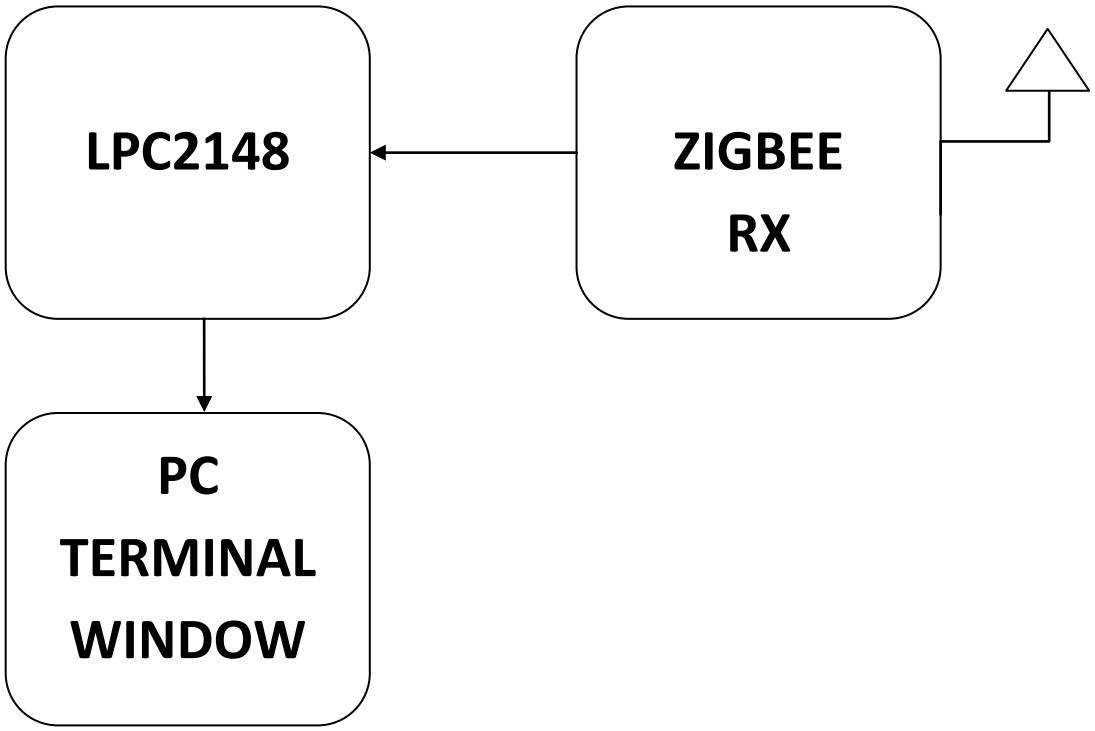


PROGRAM:

```
#include <lpc214x.h>
#include <delay.h>
#include <usart.h>

int key_pressed;
int main()
{
usart0_init();
 while(1)
        {
          delay_ms(100);
          sendchar('A');
          delay_ms(1000);
          sendchar('B');
          delay_ms(1000);
         }
}
```

ZIGBEE RECEIVER:
CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
#include <delay.h>
#include <usart.h>

int key_pressed;
int main()
{
usart0_init();
printf("\n Serial Communication \n");
        while(1)
        {
        printf("\n Value Transmitted  Is: ");
        key_pressed=getchar();
        }
}
```

RESULT:

EXP NO:	MAILBOX
DATE	

AIM:

To write and execute the program for Mailbox with ARM7 (LPC2148) processor.

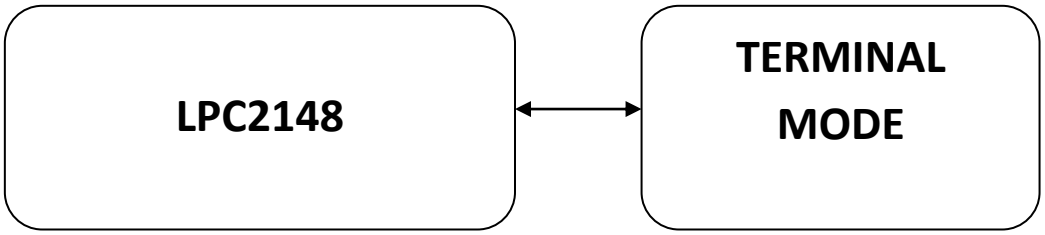
HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	Power Supply (5V, DC)	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

MAILBOX:
BLOCK DIAGRAM:



PROGRAM:

```
#include <RTL.h>
#include <LPC21xx.H>
#include <stdio.h>
OS_TID tsk1;
OS_TID tsk2;

typedef struct
 {
   float voltage;
   float current;
   U32   counter;
 }T_MEAS;

os_mbx_declare (MsgBox,16);
_declare_box (mpool,sizeof(T_MEAS),16);
__task void send_task (void);
__task void rec_task (void);

void init_serial ()
  {
     PINSEL0 = PINSEL0 | 0X00000005;
           U0LCR = 0X83;
           U0DLL = 0Xbb;
           U0DLM = 0X00;
           U0LCR = 0X03;
  }
__task void send_task (void)
 {
           T_MEAS *mptr;
           tsk1 = os_tsk_self ();
           tsk2 = os_tsk_create (rec_task, 0);
           os_mbx_init (MsgBox, sizeof(MsgBox));
           os_dly_wait (5);

           mptr = _alloc_box (mpool);
           mptr->voltage = 100.72;
           mptr->current = 17.54;
           mptr->counter = 120786;
           os_mbx_send (MsgBox, mptr, 0xffff);
           IOSET0 = 0x00010000;
           os_dly_wait (100);
```

```
            mptr = _alloc_box (mpool);
            mptr->voltage = 227.23;
            mptr->current = 12.41;
            mptr->counter = 170823;
            os_mbx_send (MsgBox, mptr, 0xffff);
            os_tsk_pass ();
            IOSET0 = 0x00020000;
            os_dly_wait (100);

            mptr = _alloc_box (mpool);
            mptr->voltage = 229.44;
            mptr->current = 11.89;
            mptr->counter = 237178;
            os_mbx_send (MsgBox, mptr, 0xffff);
            IOSET0 = 0x00040000;
            os_dly_wait (100);
            os_tsk_delete_self ();
}
__task void rec_task (void)
 {
   T_MEAS *rptr;
   for (;;)
         {
            os_mbx_wait (MsgBox, (void **)&rptr, 0xffff);
            printf ("\nVoltage: %.2f V\n",rptr->voltage);
            printf ("Current: %.2f A\n",rptr->current);
            printf ("Number of cycles: %d\n",rptr->counter);
            _free_box (mpool, rptr);
   }
 }

int main (void)
 {
  IODIR0 = 0x00FF0000;
  IOCLR0 = 0x00FF0000;
  init_serial ();
  _init_box (mpool, sizeof(mpool),sizeof(T_MEAS));
  os_sys_init (send_task);
 }
```

RESULT:

EXP NO:	PERFORMANCE CHARACTERISTICS OF ARM
DATE	

AIM:

To write and execute the program for Interrupt Performance Characteristics of ARM with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor	1
2	USB/FRC Connector	1
3	Power Supply (5V, DC)	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program
11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

PERFORMANCE CHARACTERISTICS OF ARM:

BLOCK DIAGRAM:

PROGRAM:

```
#include <lpc214x.h>
#include <string.h>
#include <lcd.h>
#include <pll.h>
void CONVERT_DATA(unsigned int);
void SEC_DELAY(void);

unsigned int count=0;

void ext_interrupt(void)__irq
 {
        EXTINT = 0X02;                          /* Clear interrupt flag        */
        count++;
        VICVectAddr = 0x00000000;               /* Acknowledge Interrupt       */
 }

void init_ext(void)
 {
        PINSEL0 |= 0X20000000;                  /*  enable EXT1                */
        EXTMODE = 0X02;                         /*  edge sensitive             */
        EXTPOLAR = 0X02;                        /*  on rising edge             */
        VICVectAddr0 =(unsigned int)ext_interrupt;   /*  Set Interrupt Vector in 0   */
        VICVectCntl0 = 0x0000002F;              /*  Use it for EXT1 Interrupt   */
        VICIntEnable = 0x00008000;              /*  enable EXT1 INTERRUPT       */
 }


int main()
 {
        init_ext();
          LCD_INIT();
        LCDSTR(0x00000080,"Counter:       ");
        LCDSTR(0x000000C0,"ARM PERFORMANCE");
      while(1)
       {
              SEC_DELAY();
              LCD_CMD(0x00000088);
              CONVERT_DATA(count);
              count=0;
       }
```

```
 }


void CONVERT_DATA(unsigned int DATA)
  {
                LCD_DATA((DATA % 100000000/10000000) | 0x30);
                LCD_DATA((DATA % 10000000/1000000) | 0x30);
                LCD_DATA((DATA % 1000000/100000) | 0x30);
                LCD_DATA((DATA % 100000/10000) | 0x30);
                LCD_DATA((DATA % 10000/1000) | 0x30);
                LCD_DATA((DATA % 1000/100) | 0x30);
                LCD_DATA((DATA % 100/10) | 0x30);
                LCD_DATA((DATA % 10/1) | 0x30);
                return;
        }

void SEC_DELAY()
{
  T0PR  = 0x0000270F;                  //0x0000270F
  T0MR0 = 0x00000BA1 ;              //0x000004B0
  T0TCR = 0x01;
  while(T0TC != T0MR0);
  T0MCR = 0x00000002;
  T0TCR = 0x02;
}
```

RESULT:

HEADER FILES

lcd.h:

```
extern void DELAY(unsigned int);
extern void LCDSTR(unsigned char ADDRESS,char *MSG);
extern void LCD_DATA(unsigned char);
extern void LCD_CD(unsigned int);
extern void LCD_INIT(void);

unsigned int  E  = 0x00000800;       // P1.11          ENABLE LCD
unsigned int  RW = 0x00000400;       // P1.10          READ/WRITE LCD
unsigned int  RS = 0x00000200;       // P1.09          REGISTER SELECT LCD

void DELAY(unsigned int VALUE)
 {
   unsigned int i,j;
   for(i=0;i<VALUE;i++)
     {
       for(j=1;j<1200;j++);
     }
 }

void LCD_DATA(unsigned char VALUE)
 {
    unsigned int b,TEMP;
            TEMP = VALUE;
         for(b=0;b<16;b++)
           {
             TEMP = TEMP<<1;
                                      }
         IO1SET = TEMP;
         IO1CLR = ~TEMP;
         IO0CLR = RW;
         IO0SET = RS;
         IO0SET = E;
         DELAY(50);
         IO0CLR = E;
 }
void LCD_CMD(unsigned int LCMD)
 {
    unsigned int b;
         for(b=0;b<16;b++)
           {
             LCMD = LCMD<<1;
                  }
         IO1SET = LCMD;
         IO1CLR = ~LCMD;
```

```
        IO0CLR = RW;
        IO0CLR = RS;
        IO0SET = E;
        DELAY(50);
        IO0CLR = E;
 }
void LCDSTR(unsigned char ADDRESS,char *MSG)
  {
     unsigned char COUNT,LENGTH;
    LCD_CMD(ADDRESS);
     LENGTH = strlen(MSG);
       for(COUNT=0;COUNT<LENGTH;COUNT++)
        {
          LCD_DATA(*MSG);
          MSG++;
        }
  }

void LCD_INIT()
 {
         //GPIO
        IO0DIR =0x00F00E00;    // P0.9  ----- P0.11  LCD DATA
        IO1DIR =0X00FF0000;    // P1.16  --- P1.23   LCD CONTROL

 //LCD INITIALIZATION
        LCD_CMD(0x00000038);  //function set: 8-bit, 2 line, 5x7 dots
        LCD_CMD(0x0000000C);  //display on, cursor off
        LCD_CMD(0x00000001);  //clear display screen
        LCD_CMD(0x00000006);  //entry mode
 }
```

keyboard.h:

```
#define O1 0X00E00000
#define O2 0X00D00000
#define O3 0X00B00000
#define O4 0X00700000

#define I1 0x000E0000
#define I2 0x000D0000
#define I3 0x000B0000
#define I4 0x00070000

#define CLR 0x00F00000

extern char scan (int);
extern void delay1(int);
```

```
char scan(int keystatus)              /* scanning a a key      */
 {
   while((IO0PIN & 0X000F0000)==keystatus)
          {
                    delay1(50);
                    if((IO0PIN & 0X000F0000)== 0X000F0000)return(1);
          }
             return(0) ;
 }


void delay1(int n)                                  /* generates one milisecond delay  */
 {
            int i,j;
            for (i=1; i<=n; i++)
            for(j=0; j<=10000; j++);
 }
```

delay.h:

```
#ifndef __delay_h
#define __delay_h

void delay_ms(unsigned int);
void delay_us(unsigned int);

void delay_ms(unsigned long delay_ms_var1)
{
unsigned int dummy_ms_1;
unsigned int dummy_ms_2;
          while(delay_ms_var1>0)
          {
                    for(dummy_ms_2=0;dummy_ms_2<7100;dummy_ms_2++)
                    {
                    dummy_ms_2++;
                    }
          delay_ms_var1--;
          }
}

void delay_us(unsigned long delay_us_var1)
{
unsigned int dummy_us_1;
          for(dummy_us_1=0;dummy_us_1<delay_us_var1;dummy_us_1++)
          {
          dummy_us_1+=3;
          }
```

```
}
#endif
```

eeprom.h:

```
#define DEV_WR_STAT 0X18
#define DEV_RD_STAT  0X40
#define LOC_WR_STAT 0x28
#define DATA_WR_STAT 0x28
#define DEV_WRITE 0XA0
#define DEV_READ  0XA1

extern void delay(unsigned int del);
extern void i2c_init(void);
extern void i2c_start(void);
extern void i2c_write(char dat , char status);
extern void i2c_stop(void);
extern char i2c_read(void);
extern void eeprom_read(char loc,char loc_count , char *ptr);
extern void eeprom_write(char loc , char *ptr);
unsigned char I2Cdata;

void delay(unsigned int del)
 {
   while(del-- > 0);
 }
void i2c_init(void)
 {
          PINSEL0 = 0X50;
          I2CONCLR = 0X0000002C;                    /* clears AA,I2C interrupt flag,START flag bit*/
          I2CONSET = 0X00000044;                    /* enable i2c   */
          I2SCLL   = 0xFFFF;                        /* bitrate = 400kHz          */       // 70,5
          I2SCLH   = 0xFFFF;;
 }
void i2c_start()
 {
          I2CONSET |= 0x00000020;                   /* Set STA flag */
          while (I2STAT   !=        0x00000008);    /* Wait for Status Set - 0x08 */
 }
void i2c_write(char dat , char status)
 {
          I2DAT = dat;
          I2CONSET = 0X00000004;
          I2CONCLR = 0X00000028;
          while (I2STAT   !=        status);        /* Wait for Status Set */
 }
void i2c_stop(void)
```

```
 {
        I2CONSET |= 0x00000014;
        I2CONCLR  = 0x00000008;                         /* Stop I2C and Start Bit */
 }
char i2c_read(void)
 {
        I2CONSET = 0X00000004;
        I2CONCLR = 0X00000028;
        while (I2STAT   !=      0x00000050);        /* Wait for Status Set - 0x50 */
        return(I2DAT);
 }
void eeprom_read(char loc,char loc_count , char *ptr)
 {
        i2c_init();
        i2c_start();
        i2c_write(DEV_WRITE ,DEV_WR_STAT);
        i2c_write(loc , LOC_WR_STAT);
        i2c_stop();
        delay(5000);
        i2c_start();
        i2c_write(DEV_READ ,DEV_RD_STAT);
        while(loc_count-- > 0)
         {
            *ptr = i2c_read();
            ptr++;
          }
        i2c_stop();
 }
void eeprom_write(char loc , char *ptr)
 {
        i2c_init();
        i2c_start();
        i2c_write(DEV_WRITE ,DEV_WR_STAT);
        i2c_write(loc , LOC_WR_STAT);
        while(*ptr != '\0')
    {           i2c_write(*ptr , LOC_WR_STAT);
                ptr++;      }
   i2c_stop();
 }
```

rtc.h:

```
extern void SETTIME(void);

void SETTIME(void)
{
  CCR  = 0x02;
```

```
   HOUR = 12;
   MIN  = 10;
   SEC  = 20;
   CCR  = 0x11;
}
```

adc.h:

```
extern void INIT_ADC(void);
extern void PwmInit(void);
extern int READ_ADC(void);
extern void CONVERT_ADC(unsigned int);

unsigned int TEMP=0,ADC=0;
unsigned long temp_adv=0,temp_sum=0;
unsigned int count=0;
unsigned int PWM=0;

void INIT_ADC()
 {
    PINSEL1 |= 0x01000000;
    AD0CR    = 0x00200602;
 }

void PwmInit()
{
           PINSEL0  = 0X00008000;  // Enable PWM2 -- P0.7
           PWMLER   = 0X00000004;  // Set latch register
           PWMPR    = 0X00000000;          // Prescale 0
           PWMMR0          = 0X000003FF;  // 120 KHz Frequency
           PWMMR2   = 0X000003FF;  // 50% Duty Cycle                7C        0,62,124,186
           PWMMCR   = 0X00000002;  // Set only Match0 control
           PWMPCR   = 0X00000400;
           PWMTCR   = 0X00000009;
}

int READ_ADC()
 {
    int VAL;
    AD0CR |= 0x01000000;             /* Start A/D Conversion         */
      do
                      {
          VAL = AD0DR1;             /* Read A/D Data Register       */
        }
           while (!(VAL & 0x80000000));        /* Wait for end of A/D Conversion */
           AD0CR &= ~0x01000000;      /* Stop A/D Conversion          */
```

```
        VAL >>=6;
        VAL = VAL & 0x3FF;
        return(VAL);
 }
void CONVERT_ADC(unsigned int ADC)
  {
        unsigned int CNT1,CNT2,CNT3,CNT4;
     unsigned char DATA1,DATA2,DATA3,DATA4;
            CNT1 = ADC % 10000/1000;
        CNT2 = ADC % 1000/100;
            CNT3 = ADC % 100/10;
            CNT4 = ADC % 10/1;
                    DATA1 = CNT1 | 0x30;
                    DATA2 = CNT2 | 0x30;
                    DATA3 = CNT3 | 0x30;
                    DATA4 = CNT4 | 0x30;
                      LCD_DATA(DATA1);
                          LCD_DATA('.');
                      LCD_DATA(DATA2);
                   LCD_DATA(DATA3);
                      LCD_DATA(DATA4);
                return;
        }


void CONVERT_ADC1(unsigned int ADC)
  {
        unsigned int CNT1,CNT2,CNT3,CNT4;
     unsigned char DATA1,DATA2,DATA3,DATA4;
            CNT1 = ADC % 10000/1000;
        CNT2 = ADC % 1000/100;
            CNT3 = ADC % 100/10;
            CNT4 = ADC % 10/1;
                    DATA1 = CNT1 | 0x30;
                    DATA2 = CNT2 | 0x30;
                    DATA3 = CNT3 | 0x30;
                    DATA4 = CNT4 | 0x30;
                      LCD_DATA(DATA1);
                      LCD_DATA(DATA2);
                     LCD_DATA(DATA3);
                    LCD_DATA('.');
                      LCD_DATA(DATA4);
                return;
        }
```

interrupt.h:

```
extern void ext_interrupt(void)__irq;
extern void init_ext(void);
extern void DELAY(unsigned long VALUE);
void ext_interrupt(void)__irq
 {
          EXTINT = 0X02;                                          /* Clear interrupt flag        */
          DELAY(5000000);
          VICVectAddr = 0x00000000;                               /* Acknowledge Interrupt       */
 }

void init_ext(void)
 {
          PINSEL0 |= 0X20000000;                                  /*  enable EXT1                */
          EXTMODE = 0X02;                                         /*  edge sensitive             */
          EXTPOLAR = 0X02;                                        /*  on rising edge             */
          VICVectAddr0 =(unsigned int)ext_interrupt;   /*  Set Interrupt Vector in 0   */
          VICVectCntl0 = 0x0000002F;                              /*  Use it for EXT1 Interrupt   */
          VICIntEnable = 0x00008000;                              /*  enable EXT1 INTERRUPT       */
 }
void DELAY(unsigned long VALUE)
{
         while(VALUE>0)
         {
                  VALUE--;
         }
}
```

pwm.h:

```
extern void INIT_ADC(void);
extern void PWM_INIT(void);
extern int READ_ADC(void);

unsigned int PWM=0;

void INIT_ADC()
 {
   PINSEL1 |= 0x01000000;
   AD0CR    = 0x00200602;
 }

void PWM_INIT()
{
          PINSEL0  = 0X00008000;  // Enable PWM2 -- P0.7
          PWMLER   = 0X00000004;  // Set latch register
```

```
        PWMPR    = 0X00000000;          // Prescale 0
        PWMMR0         = 0X000003FF;  // 120 KHz Frequency
        PWMMR2   = 0X000003FF;  // 50% Duty Cycle              7C        0,62,124,186
        PWMMCR   = 0X00000002;  // Set only Match0 control
        PWMPCR   = 0X00000400;
        PWMTCR   = 0X00000009;
}

int READ_ADC()
 {
    int VAL;
    AD0CR |= 0x01000000;             /* Start A/D Conversion         */
      do
                    {
          VAL = AD0DR1;                  /* Read A/D Data Register       */
        }
          while (!(VAL & 0x80000000));    /* Wait for end of A/D Conversion */
          AD0CR &= ~0x01000000;             /* Stop A/D Conversion          */
          VAL >>=6;
          VAL = VAL & 0x3FF;
          return(VAL);
 }
```

usart.h:

```
#include <stdio.h>
#include <time.h>
#include <rt_misc.h>

void usart0_init(void);
void usart1_init(void);
short check_usart(void);
//int getchar_ (void);
int getchar_attempt (unsigned int);
void usart_flush ();

short usart_print;

extern int  sendchar(int ch);  /* in Serial.c */
extern int  getkey(void);      /* in Serial.c */
extern long timeval;          /* in Time.c   */



struct __FILE { int handle; /* Add whatever you need here */ };
FILE __stdout;
FILE __stdin;
```

```
int fputc(int ch, FILE *f) {
  return (sendchar(ch));
}

int fgetc(FILE *f) {
  return (sendchar(getkey()));
}


int ferror(FILE *f) {
  /* Your implementation of ferror */
  return EOF;
}


void _ttywrch(int ch) {
  sendchar (ch);
}


void _sys_exit(int return_code) {
  while (1);    /* endless loop */
}
void usart0_init(void)
{
        VPBDIV = 0x02;        // Divide Pclk by two
        PINSEL0= (PINSEL0 & (~(0x0F<<0)))|(5<<0); /*Select the AD0.3 of P0.30*/
        U0LCR &= 0xFC;//
        U0FCR=0x07;             //enable and clear FIFOs
        U0LCR=0x83;             //8-N-1, enable divisors
        U0DLL=0xc3;             //9600 baud (9615)
        U0DLM=0x00;
        U0LCR=0x03;             //8-N-1, disable divisors
}

void usart1_init(void)
{
        PINSEL0= (PINSEL0 & (~(0x0F<<16)))|(5<<16); /*Select the AD0.3 of P0.30*/
        U1LCR &= 0xFC;//
        U1FCR=0x07;             //enable and clear FIFOs
        U1LCR=0x83;             //8-N-1, enable divisors
        U1DLL=0xc3;             //9600 baud (9615)
        U1DLM=0x00;
        U1LCR=0x03;             //8-N-1, disable divisors
}
```

```
#define CR     0x0D


/* implementation of putchar (also used by printf function to output data)    */
int sendchar (int ch)
{               /* Write character to Serial Port    */
        if(usart_print==0)
        {
                if (ch == '\n')
                {
                while (!(U0LSR & 0x20));
                U0THR = CR;                     /* output CR */
                }
        while (!(U0LSR & 0x20));
        return (U0THR = ch);
        }

        else
        {
                if (ch == '\n')
                {
                while (!(U1LSR & 0x20));
                U1THR = CR;                     /* output CR */
                }
        while (!(U1LSR & 0x20));
        return (U1THR = ch);
        }

}

int getchar_attempt (unsigned int time)
{               /* Read character from Serial Port   */
        if(usart_print==0)
        {
                while (!(U0LSR & 0x01))
                {
                delay_ms(1);
                time--;
                }
        return (U0RBR);
        }
        else
        {
                while (!(U1LSR & 0x01))
                {
                delay_ms(1);
```

```
                time--;
                }
        return (U1RBR);
        }
}

void usart_flush ()
{               /* Read character from Serial Port   */
        if(usart_print==0)
        {
                while ((U0LSR & 0x01))
                {
                getchar();
                }
        return;
        }
        else
        {
                while ((U1LSR & 0x01))
                {
                getchar();
                }
        return;
        }
}

int getkey (void)
{               /* Read character from Serial Port   */
//char rtn_temp;
        if(usart_print==0)
        {
        while (!(U0LSR & 0x01));
        return(U0RBR);
        }
        else
        {
        while (!(U1LSR & 0x01));
        return(U1RBR);
        }
        //if((rtn_temp==10)||(rtn_temp==13))
        //{
        //usart_flush();
        //}
//return (rtn_temp);
}

short check_usart(void)
```

```
{
        if (U0LSR & 0x01)
        {
        return 0;
        }
        if (U1LSR & 0x01)
        {
        return 1;
        }
        return 255;
}
```

pll.h:

```
extern void INIT_PLL(void);
void INIT_PLL()

{
 PLL0CFG=0x00000024;            // TO GENERATE 60 MHZ CCLK
 PLL0CON=0x00000001;
 PLL0FEED=0x000000AA;           // UPDATE THE PLL REGISTER
 PLL0FEED=0x00000055;
 while(!(PLL0STAT & 0x00000400)); // CHECK WHETHRT THE CCLK IS GENERATED EXAXT VALUE
 PLL0CON=0x00000003;            // CONNECT PLL
 PLL0FEED=0x000000AA;           // UPDATE THE PLL REGISTER
 PLL0FEED=0x00000055;
 VPBDIV=0x00000002;             // PCLK=1/2*CCLK
}
```

serial.h:

```
extern char sendchar (char SDat);
extern void init_serial (void);
extern int getchar (void);

void init_serial (void)  {                    /* Initialize Serial Interface      */
  PINSEL0 = PINSEL0 | 0X00000005;       /* Enable RxD0 and TxD0             */
  U0LCR = 0X83;                      /*8 bits, no Parity, 1 Stop bit     */
  U0DLL = 0XC3;
  U0DLM = 0X00;                                                          /* 9600bps baud rate */
  U0LCR = 0X03;                                          /* DLAB = 0                    */
  }

char sendchar (char SDat)  {              /* Write character to Serial Port    */
  while (!(U0LSR & 0x20));
  return (U0THR = SDat);
  }
```

```
int getchar (void)  {                /* Read character from Serial Port   */
  while (!(U0LSR & 0x01));
    return (U0RBR);
  }
```

sevensegment.h:

```
unsigned int  srck  = 0x00010000;       // P0.0
unsigned int  ser   = 0x00020000;       // P1.1
unsigned int  rck   = 0x00040000;       // P1.3

void delay(int);
void convert_display(unsigned int);
void Clear (void);
int display(unsigned char);
unsigned char disp[10] = { 0xC0,0xcf,0xa4,0xB0,0x99,0x92,0x82,0xf8,0x80,0x90 };

void Clear (void)
          {
           unsigned int x;
           for (x=0;x<6;x++)
                    {
                     display(0xFF);
                    }
          }

void delay(int n)                                   /* generates one milisecond delay  */
  {
   int i,j;
   for (i=1; i<=n; i++)
   for(j=0; j<=10000; j++);
   }



int display(unsigned char value)
{
          char m;
          int buffer;
          buffer = value;
                    for(m=0;m<8;m++)
                    {
                              if(buffer&0x80) IO0SET |= ser;
                              else IO0CLR |= ser;
                              IO0SET |= srck;
```

```
                        IO0CLR |= srck;
                        buffer<<=1;
                }
        IO0SET |= rck;
        ;
        IO0CLR |= rck;
        return 0;
}


void convert_display(unsigned int value)
        {
         display(disp[(value % 10/1)]);
         display(disp[(value % 100/10)]);
         display(disp[(value % 1000/100)]);
          display(disp[(value % 10000/1000)]);
          display(disp[(value % 100000/10000)]);
          display(disp[(value % 1000000/100000)]);
          return;
        }
```

EXTRA PROGRAMS

DC MOTOR INTERFACE:

```
#include <lpc214x.h>
#include <delay.h>

int main()
{
IODIR0=(1<<0)|(1<<3);

        while(1)
        {
         IOSET0=(1<<0);
         delay_ms(1000);
         IOCLR0=(1<<0);
         delay_ms(1000);
         IOSET0=(1<<3);
         delay_ms(1000);
         IOCLR0=(1<<3);
         delay_ms(1000);
        }
}
```

SEVEN SEGMENT DISPLAY:

```
#include <lpc214x.h>
#include <sevensegment.h>

unsigned int count = 0;
int main()
{
   IO0DIR = 0x000F0000;
        Clear();
        while(1)
        {
         convert_display(count);
         count++;
         delay(100);
         if(count>999999)count=0;
         Clear();
        }
}
```

VIVA QUESTIONS

STUDY OF ARM EVALUATION SYSTEM

1) What are the basic units of ARM 7?
2) What is the address system supported by ARM?
3) Define RISK.
4) What are the instructions used to access the memory in ARM?
5) How are the instructions encoded in ARM machines?
6) What are the basic units of Microprocessor?
7) What is an Instruction?
8) What is clock cycle?
9) Define - RTOS
10) Define - Pipelining
11) What is cache bus?
12) What is cache miss?
13) What is meant by memory mapped I/O?
14) Define – interrupt
15) What is the function of device driver?
16) What is meant by Exception?
17) What is meant by little – endian mode?
18) What is meant by big – endian mode?
19) What is the function of CPSR?
20) What is register - indirect addressing?
21) What are the steps in programming GPIO pins of LPC2148?
22) How to interface an LED with a microcontroller?
23) Explain the PORT pin details of your program.
24) Explain IO0SET and IO0CLR registers.
25) How to disable all the LEDs from being programmed?

FLASHING OF LEDS

1. What is seven segment displays?
2. What are the different configurations of LED?
3. What is the function of GPIO?
4. What are the Pins which are used to connect LEDs?
5. How to identify 'Polarity' of LED?
6. Differentiate LED from LCD.
7. What is a use of Jumper?
8. How many numbers of LEDs present in Primer board?
9. Where LEDs are used?
10. What is the use of flash magic software?
11. Which port is used in ARM 7 processor kit?
12. Which diode suffers an avalanche breakdown?
13. What happens if the junction temperature of LED is increased?
14. What is an anti collision rate light?
15. What is fail-safe conditions

STEPPER MOTOR INTERFACE

1) Define GPIO.
2) What is the function of ULN2803?
3) How LPC2148 control stepper motor?
4) Which I/O port lines used to rotate stepper motor?
5) How stepper motor reacts for each pulse it receives?

6) What is serial communication?
7) What is parallel communication?
8) What is stepper motor & why it is named so?
9) How can be step angle is calculated?
10) What are the advantages and disadvantages of parallel communication?
11) What does instruction-pipelining mean?
12) What is the power supply required for ARM processor?
13) What is a unit for torque?
14) What is RPM rating for a DC motor?
15) What device that is used to obtain an accurate position control of rotating shafts in terms of steps?

TEMPERATURE SENSOR

1) What is LM35?
2) Define ISP.
3) How many Analog-to-Digital Converters available as an inner peripheral in LPC2148?
4) What is the main function of analog pin in LPC 2148?
5) What is the output voltage produced by LM35 for one degree temperature?
6) What is the difference between LM 34 and LM 35 sensors?
7) How many pins are available in LM35?
8) Why Vref is set of ADC0848 to 2.56 V if analog input is connected to the LM35?
9) What is signal conditioning?
10) What is operating the operating temperature range in LM35?
11) Why LM35 is used to Measure Temperature?
12) What Does an LM35 Do?
13) List the devices used to sense temperature.
14) What is the purpose of a thermocouple?
15) What is the output voltage of a thermocouple?

ADC

1. What are the ADC operating modes in LPC2148?
2. What is meant by conversion time?
3. Why are internal ADCs preferred over external ADCs?
4. Why has 16-bit ADC more resolution than from 10-bit and 8-bit ADC?
5. What are ADC & its Resolution?
6. What is ADC operating frequency range?
7. What is Burst conversion mode?
8. What is the function of A/D Control Register
9. What is the function of A/D Global Data Register
10. What is the function of A/D Status Register
11. What is analog-to-digital (ADC) conversion error?
12. What is the function of Sample-and-hold circuits in analog-to digital converters?
13. What is meant by flash analog-to-digital converter?
14. What is the disadvantage of the flash analog-to digital converter?
15. What is the data rate available for use on USB?

DAC

1. What are the features of LPC2148 DAC?
2. What is the function of DAC register?
3. Which pin provides a voltage reference level for the D/A converter?
4. What are the applications of DAC in signal processing applications?
5. What is GIO?

6. What is the resolution of a digital-to-analog converter (DAC)?
7. What is the major advantage of the R/2R ladder digital-to-analog (DAC), as compared to a binaryweighted
digital-to-analog DAC converter?
8. What is the resolution of a 0–5 V 6-bit digital-to-analog converter (DAC) ?
9. What are DAC configurations?
10. What are the different types of DAC Circuits?
11. What are the Limitations of the Binary Weighted DAC?
12. What are the General DAC Characteristics?
13. Define reference voltage in DAC.
14. What is full scale voltage?
15. What is settling time?

LED DIMMING USING PWM

1. What factors determine the LED dimming performance?
2. What is meant by analog dimming?
3. What is flickering effect?
4. What is the factor determines dimming of the LED
5. Why Silicon is not suitable for fabrication of light-emitting diodes?
6. Why Silicon diode is less suited for low voltage rectifier operation?
7. What is the principle of zener diode?
8. What are the materials used to make LED?
9. What type of diodes is used in FM receivers?
10. Which diode suffers an avalanche breakdown?
11. What happens if the junction temperature of LED is increased?
12. How a transfer characteristic of a diode does is related?
13. What is the requirement of a clipping action of a diode?
14. What type of rectifier circuit requires four diodes?
15. Why are the pulse width modulated outputs required in most of the applications?
16. How do the variations in an average value get affected by PWM period?
17. Name the common formats available for LED display.
18. What are the types of seven segment display?
19. What is PWM?
20. What are the applications of PWM?

REAL TIME CLOCK AND SERIAL PORT

1) What is I2C and how does it work?
2) What are the features of I2C in LPC2148 ARM7 microcontroller?
3) What is the function of I2C0CONSET register?
4) What is I2C0STAT?
5) What is the function of I2C0 Data Register?
6) What is RTC?
7) List the advantages of RTC
8) Through which port the date and time is displayed in RTC?
9) What is a serial port?
10) List the registers used to transfer data in serial port.
11) List the modes used for data transmission.
12) What is a simplex mode?
13) Which pin is used to transmit a character?
14) What is the baud rate of serial port?
15) What is the disadvantage of RS-232C?

KEYPAD AND LCD

1. What is Matrix keypad?
2. What is the concept behind keypad interface?
3. What are the functions of pull- up resistors?
4. What is key de bouncing?
5. List the steps involved when the key in a 4 x 4 keyboard matrix is being pressed.
6. What is the value obtained if no key is pressed?
7. What kind of interrupt is generated if a key has to be operated in an interrupt mode?
8. How will you identify that the key is pressed?
9. What are the steps involved in Keyboard Interfacing?
10. List the registers used to store the keyboard, display modes and other operations programmed by CPU.
11. Name the mode of operation for a de bounce logic.
12. Name the mode when a data is entered from the left side of the display unit.
13. How many rows and columns are present in a 16 x 2 alphanumeric LCD?
14. How many data lines are there in a 16 x 2 alphanumeric LCD?
15. Which pin of the LCD is used for adjusting its contrast?
16. Which command of an LCD is used to shift the entire display to the right?
17. Which command is used to select the 2 lines and 5 x 7 matrix of an LCD?
18. What changes are to be made to send data to an LCD?
19. For reading operation from an LCD what changes in the software are introduced?
20. Which instruction is used to select the first row first column of an LCD?

EEPROM AND INTERRUPT

1) How does I2C protocol work?
2) What is EEPROM?
3) What is meant by master slave mode?
4) What is meant by non-volatile memory?
5) What is an interrupt?
6) How does an interrupt request works?
7) What is a nested interrupt?
8) What are the registers used for enabling an interrupt?
9) What is PROM?
10) What is EPROM?
11) What is mask ROM?
12) Which memory allows simultaneous read and write operations?
13) Which memory has the shortest access times?
14) Which interrupt has the highest priority?
15) What is non maskable interrupt?
16) Which is the first level of memory access by the microprocessor?
17) List the types of cache memories.
18) What is the use of an interrupt?
19) What is the use of converting an interrupt to threads in a microprocessor?
20) What are the bits used to control an external interrupts?

MAILBOX FUNCTION

1) What is IPC?
2) Define Queue.
3) What are the different types of mail box functions?
4) What is half duplex mode?
5) How many synchronous and asynchronous modes are there in serial port?
6) List the bits used to generate serial port interrupt

7) What is branch prediction?
8) Mention any two features of µC/OS – II.
9) Write the function for creating task in µC/OS – II.
10) Mention any two support devices for µC/OS – II.
11) How many user tasks can be handled by µC/OS – II?

INTERRUPT PERFORMANCE

1. When is an Interrupt Request received?
2. What is ISR?
3. How do you initialize interrupt?
4. What is Vectored Interrupt Controller?
5. What are the kinds of protection available for SRAMS ?
6. What is interrupt pipelining?
7. What is pipeline shutdown?
8. What is branch prediction?
9. What is the use of Neon Floating point engine?
10. What is the use of –vectorize option?
11. What is PTM?
12. What is ITM?
13. What is ETM?
14. What is I2S interface?
15. What is the use of "SWI" in ARM assembly?

ZIGBEE WIRELESS MODULE

1. How many UART ports available in LPC2148?
2. What is the main function of voltage convertors in UART?
3. Why Zigbee based is preferred for wireless communication?
4. What is the IEEE standard for Zigbee protocol?
5. Write the two modes of communication are used in a ZigBee network.
6. What is the function of a scheduler?
7. Mention any two features of µC/OS – II.
8. Write the function for creating task in µC/OS – II.
9. Mention any two support devices for µC/OS – II.
10. How many user tasks can be handled by µC/OS – II?

ADDITIONAL QUESTIONS:

BUZZER

1. Define Buzzer.
2. Which pin in ARM can be used to modulate Buzzer?
3. What is the function Flash magic software?
4. How the Buzzer can be disconnected in LPC2148?
5. How the volume of sound will be changed in Buzzer?
6. What is a buzzer
7. What are the applications of buzzer?
8. How buzzer is used in embedded system?
9. What is race round condition?
10. What is digital signal controller?
11. What are the features of buzzer circuit?
12. What is a semaphore?
13. What is mutexes?
14. What is the role of segment register?
15. What is watch dog timer?

RELAY

1. What is the function of relay?
2. What is the function of Opto-isolator?
3. What is the function of power transistor in LPC2148?
4. Which pin acts as an output port for relay?
5. What is a relay circuit
6. what are the applications of relay circuit
7. what are the advantages of relay circuit
8. How does combination of functions reduce memory requirements in embedded systems?
9. What is the need for DMAC in ES?
10. Mention how I/O devices are classified for embedded system?
11. Explain what is interrupt latency? How can you reduce it?
12. What does DMA address will deal with?
13. List out various uses of timers in embedded system?
14. List out some of the commonly found errors in Embedded Systems?
15. What is the difference between macro and line function.