INTRODUCTION

❖ KEIL µVISION 5 SOFTWARE

The µVision5 IDE is a Windows-based software development platform that combines a robust editor, project manager, and makes facility. µVision5 integrates all tools including the C compiler, macro assembler, linker/locator, and HEX file generator. µVision5 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 toKeil ULINK
- ✓ Flash programming utility for downloading the application program into Flash ROM
- ✓ Links to development tools manuals, device datasheets & user's guides

The µVision5 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 youachieve your design goals.

❖ Getting started with ARM LPC2148 using Keil uVision IDE

There are various development environments available in the market for ARM processors.

Some of these are mentioned below:

- CrossWorks for Arm
- Keil µVision
- IAR Embedded Workbench

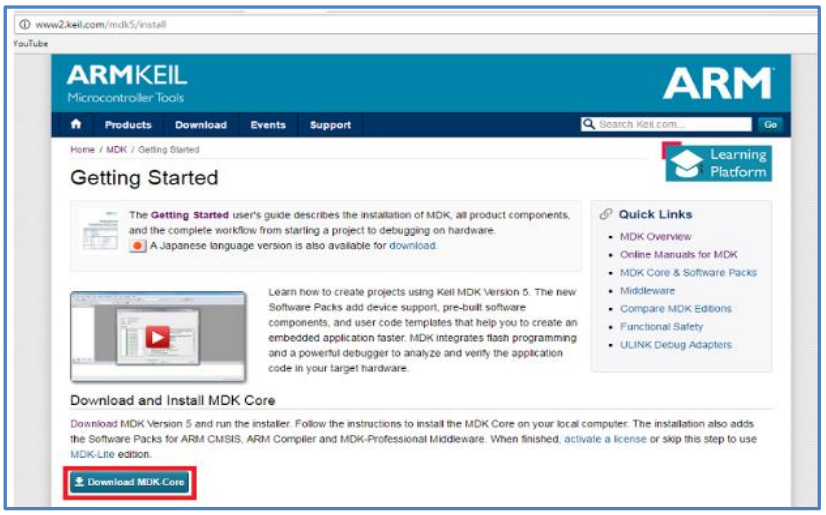
❖ Downloading and installation **µVision IDE by Keil.**

Follow the steps given below:

STEP-1: Download the MDK-lite (Microcontroller Development Kit) by Keil from their website. Here is the link to the page from where you can download this:

http://www2.keil.com/mdk5/install

Click on **Download MDK-Core**.

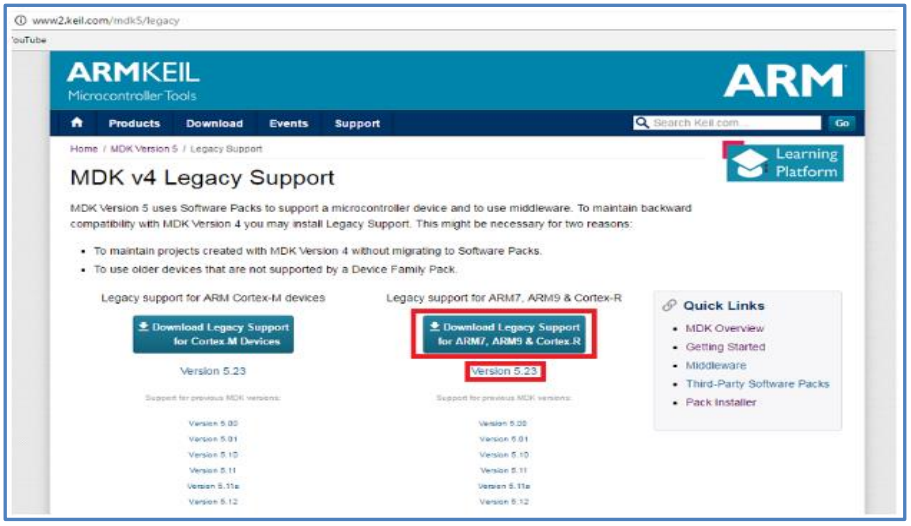


Install the software by following the simple instructions provided during installation process.

STEP-2: The new version µVision5 does not support many of the devices that were supported in the older versions yet. LPC2148 is one of the devices that are not supported. Hence, we need to add this device after successfully installing µVision5.

To do this, go to the following link and download the executable file for Legacy Support for ARM7, ARM9, and Cortex-R: **http://www2.keil.com/mdk5/legacy**

Download the Legacy support for the version of MDK downloaded and installed.



Install the executable file that will be downloaded. Follow the simple instructions provided during the installation process.

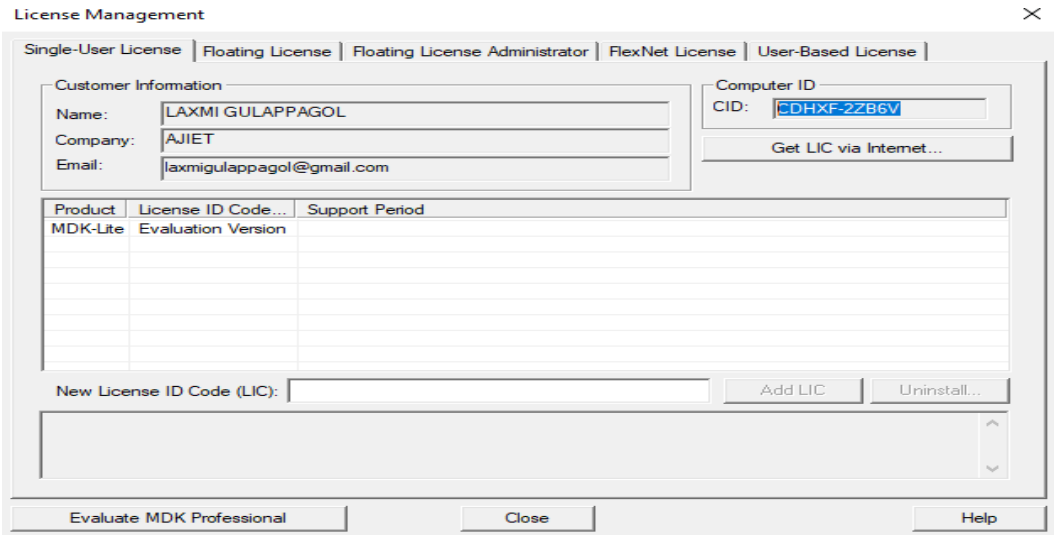
When the above-described steps are completed, we will have the IDE installed and ready to use with support for the device we intend to use, i.e. LPC2148.

1. Double click on Keil µvision5 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**.

STEP-3: Go to File > License Management... and select the Single-User License tab.

https://www.keil.arm.com/mdk-community/



STEP-4: Click Get LIC via Internet..., then click OK to register the product. The License Management page on the Keil web site will open.

STEP-5: Enter the following serial number in the Product Serial # (PSN) field:

42B2L-JM9GY-LHN8C

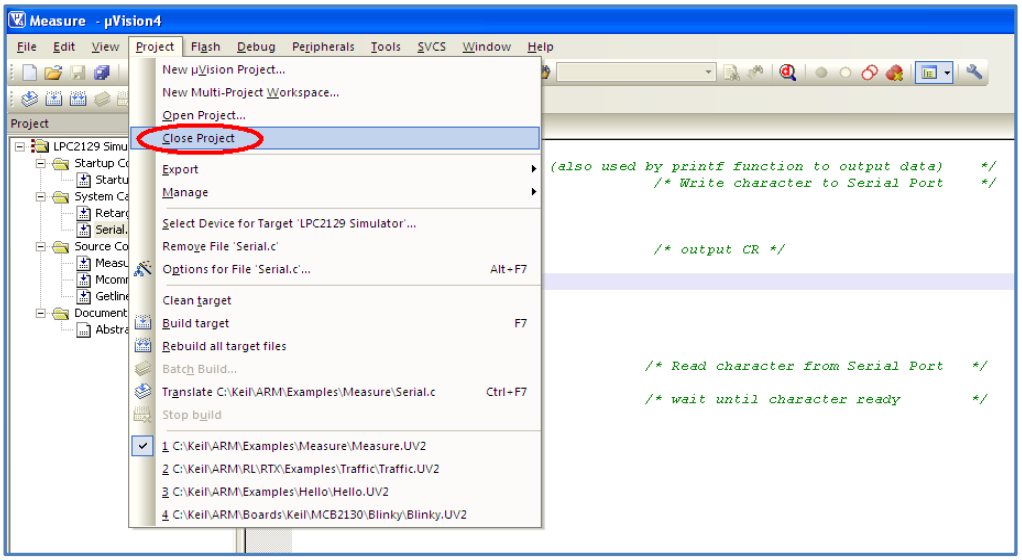
STEP-6: Complete and submit the rest of the form.

You will receive a License ID Code (LIC) to your email address. Enter this in the New License ID Code (LIC) field in µVision and click 'Add LIC'.

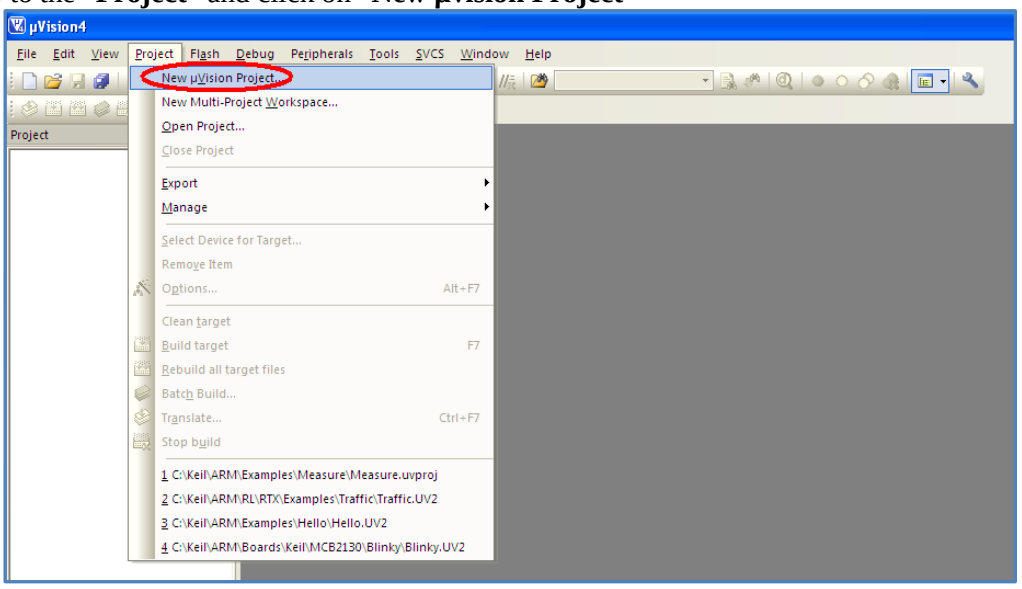
- Using µVision IDE

STEP 1: Open Keil µVision from the icon created on your desktop.

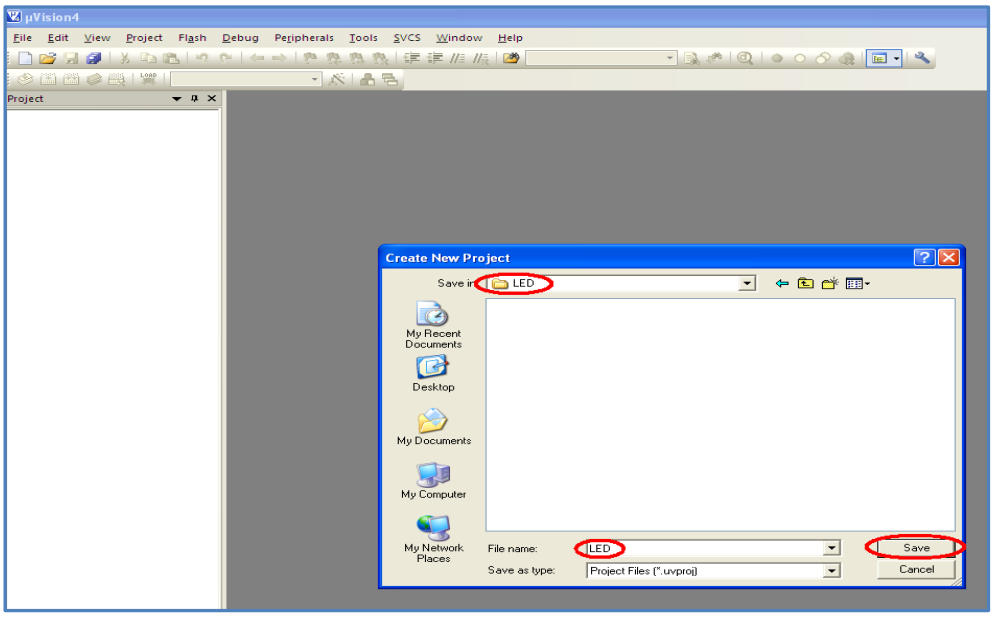
Go to "**Project**" and close the current project "**Close Project**".



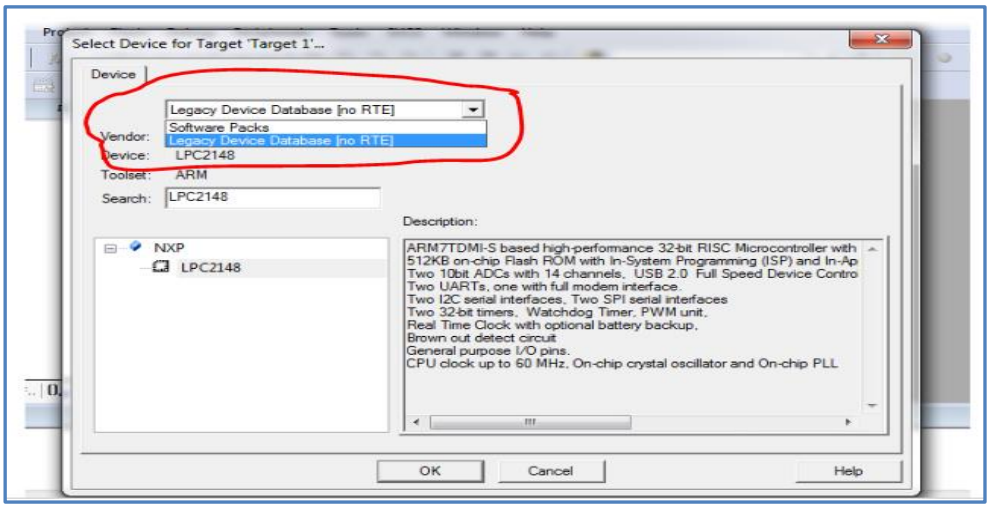
STEP 2: Go to the "**Project**" and click on "**New µvision Project**"



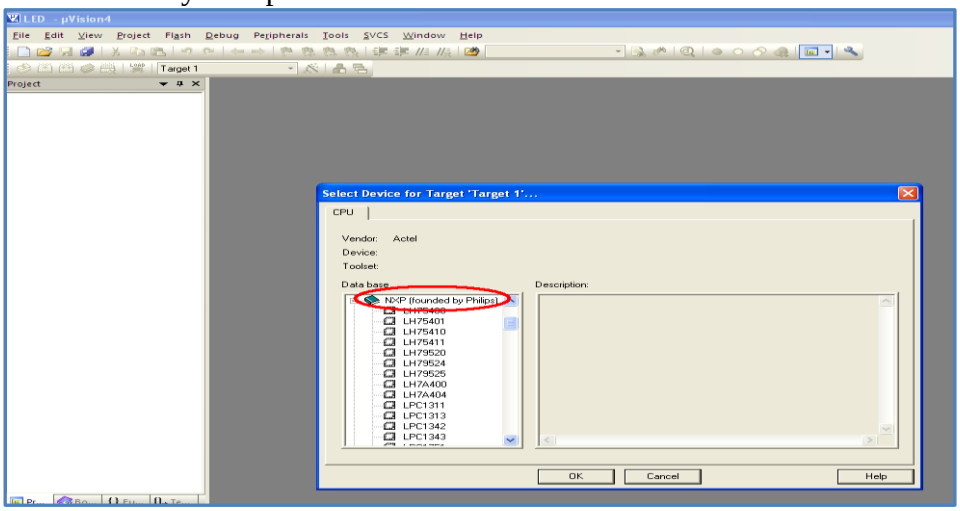
STEP 3: **Create New Project** window will pop up. Select the folder where you want to create project and give a suitable name to the project. Then click on **Save**.



STEP 4: **Select Device for Target: 'Target1'...**window will pop up next. It has a select window to choose between Software Packs or Legacy Device Database. As LPC2148 is in Legacy Device Database, choose **Legacy Device Database.**



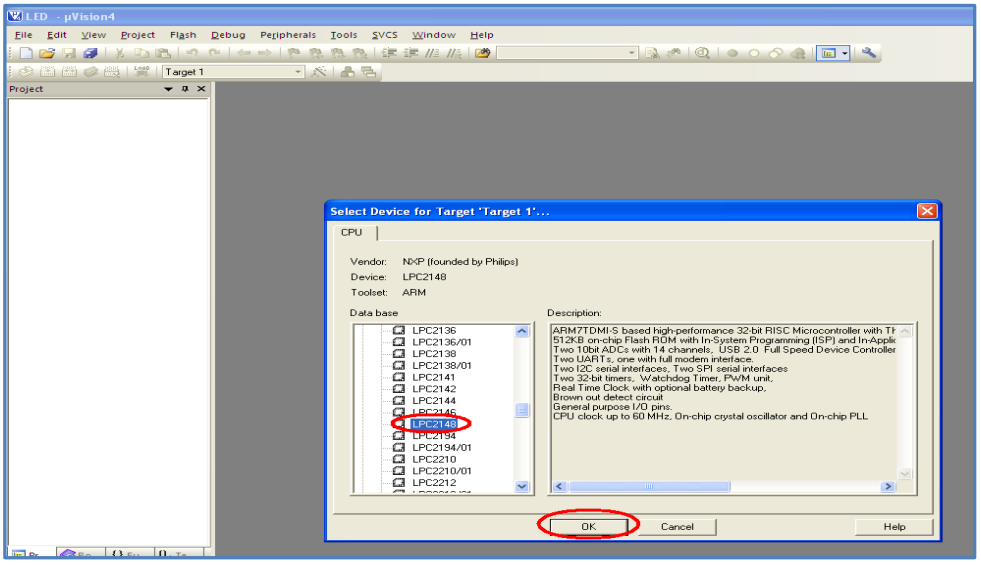
STEP 5: A small window will pop up with name "**Select Device for Target 'Target 1'**", select the data base NXP founded by Philips.



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

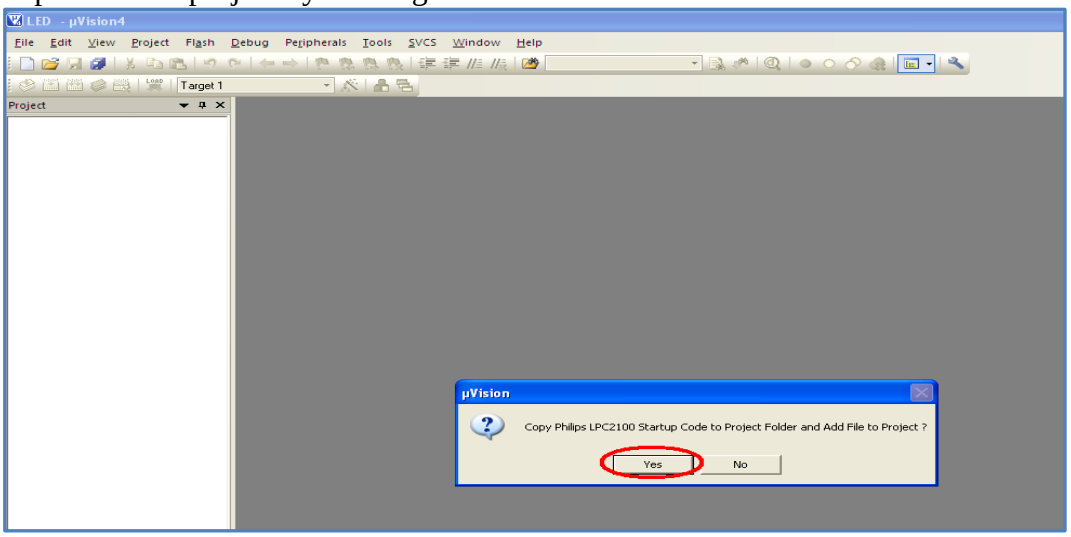
OR

Type in LPC2148 in search and select the device under NXP with the name LPC2148 and click on OK.

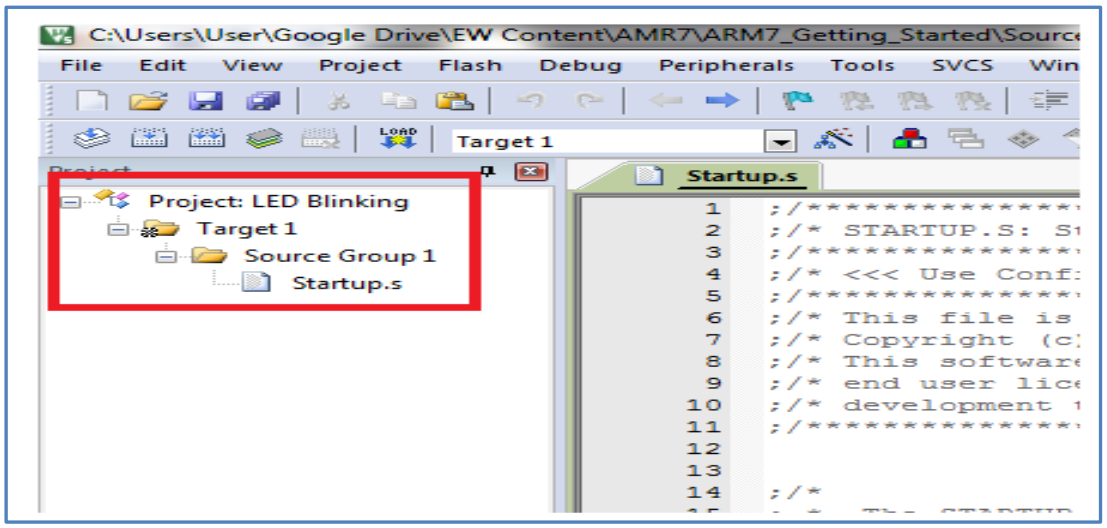


STEP 7: A window will pop up asking whether to copy Startup.s to project folder and add file to project. Click on **Yes**.

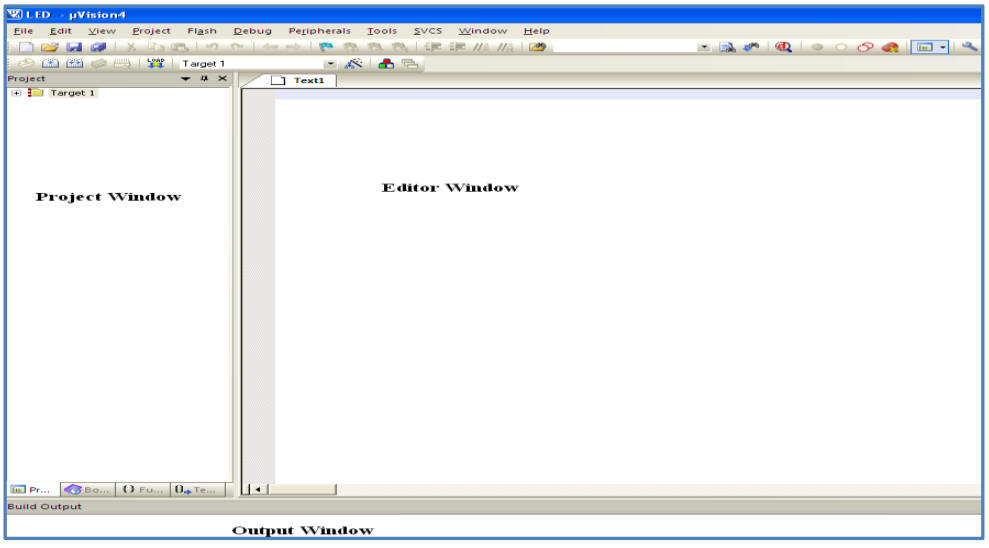
Add Startup file to the project by clicking "**Yes**"→



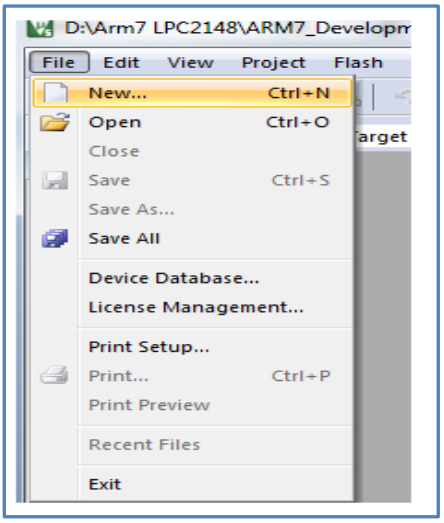
STEP 8: The project name and its folders can be seen on the left side in the project window after the previous step is completed as shown below.



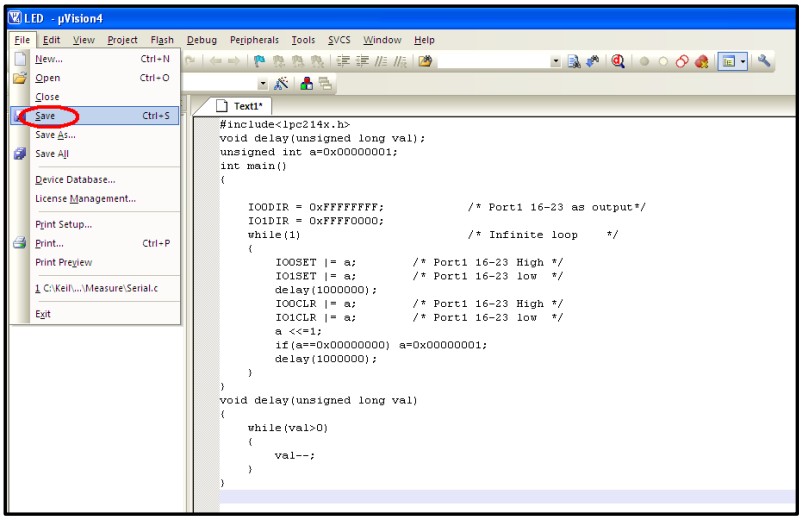
There are the main three windows are available in the keil IDE. First one is Project Workspace, second one is Editor Window and third one is Output Window.



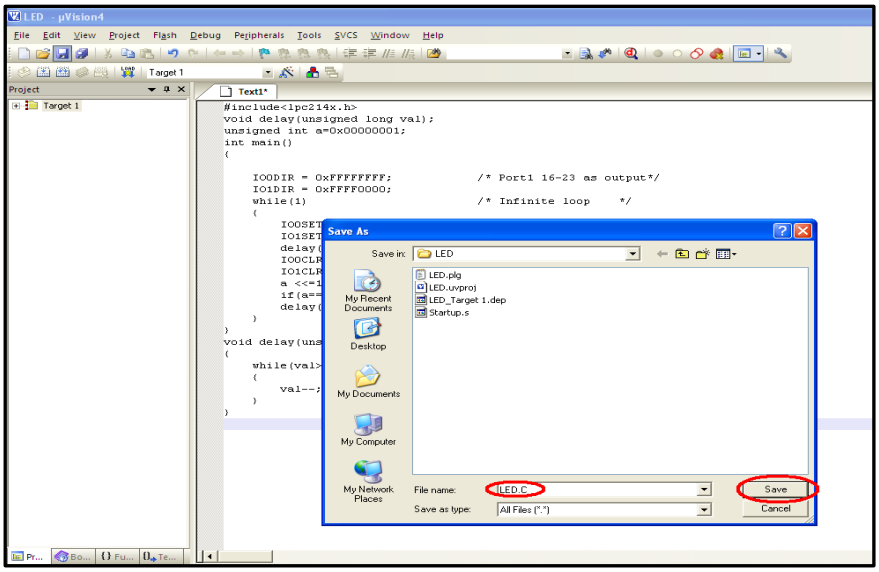
STEP 9: Now go to File tab and add **New** file from the menu.



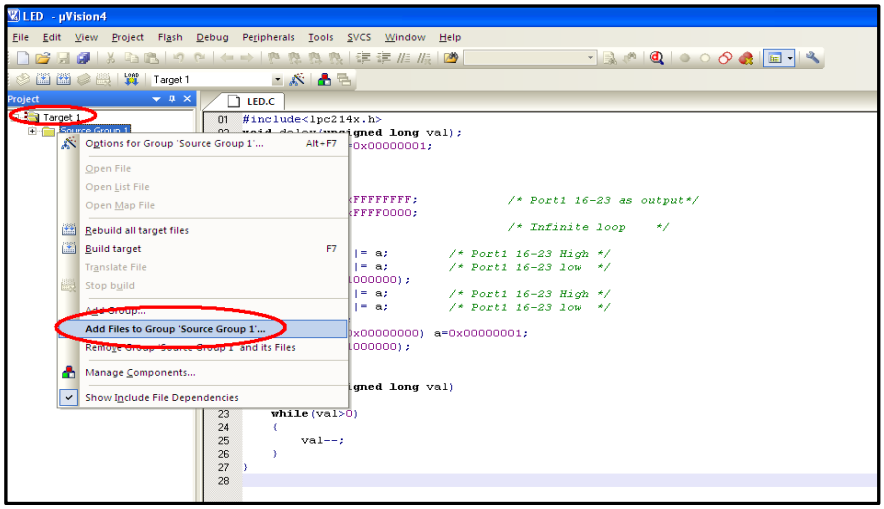
STEP 10: Can be start to write asm/c code on the editor window.



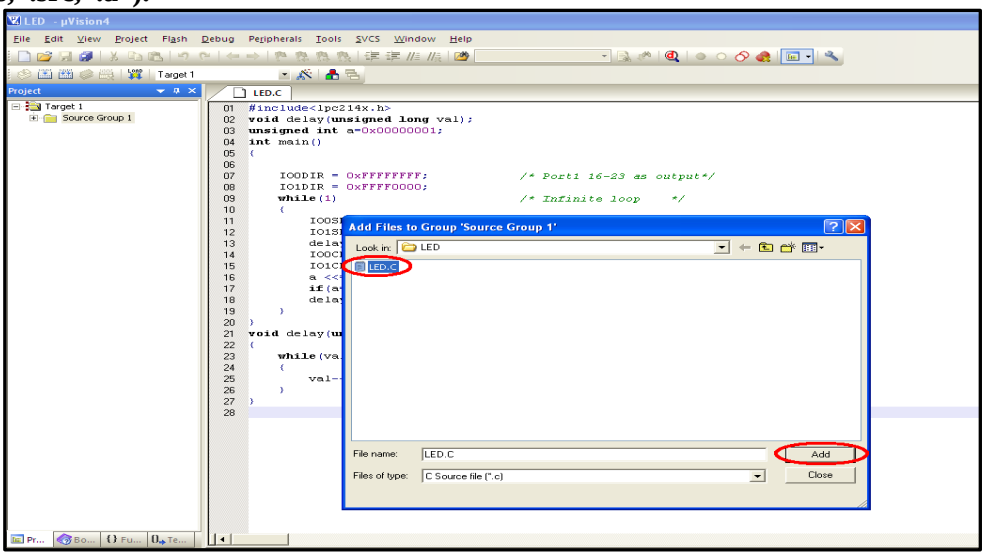
STEP 11: Can be save the file, if the program is in "**C**" save as "**filename. C**" or else save as "**filename.ASM**".



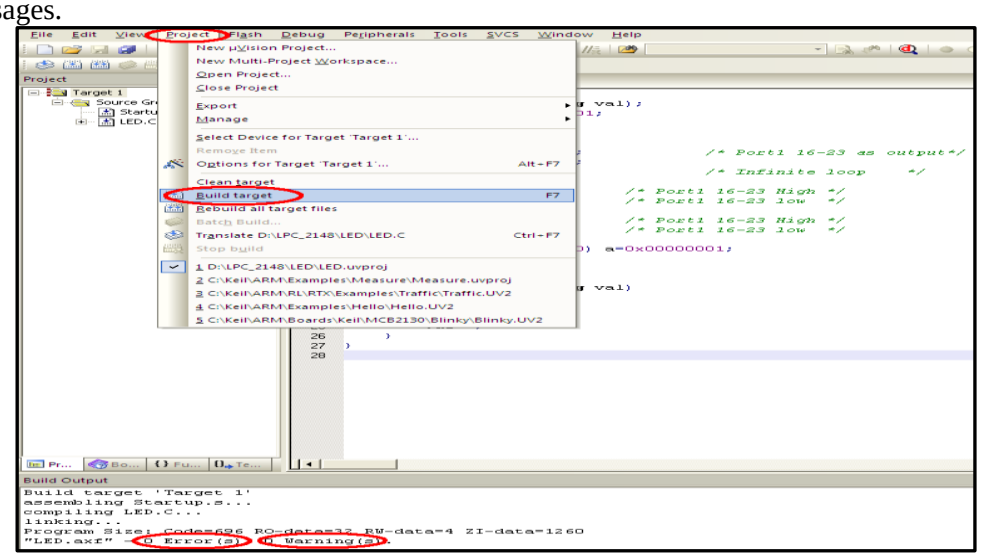
STEP 12: Add this source file to Group1, Go to the "**Project Workspace**" drag the +mark "**Target 1**" in that right click on "**Source Group1**" and click on "**Add Files to Group "Source Group1"**".



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*).**



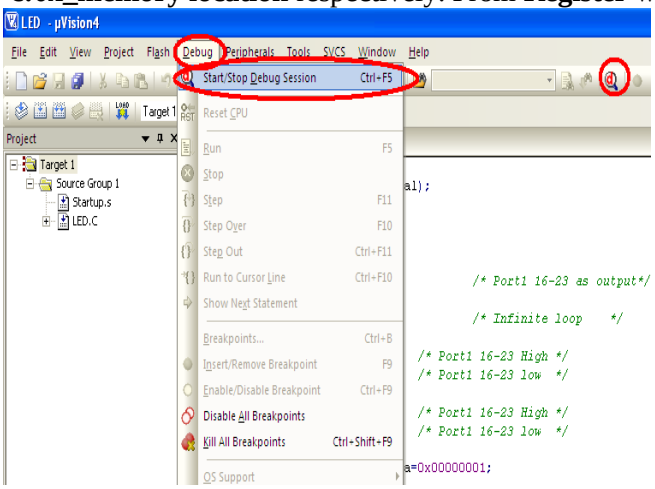
STEP 14: Then go to "**Project**" click on "**Build Target**" or **F7**. Output window will display related error and warning messages.

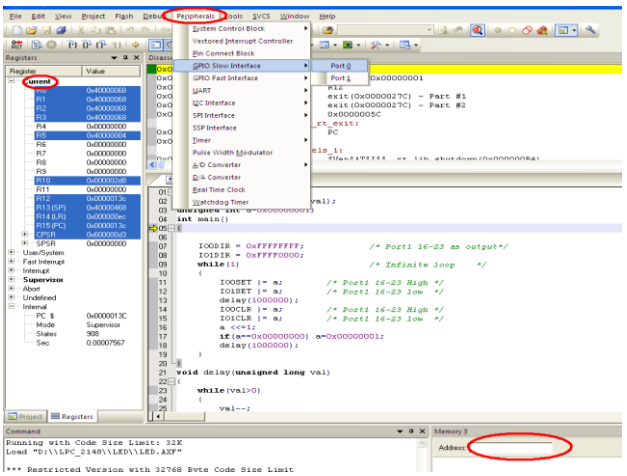


Simulation Part:

STEP 15: Go to "**Project**" menu, click on "**Rebuild all target Files**" and start **Debug**. From **View** menu can

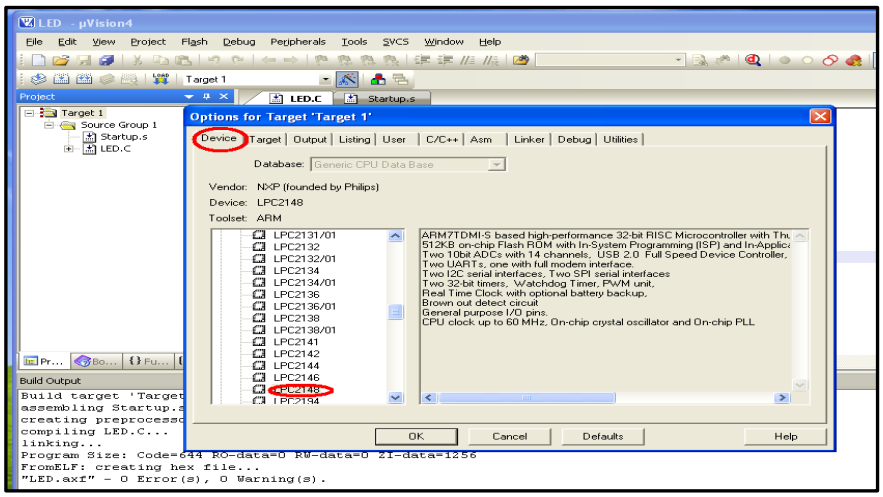
get **Memory Window** and from **Peripherals** can get I/O ports, Serial etc. For access internal memory type **i:0x_memory** location example: **i:0x30** and for external and program memory **x:0x_memory location**, **c:0x_memory location** respectively. From **Register** window we can edit and access the values also.



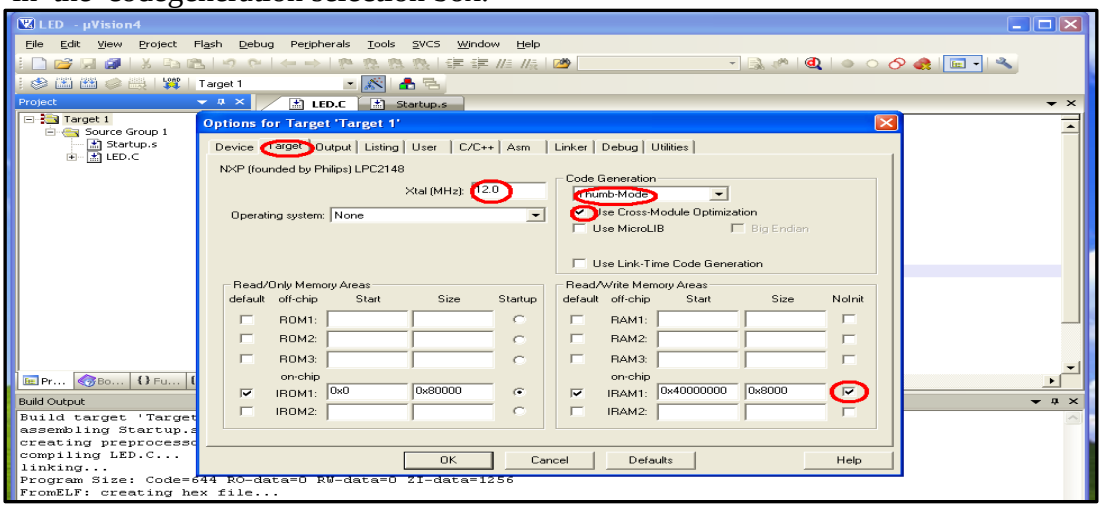


HEX file creation:

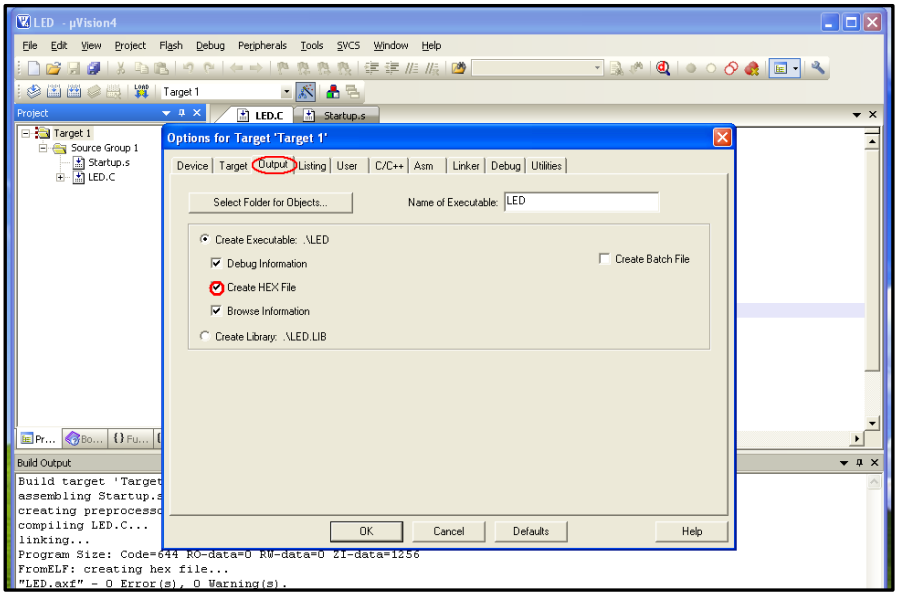
STEP 16: Follow the **STEP** up to **14**, then go to "**Project**" and click on "**Option for Group 'Source Group1**"". There a small window will open with name "**Option for Target 'Target1**"". In that window, go to first menu "Device", can be select LPC2148.



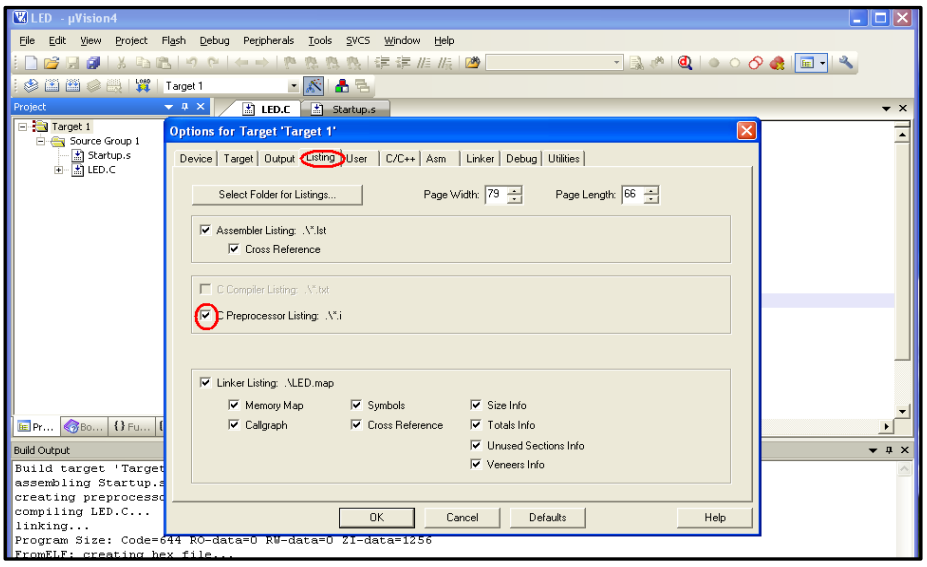
STEP 17: Next go to **Target** menu, set clock frequency as 12.0 MHz and select Thumb mode in the codegeneration selection box.



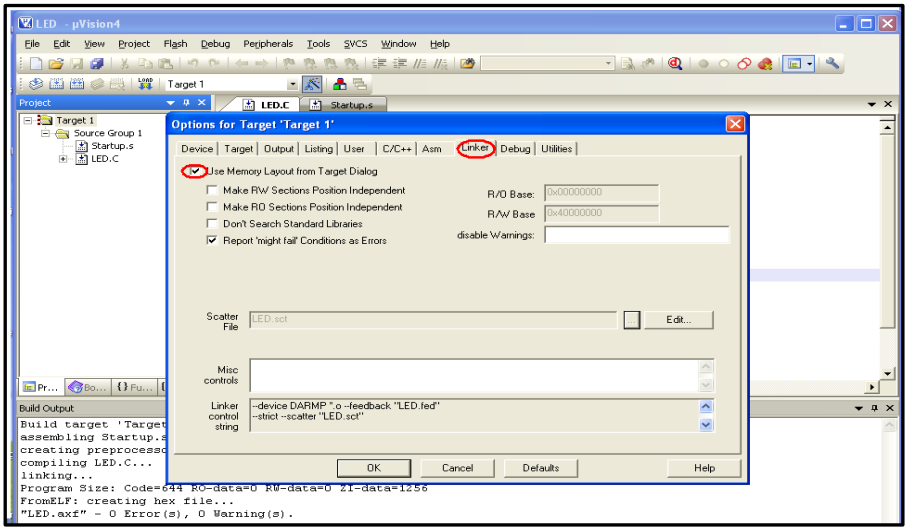
STEP 18: Then go to Output menu and click on create HEX file



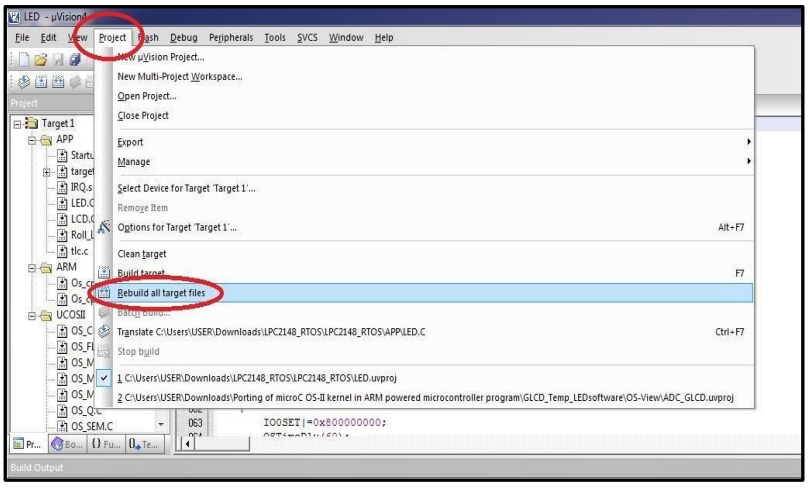
STEP 19: Then go to **Listing** menu and select C preprocessor Listing.



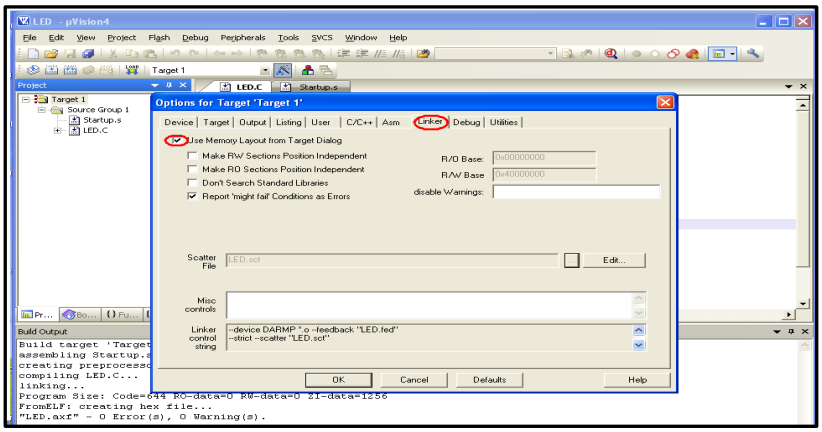
STEP 20: Finally in the **Linker menu,** click on use memory layout from target dialog and click ok.



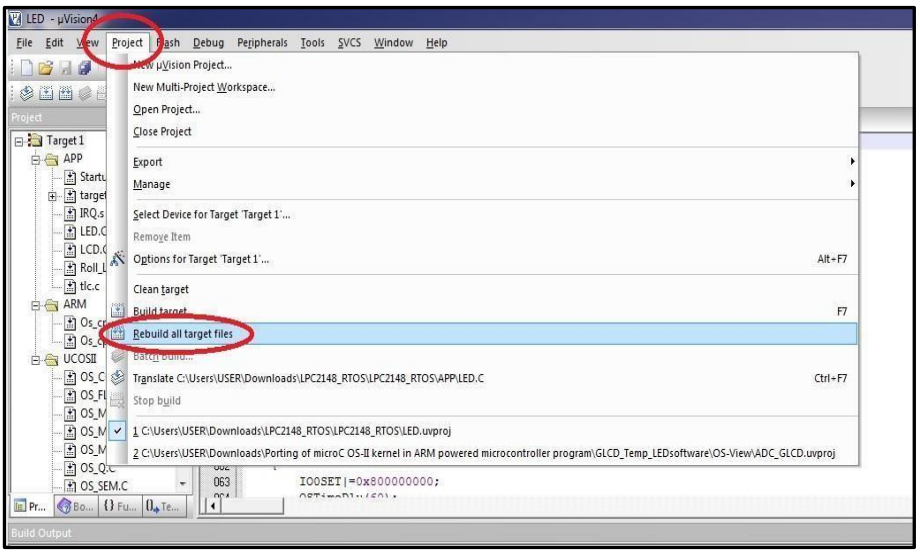
STEP 21: For creating Hex file go to "**Project**" menu and click on "**Rebuild all target Files**"



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

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.

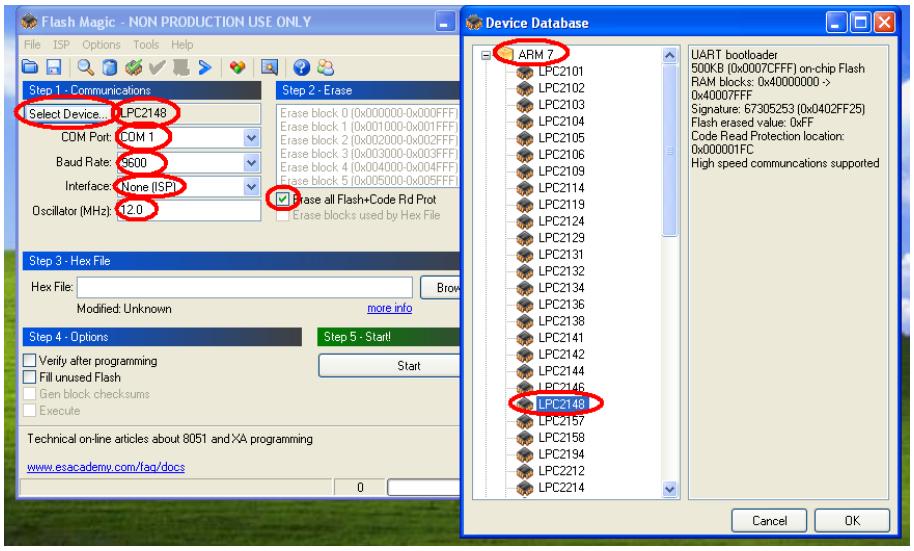
❖ Installation Procedure for Flash Magic software:

Installation steps are given below:

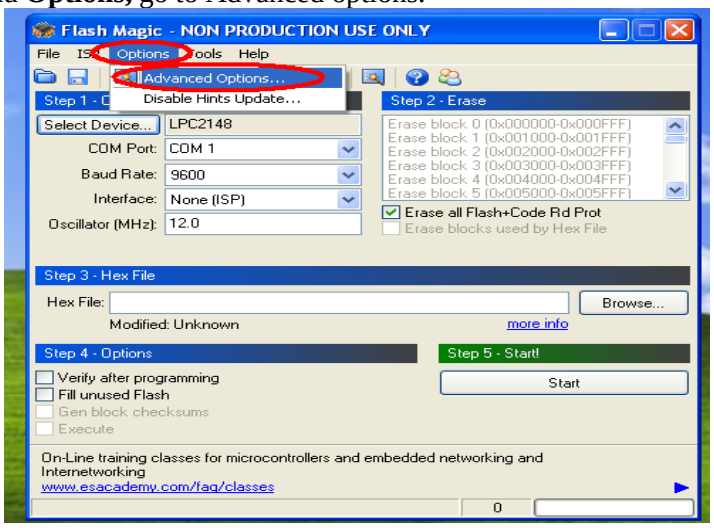
1. Double click on Flash Magic.exe.
2. Then click on Next.
3. Next accept the agreement and click on Next.
4. Select the destination folders and click on Next then Install.
5. Finally click on Finish.

Programming with communication port(COM1):

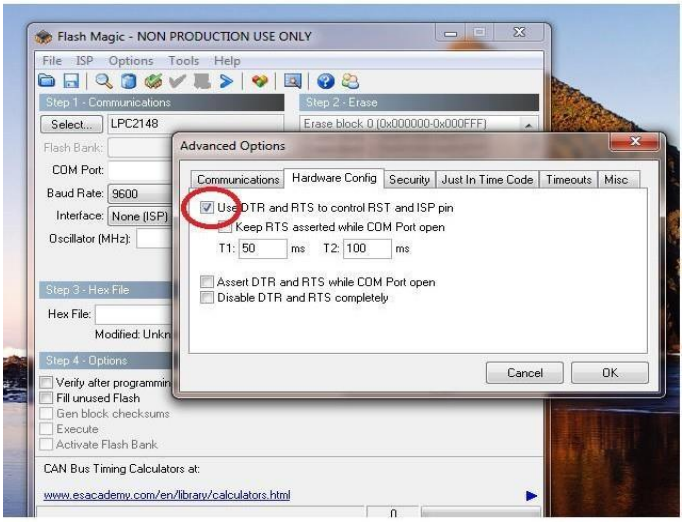
STEP 22: For programming with communication port, first select the device LPC2148 in ARM 7 category, COM port will be COM 1, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and clickon erase of flash code Rd plot.



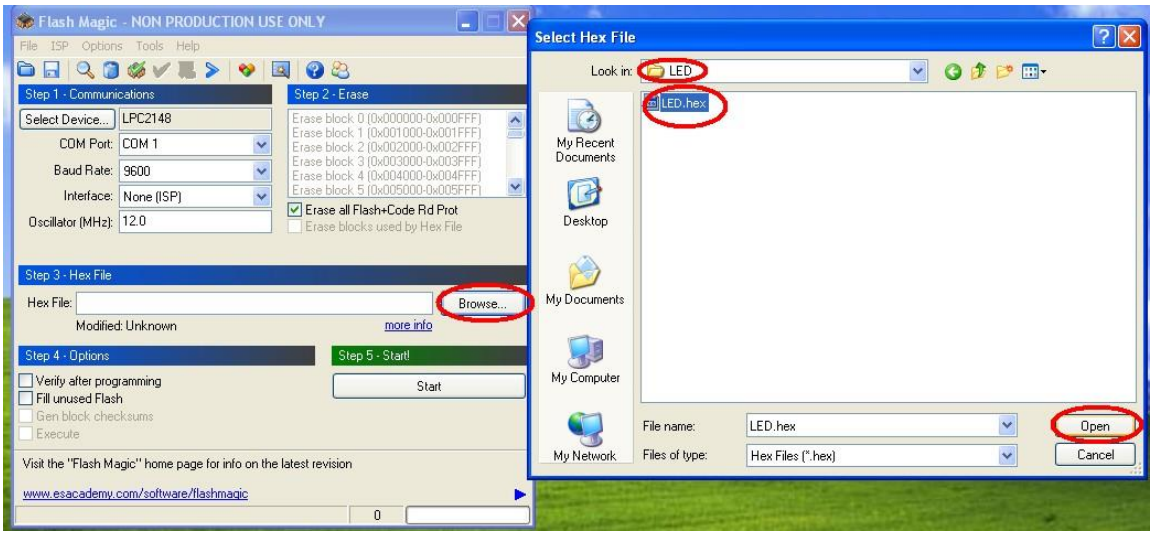
STEP 23: Under the menu **Options,** go to Advanced options.



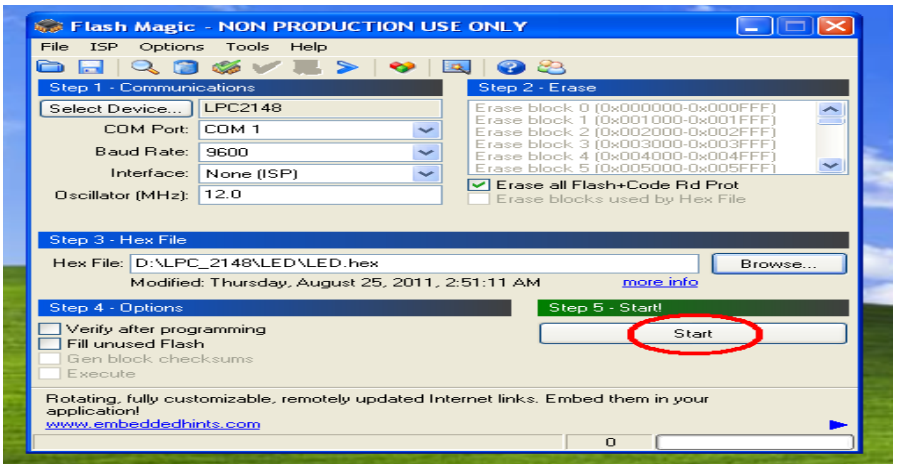
STEP 24: Under the menu Advanced options, go to Hardware configuration, click on the Use DTR andRTS to control RST and ISP pin.



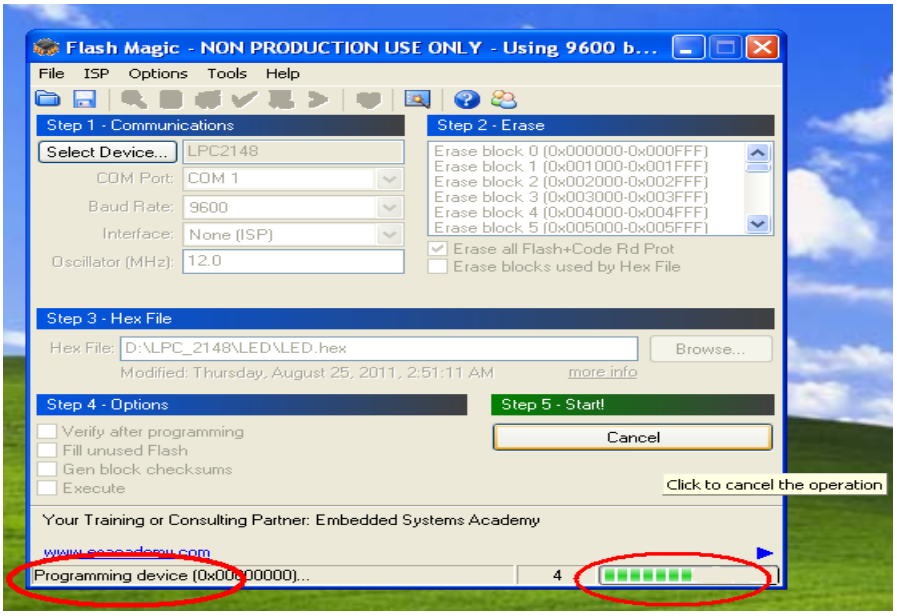
STEP 25: Next browse the path of hex file and select the file**.**



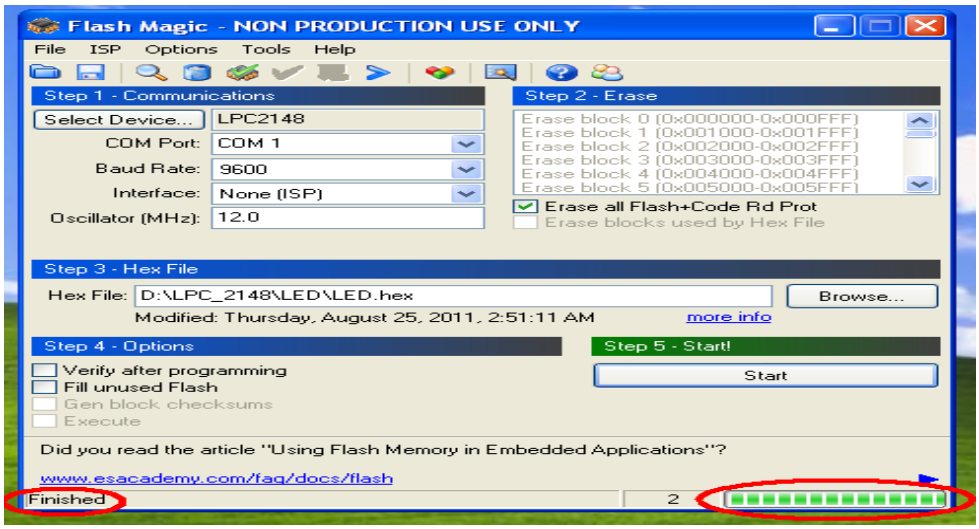
STEP 26: After selecting ISP mode on the Hardware Kit and click on start.



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.

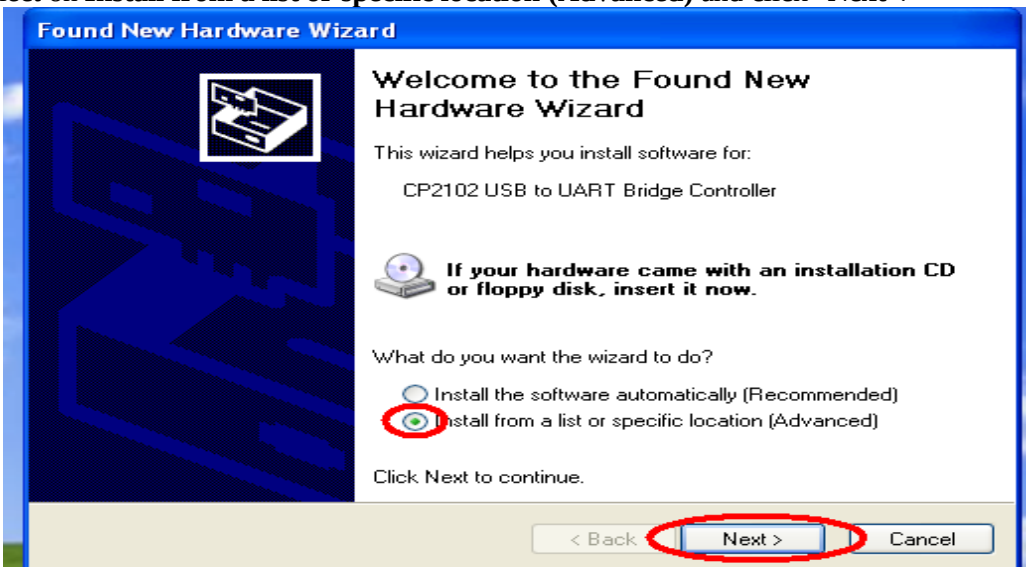


❖ Programming through USB:

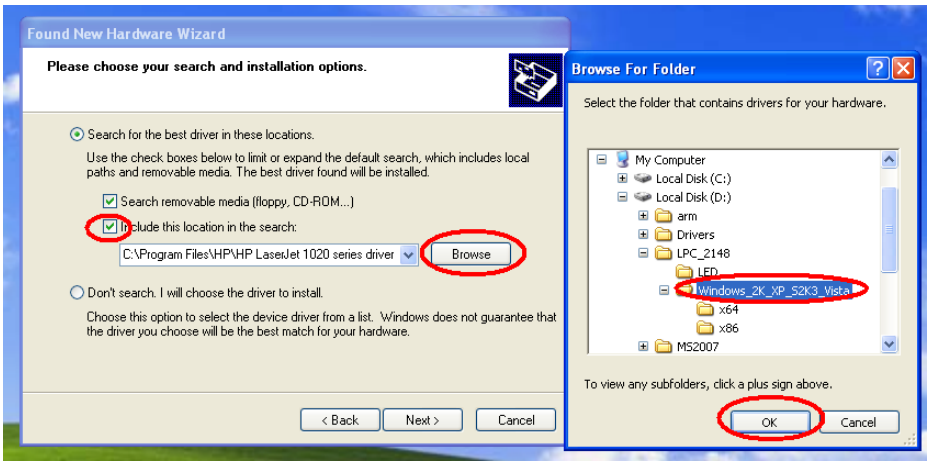
Installation Procedure for USB Cable Driver: The installation steps are givenbelow:

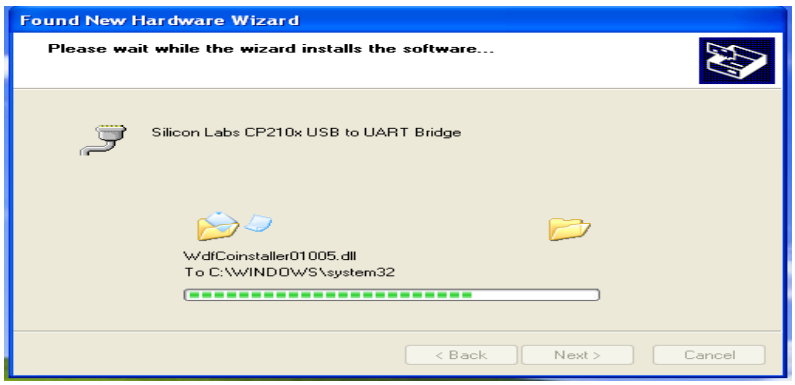
STEP 1: Connect the USB cable between ARM-7 LPC2148 Trainer Kit and PC. Connect the power supply to the trainer kit and power up. After can see a popup window with name "**Found New Hardware CP2102 USBto UART Bride Controller**".

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



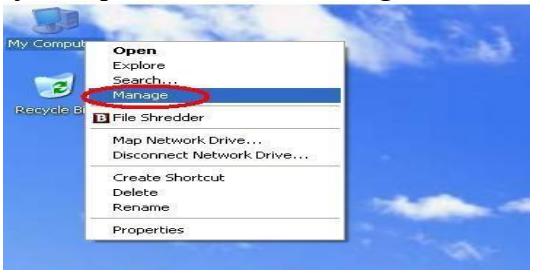
STEP 3: Browse for the driver file location and select the folder, click next.



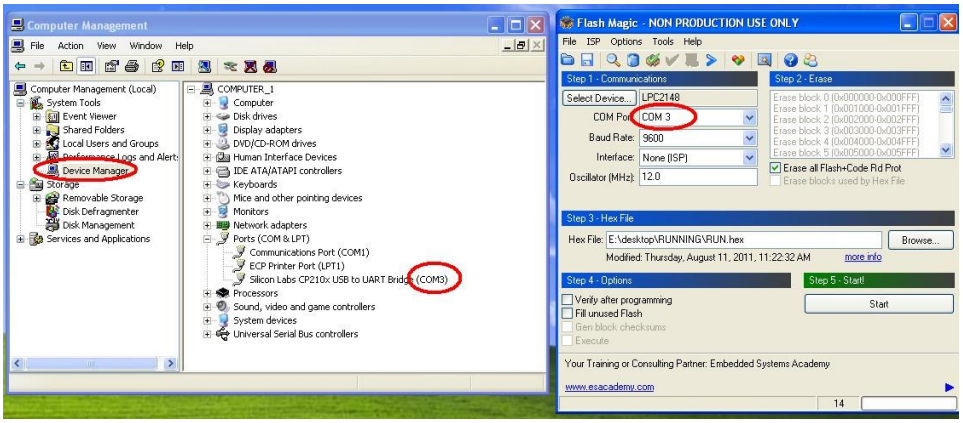


STEP 4: After that can see New Hardware wizard, and then click on finish.

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 COMhas to be select for Flash Magic.



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

ARM-7 LPC 2148 User Manual VT-2148-4.1

Features:

 Speed: 12 MHZ default, up to 60 MHZ with on Chip PLL.

 Flash-512 KB SRAM- 32 KB. USB device interface with USB connect LED

 Serial Ports: 2 UARTS; UART0 and UART1

 ADC: 2 Channels on Connector for external analog input. Facility to give POT and Lm35 input.

 DAC One 10 bit DAC output available as Test Point.

 8 digit Surface Mounted LEDs to display Digital Output.

 8 Push Button Switches to give digital input.

 Four Digit Seven Segment Display on board.

 On Board HEX Key Pad Interface (4 * 4 Matrix Keyboard).

 16*2 Alphanumeric LCD

 Relay with LED indication.

 On board Speaker (Buzzer) interface for Musical Tone Generation.

 On-board Stepper Motor interface.

 One PWM output on Test point with LED for DC Motor interface.

 One Push Button switch for External interrupt input with LED indication.

 JTAG Connector.

 Microcontroller (LPC2148) provided as Daughter Board for easy maintenance

 24 Port lines available on connector for expansion.

General Description:

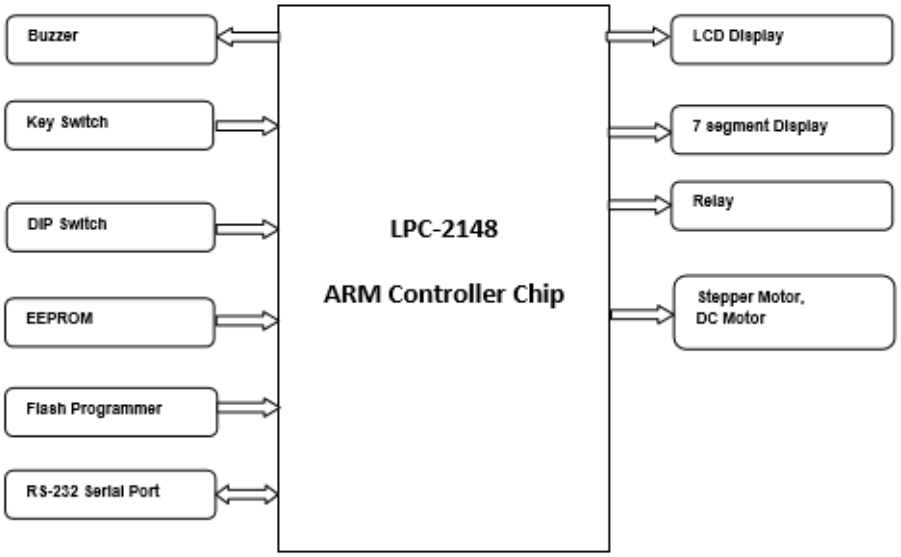
VT-2148-4.1 provides a hardware platform for developing embedded system using LPC2148device. These features make VT-2148-4.1 board ideal for instrumentation, communication

and other demanding application areas where flexibility and in-circuit hardware pgrade abilityis of paramount important.

VT-2148-4.1 board comes with complete drivers for Windows XP/7/10.LPC-2148 code examples and windows DLL interface which can be used to interface it to most common programming language.

It comes with complete Kiel development software and sample codes. It is also ideal forclassroom training in colleges and universities. Engineers can upgrade their hardware development Skills using ARM.

Block Diagram of VT 2148-2.0 ARM kit



DC Motor Pin Configuration

DC (direct current) motor rotates continuously. It has two terminals positive and negative. Connecting DC power supply to these terminals rotates motor in one direction and reversing the polarity of the power supply reverses the direction of rotation.

DC Motor Pin connection table

Motor Selection	Motor Direction	LPC2148 Pin No	LPC2148 Port No
Motor	DCM0-Clk	29 &30	P0.5=1 & P0.6=0
	DCM1-Aclk		P0.5=0 & P0.6=1
	DCM_EN	31	P0.7=1

The speed of Dc motor can maintained at a constant speed for a given load by using “Pulse Width Modulation (PWM)” technique. By changing the width of the pulse of applied to dc motor, the power applied is varied thereby DC motor speed can be increased or decreased. Wider the pulse Faster is the Speed, Narrower is the Pulse, and Slower is the Speed

Stepper Motor Configuration:

A stepper motor is a special type of electric motor that moves in increments, or steps, rather than turning smoothly as a conventional motor does. Typical increments are 0.9 or 1.8 degrees, with 400 or 200 increments thus representing a full circle. The speed of the motor isdetermined by the time delay between each incremental movement.

Stepper Motor Pin connection table

Stepper Motor Coil	LPC2148 Pin No	LPC2148 Port No
A	32	P1.24
B	28	P1.25
C	24	P1.26
D	64	P1.27

Relay Configurations:

A relay is an electrically operated switch. Many relays use an electromagnet to operate a switching mechanism mechanically, but other operating principles are also used. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electricalisolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits, repeating the signal coming in from one circuit and re-transmitting it to another. Relays wereused extensively in telephone exchanges and early computers to perform logical operations. Relay connection table.

Relay coil	LPC-2148 Pin No	LPC-2148 Port No
Relay	27	P0.4

Buzzer connection table.

Buzzer	LPC-2148 Pin No	LPC-2148 Port No
Buzzer	15	P0.30

ADC connection table.

ADC	LPC-2148 Pin No	LPC-2148 Port No
ADC0	13	P0.28
ADC1	14	P0.29

DAC connection table

DAC	LPC-2148 Pin No	LPC-2148 Port No
DAC	9	P0.25

External Interrupt connection table

External Interrupt	LPC-2148 Pin No	LPC-2148 Port No
EINT2	31	P0.7

HEX KEY PAD Pin configurations:

The hex keypad is a peripheral that is organized in rows and Columns. Hex key Pad 16 Keysarranged in a 4 by 4 grid, labeled with the hexadecimal digits 0 to F. Internally, the structureof the hex keypad is very simple. Wires run in vertical columns (we call them C0 to C3) andin horizontal rows (called R0 to R3). These 8 wires are available externally, and will be connected to the lower 8 bits of the port. Each key on the keypad is essentially a switch that connects a row wire to a column wire. When a key is pressed, it makes an electrical connection between the row and column. Table shows connections for HEX KEY Pad matrix.

HEX KEY Pad connection table

ROW/COLOUMNS	ROW1	ROW2	ROW3	ROW4	COL1	COL2	COL3	COL4
LPC-2148 Pin No	16	12	8	4	48	44	40	36
LPC-2148 Port No	P1.16	P1.17	P1.18	P1.19	P1.20	P1.21	P1.22	P1.23

USER KEY PAD Pin configurations:

VT- 2148-4.1 board provides eight individual KEYS connected to LPC-2148 device throughPORT1. S1 to S6,S11and S16 are connected to general purpose I/O pins on 2148 device as shown in table. 2148 device port pin drives to Logic „0“ when the corresponding key is pressed. Table shows connections for USER KEY Pad connection.

USER KEY Pad connection table

USER DEFINED KEYS	S1(K1)	S2(KY2)	S3(KY3)	S4(KY4)	S5(KY5)	S6(KY6)	S11(KY7)	S16(KY8)
LPC-2148 Pin No	16	12	8	4	48	44	40	36
LPC-2148 Port No	P1.16	P1.17	P1.18	P1.19	P1.20	P1.21	P1.22	P1.23

LED Pin Configuration

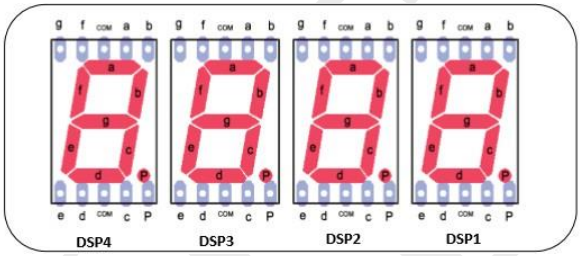
VT-2148-4.1 board provides eight individual SMD led"s connected to 2148 device through 74HC151driver IC.D3 to D10 are connected to general purpose I/O pins on 2148 device asshown in table. When 2148 device drives Logic „1" the corresponding LED turns on.

LED Pin connection table

LED	D1	D2	D3	D4	D5	D6	D7	D8
LPC-2148 Pin No	32	28	24	64	60	56	52	20
LPC-2148 Port No	P1.24	P1.25	P1.26	P1.27	P1.28	P1.29	P1.30	P1.31

SEVEN SEGMENT DIPLAY Pin Configuration:

D11, D12, D13 and D14 are Common Cathode segments connected to 2148 device so that each segment is individually controlled by a general purpose I/O pin. When the 2148 devicedrives logic '0' the corresponding segment turns on. See fig and table for more details



Seven Segment Pin connection table

Seven Segment Data Lines	g	f	a	b	p	c	d	e
2148 Pin No	45	46	47	53	54	55	1	2
2148 Port No	P0.15	P0.16	P0.17	P0.18	P0.19	P0.20	P0.21	P0.22

SEVEN SEGMENT Selection Pin Configurations:

As VT-2148-4.1 board comes with 4 digit seven segment unit. Displays connected to the microcontroller usually occupy a large number of valuable I/O pins, which can be a big problem especially if it is needed to display multi digit numbers. The problem is more than obvious if, for example, it is needed to display four digit numbers (a simple calculation showsthat 32 output pins are needed in this case). The solution to this problem is called MULTIPLEXING. This is how an optical illusion based on the same operating principle as a film camera is made. Only one digit is active at a time, but they change their state so quickly making impression that all digits of a number are simultaneously active. Each digit can made active using switching transistors Q1, Q2, Q3 and Q4 and these on switched on and off by selection lines which are in turn connected to 2148 ports. Table shows the details of seven segment selection lines.

Seven Segment Selection table.

Seven Segment Selection Lines	Disp1 (DIG 1)	Disp2 (DIG 2)	Disp3 (DIG 3)	Disp4 (DIG 4)
2148 Pin No	*35*	*37*	*38*	*39*
2148 Port No	*P0.10*	*P0.11*	*P0.12*	*P0.13*

Display Encoding:

Digits to display	**Display Segments**								**Hex code**
	g	*f*	*a*	*b*	*p*	*c*	*d*	*e*	
0	1	0	0	0	1	0	0	0	88
1	1	1	1	0	1	0	1	1	EB
2	0	1	0	0	1	1	0	0	4C
3	0	1	0	0	1	0	0	1	49
4	0	0	1	0	1	0	1	1	2B
5	0	0	0	1	1	0	0	1	19
6	0	0	0	1	1	1	1	0	18
7	1	1	0	0	1	0	1	1	CB
8	0	0	0	0	1	0	0	0	08
9	0	0	0	0	1	0	0	1	09
A	0	0	0	0	1	0	1	0	0A
B	0	0	1	1	1	0	0	0	38
C	1	0	0	1	1	0	0	0	98
D	0	1	1	0	1	0	0	0	68
E	0	0	0	1	1	1	0	0	1C
F	0	0	0	1	1	1	1	0	1E

16 X 2 LCD Display

An LCD display is specifically manufactured to be used with microcontrollers, which meansthat it cannot be activated by standard IC circuits. It is used for displaying different messages on a miniature liquid crystal display. It displays all the letters of alphabet, Greek letters, punctuation marks, mathematical symbols etc. In addition, it is possible to display symbols made up by the user. Other useful features include automatic message shift (left and right), cursor appearance, LED backlight etc.

There are pins along one side of a small printed board. These are used for connecting to the microcontroller. There are in total of 14 pins marked with numbers (16 if it has backlight). Their function is described in the table below

LCD Pin Configuration

LCD Pin No	LCD Pin Functions	LPC2148 Pin No	LPC2148 Port No
1	GND	-	-
2	VCC	-	-
3		-	-
4	RS	26	P0.3
5	R/W	GND	GND
6	EN	22	P0.2
7	DAT1	45	P0.15
8	DAT2	46	P0.16
9	DAT3	47	P0.17
10	DAT4	53	P0.18
11	DAT5	54	P0.19
12	DAT6	55	P0.20
13	DAT7	1	P0.21
14	DAT8	2	P0.22
15	VCC	-	-
16	GND	-	-

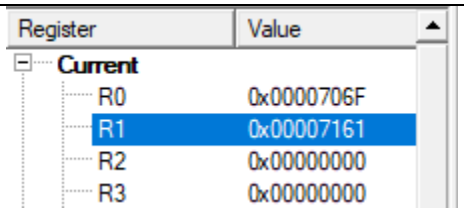
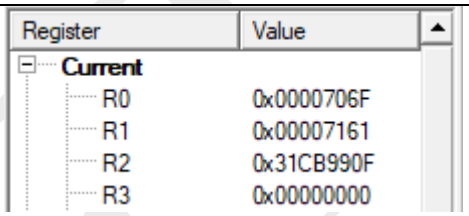
MODULE-1

1) Write a program to multiply two 16 bit binary numbers

Program:

```
        AREA  multi , CODE , READONLY
        LDR R0,=0X706F ; load 16 bit no. to R0
        LDR R1,=0X7161  ; load 16 bit no. to R1
        MUL R2,R1,R0       ; multiply R1 with R0 and store result in destregister R2
STOP B STOP
        END
```

Result:

Before Execution	After Execution
	

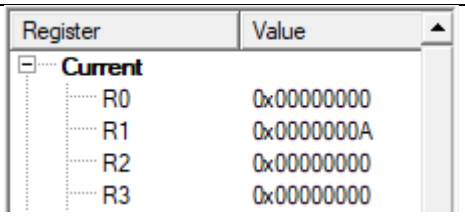
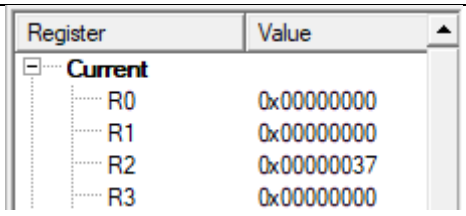
MODULE-2

2) Write a program to find the sum of first 10 integer numbers.

Program:

```
        AREA sum,CODE,READONLY
        MOV R1,#10 ; load 10 to register
        MOV R2,#0 ; empty R2 register to store result
loop    ADD R2,R1,R2 ; add the content of R1 with result at R2
        SUBS R1,#0x01 ; Decreament R1 by 1
        BNE loop ; repeat till R1 goes 0
STOP B STOP
        END
```

Result:

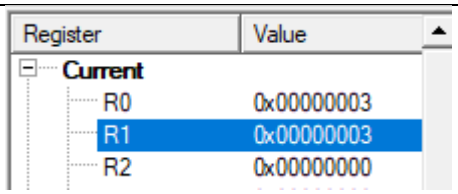
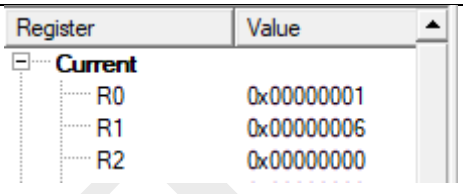
Before Execution	After Execution
	

3) Write a program to find factorial of a number.

Program:

```
        AREA Fact,CODE,READONLY
        MOV R0,#3 ; load N to register
        MOV R1,R0
LOOP  CMP R0,#1
        BEQ STOP
        SUB R0,R0,#1
        MUL R1,R0,R1
        B LOOP
STOP B STOP
        END
```

Result:

Before Execution	After Execution
	

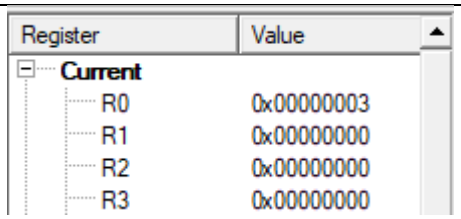
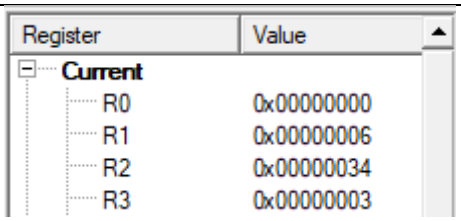
4) Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM

Program:

```
        AREA addarray,CODE,READONLY
        MOV R0,#3 ; load N to register
```

```
        MOV R1,#0
        LDR R2,=DATA1
LOOP    LDR R3,[R2]
        ADD  R1,R3,R1
        ADD  R2,R2,#4
        SUBS R0,R0,#1
        BGT LOOP
STOP B STOP
        AREA ARRAY,DATA,READONLY
DATA1
        DCW 1
        ALIGN
        DCW 2
        ALIGN
        DCW 3
        END
```

Result:

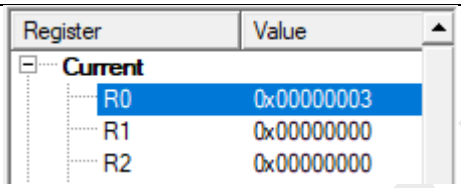
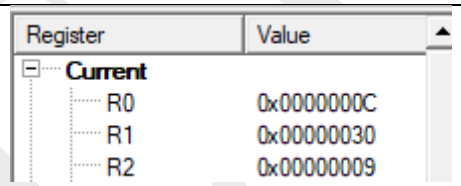
Before Execution	After Execution
	

5) Write a program to find the square of a number (1 to 10) using look-up table.

Program:

```
        AREA SQURE,CODE,READONLY
        MOV R0,#3 ; load N to register
        LDR R1,=DATA1
        MOV R2,#4
        MUL R0,R2,R0
        ADD R1,R0,R1
        MOV R2,#0; RESULT
        LDR R2,[R1]
STOP B STOP
        AREA ARRAY,DATA,READONLY
DATA1 DCD 0,1,2,9,16,25,36,49,64,81,100
        END
```

Result:

Before Execution	After Execution
	

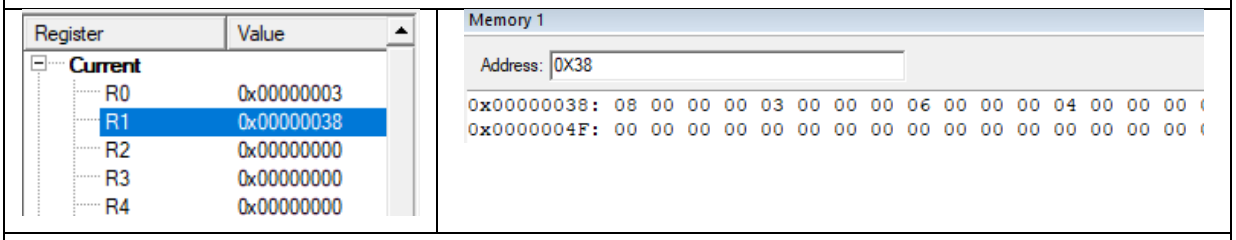
6a) Write a program to find the largest number in an array of 32 numbers .

Program:

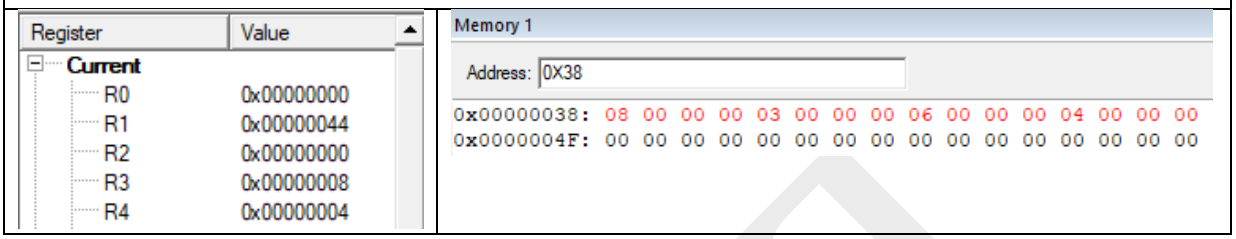
```
        AREA SMALL,CODE,READONLY
        MOV R0,#3 ; load N-1 to register
        LDR R1,=DATA1
        LDR R3,[R1]
UP      ADD R1,R1,#4
        LDR R4,[R1]
        CMP R3,R4
        BHI DOWN
        MOV R3,R4
DOWN
        SUBS R0,R0,#1
        CMP R0,#0
        BNE UP
        STR R3,[R2]
STOP B STOP
        AREA ARRAY,DATA,READONLY
DATA1
        DCD 8,3,6,4
        END
```

Result:

Before Execution



After Execution



6b) Write a program to find the smallest number in an array of 32 numbers .

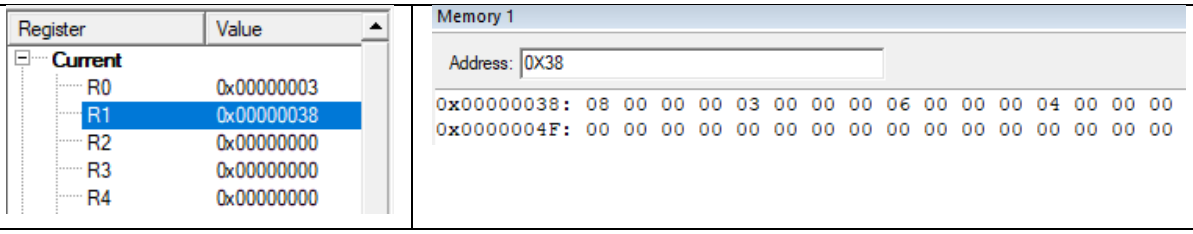
Program:

```
        AREA SMALL,CODE,READONLY
        MOV R0,#3 ; load N-1 to register
        LDR R1,=DATA1
        LDR R3,[R1]
UP      ADD R1,R1,#4
```

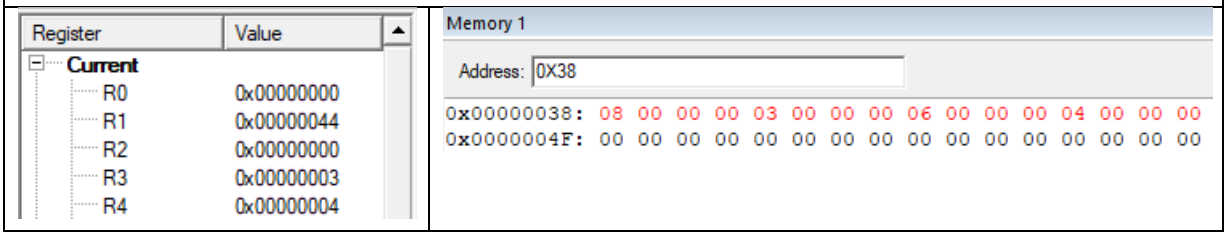
```
        LDR R4,[R1]
        CMP R3,R4
        BLO DOWN
        MOV R3,R4
DOWN
        SUBS R0,R0,#1
        CMP R0,#0
        BNE UP
        STR R3,[R2]
STOP B STOP
        AREA ARRAY,DATA,READONLY
DATA1
        DCD 8,3,6,4
        END
```

Result:

Before Execution



After Execution



MODULE-3

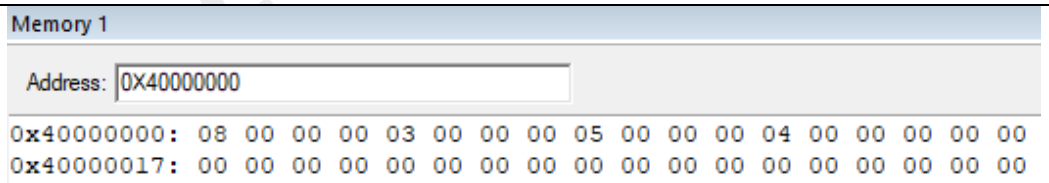
7a) Write a program to arrange a series of 32 bit numbers in ascending order.

Program:

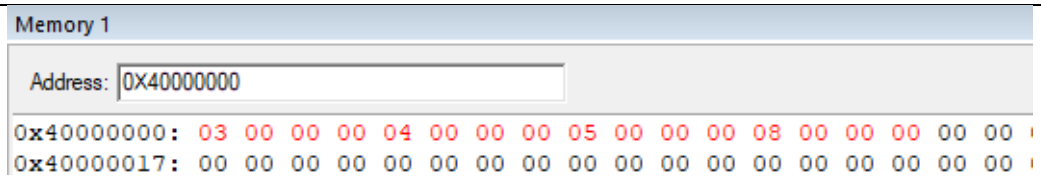
```
        AREA ASSEND, CODE, READONLY
        MOV R0, #3 ; load N-1 to
        registerMOV R5,R0
UP2     MOV R1, #0x40000000
UP      LDR R2, [R1], #4      ;ADD R1,R1,#4
        LDR R3, [R1]
        CMP R2, R3
        BLO DOWN
        STR R2, [R1], #-4     ;SUB R1,R1,#4
        STR R3,[R1],#4        ;ADD R1,R1,#4
DOWN
        SUBS R0, R0, #1
        CMP R0, #0
        BNE UP
        MOV R0, R5
        SUBS R5,R5,#1
        CMP R5,#0
        BNE UP2
STOP B STOP
        END
```

Result:

Before Execution



After Execution

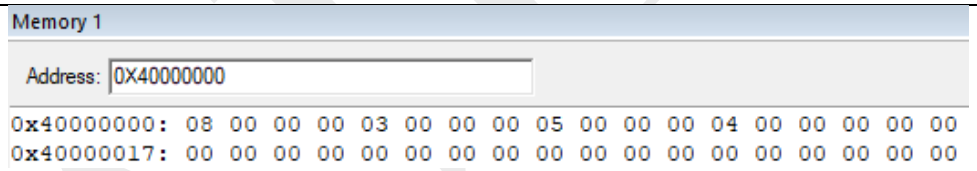


7b) Write a program to arrange a series of 32 bit numbers in ascending order.Program:

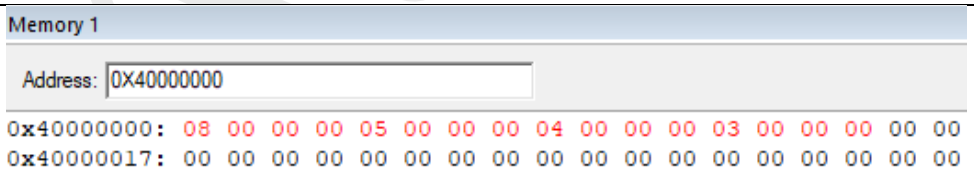
```
        AREA ASSEND, CODE, READONLY
        MOV R0, #3 ; load N-1 to
        registerMOV R5,R0
UP2     MOV R1, #0x40000000
UP      LDR R2, [R1], #4      ;ADD R1,R1,#4
        LDR R3,[R1]
        CMP R2, R3
        BHI DOWN
        STR R2, [R1], #-4     ;SUB R1,R1,#4
        STR R3,[R1],#4        ;ADD R1,R1,#4
DOWN
        SUBS R0, R0, #1
        CMP R0, #0 BNE
        UP
        MOV R0, R5
        SUBS R5, R5,#1
        CMP R5,#0 BNE
        UP2
STOP B STOP
        END
```

Result:

Before Execution



After Execution



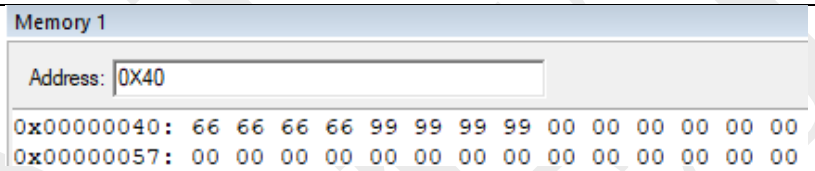
8) Write a program to count the number of ones and zeros in two consecutive memory locations

Program:

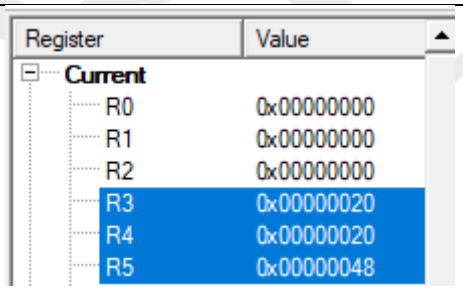
```
        AREA ONEZERO, CODE, READONLY
        MOV R6, #2
        LDR R5, =DATA1
UP2     LDR R0, [R5]
        MOV R1, #32
UP      MOVS R0, R0, LSR #1
        ADDCS R3, R3, #1; r3 HOLDS COUNT OF ONES
        ADDCC R4, R4, #1 ;r4 HOLDS COUNT OF
        ZEROSSUB R1,R1,#1
        CMP R1, #00
        BNE UP
        ADD R5, R5, #4
        SUB R6, R6, #1
        CMP R6, #00
        BNE UP2
STOP B STOP
        AREA ARRAY, DATA, READONLY
DATA1 DCD 0X66666666,0X99999999
        END
```

Result:

Before Execution



After Execution



9) Display "Hello World" message using Internal UART

Program:

```
#include <lpc214x.h>
void initClocks(void);
void initUART0(void);
void U0Write(char data);
void Send_String(char* StringPtr);
char String[]="    hello world \n\r\n";
unsigned int delay;
int main(void)
{
        initClocks(); // Set CCLK=60Mhz and PCLK=60Mhz
        initUART0();
        {
        Send_String(String);    //Pass the string to the USART_putstring function and sends it over the serial
        for(delay=0; delay<500000; delay++); // delay
        }
}
void initUART0(void)
{
        PINSEL0 = 0x5; /* Select TxD for P0.0 and RxD for P0.1 */
        U0LCR = 0x83; /* 8 bits, no Parity, 1 Stop bit | DLAB set to 1 */
        U0DLL = 110;
        U0DLM = 1;
        U0FDR = 0xF1; /* MULVAL=15(bits - 7:4) , DIVADDVAL=0(bits - 3:0)*/
        U0LCR &= 0x0F; // Set DLAB=0 to lock MULVAL and DIVADDVAL
        //BaudRate is now ~9600 and we are ready for UART communication!
}
void U0Write(char data)
{
        while (!(U0LSR & (1<<5))); // wait till the THR is empty
        // now we can write to the Tx FIFO
        U0THR = data;
}
void initClocks(void)
{
        PLL0CON = 0x01;  //Enable PLL
        PLL0CFG = 0x24; //Multiplier and divider setup
        PLL0FEED = 0xAA;  //Feed sequence
        PLL0FEED = 0x55;
        while(!(PLL0STAT & 0x00000400)); //is locked?
        PLL0CON = 0x03;  //Connect PLL after PLL is locked
        PLL0FEED = 0xAA;  //Feed sequence
        PLL0FEED = 0x55;
```

```
        VPBDIV = 0x01;    // PCLK is same as CCLK i.e.60 MHz
}

void Send_String(char* StringPtr)
{
  while(*StringPtr != 0x00)
        {
                U0Write(*StringPtr);
                StringPtr++;
        }
}
```

MODULE-4

10) Interface and Control a DC Motor.

Program:

```
#include <LPC214x.H>
int main (void)
{
 IO1DIR &= (0<<16)& (0<<17)&(0<<18);
 IO0DIR = (1<<5)|(1<<6);
 while(1)
 {
        if (!(IO1PIN & (1<<16)))       //Clock wise key pressed
                {
                 IO0PIN = 0X00000040 ;
                 }
               if (!(IO1PIN & (1<<18)))       //OFF key pressed
                {
                        IO0PIN = 0X00000060 ;
                 }
               if (!(IO1PIN & (1<<17)))       // Counterclockwise key pressed
                {
                        IO0PIN = 0X00000020 ;
                 }
        }
}
```

11) Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.

Program:

```
#include <LPC214x.H>
void delay(unsigned int count)
{
 int j=0,i=0;
 for(j=0;j<count;j++)
 {
  for(i=0;i<500;i++);
 }
}
int main (void)
{
        unsigned char run=1;
        IO1DIR |= (1<<24)|(1<<25)|(1<<26)|(1<<27); //|0x0F000000;//make Port P1.31 to P1.24 as output
        PINSEL2 = 0x0; // to ensure RTCK/P1.23  behaves like GPIO, no accidental JTAG
{
while(1)
{
```

```
        if (!(IO1PIN & (1 << 16)) )
                run = 1;
        if (!(IO1PIN & (1 << 17)))
                run = 0;
        if (run==1)
                {
                        IO1PIN = 0X01000000;
                        delay(100);
                        IO1PIN = 0X02000000;
                        delay(100);
                        IO1PIN = 0X04000000;
                        delay(100);
                        IO1PIN = 0X08000000;
                        delay(100);
                }
        else
                {
                        IO1PIN = 0X08000000;
                        delay(100);
                        IO1PIN = 0X04000000;
                        delay(100);
                        IO1PIN = 0X02000000;
                        delay(100);
                        IO1PIN = 0X01000000;
                        delay(100);}
                }
        }
}
```

12) Determine Digital output for a given Analog input using Internal ADC of ARMcontroller.

Program:

```
#include <LPC214x.H>
#include <stdio.h>
#include "lcd.h"
unsigned int adc_value = 0;
char buf[] ={2};

int main (void)
{
          PINSEL1 = 0X01000000;
         AD0CR = 0x00200D02;  //0x00200D04;
          init_lcd();
  while(1)
  {
         AD0CR |= 0x01000000; // Start A/D Conversion
   do
     {
       adc_value  = AD0GDR; // Read A/D Data Register
     } while ((adc_value & 0x80000000) == 0); // Wait for end of A/D Conversion

         adc_value = (adc_value >> 6) & 0x03FF; // bit 6:15 is 10 bit AD value
         sprintf((char *)buf, "%3d", adc_value);
         lcd_putstring16(0,"ADC VAL = 000 ");
         lcd_gotoxy(0,10);
         lcd_putstring(buf);
         delay(50000);
  }
}
```

13 a) Interface a DAC and generate Square waveforms.

Program:

```
  DACR = 0;
}

////////// Write DAC /////////////////
Write_DAC(unsigned int dacval)
{
         DACR = dacval << 6;
}
void delay(unsigned int count)
{
  int j=0,i=0;

  for(j=0;j<count;j++)
```

```
  {
    for(i=0;i<120;i++);
  }
}
////////// MAIN /////////////////
int main (void)
{

   Init_DAC();

  while(1)
  {
        Write_DAC(00);
        delay(100);                    // change this value to change Frequency
        Write_DAC(1023);    // change this value to change Amplitude
        delay(100);                 // change this value to change Frequency
  }
}
```

13 b) Interface a DAC and generate Triangular waveforms.

Program:

```
#include <LPC214x.H>
Init_DAC()
{
  PINSEL1 = 0X00080000; // Convert Port pin 0.25 to function as DAC
  DACR = 0;
}
Write_DAC(unsigned int dacval)
{
        DACR = dacval << 6;
}

int main (void)
{
unsigned int i;
Init_DAC();
while(1)
  {
        for(i=0;i<1024;i++)
                Write_DAC(i);
        for(i=1023;i>0;i--)
                Write_DAC(i);
  }
}
```

14) Interface a 4x4 keyboard and display the key code on an LCD.
Program:

```
#include <LPC214x.H>                    /* LPC214x definitions */
#include "lcd.h"
int main (void)
{
unsigned char key, i;
unsigned char rval[] = {0x7,0xB,0xD,0xE,0x0};
unsigned char keyPadMatrix[] =
{
   'C','8','4','0',
   'D','9','5','1',
   'E','A','6','2',
   'F','B','7','3'
};
 init_lcd();
 IO1DIR |= 0X000F0000;  //Set COLs as Outputs and Set ROW lines as Inputs
while (1)
 {
   key = 0;
   for( i = 0; i < 4; i++ )
   {
      IO1CLR = ((1 << 16) | (1 << 17) | (1 << 18) | (1 << 19));
      IO1SET = rval[i]<<16;
      // read rows - break when key press detected
      if (!(IO1PIN & (1 << 20)))
         break;
      key++;
      if (!(IO1PIN &(1 << 21)))
         break;
      key++;
      if (!(IO1PIN & (1 << 22)))
         break;
      key++;
      if (!(IO1PIN & (1 << 23)))
         break;
      key++;
   }
         if (key == 0x10)
                  {key =0;}
         else
                  {
                  lcd_putstring16(0,"Key Pressed = ");
                  lcd_gotoxy(0,14);
```

```
                lcd_putchar(keyPadMatrix[key]);
                }
}}
```

15)Demonstrate the use of an external interrupt to toggle an LED On/Off.

Program:

```
#include <LPC214x.H>
void delay(int count);
__irq void Ext_ISR(void);
int main (void)
{
 EXTMODE = 0x4;          //Edge sensitive mode on EINT2
 EXTPOLAR &= ~(0x4);     //Falling Edge Sensitive
 PINSEL0 = 0x0000C000;   //Select Pin function P0.15 as EINT2
   /* initialize the interrupt vector */
 VICIntSelect &= ~ (1<<16);         // EINT2 selected as IRQ 16
 VICVectAddr5 = (unsigned int)Ext_ISR; // address of the ISR
 VICVectCntl5 = (1<<5) | 16;     //
 VICIntEnable = (1<<16);            // EINT2 interrupt enabled
 EXTINT &= ~(0x4);
   while (1)
 {     }
}
__irq void Ext_ISR(void) // Interrupt Service Routine-ISR
{
 IO1DIR |=0xFF000000;//make Port P1.31 to P1.24 as output
 IO1PIN ^= 0x01000000;  // Turn ON Buzzer
 //delay(10);
 //IO1PIN |= 0x00000000;//(1<<25);    // Turn OFF Buzzer
 EXTINT |= 0x4;      //clear interrupt
 VICVectAddr = 0;    // End of interrupt execution
}
```

16) Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in Between

Program:

```
#include <LPC214x.H>
unsigned char dig[] =
{0x88,0xeb,0x4c,0x49,0x2b,0x19,0x18,0xcb,0x8,0x9,0xa,0x38,0x9c,0x68,0x1c,0x1e};
void delay(unsigned int count)
{
  int j=0,i=0;
  for(j=0;j<count;j++)
  {
    for(i=0;i<120;i++);
```

```
  }
}
int main (void)
{
unsigned char count=0;
unsigned int i=0;
IO0DIR |= (1 << 11);  //Set Digit control lines as Outputs
IO0SET = (1 << 11);
IO0DIR |= 0x007F8000;
  while(1)
 {
                    count++;
                    if(count ==16) count = 0;
                    for (i=0; i < 800; i++) //change to inc/dec speed of count
                     {
                      IO0CLR = 0x007F8000;
                      IO0SET = (dig[count] << 15);
                     delay(200);
                    }
  }}
```

MODULE-5

Demonstration of IoT applications by using Arduino and Raspberry Pi

VIVA QUESTIONS

1. Name the number of ports in ARM

2. Name the number of pins in ARM

3. How to configure the ports as input and output ports

4. Name the important peripherals in ARM

5. List the flags present in ARM

6. Name the software and version of software used for interfacing

7. What is PWM and how to configure it

8. What is the step angle of a stepper motor used in lab.

9. How to vary the speed and direction of a stepper motor.

10. What is baud rate and how to select the baud rate

11. Name the different baud rates available

12. What is serial data transfer. What type of data transfer is used in experiment.

13. What is SPI? Name the IC used for interfacing

14. What are the digital inputs required for turning on Common Anode and common Cathode display

15. Explain the logic behind the generation of square and triangular waveform using DAC

16. Name the different pins of LCD

17. What is the meaning of 16X2 in LCD

18. Name the number of scanning lines and return lines in the keypad

19. What is the difference between binary and BCD. Write the binary and BCD value for 0xC9

20. Name the profile used in ARM

Appendix

Register description

LPC2148 has two 32-bit General Purpose I/O ports. Total of 30 input/output and a single output only pin out of 32 pins are available on PORT0. PORT1 has up to 16 pins available for GPIO functions. PORT0 and PORT1 are controlled via two groups of 4 registers namely

1. IOPIN (port pin value register).
2. IOSET (port output set value register).
3. IODIR (port direction control register).
4. IOCLR (port output clear register).

GPIO register map

Generic Name	Description	Access	Reset Value	PORT0 Address & Name	PORT1 Address & Name
IOPIN	GPIO Port Pin value register. The current state of the GPIO configured port pins can always be read from this register, regardless of pin direction.	R/W	NA	0xE002 8000 IO0PIN	0xE002 8010 IO1PIN
IOSET	GPIO Port Output Set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect	R/W	0	0xE002 8004 IO0SET	0xE002 8014 IO1SET
IODIR	GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	0xE002 8008 IO0DIR	0xE002 8018 IO1DIR
IOCLR	GPIO Port Output Clear register. This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	WO	0	0xE002 800C IO0CLR	0xE002 801C IO1CLR

The Pin Control Module contains 3 registers

Name	Description	Access	Reset value	Address
PINSEL0	Pin function select register 0.	Read/Write	0x0000 0000	0xE002 C000
PINSEL1	Pin function select register 1.	Read/Write	0x0000 0000	0xE002 C004
PINSEL2	Pin function select register 2.	Read/Write	-	0xE002 C014

Pin function select register value

PINSEL0 and PINSEL1 Values	Function	Value after Reset
00	Primary (default) function, typically GPIO port	00
01	First alternate function	
10	Second alternate function	
11	Reserved	

UARTs

The LPC2148 each contain two UARTs. In addition to standard transmit and receive data lines, the LPC2148 UART1 also provides a full modem control handshake interface.

Compared to previous LPC2000 microcontrollers, UARTs in LPC2148 introduce a fractional baud rate generator for both UARTs, enabling these microcontrollers to achieve standard baud rates such as 115200 with any crystal frequency above 2 MHz. In addition, auto- CTS/RTS flow-control functions are fully implemented in hardware

Features

• 16 byte Receive and Transmit FIFOs.
• Register locations conform to _550 industry standard.
• Receiver FIFO trigger points at 1, 4, 8, and 14 bytes
• Built-in fractional baud rate generator covering wide range of baud rates without a need for external crystals of particular values.
• Transmission FIFO control enables implementation of software (XON/XOFF) flow control on both UARTs.
• LPC2148 UART1 equipped with standard modem interface signals. This module also provides full support for hardware flow control (auto-CTS/RTS).