Jawaharlal Nehru Engineering College

Laboratory Manual

EMBEDDED SYSTEM

For

Final Year Students

Manual made by
Prof. Salunke A.R.
Prof. Ukarande V.V.
Prof. Yadav V.P.

Author JNEC, Aurangabad

MGM'S
Jawaharlal Nehru Engineering College

N-6, CIDCO, Aurangabad

Department of Electronics &Telecommunication

Vision of the Department:

To develop **GREAT** technocrats and to establish centre of excellence in the field of **Electronics and Telecommunications.**

- ***G***lobal technocrats with human values
- ***R***esearch and lifelong learning attitude,
- ***E***xcellent ability to tackle challenges
- ***A***wareness of the needs of society
- ***T***echnical expertise

Mission of the Department:

1. To provide good technical education and enhance technical competency by providing good infrastructure, resources, effective teaching learning process and competent, caring and committed faculty.
2. To provide various platforms to students for cultivating professional attitude and ethical values.
3. Creating a strong foundation among students which will enable them to pursue their career choice.

Jawaharlal Nehru Engineering College

Technical Document

This technical document is a series of Laboratory manuals of Electronics & Telecommunication and is a certified document of Jawaharlal Nehru Engineering College. The care has been taken to make the document error free but still if any error is found kindly bring it to the notice of subject teacher and HOD.

Recommended by,

HOD

Approved by,

Principal

Copies:

- Departmental Library
- Laboratory
- HOD
- Principal

FOREWORD

It is my great pleasure to present this laboratory manual for final year engineering students for the subject of Embedded Systems keeping in view the vast coverage required for visualization of concepts of basic electronic circuits.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly that has been tried to answer through this manual.

Faculty members are also advised that covering these aspects in initial stage itself will greatly relieve them in future, as much of the load will be taken care by the enthusiastic energies of the students, once they are conceptually clear. Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions.

Prof. Salunke A.R.
Prof. Ukarande V.V.
Prof. Yadav V.P.

Author

SUBJECT INDEX

1. Do’s and Don’ts

2. Lab exercise:

    1. Study Of EDU-ARM-2148 Trainer kit
    2. Study of IDE overview-Project creation, downloading &debugging.
    3. Write a Program to Blink LED: Generate any four random patterns on LED Matrix.
    4. Write a program to interface stepper motor.
    5. ARM to PC communication via UART Transmit a message via UART of ARM and display it on terminal of PC
    6. Write a program to interface LCD
    7. Write a program to interface 4*4 matrix keyboard
    8. Write a program to interface 7SEG (7 segment display).
    9. Write a program to interface buzzer with ARM
    10. Write a program to interface 2 relays with LPC2148
    11. Write a program to generate a square wave using ARM

3. Quiz on the subject.

4. Conduction of Viva-Voce Examination.

5. Evaluation and Marking System.

1. DO's and DON'Ts in Laboratory:

1. Do not handle kit without reading the instructions/Instruction manuals.
2. Refer Help for debugging the program.
3. Go through Demos of Signal Processing tool box.
4. Strictly observe the instructions given by the teacher/Lab Instructor.

2 Instruction for Laboratory Teachers:

1. Lab work completed during prior session should be corrected during the next lab session.

2. Students should be guided and helped whenever they face difficulties.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

Experiment:1

Title- EDU-ARM-2148 Trainer kit

Aim:-Study of EDU-ARM-2148 Trainer kit.

Apparatus: - EDU-ARM-2148 Trainer kit.

Theory:-

The trainer kit contains following items:

1. EDU-ARM7-2148 board
2. Serial communication Cable
3. Power Supply Adaptor
4. SPJETs CD-ROM

Power Supply Requirement:

The power adaptor works with 230VAC. It produces approximately 9V DC, and the EDU-ARM-2148 uses on board regulators to provide 5V, 3.3V and 1.8V DC to all components on board.

Connecting the system:

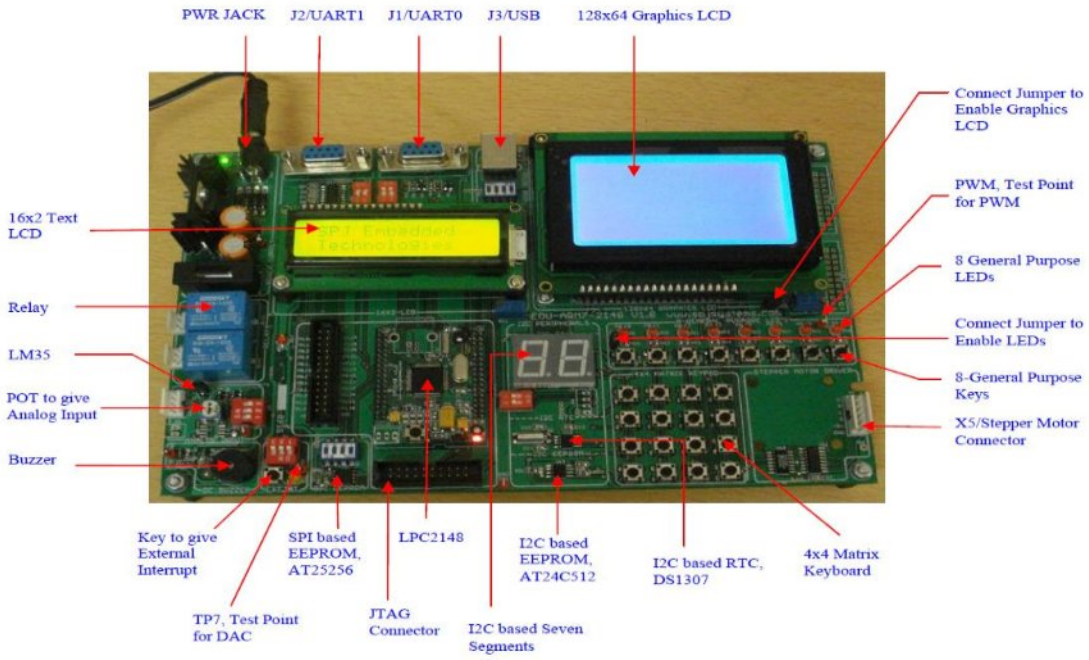
The serial communication cable supplied with the board should be used to connect the board to a PC running windows95/98/ME/2000/XP/Vista Operating system. Connect one end of the serial cable to UART0 of EDU-ARM7-2148 board and other end to PCs serial port.

Powering ON:

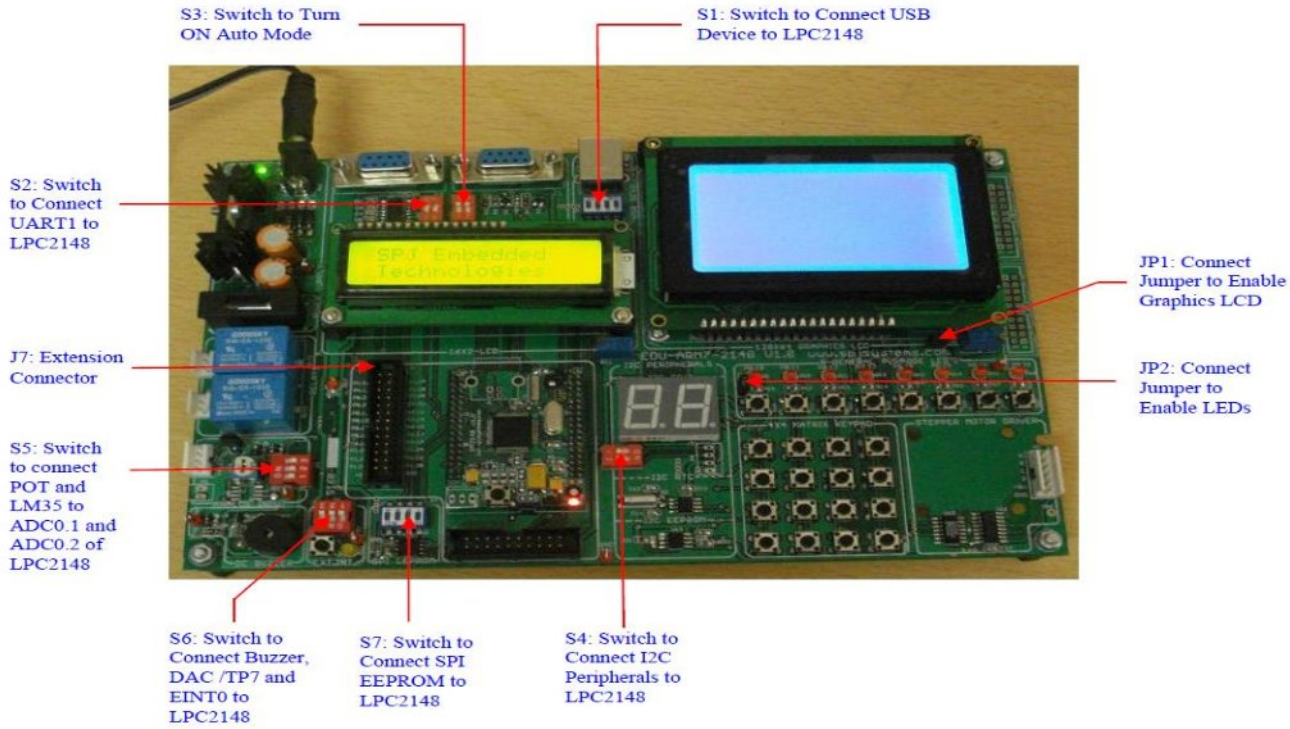
After connecting the serial communication cable as described above, you may insert the power adaptor output jack into the on-board power socket. Plug the power adaptor into 230 VAC mains outlet and turn it on. The power-on indication green LED will turn on.

A. **EDU-ARM7-2148 Block Diagram:**

Below figure shows the locations of different components on the EDU-ARM7-2148 board.



Below figure shows the locations of different switches and jumpers on the EDU-ARM7-2148 board.



B. Switches Details:

S1: Turn ON this switch to connect USB device connector to USB lines of LPC2148.

S2: Turn ON this switch to connect UART1 connector to UART1 lines (TxD1/P0.8 and RxD1/P0.9) of LPC2148.

S3: Mode selection switch. The LPC21xx micro-controllers include on-chip flash for storing user program and non-volatile data. The LPC2148 have 512KBytes flash. This flash is In-System-Programmable (ISP). The LPC21xx micro-controllers have a built-in boot-load program. Upon power-on, this bootload program takes control; it passes control to the user program if pin P0.14 is HIGH and some other conditions are satisfied. Please refer to the LPC21xx data-sheet for further details. On the EDU-ARM7-2148 board, the P0.14 pin is made available on this S3 switch. Turn ON this switch to control the Mode (ISP mode or Run mode) by Flash Magic.

S4: Turn ON this switch to connect Seven Segments, RTC (DS1307) and EEPROM (AT24C512) to I2C lines (SCL0/P0.2 and SDA0/P0.3) of LPC2148.

S5: Turn ON this switch to connect POT and LM35 to ADC0.1/P0.28 and ADC0.2/P0.29 of LPC2148.

S6: Turn ON this switch to connect Buzzer, DAC/TP7 and External Interrupt to ACOut/P0.25 and EINT0/P0.16 of LPC2148.

S7: Turn ON this switch to connect SPI EEPROM (AT25256) to SPI lines (SCK0/P0.4, MISO0/P0.5, MOSI0/P0.6 and CS/P0.7) of LPC2148.

C. Connector Details:

UART0:

This is a DB9 female connector, used for RS232 serial communication with the PC:

Pin 2 = UART0 RS232 TxD (output of mC)
Pin 3 = UART0 RS232 RxD (input to mC)
Pin 4 = RS232 DTR
Pin 5 = Ground
Pin 7 = RS232 RTS
All other pins of J1/UART0 are unused.

UART1:

This is a DB9 female connector, used for RS232 serial communication with the PC:

Pin 2 = UART1 RS232 TxD (output of mC)
Pin 3 = UART1 RS232 RxD (input to mC)
Pin 5 = Ground

16x2 LCD:

This is a 16 pin, single line connector, designed for connection to standard, text LCD modules. The pin/signal correspondence is designed to be matching with that required by such LCD modules.

Pin 1 = GND
Pin 2 = +5V
Pin 3 = Vlcd
Pin 4 = P1.25 (Used as RS of LCD)
Pin 5 = GND
Pin 6 = P1.24 (Used as EN of LCD)
Pin 7 to 10 = No Connection

Pin 11 = P0.15 (Used as D4 of LCD)
Pin 12 = P0.17 (Used as D5 of LCD)
Pin 13 = P0.22 (Used as D6 of LCD)
Pin 14 = P0.30 (Used as D7 of LCD)
Pin 15 = Backlighting
Pin 16 = GND

128x64 Graphics LCD:

This is a 20 pin, single line connector, designed for connection to standard, 128x64 Monochrome Graphics LCD modules. The pin/signal correspondence is designed to be matching with that required by such LCD modules.

Pin 1 = GND
Pin 2 = +5V
Pin 3 = Vlcd
Pin 4 = P1.25 (Used as RS of GLCD)
Pin 5 = P0.15 (Used as RW of GLCD)
Pin 6 = P1.24 (Used as EN of GLCD)
Pin 7 = P0.10 (Used as D0 of GLCD)
Pin 8 = P0.11 (Used as D1 of GLCD)
Pin 9 = P0.12 (Used as D2 of GLCD)
Pin 10 = P0.13 (Used as D3 of GLCD)
Pin 11 = P0.18 (Used as D4 of GLCD)
Pin 12 = P0.19 (Used as D5 of GLCD)
Pin 13 = P0.20 (Used as D6 of GLCD)
Pin 14 = P0.21 (Used as D7 of GLCD)
Pin 15 = P0.22 (Used as CS1 of GLCD)
Pin 16 = P0.30 (Used as CS2 of GLCD)
Pin 17 =
Pin 18 =
Pin 19 = +5V
Pin 20 = GND

JTAG Connector:

This standard 20 pin JTAG connector provides debugging support for the LPC21xx. This connector is mounted on top side of the board as shown in figure1. JTAG cables like SJT-S or SJT-U can be connected to this connector, while other end of the cable can be connected to PC COM port or USB port, respectively. Debugger software (like the debugger built into SCARM) allows JTAG based debugging. It is also possible to use third party JTAG based emulators /debuggers. The pin-out of JTAG Connector is given below:

Pin	Signal name		Pin	Signal name
1	3.3V		2	3.3V
3	P1.31/NTRST		4	GND
5	P1.28/TDI		6	GND
7	P1.30/TMS		8	GND
9	P1.29/TCK		10	GND
11	P1.26/RTCK		12	GND
13	P1.27/TDO		14	GND
15	NRST		16	GND
17	GND		18	GND
19	GND		20	GND

J7:

This is 26 pin dual line headers. It brings out I/O and most of the pins of the LPC21xx micro-controller. Further, 5V and GND are also made available on these connectors. These connectors are intended for use to connect external peripherals.

The pin/signal details of J7 are as below:

Pin	Signal name		Pin	Signal name
1	P1.16		2	P1.17
3	P1.18		4	P1.19
5	P0.28		6	P0.29
7	P0.30		8	P0.31
9	P0.10		10	P0.11
11	P0.12		12	P0.13
13	P0.14		14	P0.15
15	P0.16		16	P0.17
17	P1.18		18	P1.19
19	P1.25		20	P1.24
21	P1.23		22	P1.22
23	P1.21		24	P1.20
25	5V		26	GND

Conclusion- In this experiment we have studied EDU-ARM-2148 Trainer kit

Experiment:2

Title: IDE overview

Aim: - Study of IDE overview (Project creation, downloading & debugging)

Apparatus:- EDU-ARM-2148 Trainer kit.

Theory:-

A. SCARM Installation:

As a part of the SCARM software package, you should have received a CDROM. Please insert it into the CD-ROM drive and run SETUP.EXE from it. Once you start the SETUP program, follow the instructions on the screen to complete the installation. The setup program will:

· Copy the required files to your hard disk
· If there are any compressed files, un-compress them
· Create a program group for SCARM

Once you have successfully installed the software on your hard disk, you can run it by clicking on **Start/Programs/SPJ - SCARM/SIDEARM**. However, we recommend, that you go through this user's manual before you actually start using it.

SCARM is **SPJETs'** C Compiler for ARM. It also includes an **IDE** and other tools like **Debugger**, **Visual Code Generator (VCG)** and **Terminal Emulation Utility (SPJTerm)**. This document describes steps to create ARM applications in C using the SCARM.

About "Project":

What is a project?

A project is a file in which SIDEARM stores all information related to an application. E.g. it stores the name(s) of 'C' and/or Assembler source file(s), memory size to be used and other options for compiler, assembler and linker.

Opening a project:

To open an existing project file, select **Project / Open Project** from the menu.

Creating a new project: To create a new project, select **Project / New Project** from the menu.

Changing project settings: To change the project settings (such as adding or removing 'C' and/or Assembler source file(s), changing memory settings etc.), select Project / Settings from the menu.

B. SCARM Quick Start for creating 'C' applications:

1. Start the **SIDE_ARM** program (i.e. the Integrated Development Environment) from start\Programs\SPJ-SCARM.

2. From Project menu, select **Close project** (if any project is open).

3. From Project menu, select **New Project**. The Open dialog window will be displayed. Select the desired path where you wish to create this new project. (For example, C:\SPJ). CAUTION: The path and filename must not contain space or other special characters such as tab, comma, semicolon etc. In the "File name" field, type the name of the project, without any extension. For example, you may type "**PROG1**". Then click on the "**Open**" button.

4. The action in the previous step will display the "**Project Settings**" dialog window. This dialog window has 3 different parts named "**Compiler Options**", "**Linker Options**", and "**Source Files**". Any of these 3 parts can be displayed by clicking on the corresponding name near the top of this dialog window. Currently, the "**Compiler Options**" will be automatically displayed. If the target micro-controller (must be a member of ARM family) is known, you may select the appropriate Manufacturer from the list; and then select the appropriate micro-controller from the device list. If the target micro-controller is not known or if you cannot find it in the list, then you may simply select "**Philips**" as the manufacturer and "**LPC2148**" as the micro-controller.

5. Click on "**Linker Options**" to display that part of the dialog window. In this window, you will see a list of 8 "Memory Banks", with names such as "Memory #1", "Memory #2" and so on. In your target hardware, there may be none or 1 or more number of contiguous memory blocks connected to the ARM micro-controller. Check the appropriate number of memory banks to reflect the target's memory blocks. For each checked memory bank, specify memory start address (in Hexadecimal) and memory block size (in decimal). Size maybe specified either in number of Kilobytes (KB) or Megabytes (MB). Some of the memory blocks maybe "read-only" (e.g. flash or conventional EPROM). Accordingly, you may check or uncheck the "Read only" box. Based on this information about memory banks, the IDE will automatically create the Linker Script. This auto-generated script is adequate for most users. However, if you wish to use your own script file instead of this autogenerated script, you may check the "Use different linker script" box and further click on the browse button (marked "…") and select appropriate linker script file.

6. Click on "**Source Files**" to display that part of the dialog window. This window will indicate that IDE has automatically added 2 files in this new project: **PROG1.C** and **STARTUP.ASM**. The STARTUP.ASM file is automatically created by the IDE and is required for all C projects. Similarly, the IDE has automatically created an empty C file (PROG1.C). If the file PROG1.C already exists in the same path, then IDE would neither create/overwrite it nor modify it; but it

will anyway add it to the project automatically. If you wish to add more files in this project, then click on the "**Add file**" button, select the desired filename and then click on "**Open**" button. Now the Project Settings dialog will indicate that selected file has been added into the project. When all necessary files have been added to the project, click "**OK**" button to create this new project.

7. The PROG1.C file created by the IDE will be an empty file containing only the frame of "main" function. You may write the desired program statements in this file (or other files that you may have added to the project). When done, select **Save** from **File menu**. If you have modified more than one source files, then select **Save All** from **File menu**.

8. From the **Compile** menu, select **Build**. This will invoke the Compiler to compile the file PROG1.C; and further (assuming no errors) invoke the linker to create the **.HEX** file. If there are any errors or warnings during the process of compiling, assembling or linking, then those will be displayed in the output window (below the editor window). If there are errors, then you may correct those by making appropriate changes to the program; select Save from File menu to save the changes and then again select Build from Compile menu. Repeat this until there are no errors.

9. You may inspect contents of the folder where your project files reside. When there are no errors and build has completed successfully and then you will see a filename with same name as the project name and extension .HEX (in above example, **PROG1.HEX**). This is the file that you will need to use to program your micro-controller.

C. Downloading & Running Programs:

The LPC2148 micro-controllers include on-chip flash for storing user program and non-volatile data. LPC2148 on EDU-ARM7-2148 have 512KBytes flash. This flash is In-System-Programmable (ISP). Therefore it is possible to download user program into on-chip flash of LPC2148, through serial port connected to PC. For doing so, a certain position of S3 switch is required. **S3 Switch should be continuously ON**. This section describes how to use the software Flash Magic to download program into LPC2148.

1. How to install Flash Magic:

The CD you have received with this board contains SCARM, C Compiler for ARM. Install it. After installation go to folder **C:\SCARM\Utilities**. This folder contains 5 zip files. Install Flash Magic from FlashMagic3.71.zip. Extract the **FlashMagic3.71.zip** and then run FlashMagic.exe from the extracted files. (If you have wrong version of Flash Magic already installed, then please uninstall it first and then install new version).

2. Download and Run program using Flash Magic into LPC2148:

· After installation of Flash Magic, open it.

· In Flash Magic go to Options -> Advanced Options-> Communications. Check High Speed

Communications and keep Maximum Baud Rate as 19200. Click on OK.

Again in Flash Magic go to Options -> Advanced Options-> Hardware Config. “Use DTR and RTS to control RST and P0.14” option should be checked. Click on OK.

(After doing above mentioned settings, Flash Magic stores it means for the next time just verify if these setting are proper or not. If they are proper then you can directly follow below mentioned procedure)

a) Connect the J1/UART0 connector of EDU-ARM7-2148 board to COM1 or COM2 of a PC, using the serial communication cable (supplied with the board).

b) Keep S3 switch in ON position. (You can keep S3 switch continuously ON) Switch ON power to the EDU-ARM7-2148.

c) Do proper settings in Flash Magic (COM Port: COM1 (if other choose it), Baud Rate: 38400, Device: LPC2148, Interface: None (ISP), Enable “Erase blocks used by Hex File”, Browse the file which you want to download) and click on Start button.

d) Flash Magic will download the program. Wait till Finished comes.

e) After downloading Flash Magic automatically resets the EDU-ARM7-2148 board and program executes. You can see output according to the program.

f) If again you want to Reset the board then Switch OFF and ON the power to the EDU-ARM7-2148 board. You can see output according to the program.

Note: Flash Magic can be used to download the program into other Philips Microcontrollers also. See the list in Flash Magic itself.

Conclusion- In this experiment we have studied IDE overview (Project creation, downloading & debugging)

Experiment:3

Title: LED blinking

Aim: Write a Program to Blink LED: Generate any four random pattern on LED Matrix.

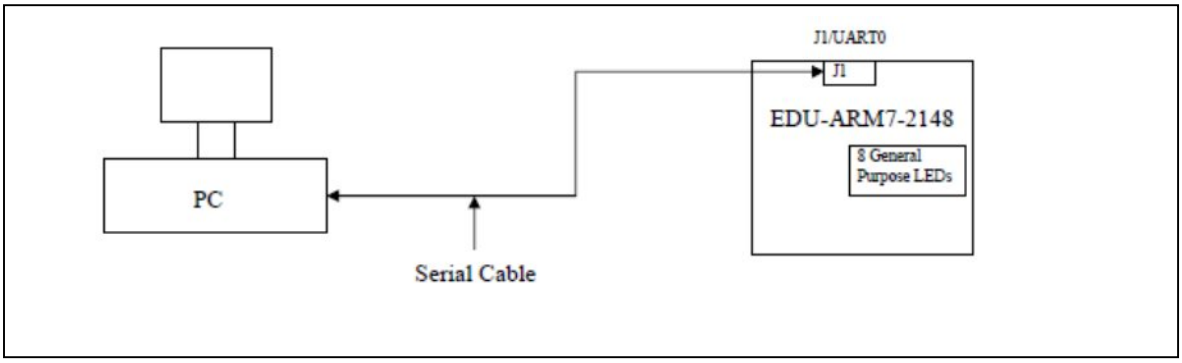
Equipment's: SCARM, PC, EDU-ARM7-2148.

Theory: - Write theory related to Interfacing of LED

Connections:
Connect Jumper JP2 which is near to LED D9.
8 LEDs (D9 to D16) present on EDU-ARM7-2148 are connected to P0.10, P0.11, P0.12, P0.13, P0.18, P0.19, P0.20 and P0.21 respectively by Common Anode method.

Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Block Diagram:



Source Code:
LED pattern 1: All Blink

```
#include <Philips\LPC2148.h>
unsignedint delay;

void main ()
{
        PINSEL0 = 0x00000000 ;
        PINSEL1 = 0x00000000 ;
        IO0DIR = 0x003C3C00 ;
```

```
	while(1)
	{
		IO0CLR = 0x003C3C00;
		for(delay=0; delay<50000; delay++)
		{}
		IO0SET = 0x003C3C00;
		for(delay=0; delay<50000; delay++)
		{}
	}

}
```

LED pattern 2: Only Four Blink

```
#include <Philips\LPC2148.h>
unsignedint delay;

void main ()
{
	PINSEL0 = 0x00000000 ;
	PINSEL1 = 0x00000000 ;
	IO0DIR = 0x003C0000 ;

	while(1)
	{
		IO0CLR = 0x003C0000;
		for(delay=0; delay<50000; delay++)
		{}
		IO0SET = 0x003C0000;
		for(delay=0; delay<50000; delay++)
		{}
	}

}
```

LED pattern 3: Alternate Blink

```
#include <Philips\LPC2148.h>

unsignedint delay;
```

```
void main ()
{
	PINSEL0 = 0x00000000 ;
	PINSEL1 = 0x00000000 ;
	IO0DIR = 0x00141400 ;

	while(1)
	{
		IO0CLR = 0x00141400;
		for(delay=0; delay<50000; delay++)
		{}
		IO0SET = 0x00141400;
		for(delay=0; delay<50000; delay++)
		{}
	}

}
```

LED pattern 4: Alternate Two Blink

```
#include <Philips\LPC2148.h>

unsignedint delay;

void main ()
{
	PINSEL0 = 0x00000000 ;
	PINSEL1 = 0x00000000 ;
	IO0DIR = 0x000C0C00 ;

	while(1)
	{
		IO0CLR = 0x000C0C00;
		for(delay=0; delay<50000; delay++)
		{}
		IO0SET = 0x000C0C00;
		for(delay=0; delay<50000; delay++)
		{}
	}
}
```

Note: Remove jumper JP2 to save power, after execution of program.

Output:
You can see blinking of LEDs

Experiment:4

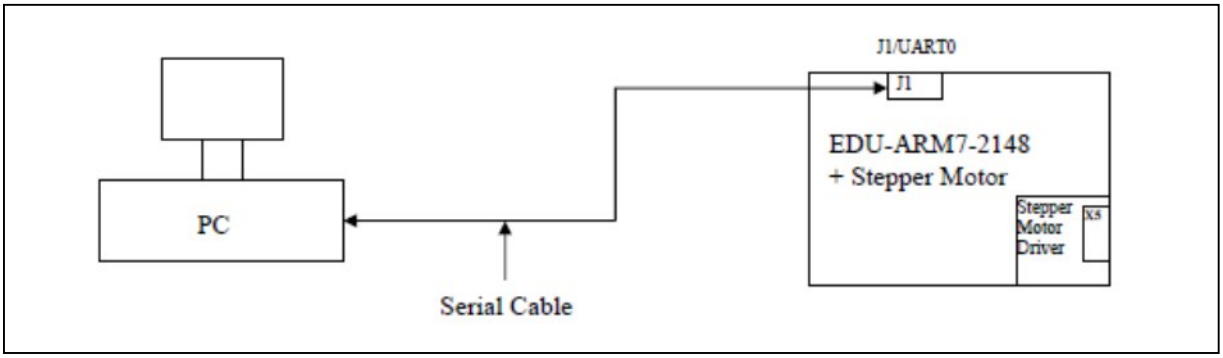
Title: Stepper motor Interfacing.

Aim: Write a program to interface stepper motor.

Equipment's: SCARM, PC, EDU-ARM7-2148 with Stepper Motor.

Theory: - Write theory related to Interfacing of stepper motor

Block Diagram:



Connections:
Connect Jumper JP2 which is near to LED D9 in 8 General Purpose LEDs region.
Stepper Motor is connected to P0.10, P0.11, P0.12 and P0.13 through LEDs D9 to D12. Connect Stepper motor connector to X5.

Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Source Code:

```
#include <Philips\LPC2148.h>

#define PHASEA 0x00002400
#define PHASEB 0x00001400
#define PHASEC 0x00001800
#define PHASED 0x00002800

unsignedint delay ;
```

```
void main ()
{
	PINSEL0 = 0x00000000 ;
	PINSEL1 = 0x00000000 ;
	IO0DIR = 0x003C3C00 ;
	IO0SET = 0x003C0000 ;

	//PINSEL2 = 0x00000000 ;
	//IO1DIR = 0x00000000 ;

	while(1)
	{
		IO0SET = PHASEA ;
		IO0CLR = (~PHASEA) &0x00003C00 ;
		for(delay=0; delay<300000; delay++) ;

		IO0SET = PHASEB ;
		IO0CLR = (~PHASEB) &0x00003C00 ;
		for(delay=0; delay<300000; delay++) ;

		IO0SET = PHASEC ;
		IO0CLR = (~PHASEC) &0x00003C00 ;
		for(delay=0; delay<300000; delay++) ;

		IO0SET = PHASED ;
		IO0CLR = (~PHASED) &0x00003C00 ;
		for(delay=0; delay<300000; delay++) ;

	}
}
```

Note: Remove jumper JP2 and Stepper Motor connector from X5 to save power, after execution of program.

Output:

You can see stepper motor moving in a particular direction and corresponding phase changes you can observe on LEDs D9 to D12.

Experiment:5

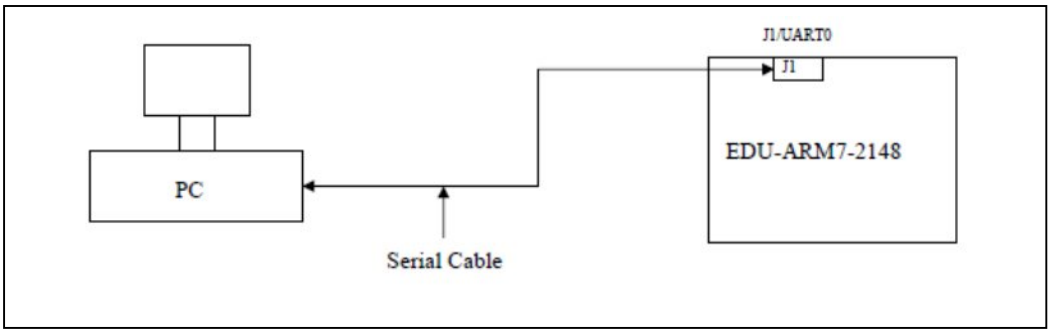
Title: ARM to PC communication via UART Transmit a message via UART of ARM and display it on terminal of PC

Aim: Write a Program to transfer massage “Hello world!” serially at 19200-baud rate 8-bit data and 1 stop-bit using UART0.

Equipment’s: SCARM, PC, EDU-ARM7-2148.

Theory: - Write theory related to Interfacing of LED

Block Diagram:



Connections:
Connect PCs serial port to J1/UART0 connector on EDU-ARM7-2148 by the cable provided to youwith EDU-ARM7-2148.

Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Source Code:

```
#include <Philips\LPC2148.h>
#include <stdio.h>

void main ()
{
        PINSEL0 = 0x00000005 ;
```

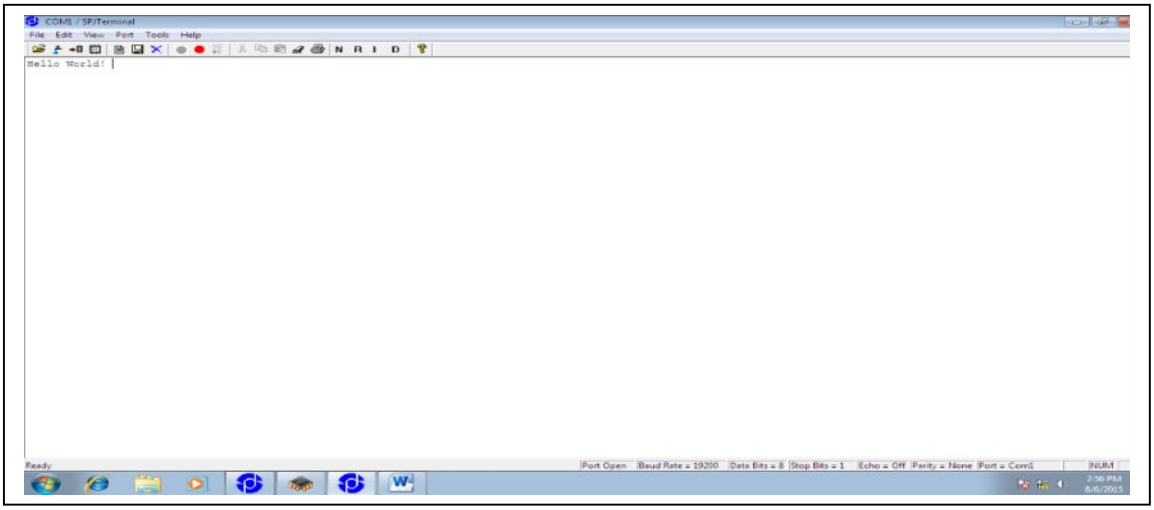
```
	InitUart0();

	printf("Hello World! \n");

	while(1)
	{
		putchar(getchar());
	}
}
```

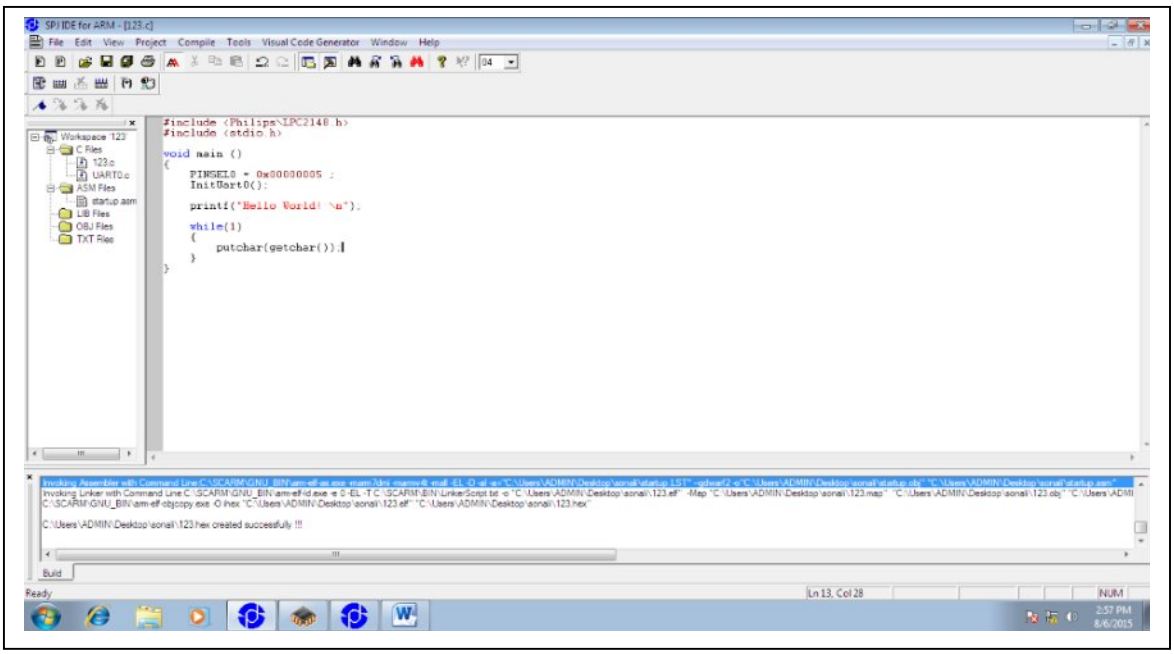
UART Drivers add to main program

```
#include <Philips\LPC2148.h>
//#include <Philips\LPC2138.h>

#defineVPBDIV			(*((volatile WORD32 *) 0xE01FC100))
#defineU0RBR			(*((volatile WORD32 *) 0xE000C000))

#define DESIRED_BAUDRATE 19200

#define CRYSTAL_FREQUENCY_IN_HZ 12000000
#define PCLK CRYSTAL_FREQUENCY_IN_HZ  /* since VPBDIV=0x01	*/
#define DIVISOR (PCLK/(16*DESIRED_BAUDRATE))

void InitUart0(void)
{
/*	U0LCR: UART0 Line Control Register
	0x83: enable Divisor Latch access, set 8-bit word length,
	1 stop bit, no parity, disable break transmission		*/
	U0LCR=0x83;

/*	VPBDIV: VPB bus clock divider
	0x01: PCLK = processor clock		*/
	VPBDIV=0x01;

/*	U0DLL: UART0 Divisor Latch (LSB)	*/
	U0DLL=DIVISOR&0xFF;
```

```
/*      U0DLM: UART0 Divisor Latch (MSB)        */
        U0DLM=DIVISOR>>8;

/*      U0LCR: UART0 Line Control Register
        0x03: same as above, but disable Divisor Latch access        */
        U0LCR=0x03 ;

/*      U0FCR: UART0 FIFO Control Register
        0x05: Clear Tx FIFO and enable Rx and Tx FIFOs               */
        U0FCR=0x05 ;
}

charputchar(char ch)
{
        if (ch=='\n')
        {
                //wait until Transmit Holding Register is empty
                while (!(U0LSR&0x20)) {}

                //then store to Transmit Holding Register
                U0THR='\r';
        }

        //wait until Transmit Holding Register is empty
        while (!(U0LSR&0x20)) {}
        //then store to Transmit Holding Register
        U0THR=ch;

        returnch;
}

chargetchar(void)
{
        charch;

        //wait until there's a character to be read
        while (!(U0LSR&0x01)) {}

        //then read from the Receiver Buffer Register
```

```
    ch=U0RBR;
    returnch;
}
```





Note: Keep S2 switch in OFF position if the program uses UART0, otherwise it should be ON. Go to Port -> Settings. Do proper settings (Baud Rate: 19200, Data Bits: 8, Stop Bits: 1, Echo: Off, Parity: None, Com Port: Com 1 (if other choose it)). Click on OK. Go to Port -> Open. If required Reset the EDU-ARM7-2148 board.

Output:

You can see output on SPJ Terminal. Therefore Open SPJ Terminal. It will transmit the message "Hello world!".

Experiment:6

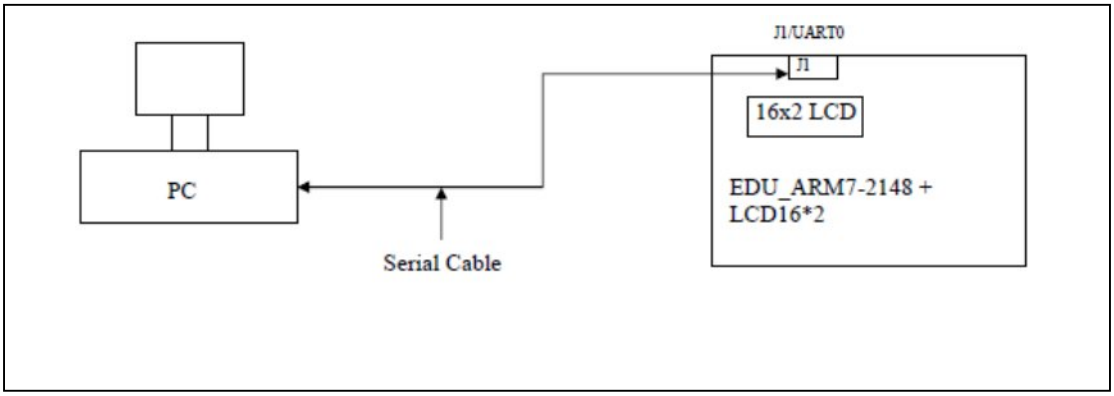
Title: Interface LCD with ARM

Aim: Write a program to interface LCD.

Equipment's: SCARM, PC, EDU-ARM7-2148 with 16x2 LCD.

Theory: - Write theory related to Interfacing of LCD

Block Diagram:

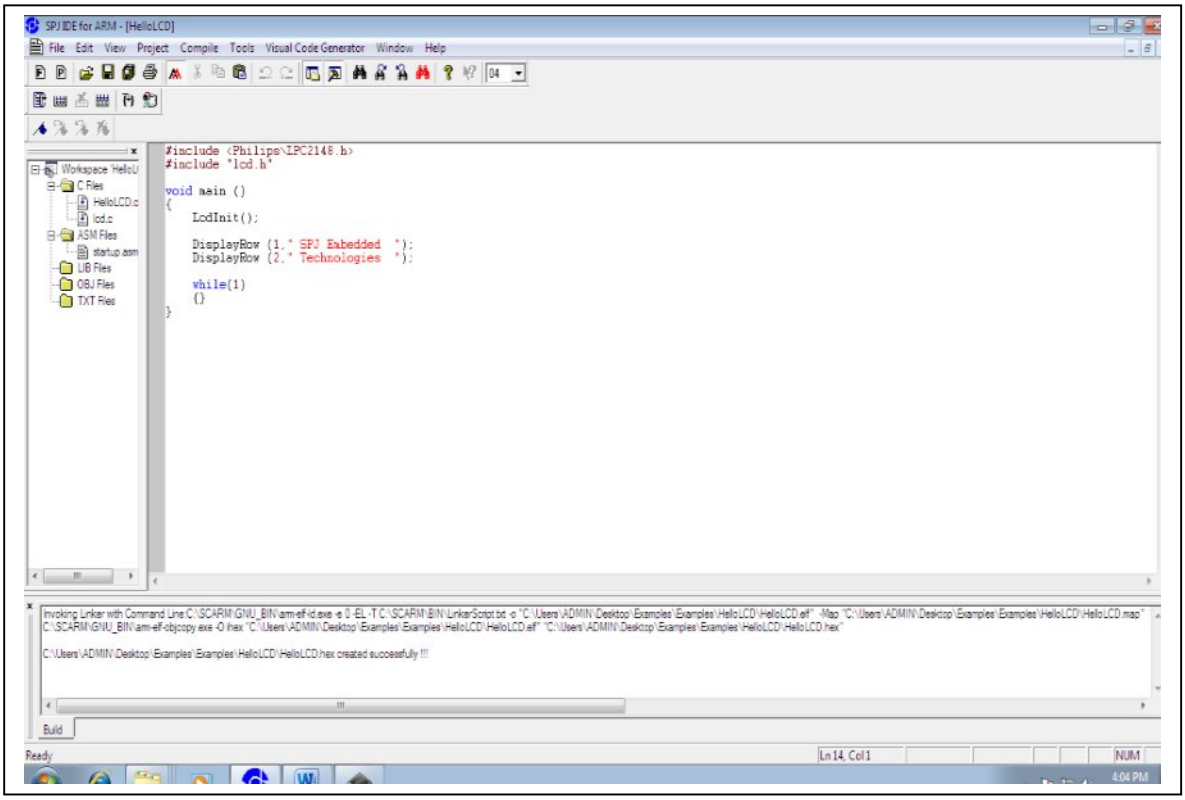


Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Source Code:
Program1

```
#include <Philips\LPC2148.h>
#include "lcd.h"

void main ()
{
        LcdInit();

        DisplayRow (1," SPJ Embedded  ");
        DisplayRow (2," Technologies  ");
```

```
        while(1)
        {}
}
```

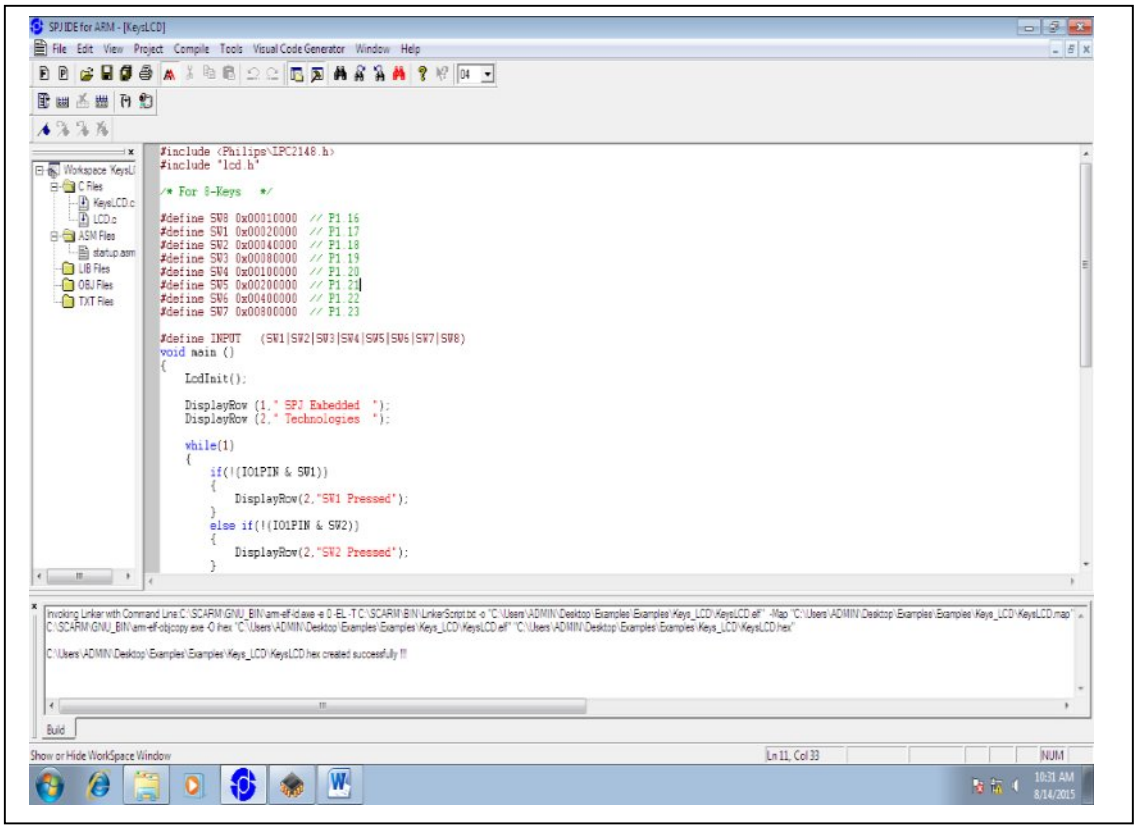


Program 2:

```
#include <Philips\LPC2148.h>
#include "lcd.h"

/* For 8-Keys  */

#define SW8   0x00010000    // P1.16
#define SW1   0x00020000    // P1.17
#define SW2   0x00040000    // P1.18
#define SW3   0x00080000    // P1.19
#define SW4   0x00100000    // P1.20
#define SW5   0x00200000    // P1.21
#define SW6   0x00400000    // P1.22
#define SW7   0x00800000    // P1.23

#defineINPUT(SW1|SW2|SW3|SW4|SW5|SW6|SW7|SW8)
void main ()
```

```
{
        LcdInit();

        DisplayRow (1," SPJ Embedded  ");
        DisplayRow (2," Technologies  ");

        while(1)
        {
                if(!(IO1PIN & SW1))
                {
        DisplayRow(2,"SW1 Pressed");
                }
                else if(!(IO1PIN & SW2))
                {
        DisplayRow(2,"SW2 Pressed");
                }
                else if(!(IO1PIN & SW3))
                {
        DisplayRow(2,"SW3 Pressed");
                }
                else if(!(IO1PIN & SW4))
                {
        DisplayRow(2,"SW4 Pressed");
                }
                else if(!(IO1PIN & SW5))
                {
        DisplayRow(2,"SW5 Pressed");
                }
                else if(!(IO1PIN & SW6))
                {
        DisplayRow(2,"SW6 Pressed");
                }
                else if(!(IO1PIN & SW7))
                {
        DisplayRow(2,"SW7 Pressed");
                }
                else if(!(IO1PIN & SW8))
                {
        DisplayRow(2,"SW8 Pressed");
                }
```

```
        }
}
```



Output:

You can see the message **Hello World** on LCD. If required reset the board.

Experiment:7

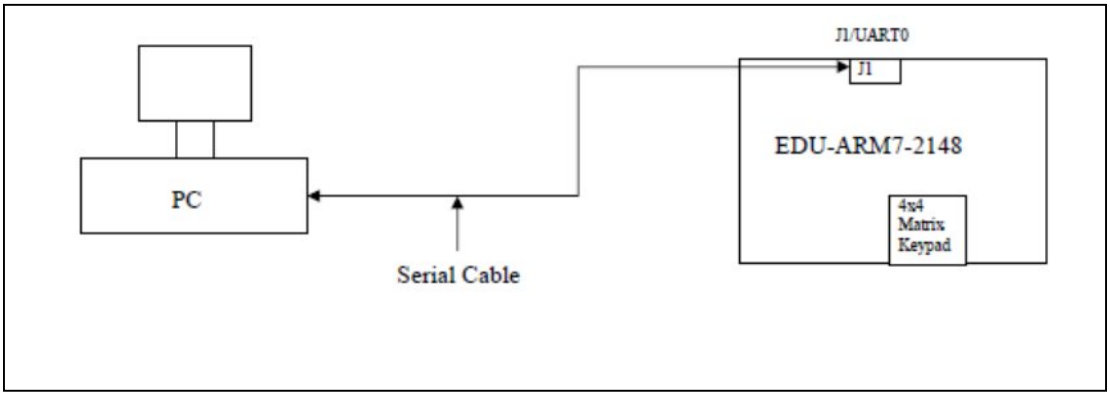
Title: Keyboard interfacing sense key and display the appropriate code on LCD

Aim: Write a program to interface 4*4 matrix keyboard.

Equipment's: SCARM, PC, EDU-ARM7-2148.

Theory: - Write theory related to Interfacing of LED

Block Diagram:



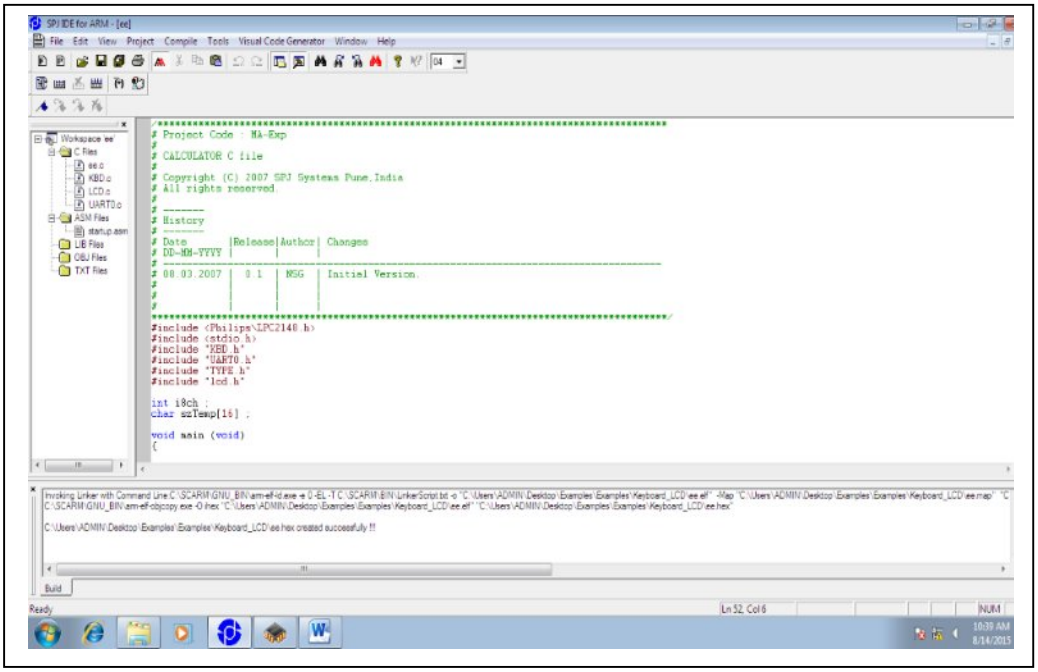
Connections:
16 Keys (SW9 to SW25) present in 4x4 Matrix Keypad region on EDU-ARM7-2148 are connected to
P1.16 to P1.23.

Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Source Code:

```
#include <Philips\LPC2148.h>
#include <stdio.h>
#include "KBD.h"
#include "UART0.h"
#include "TYPE.h"
#include "lcd.h"
int i8ch ;
```

```
charszTemp[16] ;
void main (void)
{
        PINSEL0 |= 0x00000005 ;
        PINSEL1 |= 0x00000000 ;

        UART0_Init();
        LcdInit();
        DisplayRow(1,"Keypad Test ");
        DisplayRow(2,"Press Any Key  ");

        KBD_Init();
        {

                puts("This is a keyboard test program\nPress any key\nKeycode will be
displayed\n") ;
                while(1)
                {
                        i8ch = KBD_rdkbd() ;
                        printf("Keycode = %02X\n", i8ch) ;
                        sprintf(szTemp,"KeyCode = %02X",i8ch);
                        DisplayRow(2,szTemp) ;
                }
        }
}
`
```



Output:

In this program after pressing any key, its code is sent to serial port using UART0. You can see outputon SPJTerminal. Therefore Open SPJTerminal. Go to Port -> Settings. Do proper settings (Baud Rate:19200, Data Bits: 8, Stop Bits: 1, Echo: Off, Parity: None, Com Port: Com 1 (if other choose it)). Clickon OK. Go to Port -> Open. If required Reset the EDU-ARM7-2148 board. After pressing key frommatrix keypad its corresponding key code will be displayed on SPJTerminal.

Experiment:8

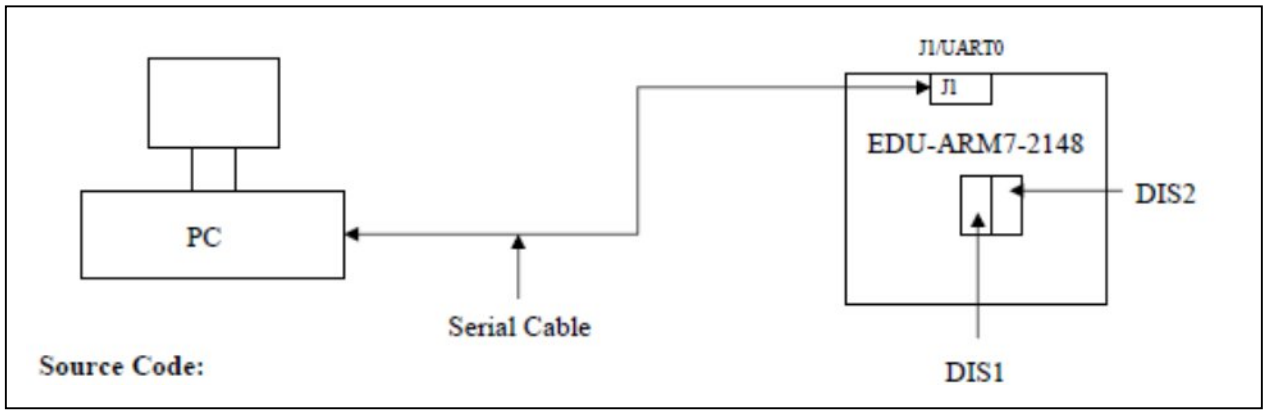
Title: Seven segment display

Aim: Write a program to interface 7SEG (7 segment display).

Equipment's: SCARM, PC, EDU-ARM7-2148.

Theory: - Write theory related to Interfacing of seven segment display

Block Diagram:



Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Source Code:

```
#include <Philips\LPC2138.h>
#include <Stdio.h>
#include "sevenseg.h"

/*Initializes the I2C protocol and port pins.*/
void I2C_Init (void)
{
        // Power on I2C0 peripheral
        PCONP          |= 0x00000080;

        // Define port pin as SDA and SCL
        PINSEL0        |= 0x00000050 ;

        I2C0CONCLR          = 0x6C;          // clear all I2C config bits
```

```
	I2C0CONSET = 0x40;		// set I2EN

	// I2C Clock Duty Cycle (high and low), Bit freq.= 100KHz
	I2C0SCLH	= 60;
	I2C0SCLL	= 60;
}

/*
Waits until given status occured.
Return:True on status occured and
		False on time out
*/

/*Main function.*/
voidSevenSeg()
{
		unsignedint i;
		I2C0CONSET = 0x20;				// Start set
		I2C0CONCLR	= 0x2C;				// clear all except I2EN
		I2C0DAT		= I2CEXPANDER_ADDR;// addr[0]=0 means I2C write
		for(i=0;i<300;i++);
		I2C0CONCLR	= 0x2C;				// clear all except I2EN

		I2C0DAT		= DISPLAY_1 ;			// Data on DIS1
		for(i=0;i<300;i++);
		I2C0CONCLR	= 0x2C;				// clear all except I2EN

		I2C0DAT		= DISPLAY_2 ;			// Data on DIS2
		for(i=0;i<300;i++);
		I2C0CONCLR	= 0x2C;				// clear all except I2EN

		I2C0CONSET = 0x10;				// generate stop condition
		for(i=0;i<300;i++);
		I2C0CONCLR	= 0x2C;
}

void main(void)
{
	I2C_Init();
```

```
        while(1)
        {
                SevenSeg();
        }
}
```

Output:
You can see 1 and 2 numbers on Seven Segments.

Experiment:9

Title: Buzzer interfacing with ARM

Aim: Write a program to interface buzzer with ARM

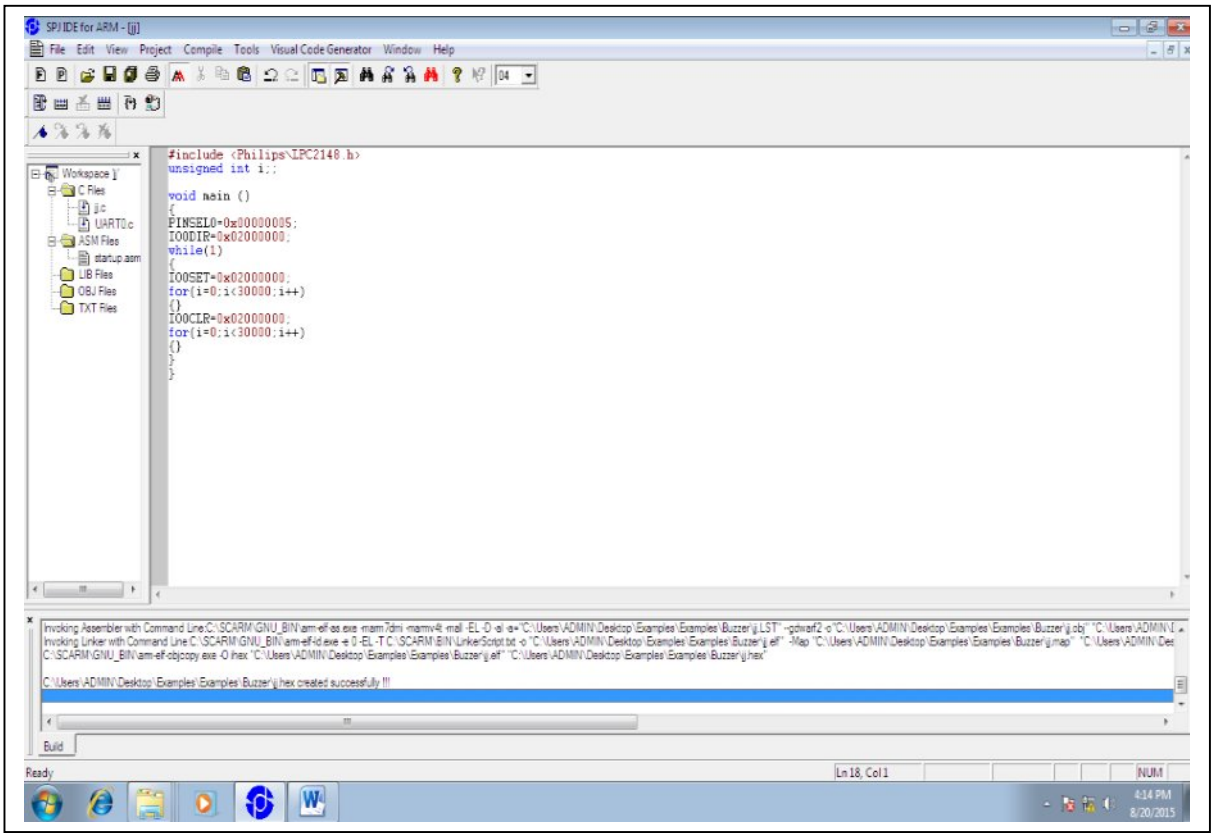
Equipment's: SCARM, PC, EDU-ARM7-2148.

Theory: - Write theory related to Interfacing of Buzzer

Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Source code:

```
include <Philips\LPC2148.h>
unsigned int i;;

void main ()
{
PINSEL0=0x00000005;
IO0DIR=0x02000000;
while(1)
{
IO0SET=0x02000000;
for(i=0;i<30000;i++)
{}
IO0CLR=0x02000000;
for(i=0;i<30000;i++)

{}

}

}
```



Output:

You hear buzzer sound on and off depending on delay duration

Experiment:10

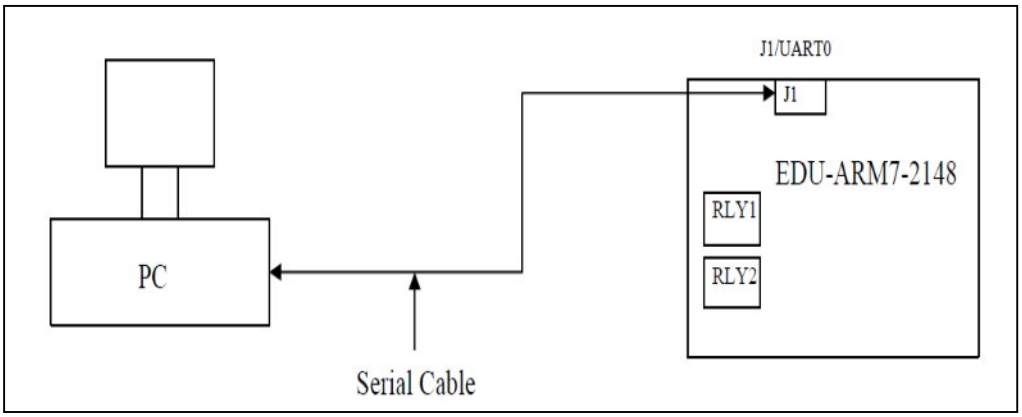
Title: Interfacing of Relays.

Aim: Write a program to interface 2 relays with LPC2148.

Equipment's: SCARM, PC, EDU-ARM7-2148.

Theory: - Write theory related to Interfacing of LED

Block Diagram:



Procedure:
To Edit / Compile/ Generate Hex file, download and run this program: Refer Experiment no. 2.

Source Code:

```
#include <Philips\LPC2148.h>

unsigned int delay;

void main ()
{
        PINSEL0 = 0x00000000 ;
        PINSEL1 = 0x00000000 ;
        IO0DIR = 0x000C0000 ;

        while(1)
        {
                IO0CLR = 0x000C0000;
```

```
                for(delay=0; delay<50000; delay++)
                {}
                IO0SET = 0x000C0000;
                for(delay=0; delay<50000; delay++)
                {}
        }

}
```

Output:

Relay gets on and off depending upon delay.

Experiment:11

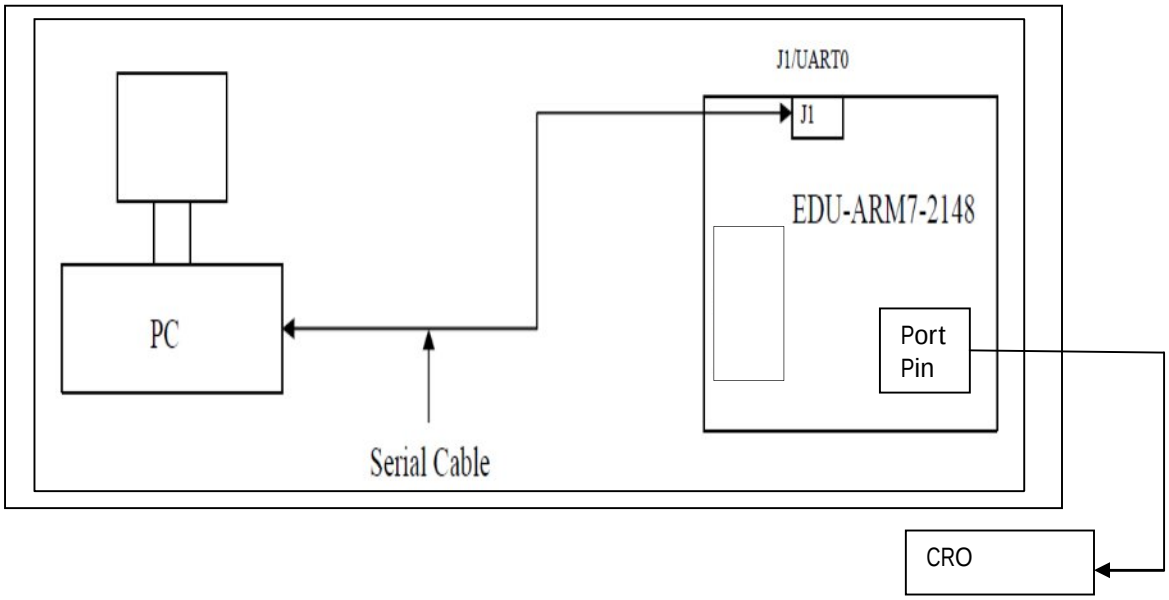
Title: Square wave generation.

Aim: Write a program to generate square wave using ARM.

Equipment's: SCARM, PC, EDU-ARM7-2148.

Theory: - Write theory related to square wave generation

Block Diagram:



Source Code:

```
#include <Philips\LPC2148.h>
Unsigned nt delay;

void main ()
{
        PINSEL0 = 0x00000000 ;
        PINSEL1 = 0x00000000 ;
        IO0DIR = 0x00003C00 ;

        while(1)
        {
                IO0CLR = 0x00003C00;
```

```
                for(delay=0; delay<50000; delay++)
                {}
                IO0SET = 0x00003C00;
                for(delay=0; delay<50000; delay++)
                {}
        }

}
```

Output:

You can see on square wave on the CRO.

3. Quiz on the subject

1. Which is the most commonly used language(s) used in embedded system?

2. An embedded system must have
(a) hard disk (b) processor and memory
(c) operating system (d) processor and input-output unit(s).

3. An embedded system hardware can
(a) have microprocessor or microcontroller or single purpose processor
(b) have digital signal processor
(c) one or several microprocessor or microcontroller or digital signal processor or single purpose processors
(d) not have single purpose processor (s)

4. An embedded system has RAM memory
(a) for storing the variables during program run, stack and input or output buffers, for example, for speech or image
(b) for storing all the instructions and data
(c) for storing the programs from external secondary memory
(d) for fetching instructions and data into cache(s).

5. A system might be connected to a number of other devices and systems.
(i) A bus consists of a common set of lines to connect multiple devices, hardware units and systems
(ii) A bus is used for communication between two of these at any given instance.
(iii) A bus is used for communication between all of these at the same instance
(iv) A bus may be serial bus or parallel bus to transfer one bit or multiple data bits at an instance, respectively.
(a) i, ii and iv correct (b) iii correct (c) iii and iv correct (d) all are correct.

6. A system must have an interrupt handling mechanism for executing the interrupt service routines in case of the interrupts from
(a) physical devices
(b) interfaced circuits or systems, software interrupt instructions and software exceptions
(c) physical devices or interfaced circuits or systems
(d) physical devices or interfaced circuits or systems, software interrupt instructions and software exceptions

7. (i) A compiler generates an object file. (ii) The object file is linked with library functions using linker. (iii) After re-allocation of addresses a locator sends the codes to device programmer for burning as ROM image in embedded system ROM. (iv) After re-allocation of addresses a loader loads the codes to device programmer for burning as ROM image in embedded system ROM. (v) After re-allocation of addresses a loader loads the codes in RAM. Steps for embedded system development are steps

   (a) i, ii and iv (b) i, iii, iv and vi (c) Steps i, ii and iii (d) i , ii and vi.

8. In a multitasking OS, (i) each process (task) has a distinct process control block (ii) each process (task) has memory allocation of its own (iii) a task has one or more functions or procedures for a specific job. (iv) a task may share the memory (data) with other tasks. (iv) processor may process multiple tasks separately or concurrently (v) each process (task) has a separate stack in memory (vi) a process calls another process, which can call another process, similar to nested call of the functions.

   (a) i, ii, iv and vi correct
   (b) all are correct except vi
   (c) iii, iv and v correct
   (d) ii, iii and vi correct.

9. RTOS is used in most embedded systems when the system does (a) concurrent processing of multiple real time processes (b) sequential processing of multiple processes when the tasks have real time constraints (c) real time processing of multiple processes (d) the concurrent processing of multiple processes, tasks have real time constraints and deadlines, and high priority task preempts low priority task as per the real time constraints.

10. A device driver is software for (a) opening or connecting or binding or reading or writing or closing or other actions of the device (b receiving input or sending outputs from device (c) access to parallel or serial port by the device (d) controlling and configuring the device for read and write functions.

4. Conduction of Viva-Voce Examination:

Teacher should conduct oral exams of student with full preparation. Normally the objective questions with guesses are to be avoided. To make it meaningful, the questions should be such that depth of the students in the subject is tested. Oral examinations are to be conducted in cordial environment amongst the teacher taking the examination. Teachers taking such examinations should not have ill thoughts about each other and courtesies should be offered to each other in case of difference of opinion, which should be critically suppressed in front of the students.

5. Evaluation and Marking systems:

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination system to become successful. It is wrong approach or concept to award the students by way of easy making to get cheap popularity among the students which they do not deserve. It is a primary responsibility of the teacher to see that right students are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking pattern should be justifiable to the students without any ambiguity and teacher should see that the students are faced with just circumstances.