ST. ANNE'S COLLEGE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC

(Approved by AICTE, New Delhi. Affiliated to Anna University, Chennai)

ANGUCHETTYPALAYAM, PANRUTI – 607 106.

LABORATORY MANUAL

EC3491 EMBEDDED SYSTEMS AND IOT LAB MANUAL

OBSERVATION NOTE

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

NAME :

REGISTER NO :

YEAR / SEMESTER :

PERIOD :

AS PER ANNA UNIVERSITY (CHENNAI) SYLLABUS

2021 REGULATION

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PREPARED BY

B.ARUN KUMAR AP/ECE

EC3491 EMBEDDED SYSTEMS AND IOT LAB

SYLLABUS

PRACTICAL EXERCISES: **30 PERIODS**

Experiments using 8051

1. Programming Arithmetic and Logical Operations in 8051.
2. Generation of Square waveform using 8051.
3. Programming using on – Chip ports in 8051.
4. Programming using Serial Ports in 8051.
5. Design of a Digital Clock using Timers/Counters in 8051.

Experiments using ARM

1. Interfacing ADC and DAC
2. Blinking of LEDs and LCD
3. Interfacing keyboard and Stepper Motor.

Miniprojects for IoT

1. Garbage Segregator and Bin Level Indicator
2. Colour based Product Sorting
3. Image Processing based Fire Detection
4. Vehicle Number Plate Detection
5. Smart Lock System

COURSE OUTCOMES:

CO1: Explain the architecture and features of 8051.

CO2: Develop a model of an embedded system.

CO3: List the concepts of real time operating systems.

CO4: Learn the architecture and protocols of IoT.

CO5: Design an IoT based system for any application.

EXP NO:	BASIC ARITHMETIC AND LOGICAL OPERATIONS USING 8051
DATE:	A. 8 BIT ADDITION

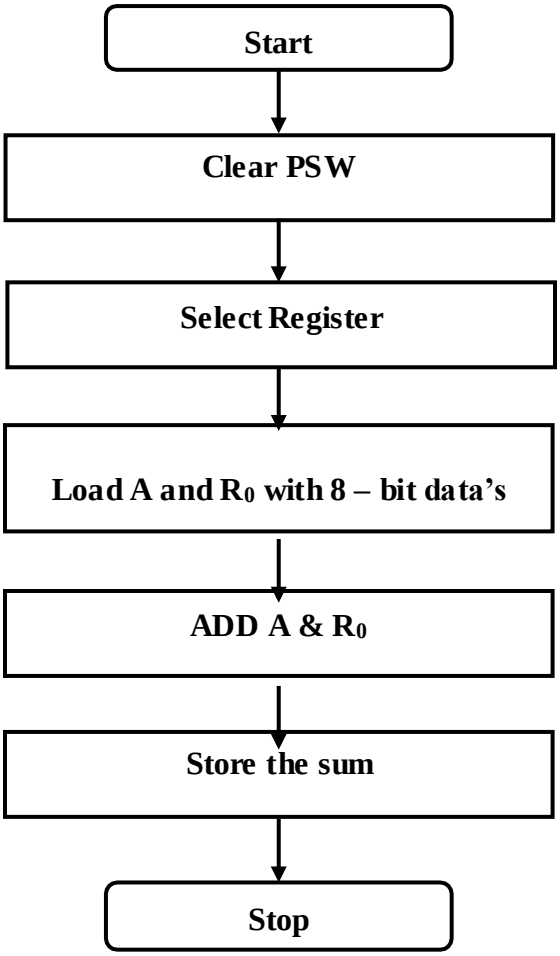
AIM:

To write a program to add two 8-bit numbers using 8051 microcontrollers.

ALGORITHM:

1. Clear Program Status Word.
2. Select Register bank by giving proper values to RS1 & RS0 of PSW.
3. Load accumulator A with any desired 8-bit data.
4. Load the register R 0 with the second 8- bit data.
5. Add these two 8-bit numbers.
6. Store the result.
7. Stop the program.

FLOW CHART:


Start
Clear PSW
Select Register
Load A and R0 with 8 – bit data's
ADD A & R0
Store the sum
Stop

PROGRAM:

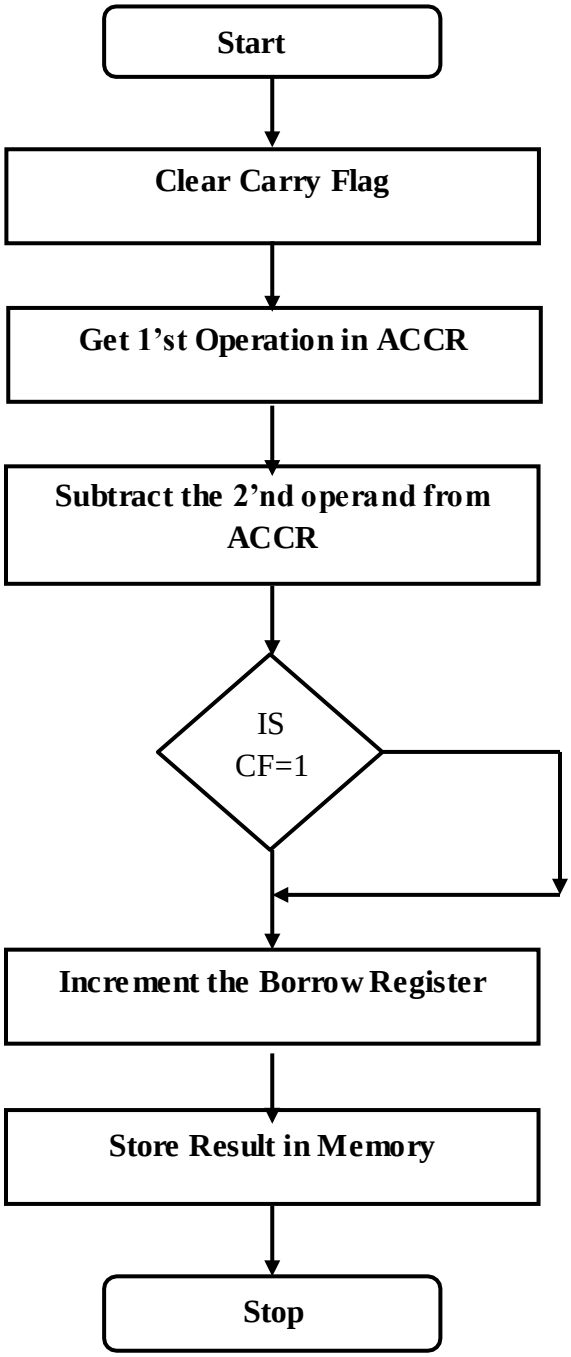
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	CLR	C	C3	Clear CY Flag
	4101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	4103	ADDC	A,#10	34 10	Add the data1 with data 2
	4105	MOV	DPTR,#4500	90 45 00	Initialize the memory location
	4108	MOVX	@DPTR,A	F0	Store the result in memory location
L1	4109	SJMP	L1	80 FE	Stop the program

Address	**Output**
4500	**1A(LSB)**
4501	**00(MSB)**

RESULT:

Thus the 8051 Assembly Language Program for addition of two 8 bit numbers was executed.

FLOW CHART:


Start
Clear Carry Flag
Get 1'st Operation in ACCR
Subtract the 2'nd operand from ACCR
IS
CF=1
Increment the Borrow Register
Store Result in Memory
Stop

B. 8 BIT SUBTRACTION

AIM:

To perform subtraction of two 8 bit data and store the result in memory.

ALGORITHM:

1. Clear the carry flag.
2. Initialize the register for borrow.
3. Get the first operand into the accumulator.
4. Subtract the second operand from the accumulator.
5. If a borrow results increment the carry register.
6. Store the result in memory.

PROGRAM:

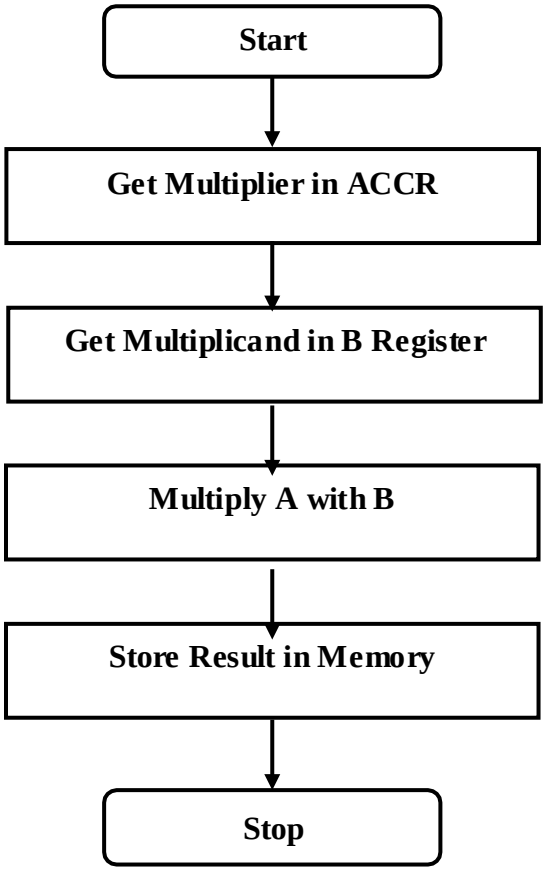
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	CLR	C	C3	Clear CY Flag
	4101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	4103	SUBB	A,#05	94 05	Subtract data2 from data1
	4105	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4108	MOVX	@DPTR,A	F0	Store the difference in memory location
L1	4109	SJMP	L1	80 FE	Stop the program

Address	**Output**
4500	**05**

RESULT:

Thus the 8051 Assembly Language Program for subtraction of two 8 bit numbers was executed.

FLOW CHART:


Start
Get Multiplier in ACCR
Get Multiplicand in B Register
Multiply A with B
Store Result in Memory
Stop

C. 8 BIT MULTIPLICATION

AIM:

To perform multiplication of two 8 bit data and store the result in memory.

ALGORITHM:

1. Get the multiplier in the accumulator.
2. Get the multiplicand in the B register.
3. Multiply A with B.

Store the product in memory

PROGRAM:

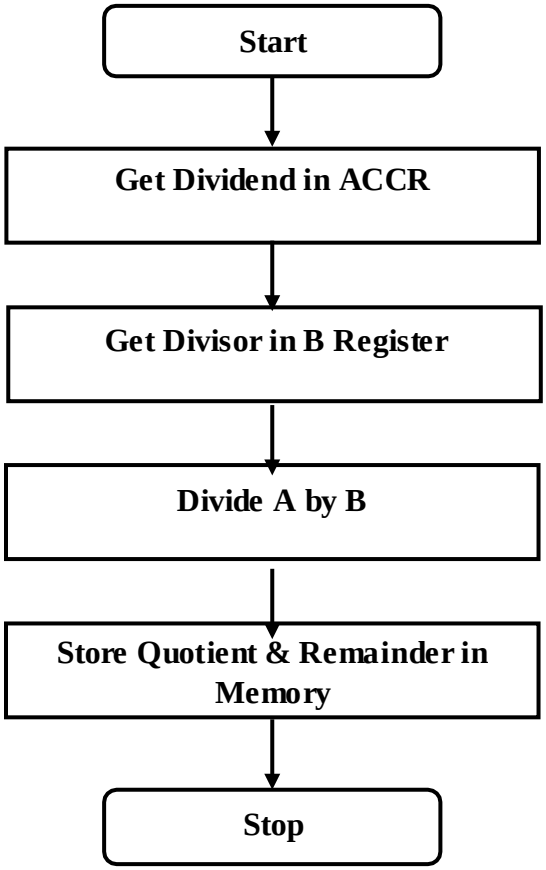
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	MOV	A,#05	74 05	Store data1 in accumulator
	4102	MOV	B,#03	75 F0 03	Store data2 in B register
	4105	MUL	AB	A4	Multiply both
	4106	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4109	MOVX	@DPTR,A	F0	Store lower order result
	410A	INC	DPTR	A3	Go to next memory location
	410B	MOV	A,B	E5 F0	Store higher order result
	410D	MOVX	@DPTR,A	F0	
L1	410E	SJMP	L1	80 FE	Stop the program

Address	**Output**
4500	**0F(LSB)**
4501	**00(MSB)**

RESULT:

Thus the 8051Assembly Language Program for multiplication of two 8 bit numbers was executed.

FLOW CHART:


Start
Get Dividend in ACCR
Get Divisor in B Register
Divide A by B
Store Quotient & Remainder in Memory
Stop

D. 8 BIT DIVISION

AIM:

To perform division of two 8 bit data and store the result in memory.

ALGORITHM:

1. Get the Dividend in the accumulator.
2. Get the Divisor in the B register.
3. Divide A by B.

Store the Quotient and Remainder in memory

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	MOV	A,#15	74 15	Store data1 in accumulator
	4102	MOV	B,#03	75 F0 03	Store data2 in B register
	4105	DIV	AB	84	Divide
	4106	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4109	MOVX	@DPTR,A	F0	Store remainder
	410A	INC	DPTR	A3	Go to next memory location
	410B	MOV	A,B	E5 F0	Store quotient
	410D	MOVX	@DPTR,A	F0	
L1	410E	SJMP	L1	80 FE	Stop the program

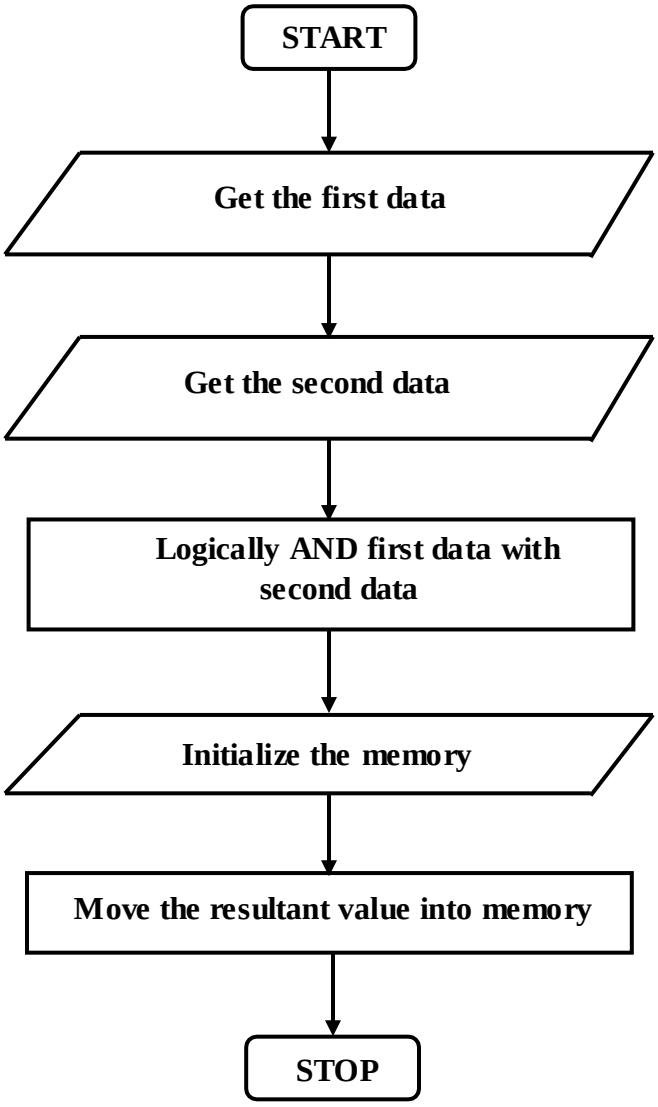
Input	
Memory Location	**Data**
4500 (dividend)	**0F**
4501 (divisor)	**03**

Output	
Memory Location	**Data**
4502 (remainder)	**05**
4503 (quotient)	**00**

RESULT:

Thus the 8051 8051Assembly Language Program for division of two 8 bit numbers was executed.

FLOW CHART:


START
Get the first data
Get the second data
Logically AND first data with second data
Initialize the memory
Move the resultant value into memory
STOP

D. MASKING BITS IN AN 8 – BIT NUMBER

AIM:

To write an assembly language program to mask bits o and 7 of an 8 – bit number and store the result in memory using 8051 microcontrollers.

APPARATUS REQUIRED:

8051 microcontroller kit

ALGORITHM:

Masking bits in a 8 bit number

- Start the process
- Get the two data values
- Get the second data
- Logically 'AND' the two data values.
- Initialize the memory value and store the result in memory.
- Start the process

PROGRAM:

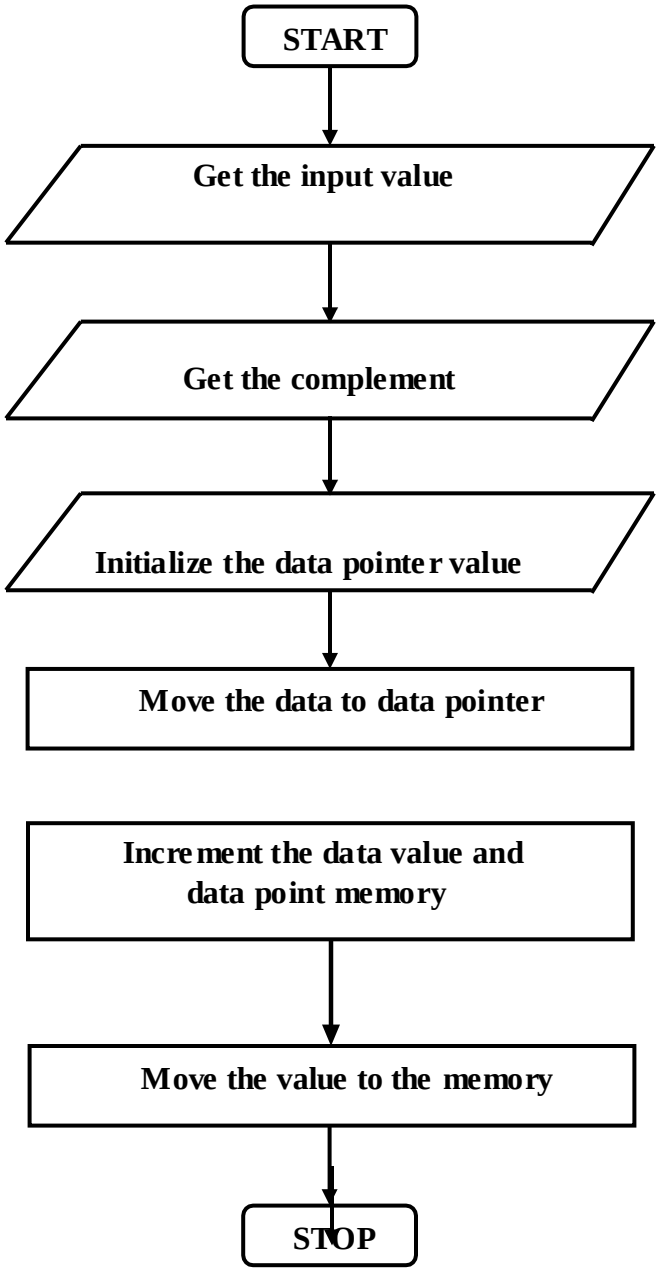
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	4100	MOV	A,#87	74 87	
	4102	ANL	A,#7E	54 7E	
	4104	MOV	DPTR,#4500	90 45 00	
	4107	MOVX	@DPTR,A	F0	
L1	4108	SJMP	L1	80 FE	

Output	
Memory Location	**Data**
4500	**06**

RESULT:

Thus the 8051assembly language program for masking bits was executed and verified

a) 1's and 2's complement



EXP NO:	SQUARE AND CUBE PROGRAM, FIND 2'S COMPLEMENT OF A NUMBER
DATE	

AIM:-

To write an assembly language to perform arithmetic, logical and bit manipulation instruction using 8051.

ALOGRITHM:

a) 1's and 2's complement

- Get the value
- Get the complement value of data.
- Initialize the data pointer value as memory.
- Move the complemented value to memory of data pointer.
- Increment the value and memory.
- Store the result in memory.
- Stop the process.

a) 1's and 2's complement

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	4100	MOV	A, #02	74, 02	Get the initial value
	4102	CPL	A	F4	Complement the value
	4103	MOV	DPTR, # 4200	90, 42, 00	Initialize the memory
	4106	MOVX	@ DPTR, A	F0	Move the data to memory
	4107	INC	A	04	Increment Accumulator
	4108	INC	DPTR	A3	Increment the memory
	4109	MOVX	@ DPTR, A	F0	Move the value to memory
ECE:	410A	SJMP	ECE	80, FE	Continue the process.

1’s and 2’s complement

Output	
Memory Location	**Data**
4200	**FD (1’s complement**
4201	**FE(2’S Complement)**

Square of a number

Input		Output	
Memory Location	**Data**	**Memory Location**	**Data**
4200	**89**	**4201**	**51**
		4202	**49**

b) SQUARE PROGRAM FOR 8051

```
$MOD51
ORG 4100H
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
MUL AB
INC DPTR
MOVX @DPTR,A
INC DPTR
MOV A,B
MOVX @DPTR,A
L:SJMP L
```

****

C). CUBE PROGRAM FOR 8051

```
$MOD51
ORG 4100H
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
MOV R7,A
MUL AB
MOV R0,A
MOV R1,B
MOV A,R7
ANL A,#0FH
MOV B,A
MOV A,R0
MUL AB
MOV R2,A
MOV R3,B
MOV A,R1
MOV B,A
MOV A,R7
ANL A,#0FH
MUL AB
MOV R4,A
MOV R5,B
MOV A,R3
MOV B,R4
ADD A,B
MOV R3,A
```

```
MOV A,R0
MOV B,A
MOV A,R7
ANL A,#0F0H
SWAP  A
MUL AB
MOV R4,A
MOV R6,B
MOV A,R7
ANL A,#0F0H
SWAP A
MOV B,R1
MUL AB
MOV R0,A
MOV R1,B
MOV A,R6
MOV B,R0
ADD A,B
MOV R0,A
MOV R7,A
MOV A,R4

SWAP  A
ANL A,#0F0H
MOV R6,A
MOV A,R0
SWAP  A
ANL A,#0F0H
MOV R0,A
MOV A,R4
SWAP  A
ANL A,#0FH
MOV R4,A
MOV B,R0
ADD A,B
MOV R4,A
MOV A,R1
SWAP A
MOV B,A
MOV A,R7
SWAP  A
ANL A,#0FH
ADD A,B
MOV R0,A
MOV A,R6
MOV B,R2
ADD A,B
MOV R6,A
MOV A,R3
MOV B,R4
ADDC A,B
MOV R3,A
```

```
MOV A,R0
MOV B,R5
ADDC A,B
MOV R0,A
MOV DPTR,#4500H
MOV A,R6
MOVX @DPTR,A
INC DPTR
MOV A,R3
MOVX @DPTR,A
INC DPTR
MOV A,R0
MOVX @DPTR,A
L:SJMP L
```

Cube of a number

Input		Output	
Memory Location	**Data**	**Memory Location**	**Data**
4200	**89**	**4500**	**56**
		4501	**3C**
		4502	**27**

RESULT:

Thus the assembly language program to find 2's complement, Square and cube of a number was executed and verified successfully using 8051 microcontrolle

EXP NO:	A/D INTERFACE WITH 8051
DATE	

AIM:

To write an assembly language program for interfacing of ADC with 8051.

ALGORITHM: -

(i) Start the program

(ii) Make CS=0 and send a low to high pulse to WR pin to start the conversion.

(iii) Now keep checking the INTR pin. INTR will be 1 if conversion is not finished and INTR will be 0 if conversion is finished.

(iv) If conversion is not finished (INTR=1) , poll until it is finished.

(v) If conversion is finished (INTR=0), go to the next step.

(vi) Make CS=0 and send a high to low pulse to RD pin to read the data from the ADC

(vii) Stop the program

PROCEDURE:

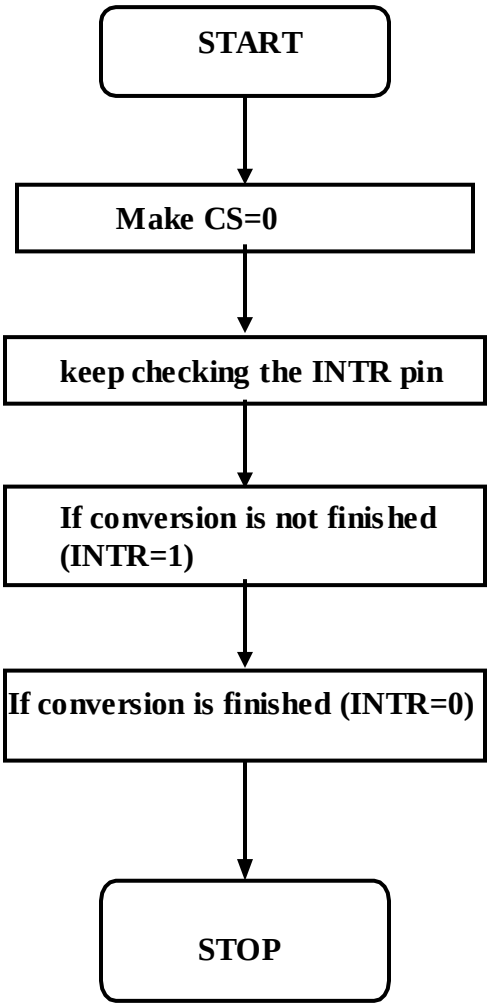
(i) Place jumper J2 in C position

(ii) Place jumper J5 in A position

(iii) Enter and execute the program

(iv) Vary the analog input (using trim pot) and view the corresponding digital value in LED display,

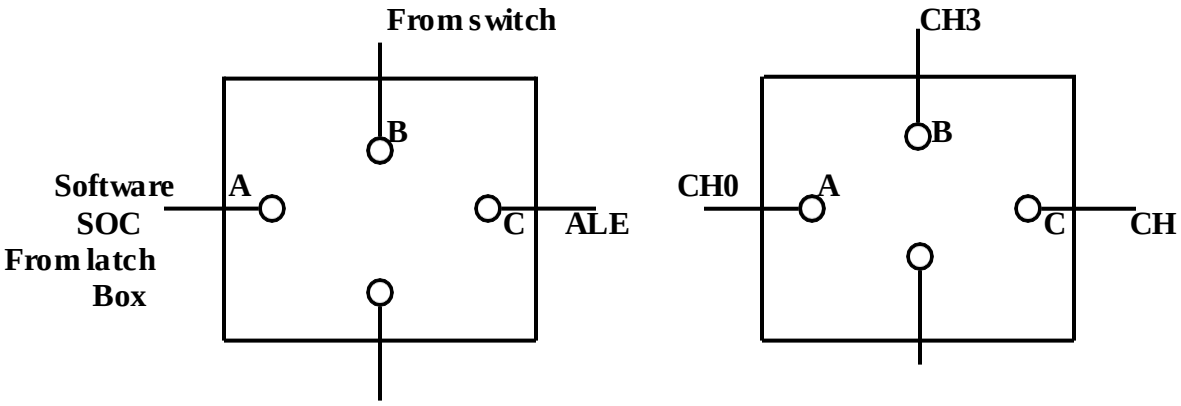
FLOWCHART:


START
Make CS=0
keep checking the INTR pin
If conversion is not finished (INTR=1)
If conversion is finished (INTR=0)
STOP

PROGRAM:

Label	Address	Mnemonics	Comments
	1000	MOV P1,#11111111B	; initiates P1 as the input port
MAIN:		CLR P3.7	; makes CS=0
	1003	SETB P3.6	;makes RD high
		CLR P3.5	
	1005	SETB P3.5	;makes WR low
WAIT:	1008	JB P3.4, WAIT	;low to high pulse to WR for starting
		CLR P3.7	;polls until INTR=0
		CLR P3.6	conversion
		MOV A,P1	;ensures CS=0
	100A	CPL A	;high to low pulse to RD for reading
		MOV P0,A	the data from ADC
		SJMP MAIN	;moves the digital data to
		END	accumulator
			;complements the digital data
	100D		;outputs the data to P0 for the LEDs
			;Jump to program
	100F		;End

Jumper Details:-



RESULT:

Thus, the assembly language program for performing the interfacing of ADC with 8051 has been verified.

EXP NO:	INTERFACING OF DAC
DATE	

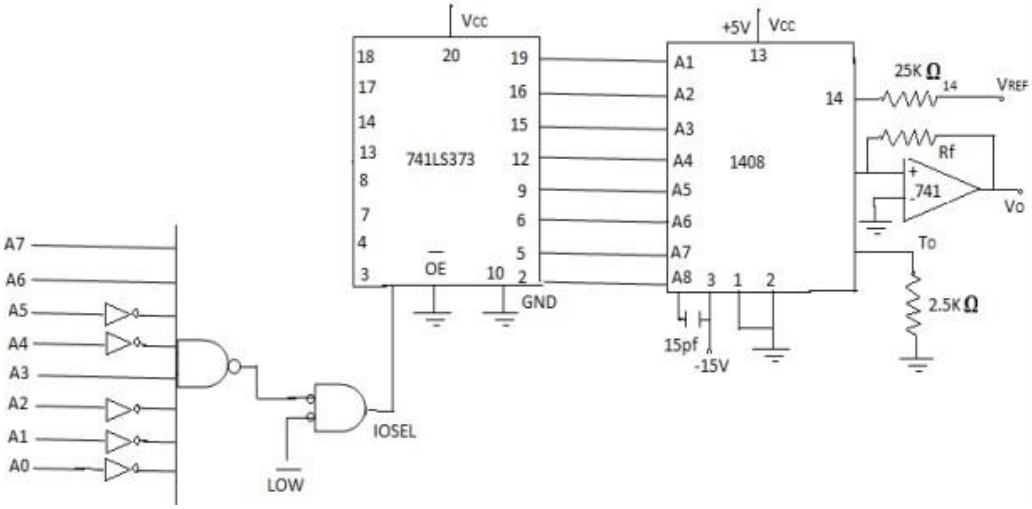
AIM:

To interface the DAC with 8051 microcontroller and generate the square wave, saw tooth wave and triangular wave.

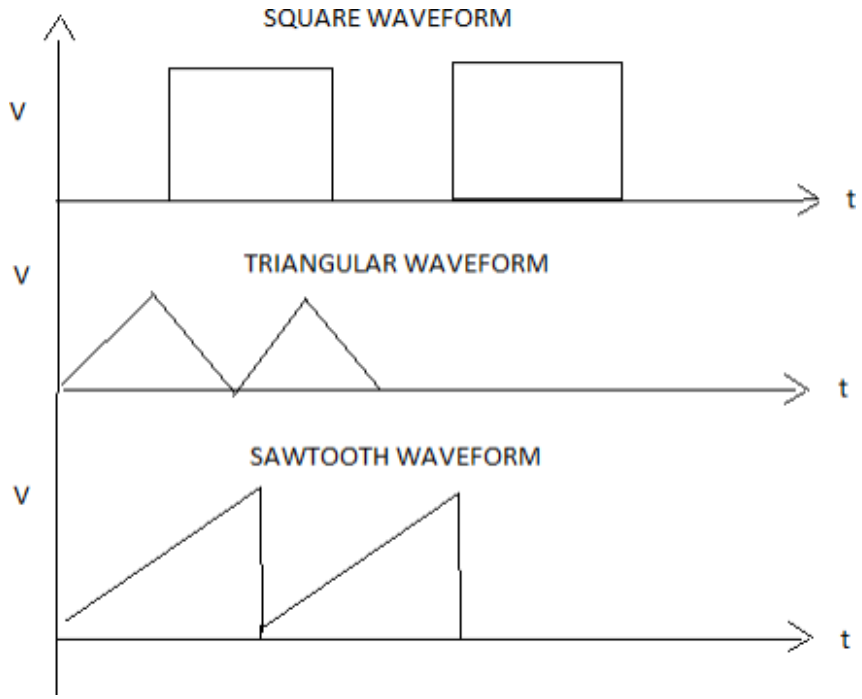
REQUIREMENTS:

S.No	Hardware & Software Requirements	Quantity
1	8051 Trainer Kit	1No
2	Power Chord	1No
3	DAC interfacing board	1 No
4	CRO	1 No

CIRCUIT DIAGRAM:



WAVEFORMS:



SQUARE WAVE

ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR, # E0C0	90,FF,C8	Move the immediate Data EOCO
4103	START	MOV A, # 00	74,00	Initialize the Accumulator To zero
4105		MOVX @ DPTR, A	F0	Long call the delay
4106		LCALL DELAY	12,41,12	Move the content of Accumulator to FF
4109		MOV A, # FF	74,FF	Long call delay
410B		MOVX @ DPTR, A	F0	Long jump to start
410C		LCALL DELAY	12,41,12	Move the 05 data To R register
410F		LJMP START	02,41,03	Decrement Jump NON zero
4112	DELAY	MOV R1, # 05	79,05	Return to main program
4114	LOOP	MOV R2, # FF	74,FF	Short jump to start
4116	HERE	DJNZ R2, HERE	DA,FE	
4118		DJNZ R1, LOOP	D9,FA	
411A		RET	22	
411C		SZMP START	80,E3	

OBSERVATION:

AMPLITUDE	TIME PERIOD

DIGITAL CLOCK

AIM:-

To display the digital clock specifically by displaying the hours, minutes and secondsusing 8051 kits

PROGRAM:

Address	Label field	Mnemonic field	Comments field
Main Program		ORG 8000H	Origin of the program from 8000H
8000		MOV DPTR,#2043H	; Load DPTR with Control port Address of 2nd 8255
8003		MOV A,#80	; (A) = 80H = control word for all ports as output port
8005		MOV R7,#16H	; (R7) = 16H = to display 16th year on years displays
8007		MOV DPTR,#2042H	; Load DPTR with port C Address of 2nd 8255
800A		MOV A,R7	; (A) = 16H
800B		MOVX @DPTR,A	; Display 16H at port C of 8255
800C		MOV R6,#00	; (R6) = 00
800E		ACALL Month	; Call month routine to display January
8010		MOV R5,#0	; (R5) = 0 = first Day on Days displays
8012		ACALL Day	; Call the routine Day to display day at port A
8014		CJNE R5,#1FH,Day1	; Compare R5 with 1FH (i.e. 31_{10}), if R5 ≠ 1FH then jump to Day1. No need of involving Accumulator in compare
8017		ACALL Month	; Call month routine to display February
8019		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
801B		ACALL Day	; Call the routine Day to display day at port A
801D		CJNE R5,#1CH,Day1	; Compare R5 with 1CH (i.e. 28_{10}), if R5 ≠ 1CH then jump to Day1. In February month there are 28 days
8020		ACALL Month	; Call month routine to display March
8022		MOV R5,#0	; (R5) = 0 = first Day on Days displays
8024		ACALL Day	; Call the routine Day to display day at port A
8026		CJNE R5,#1FH,Day1	; Compare R5 with 1FH (i.e. 31_{10}), if R5 ≠ 1FH then jump to Day1. No need of involving Accumulator in compare
8029		ACALL Month	; Call month routine to display April
802B	BACK:	MOV R5,#0	; (R5) = 0 = first Day on Days displays
802D		ACALL Day	; Call the routine Day to display day at port A
802F		CJNE R5,#1EH,Day1	; Compare R5 with 30_{10}, if R5≠1EH then jump to Day
8032		ACALL Month	; Call month routine to display May / July
8034		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
8036		ACALL Day	; Call the routine Day to display day at port A
8038		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10}, if R5 ≠ 1FH then jump to Day1.
803B		ACALL Month	; Call month routine to display June / August
803D		CJNE R6,#08,BACK	; Compare R6 with 07 (July), if R6≠07 then jump toBACK.
8040		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
8042		ACALL Day	; Call the routine Day to display day at port A
8044		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10}, if R5 ≠ 1FH then jump to Day1.
8047		ACALL Month	; Call month routine to display September
8049	BACK1:	MOV R5,#0	; (R5) = 0 = first Day on Days displays
804B		ACALL Day	; Call the routine Day to display day at port A
804D		CJNE R5,#1EH,Day1	; Compare R5 with 1EH (i.e. 30_{10}), if R5 ≠ 1EH then jump to Day1. No need of involving Accumulator in compare
8050		ACALL Month	; Call month routine to display October / December
8052		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
8054		ACALL Day	; Call the routine Day to display day at port A
8056		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10}, if R5 ≠ 1FH then jump to Day1.
8059		ACALL Month	; Call month routine to display November
805B		CJNE R6,#12,BACK1	; Compare R6 with 12H, if R6≠12H then jump to BACK1.
805E		INC R7	; Increment the year
805F		SJMP START	; Repeat the process again

Observation:

Input

1200	00
1201	00
1202	00
1203	00
1204	00

Output:

Time is displayed in the RTC board as

	! Hour ׀·	Minutes		׀· seconds	׀·
X	0	0	0	5	9

X	0	0	1	0	0

RESULT:

Thus the digital clock program has been written and executed using 8086 microprocessorkit and the output of digital clock was displayed as [hours: minutes: seconds] successfully.

INTRODUCTION TO KEIL µ VISION 4 SOFTWARE

The µ Visio n4 IDE is a Windows-based software development platform that combines a robust editor, project manager, and makes facility. µVisi on4 integrates all tools including the C compiler, macro assembler, linker/locator, and HEX file generator. µVisio n4 helps expedite the development process of your embedded applications by providing the following:

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

The µ Vision4 IDE offers numerous features and advantages that help you quickly and successfully develop embedded applications. They are easy to use and are guaranteed to help you achieve your design goals.

The installation steps for keil software are given below:

1. Double click on Keil µvisio n4.exe file.
2. Then click on **Next**.
3. Tick the check box towards to license agreements and click **Next**.
4. Select Destination folder and click **Next**.
5. Fill the necessary text boxes and click **Next**.
6. Finally click on **Finish**.

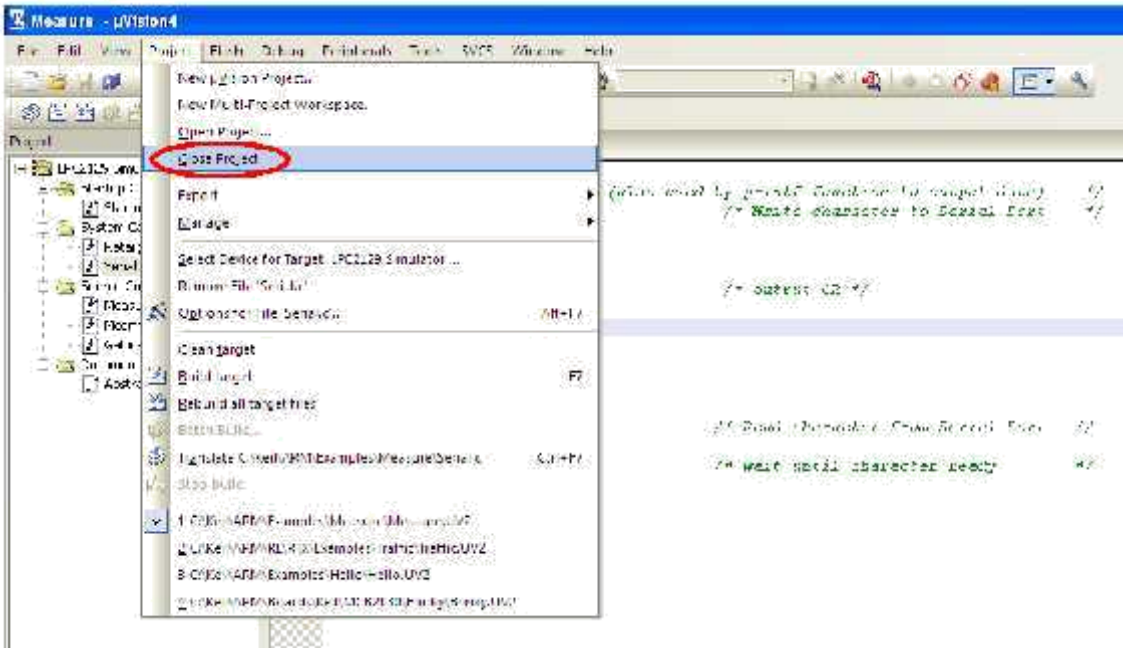
Software Flow

First open the icon keil µvis ion4 and the follow the steps are given below. The menu bar provides you with menus for editor operations, project maintenance, development tool option settings, program debugging, external tool control, window selection and manipulation, and on-line help. The toolbar buttons allow you to rapidly execute µ Vision4 commands. A Status Bar provides editor and debugger information. The various toolbars and the status bar can be enabled or disabled from the View Menu commands.

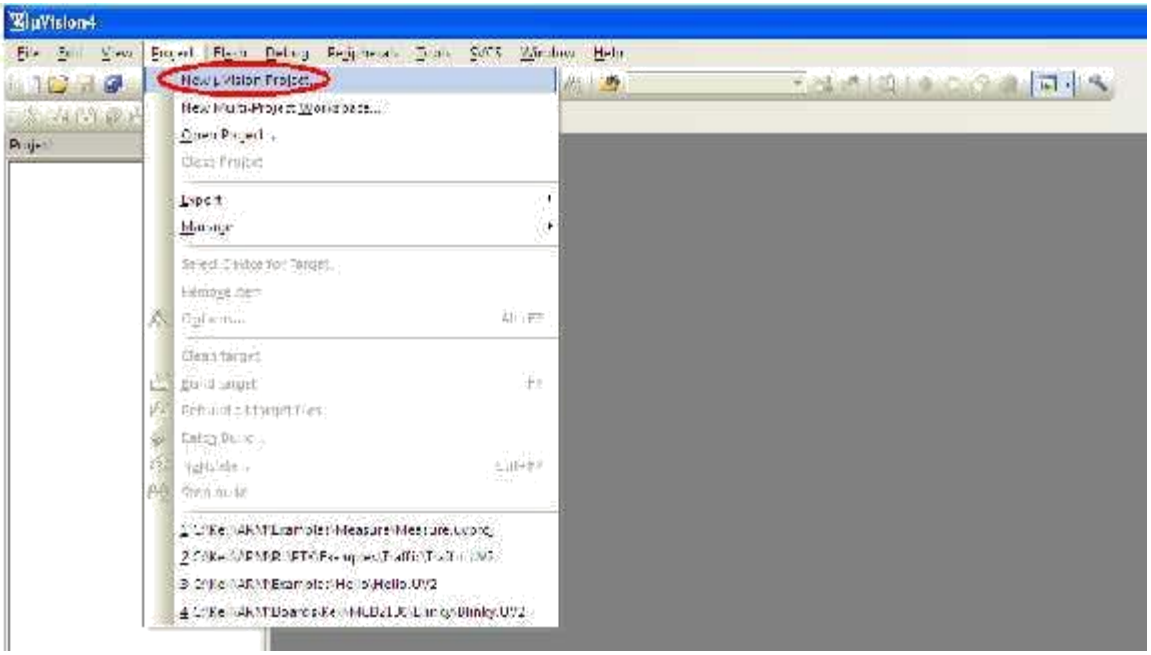
Creating a New Project

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

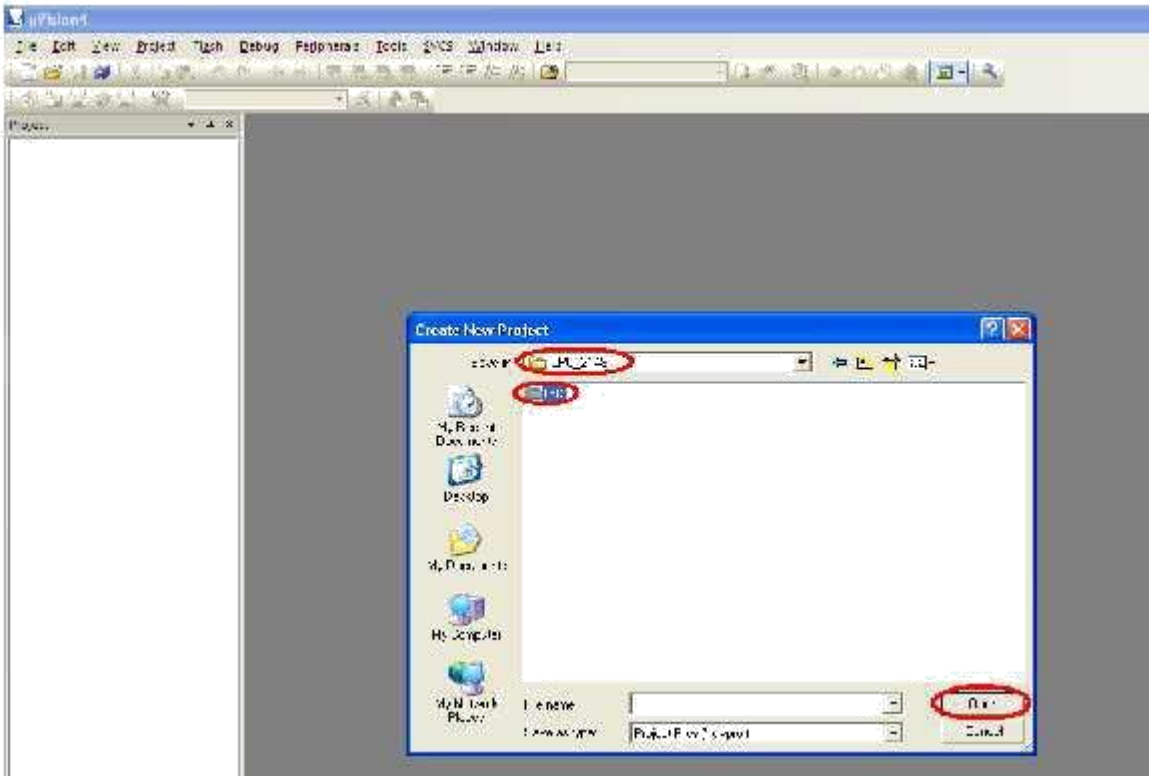
STEP 1: Go to "**Project**" and close the current project "**Close Project**".



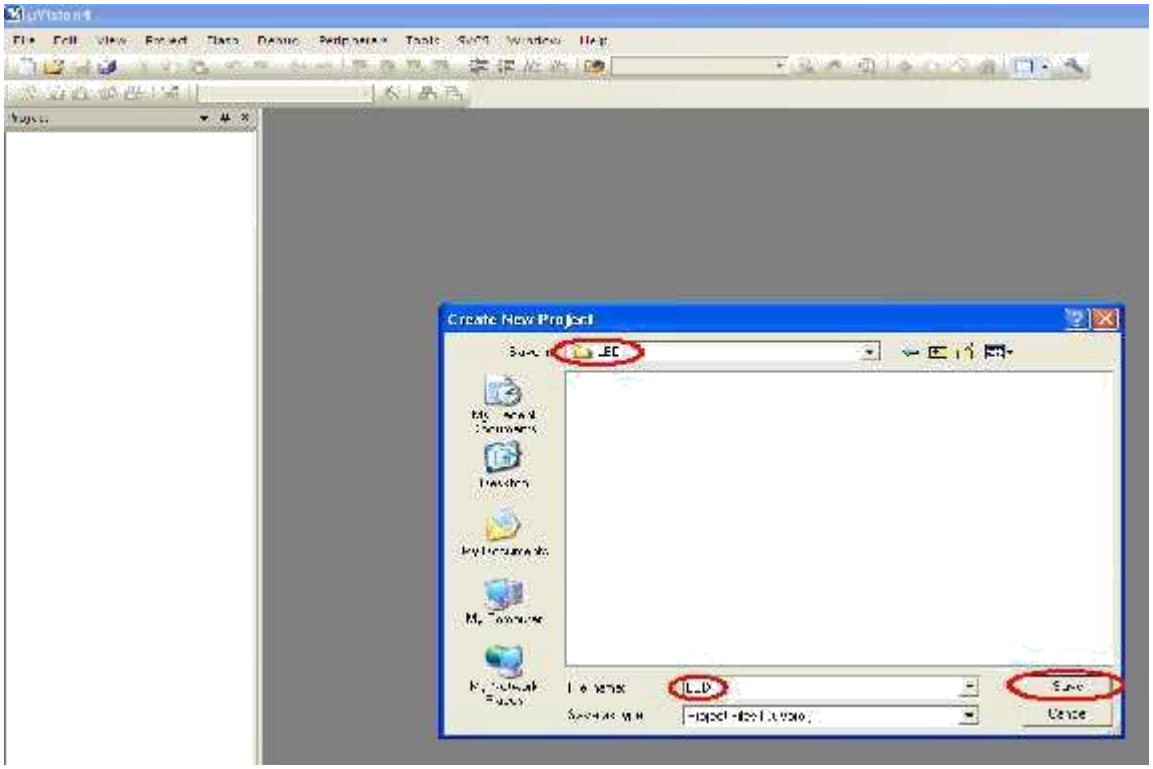
STEP 2: Go to the "**Project**" and click on "**New µ vision Project**"



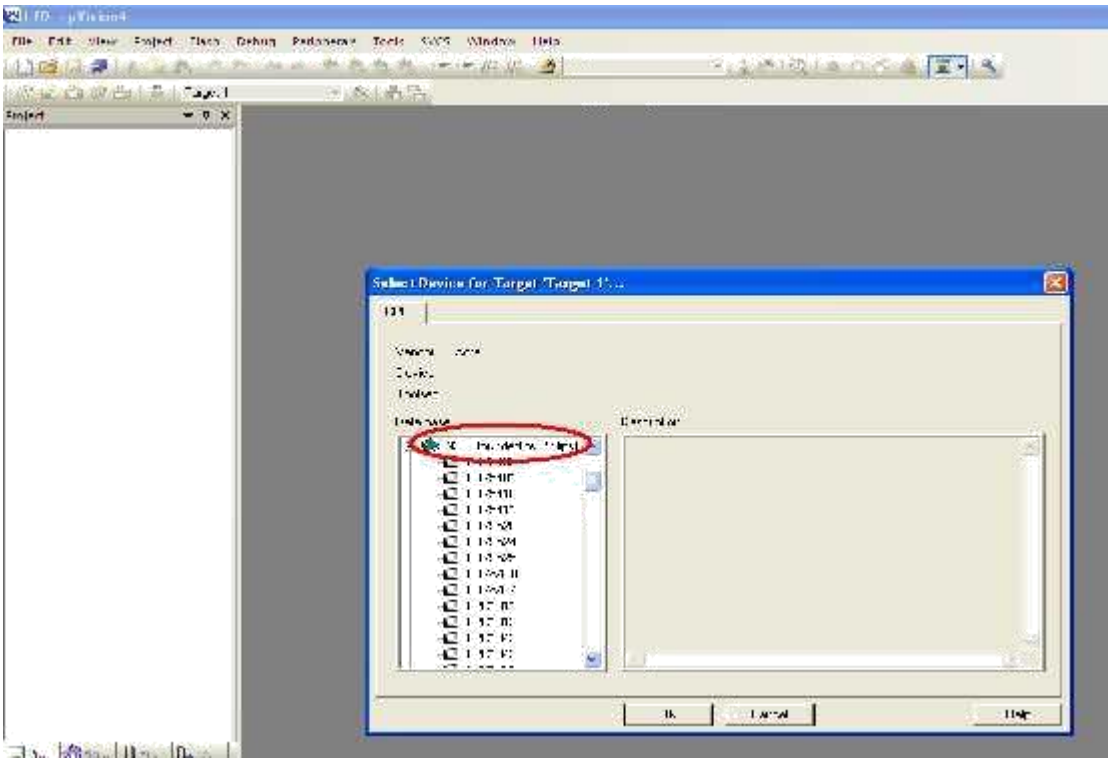
STEP 3: A small window will pop up with name "**Create New Project**" and can be create and select destination path.



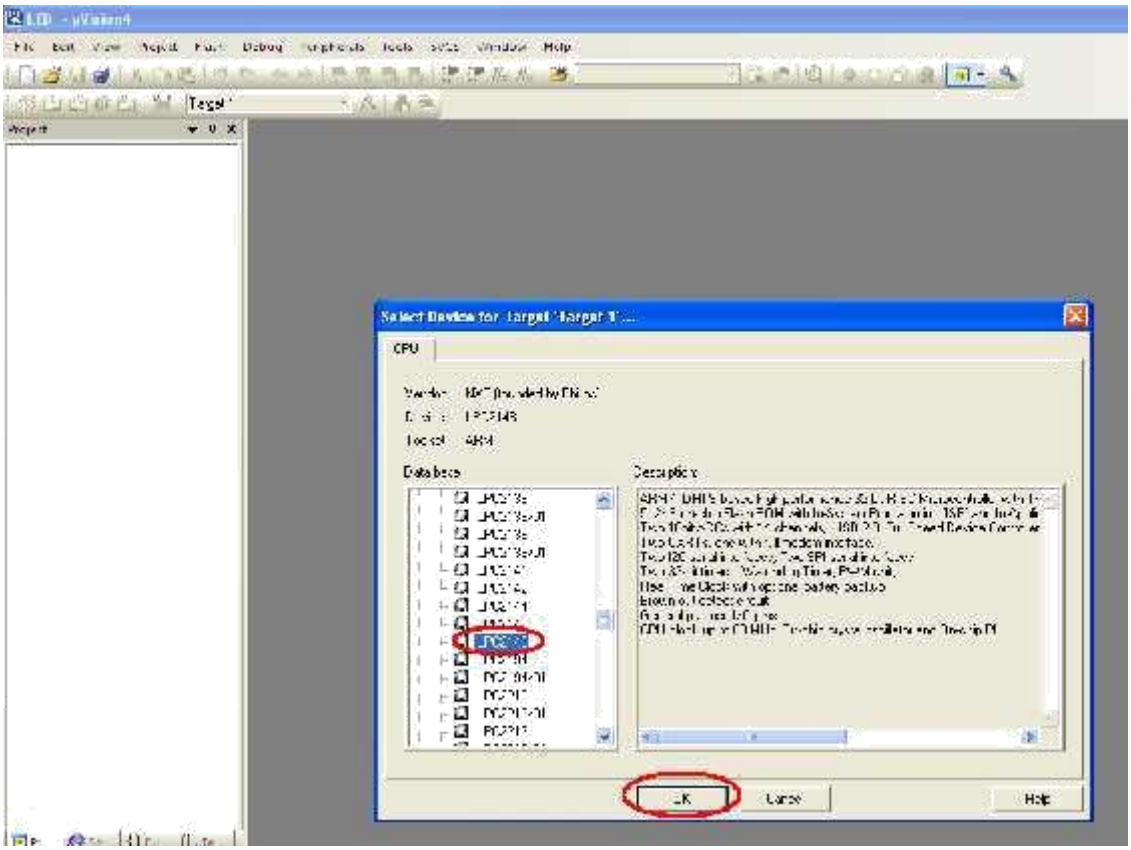
STEP 4: Create a folder and give a proper name that can be related to the Project.



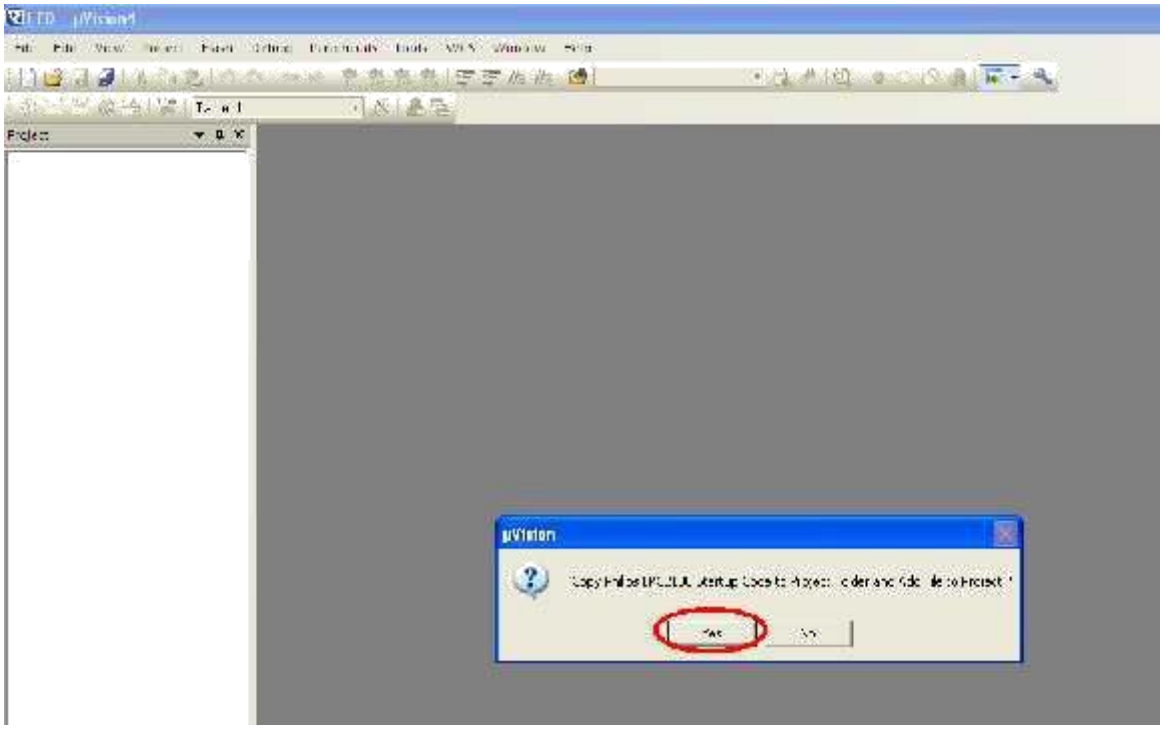
STEP 5: A small window will pop up with name "**Select Device for Target 'Target 1'**", and select the data base NXP founded by Philips.



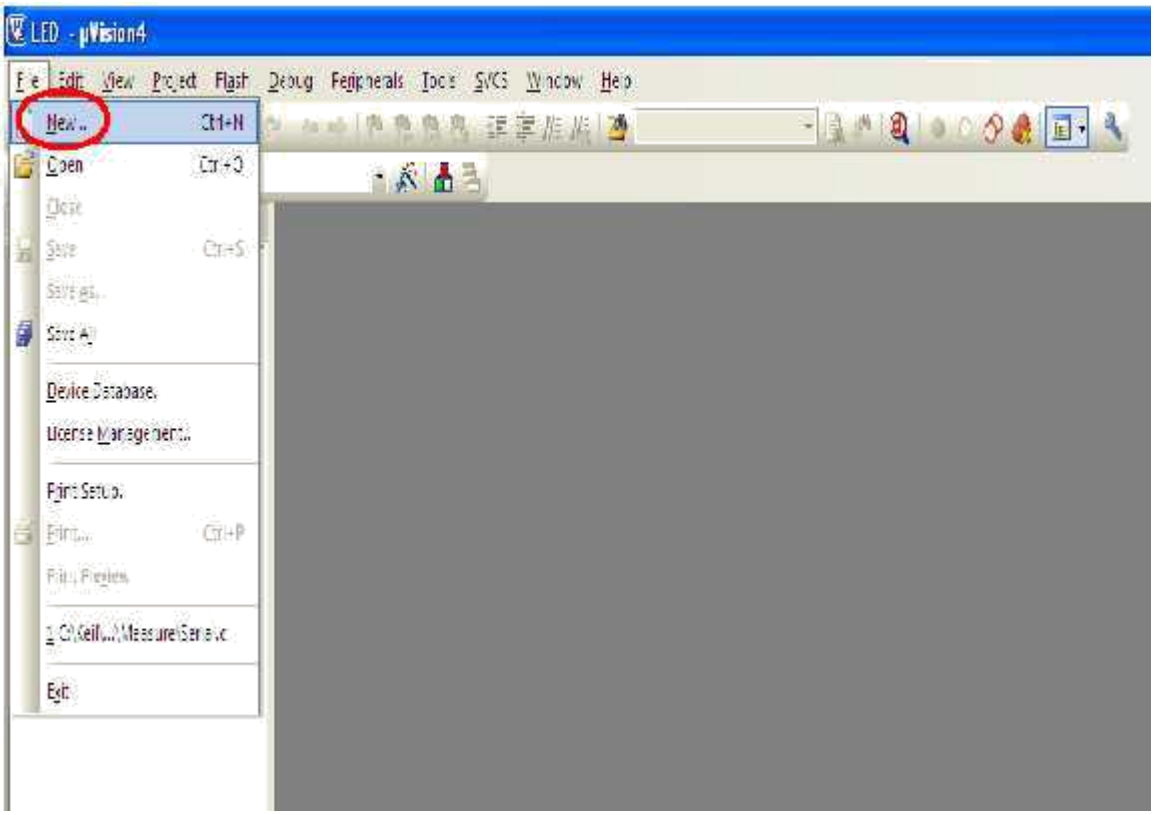
STEP 6: Within the NXP founded by Philips select **LPC2148.**



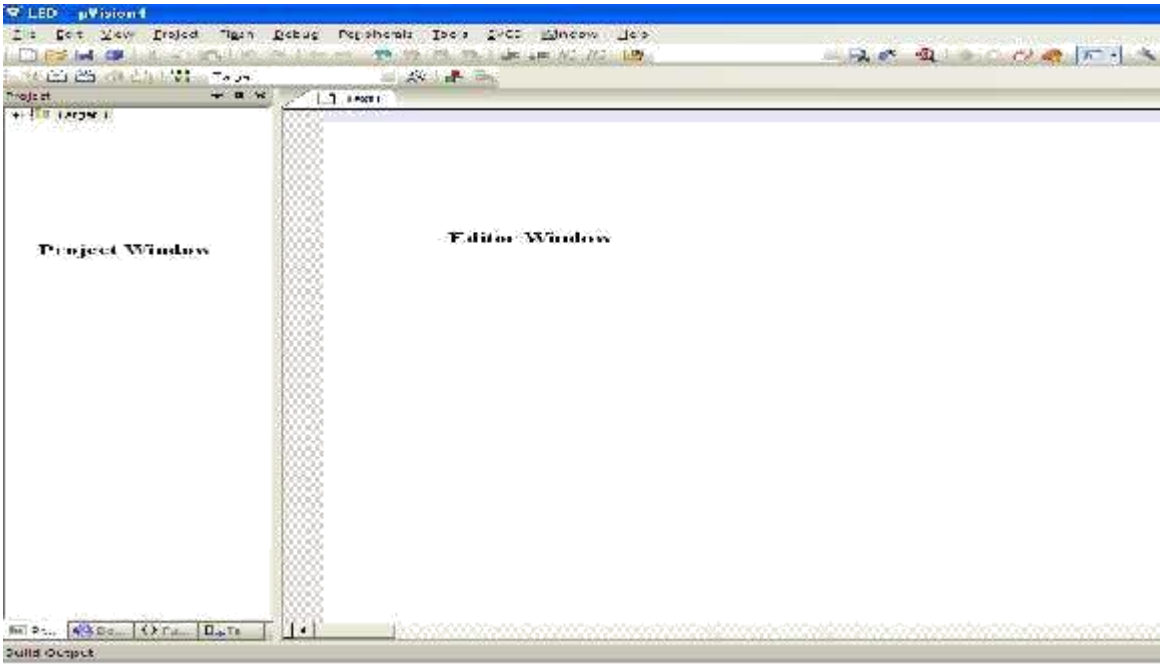
STEP 7: Add Startup file to the project by clicking "**Yes**".



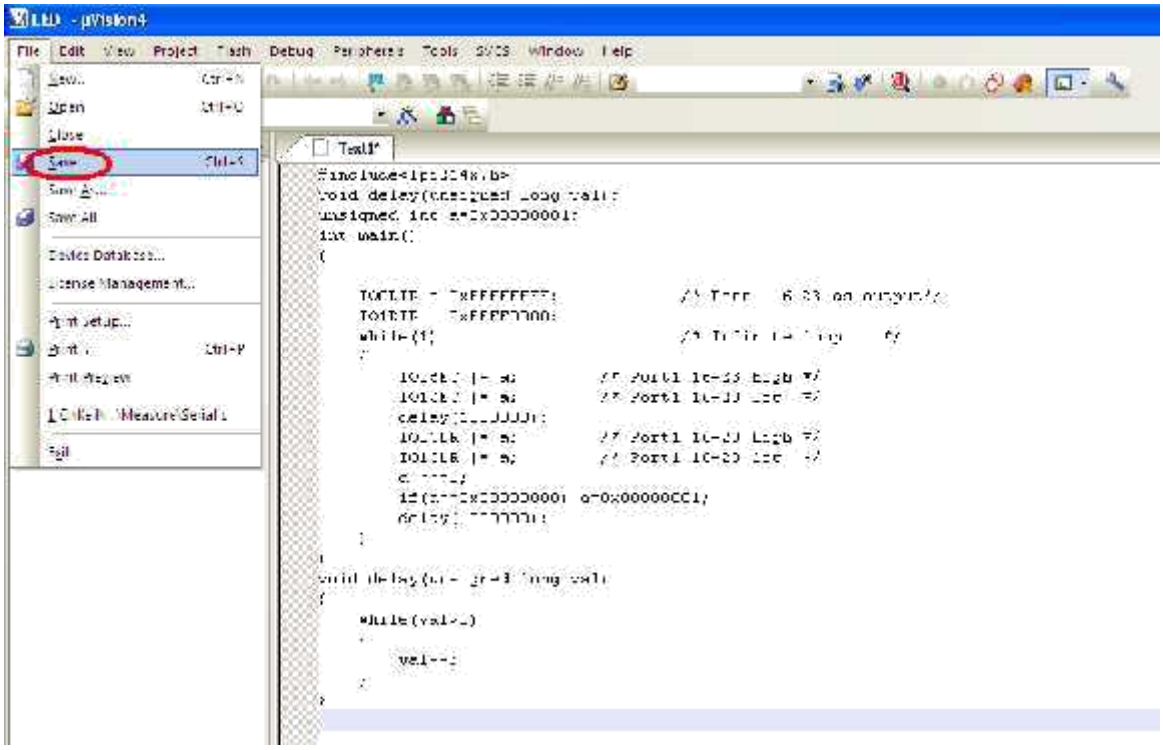
STEP 8: Next go to "**File**" and click "**New**".



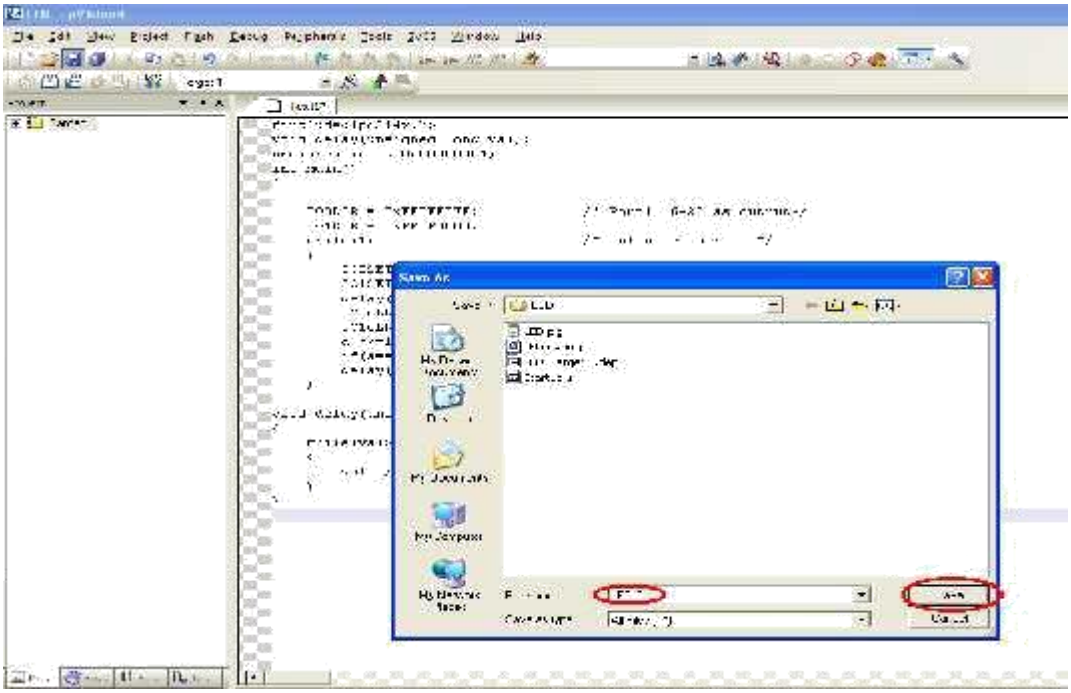
STEP 9: There are the main three windows are available in the keil IDE. First one is Project Workspace, second one is Editor Window and third one is Output Window.



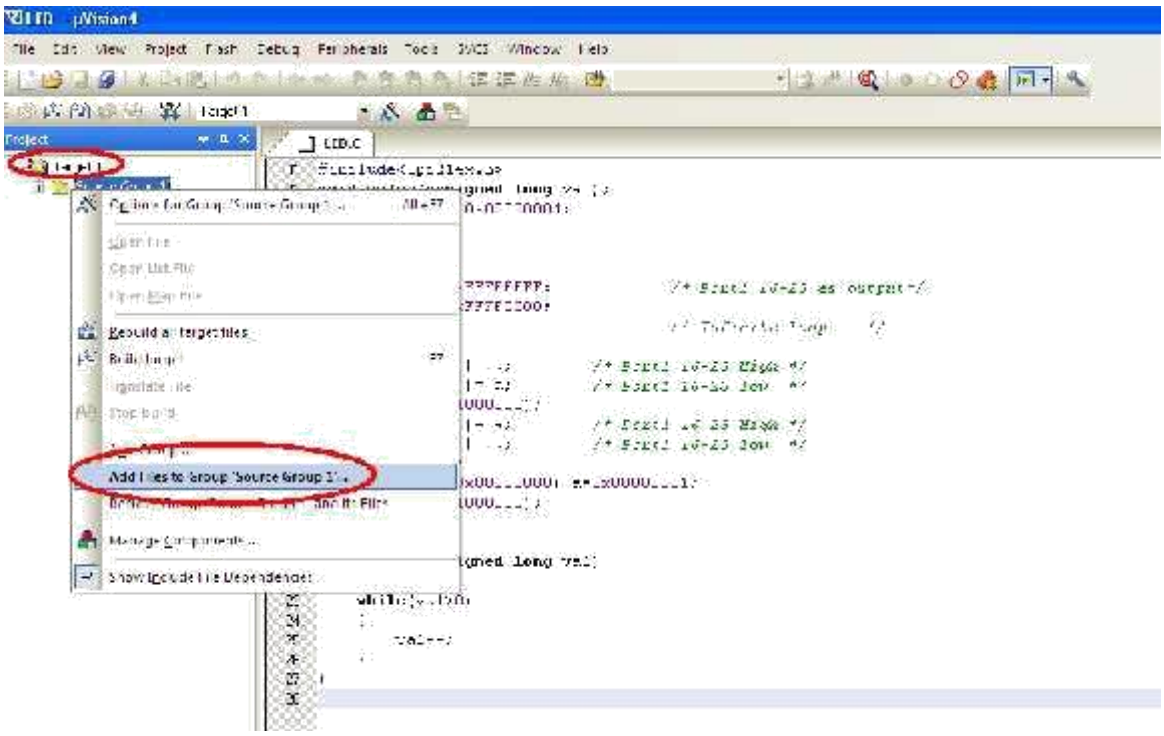
STEP 10: Can be start to write *.asm/c code on the editor window.



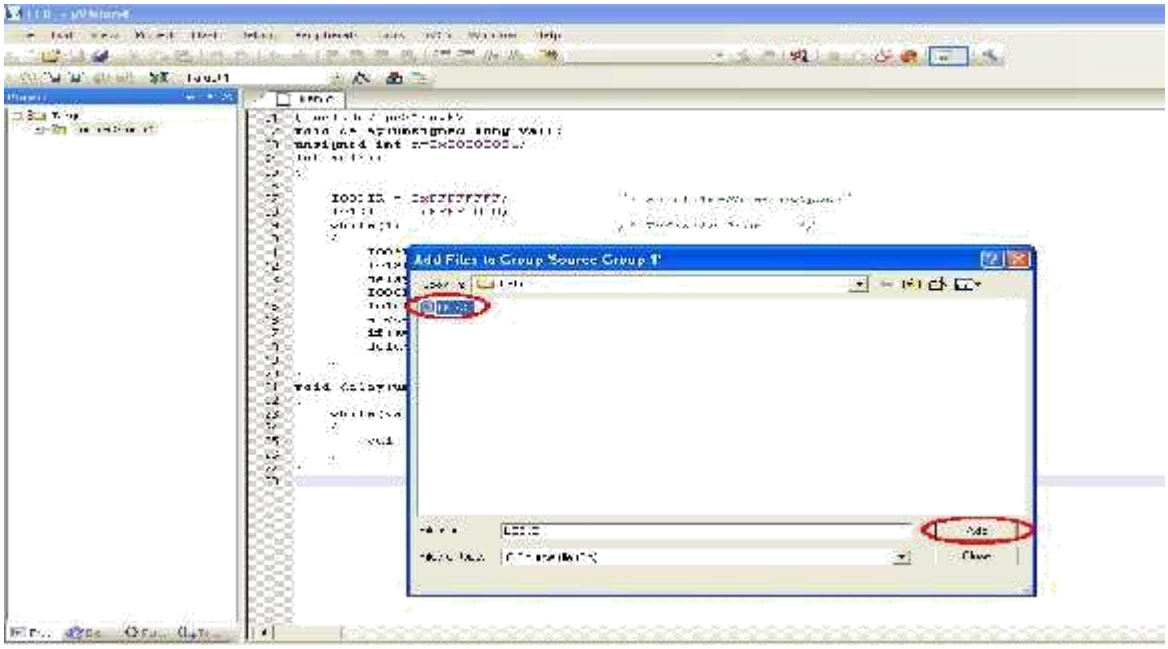
STEP 11: Can be save the file, if the program is in "**C**" save as "**filename.C**" or else save as "**filename.ASM**".



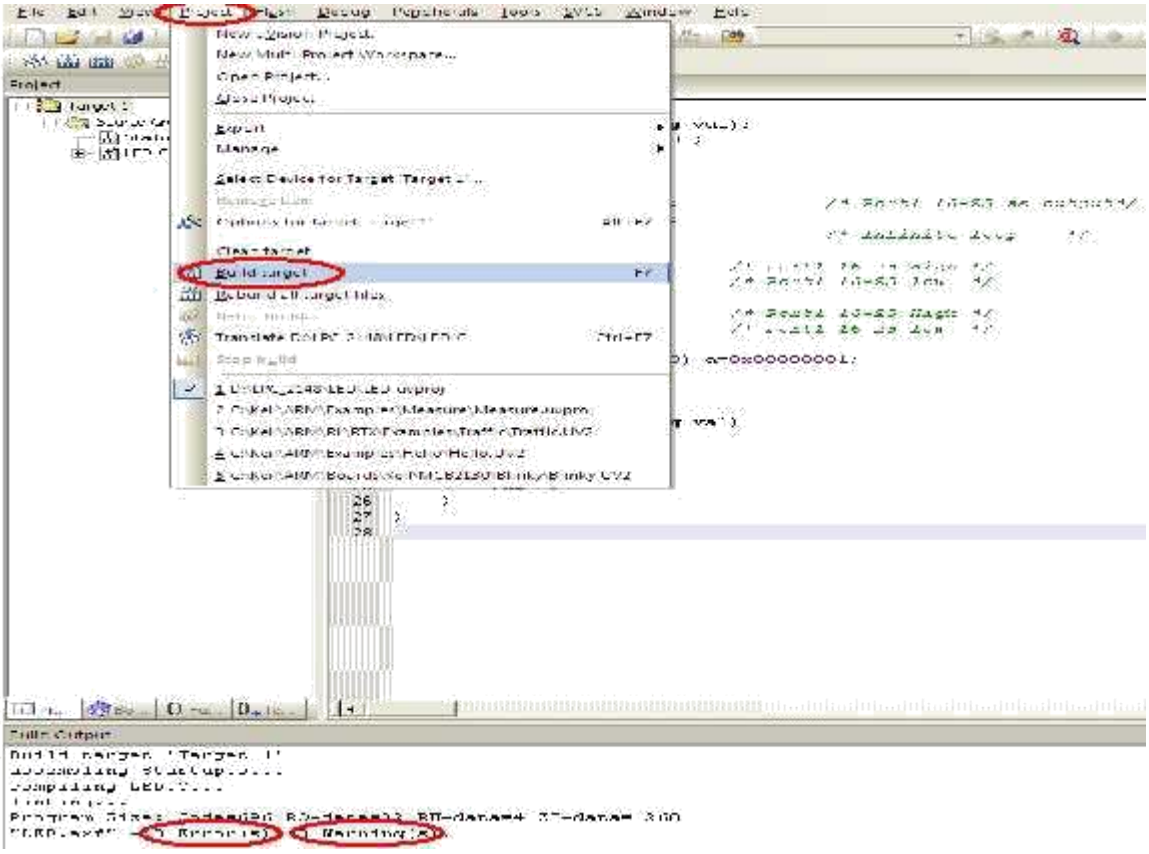
STEP 12: Add this source file to Group1, Go to the "**Project Workspace**" drag the +mark "**Target 1**" in that right click on "**Source Group1**" and click on "**Add Files to Group "Source Group1"**".



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

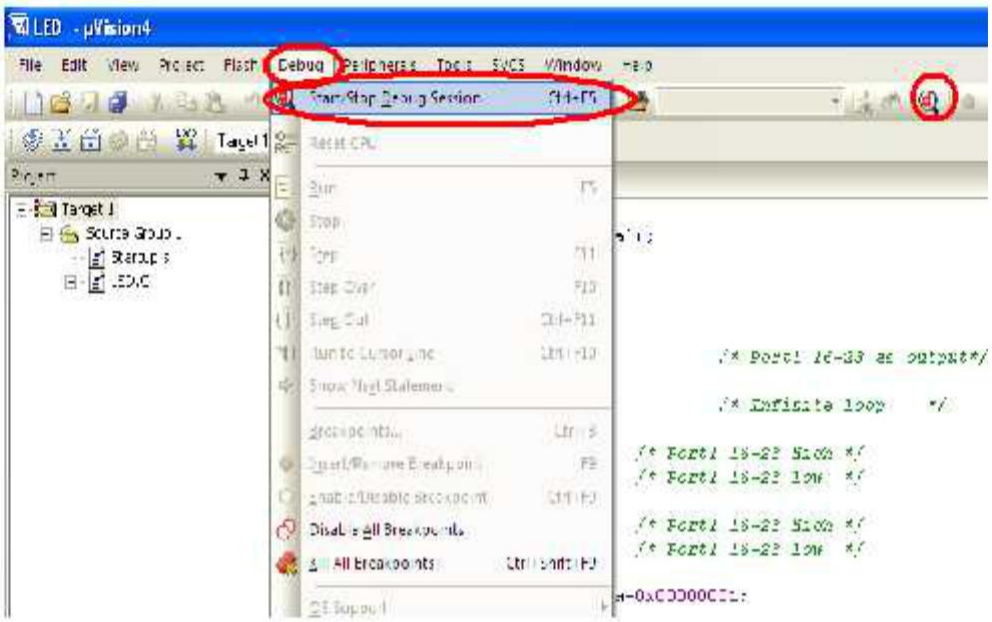


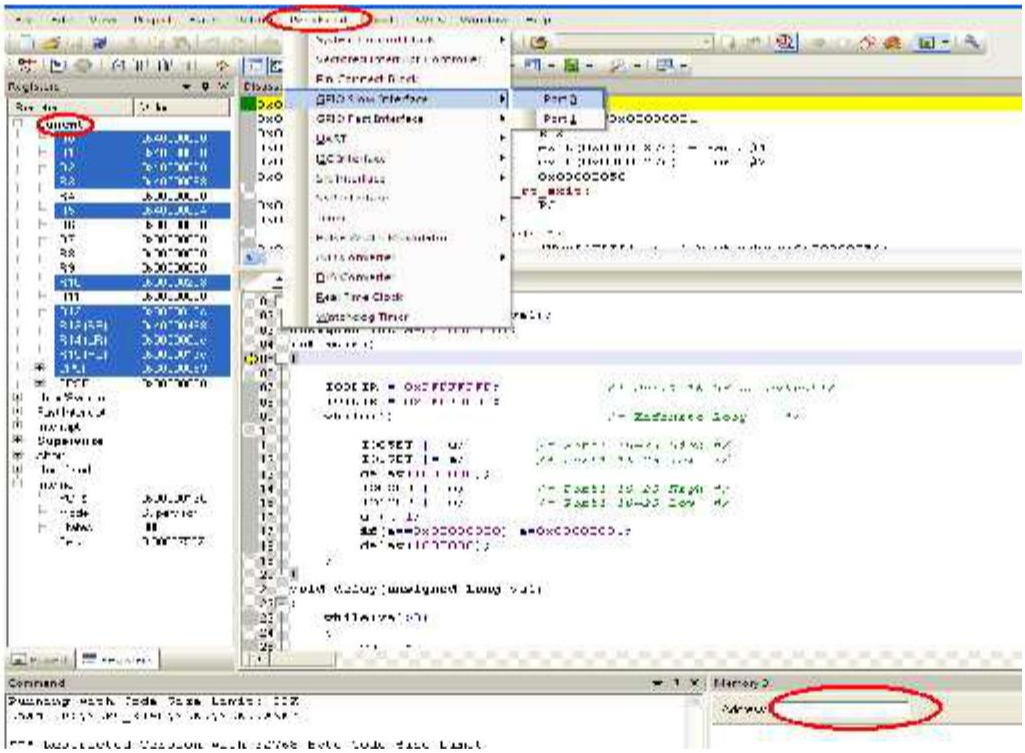
STEP 14: Then go to "**Project**" click on "**Build Target**" or **F7**. Output window will display related error and warning messages.



Simulation
Part:

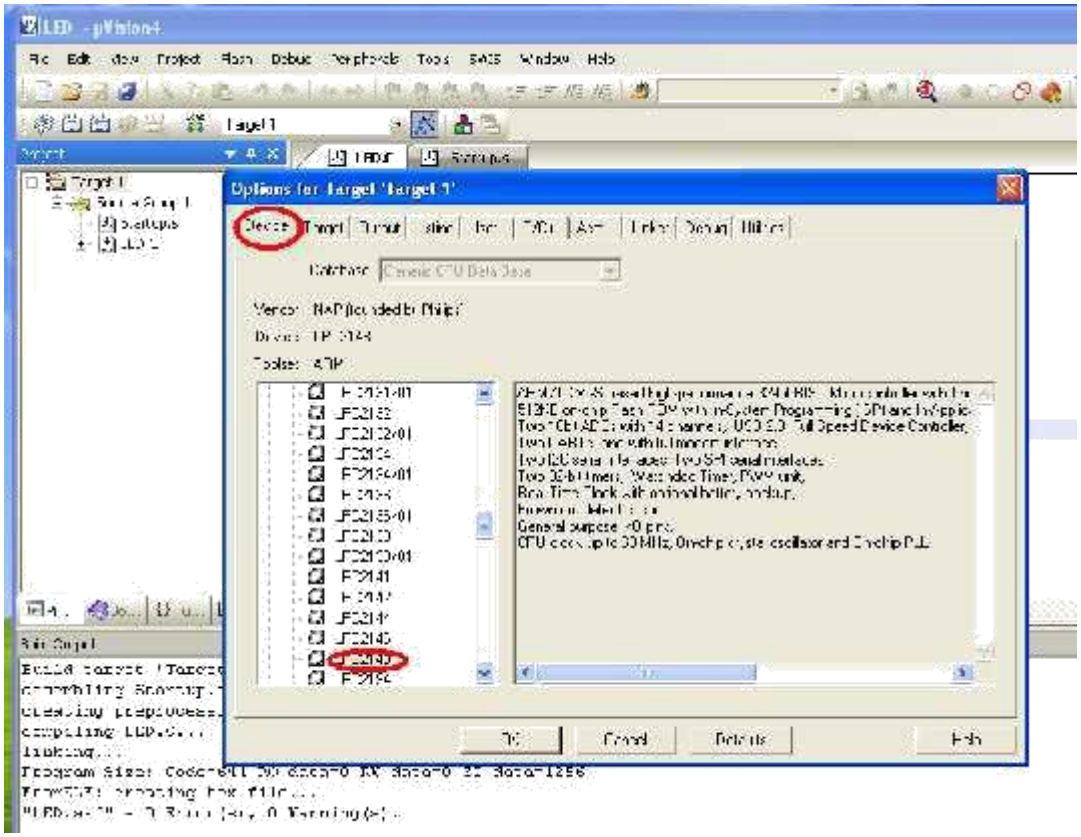
STEP 15: Go to "**Project**" menu, click on "**Rebuild all target Files**" and start **Debug**. From **View** menu can get **Memory Window** and from **Peripherals** can get I/O ports, Serial etc. For access internal memory type **i:0x_memory** location example: **i:0x30** and for external and program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



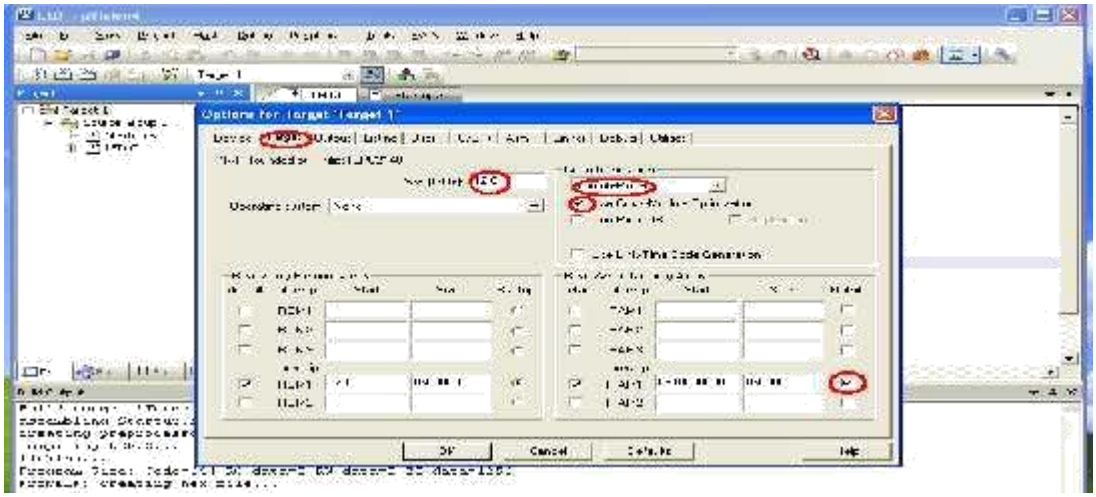


HEX file creation:

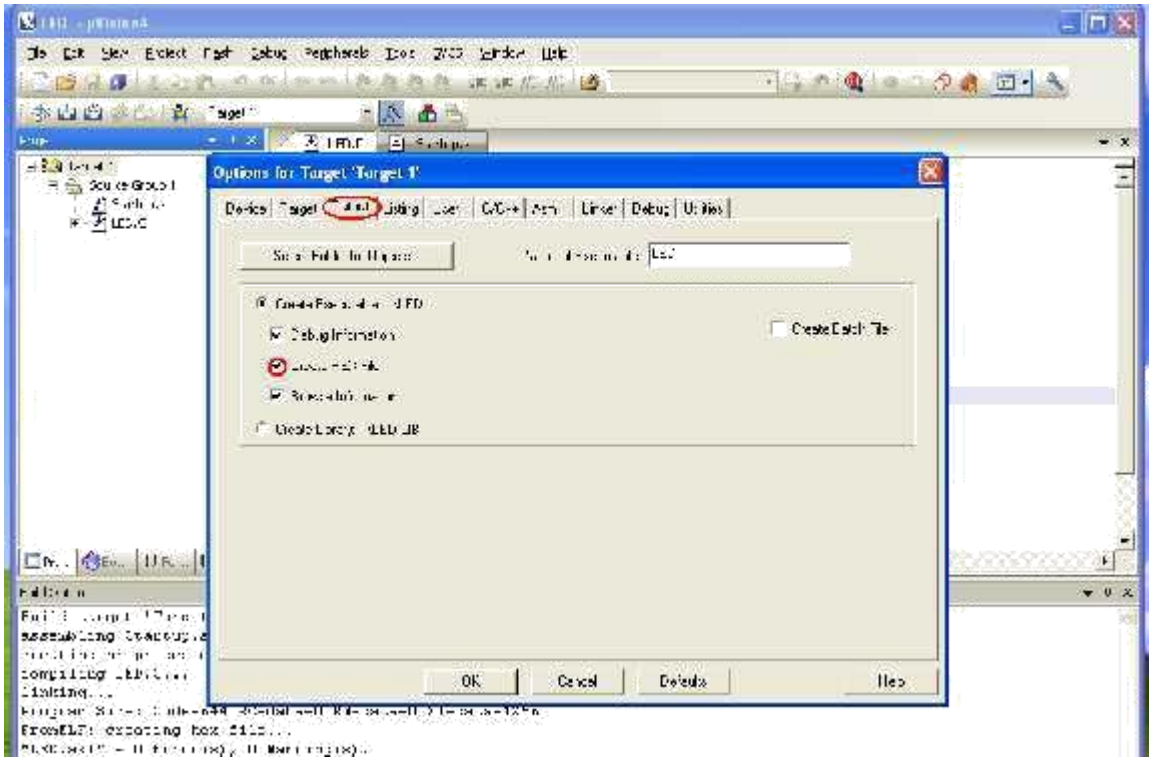
STEP 16: Follow the **STEP** up to **14**, then go to "**Project**" and click on "**Option for Group 'Source Group1**'". There a small window will open with name "**Option for Target 'Target1'**". In that window, go to first menu "Device", can be select LPC2148.



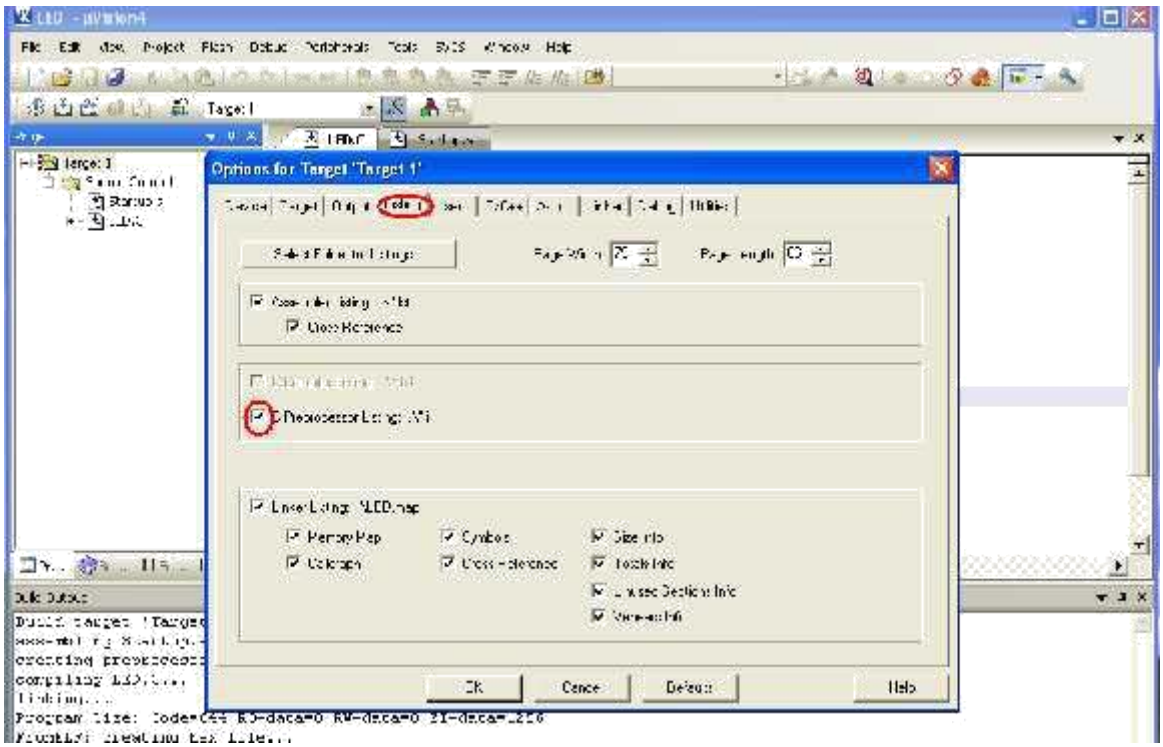
STEP 17: Next go to **Target** menu, set clock frequency as 12.0 MHz and select Thumb mode in the code generation selection box.



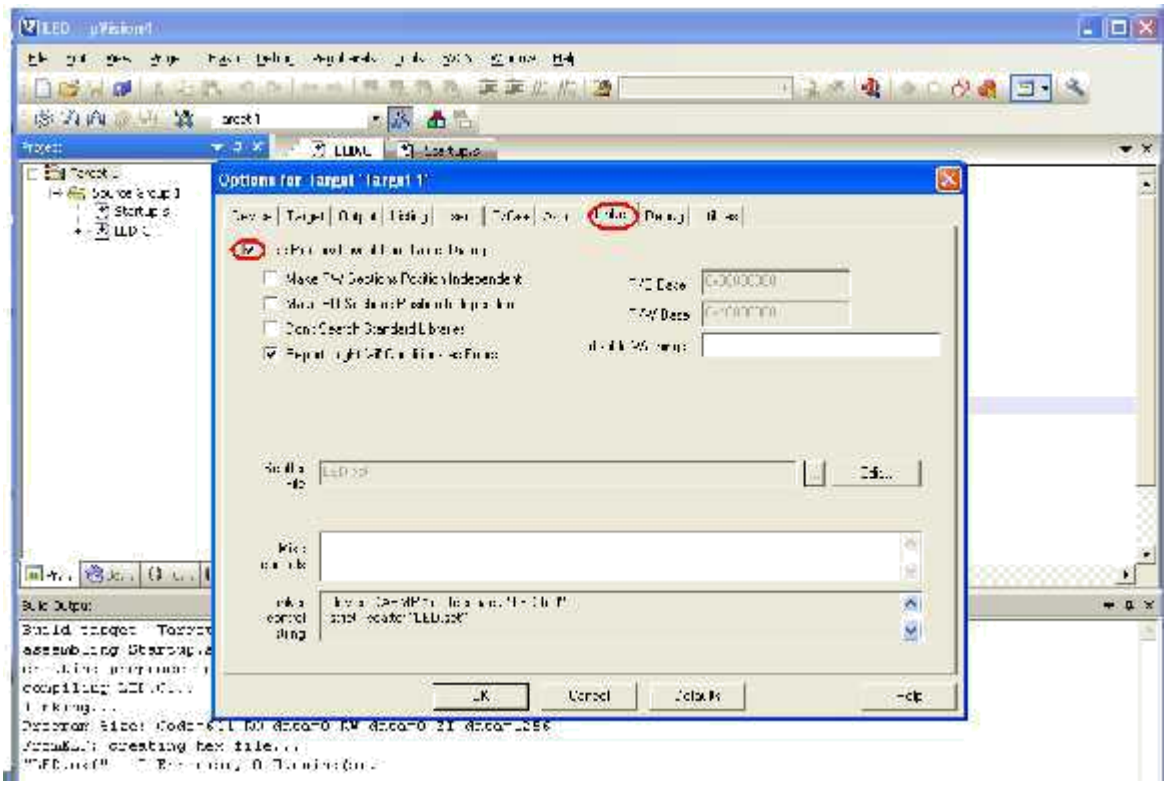
STEP 18: Then go to Output menu and click on create HEX file.



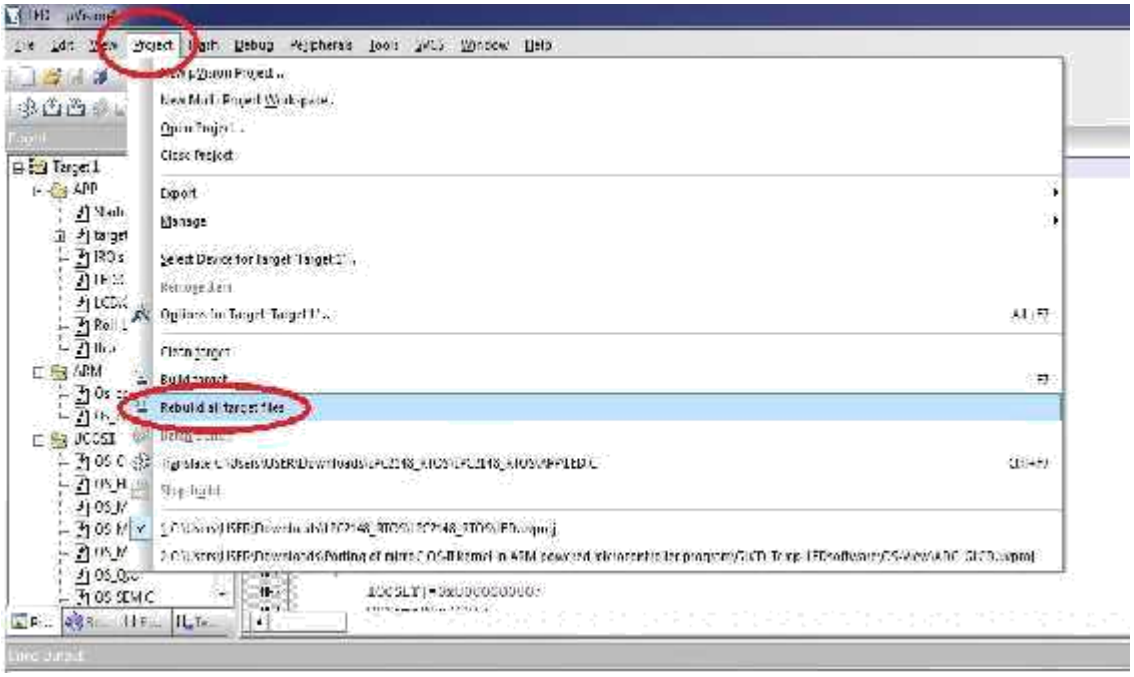
STEP 19: Then go to **Listing** menu and select C preprocessor Listing.



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



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



FLASH MAGIC

Introduction:

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

These features include:

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

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

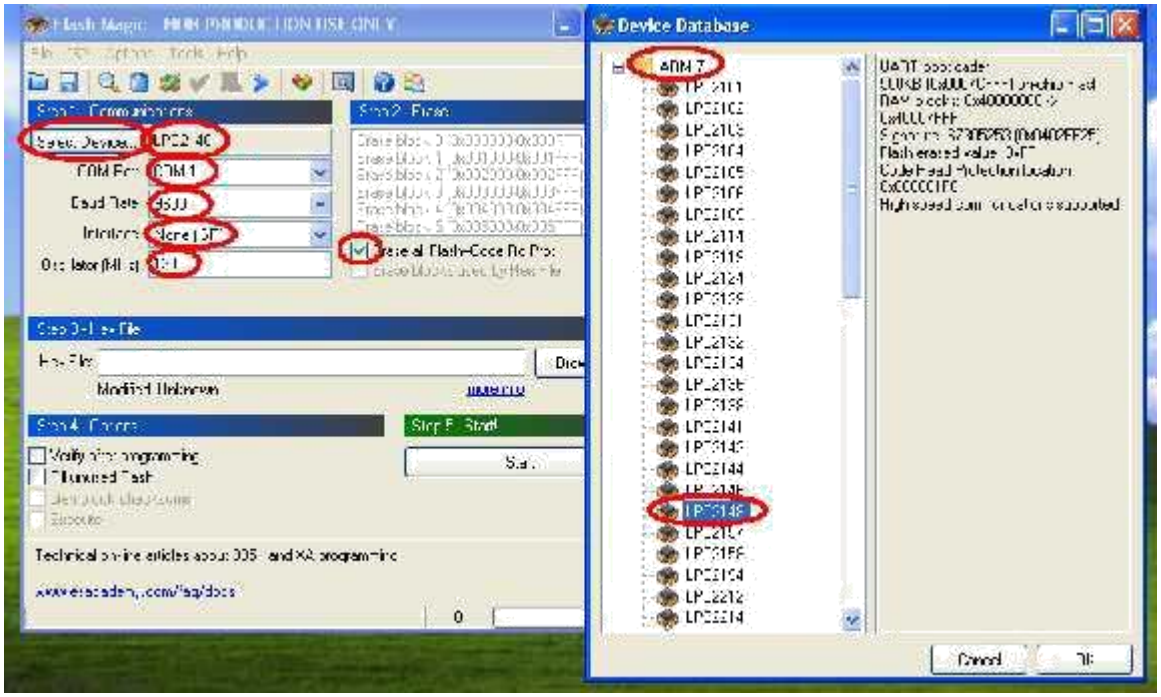
Installation Procedure for Flash Magic software:

Installation steps are given below:

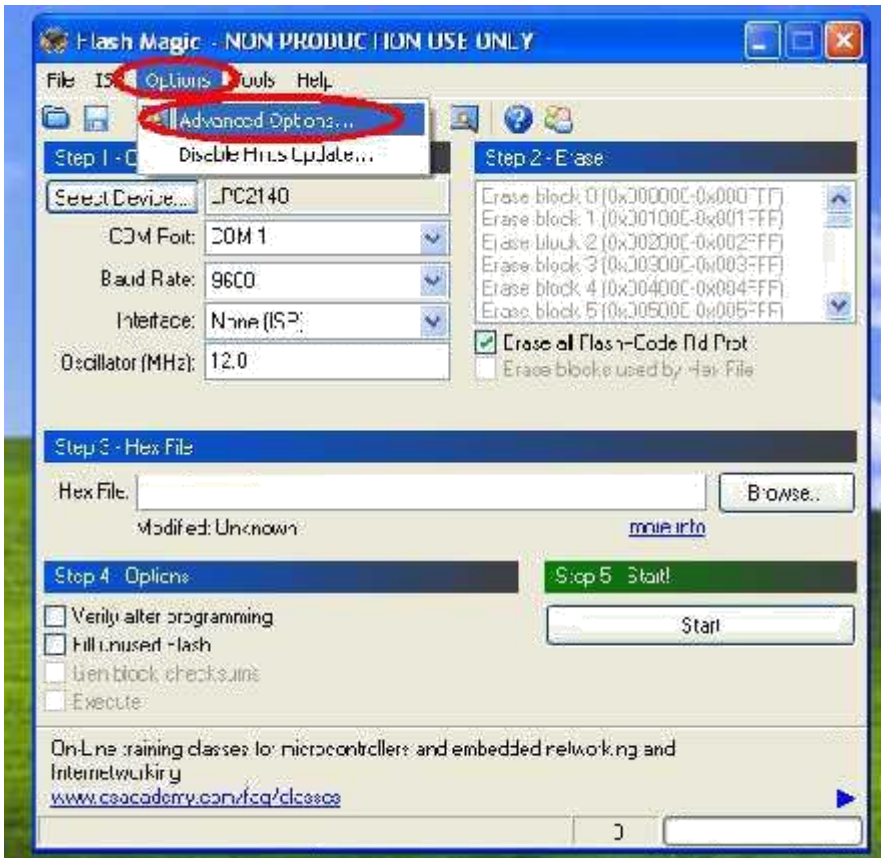
1. Double click on Flash Magic.exe.
2. Then click on Next.
3. Next accept the agreement and click on Next.
4. Select the destination folders and click on Next then Install.
5. Finally click on Finish.

Programming with communication port (COM1):

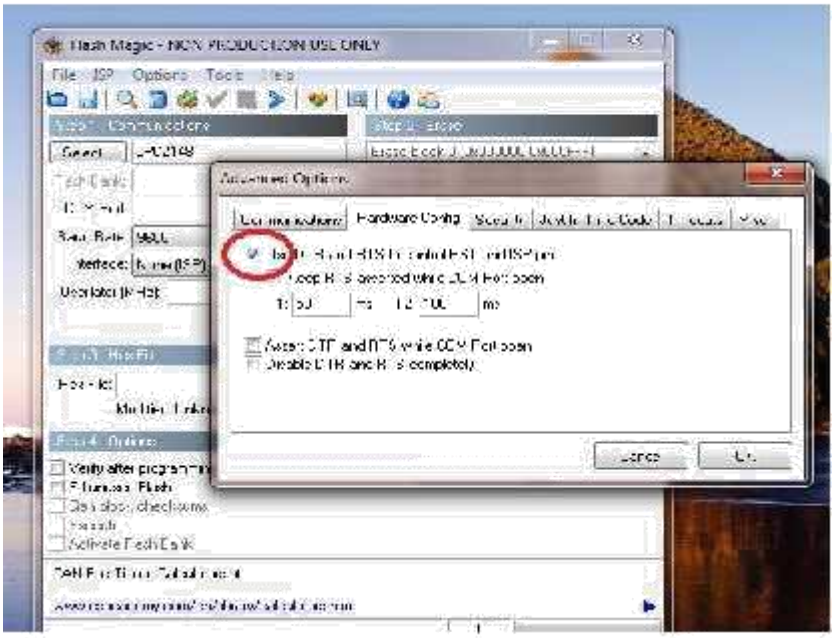
STEP 22: For programming with communication port, first select the device LPC2148 in ARM 7 category, COM port will be COM 1, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.



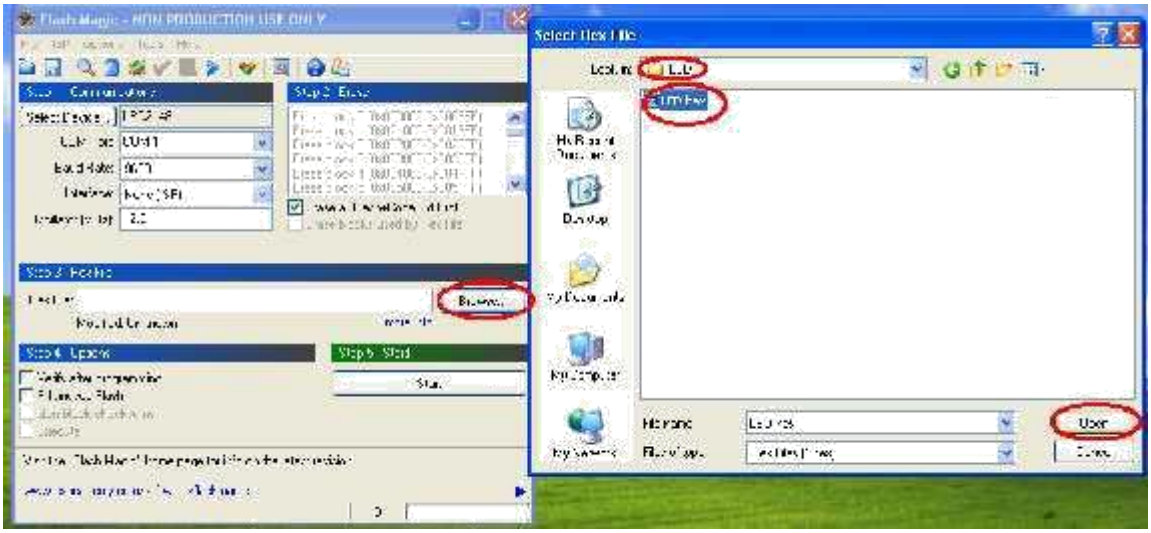
STEP 23: Under the menu **Options,** go to Advanced options.



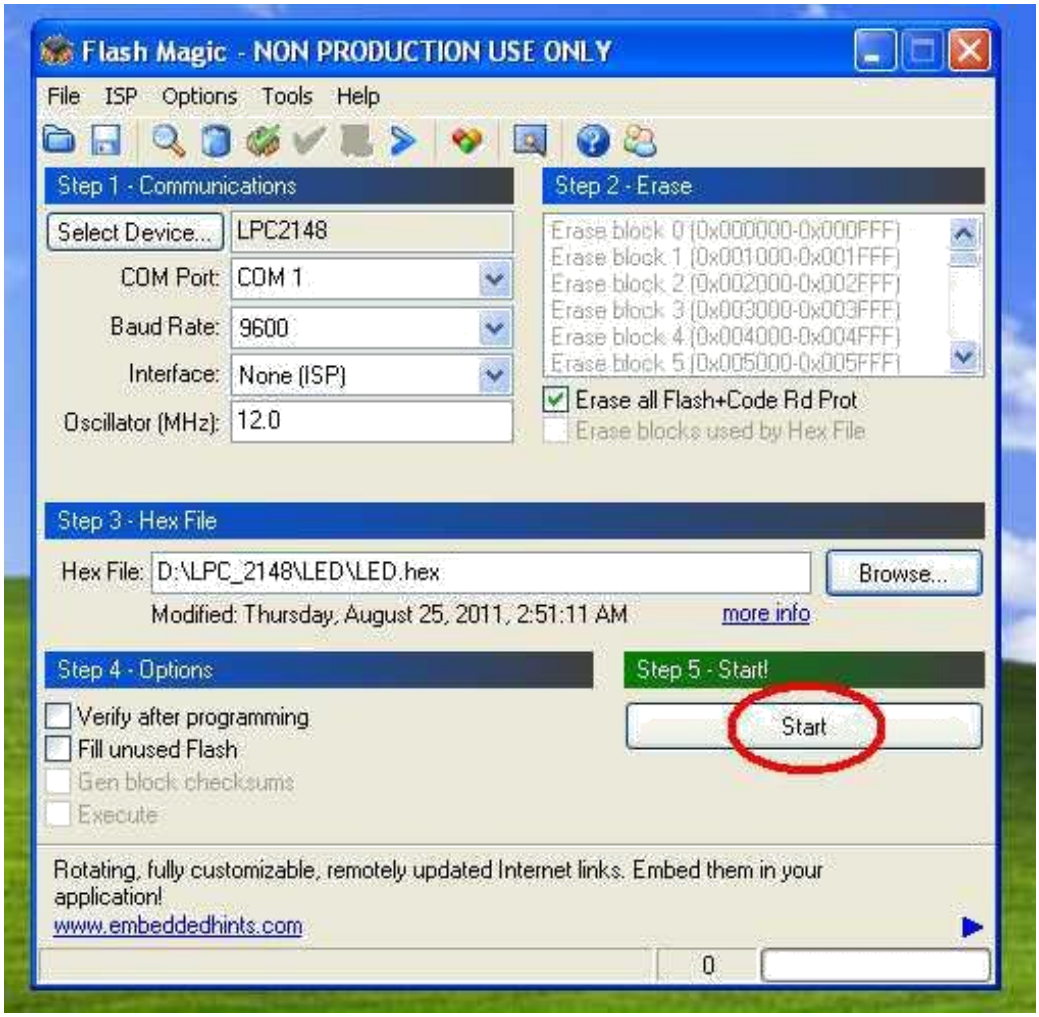
STEP 24: Under the menu Advanced options, go to Hardware configuration, click on the Use DTR and RTS to control RST and ISP pin.



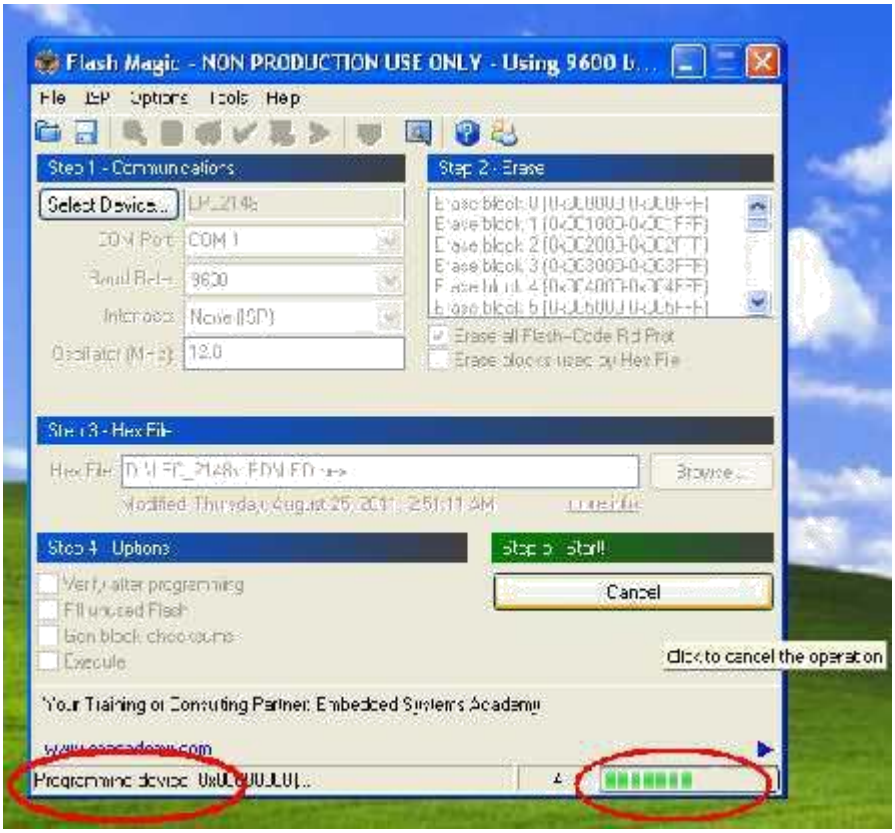
STEP 25: Next browse the path of hex file and select the file.



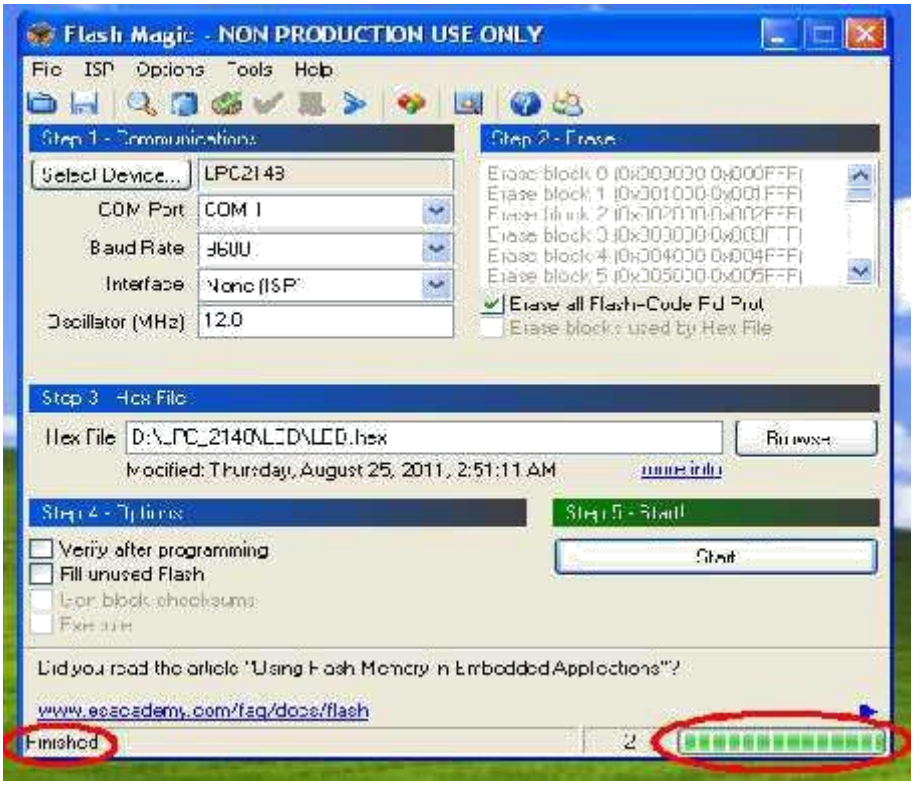
STEP 26: After selecting ISP mode on the Hardware Kit and click on start.



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



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



Programming through USB:

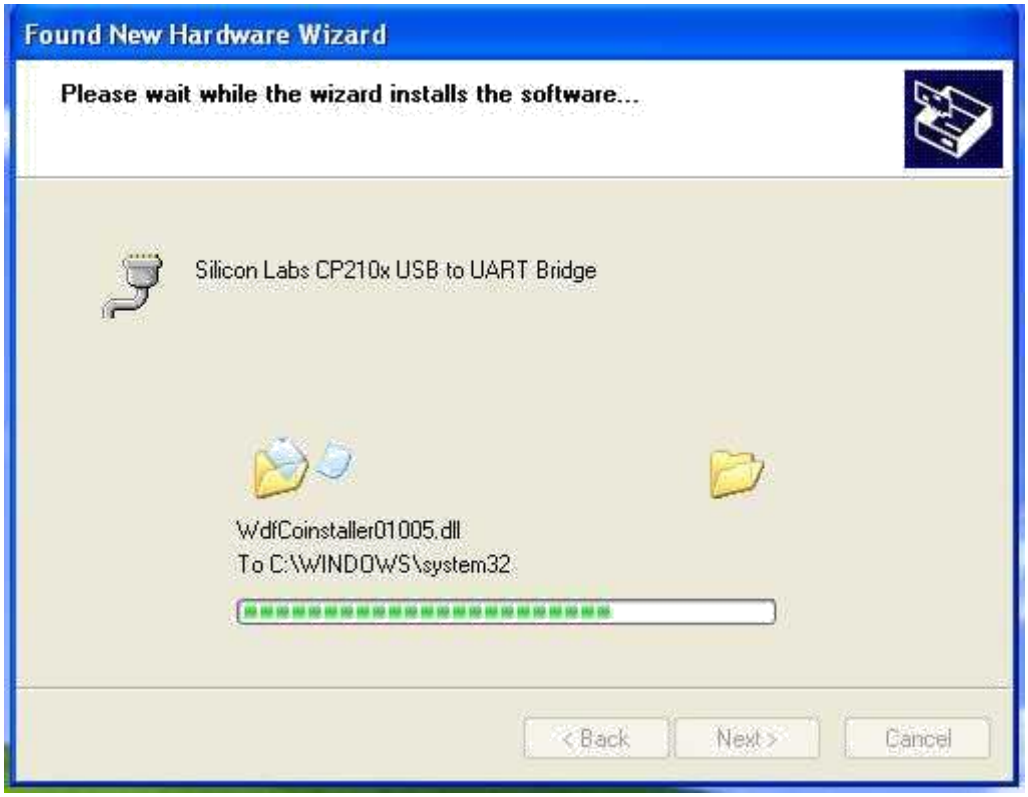
Installation Procedure for USB Cable Driver:

The installation steps are given below:

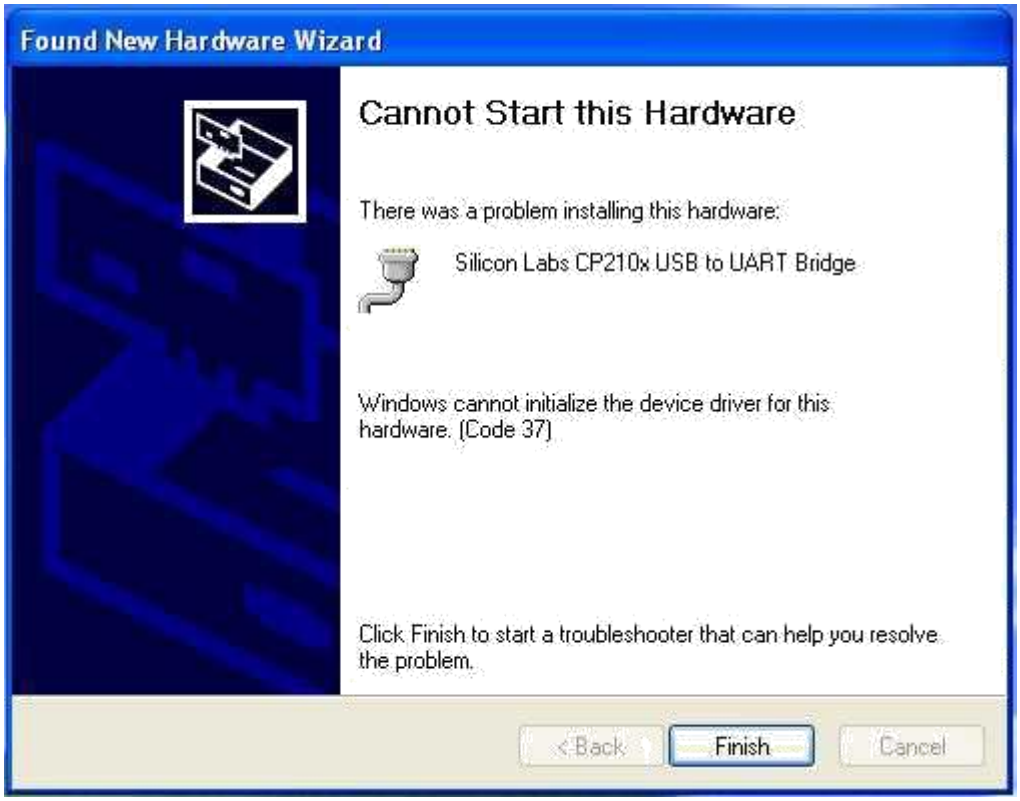
STEP 1: Connect the USB cable between ARM-7 LPC2148 Trainer Kit and PC. Connect the power supply to the trainer kit and power up. After can see a popup window with name "**Found New Hardware CP2102 USB to UART Bride Controller**".

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

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

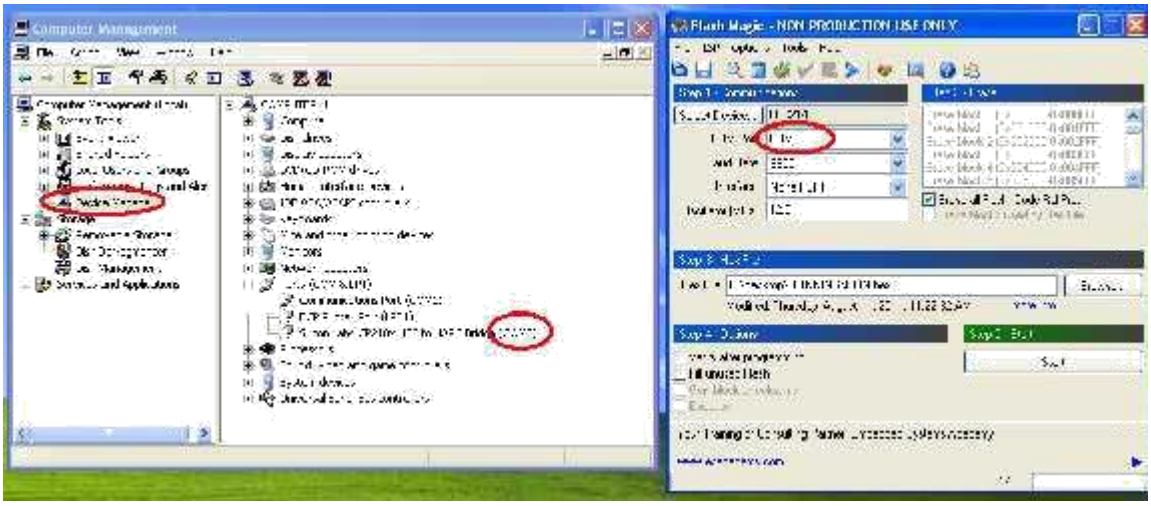


STEP 4: After that can see New Hardware wizard, and then click on finish.



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

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

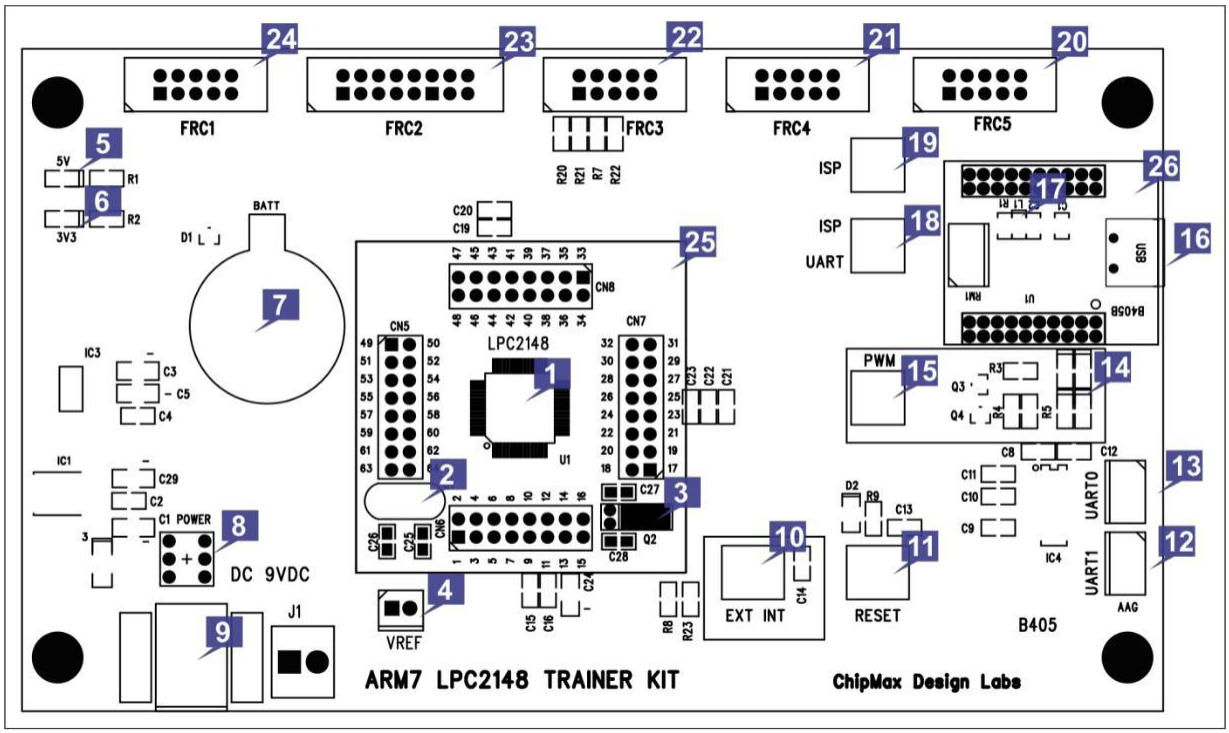


STEP 7: Repeat the **STEP** from 22 to 28

ARM 7 LPC2148 TRAINER KIT

Introduction:

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



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

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

13. UART0 connectivity.

    Optional: Can be use for ISP programming mode.

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

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

Features:

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

Device Daughter Board Details:

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

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

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

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

FRC PIN DETAILS:

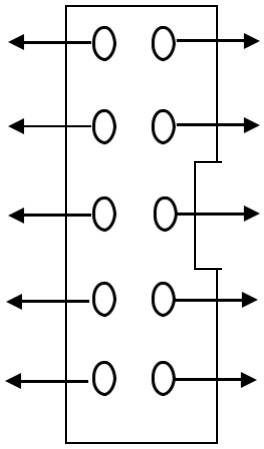
FRC 1	
PIN NO:	**DESCRIPTION**
1	P0.16
2	P0.17
3	P0.18
4	P0.19
5	P0.20
6	P0.21
7	P0.22
8	P0.23
9	+5V
10	GND

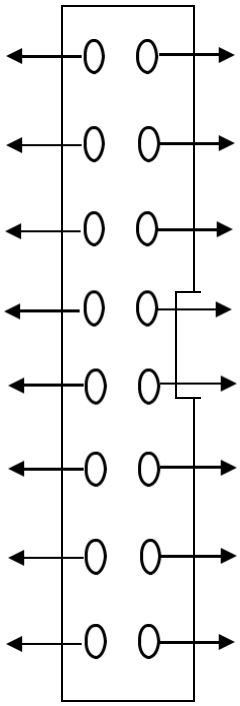
FRC 2	
PIN NO:	**DESCRIPTION**
1	P0.8
2	P0.9
3	P0.10
4	P0.11
5	P1.16
6	P1.17
7	P1.18
8	P1.19
9	P1.20
10	P1.21
11	P1.22
12	P1.23
13	+5V
14	NA
15	+3V3
16	GND

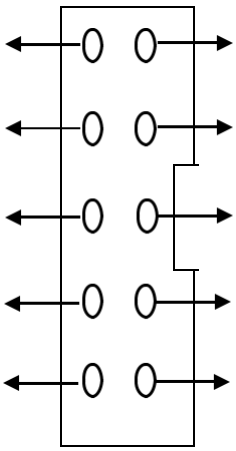
FRC 3	
PIN NO:	**DESCRIPTION**
1	P0.3(SDA0)
2	P0.2(SCL0)
3	P0.14(SDA1)
4	P0.11(SCL1)
5	P0.16 ENT1
6	P0.15 ENT2
7	RS232 TX0
8	RS232 RX0
9	+5V
10	GND

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

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







INTERFACE CARD DETAILS:

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

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

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

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

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

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

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

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

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

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

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

FRC CONNECTION DETAILS:

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

EXP NO:

DATE

LED & FLASHING OF LED'S

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

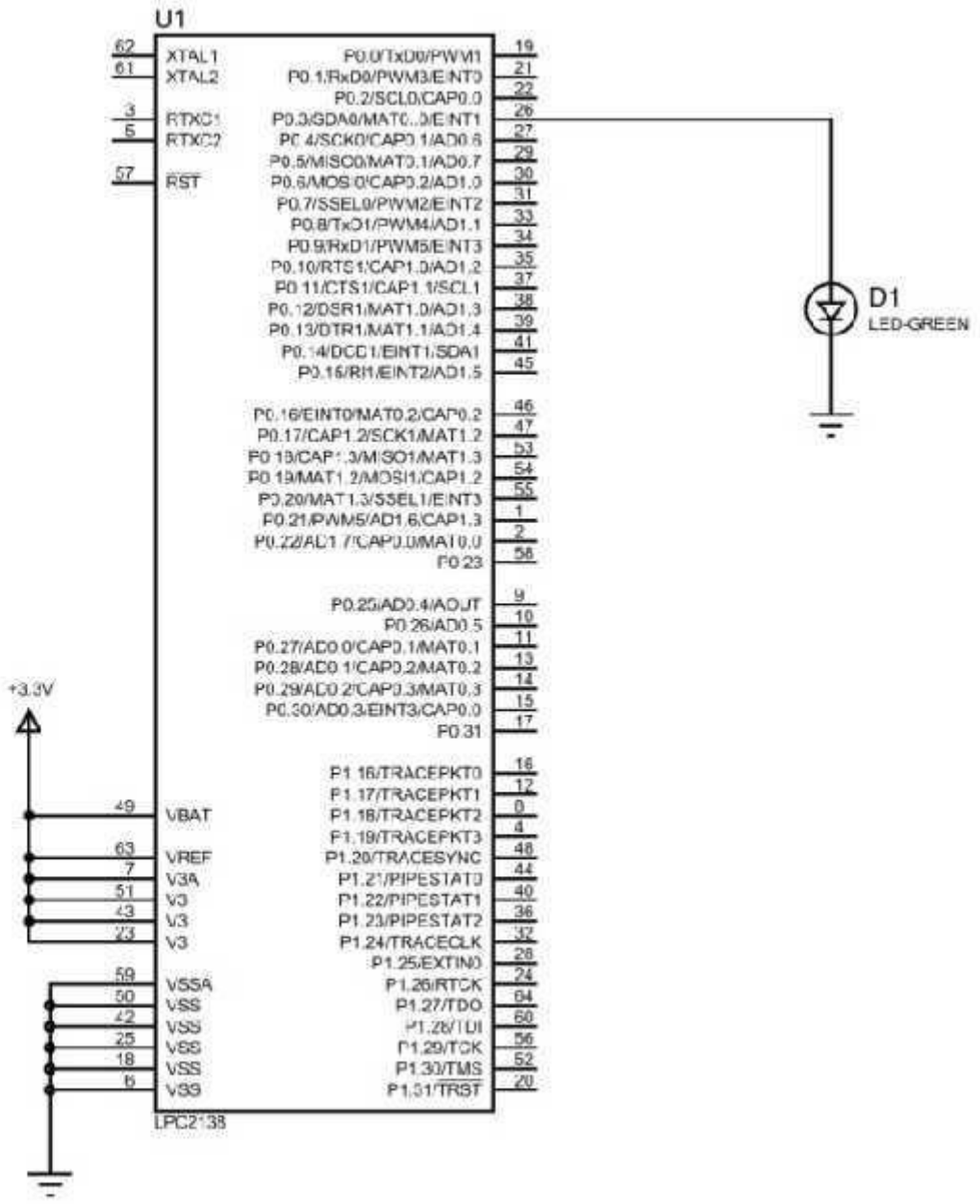
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LED Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

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

LED INTERFACING:

CIRCUIT DIAGRAM:

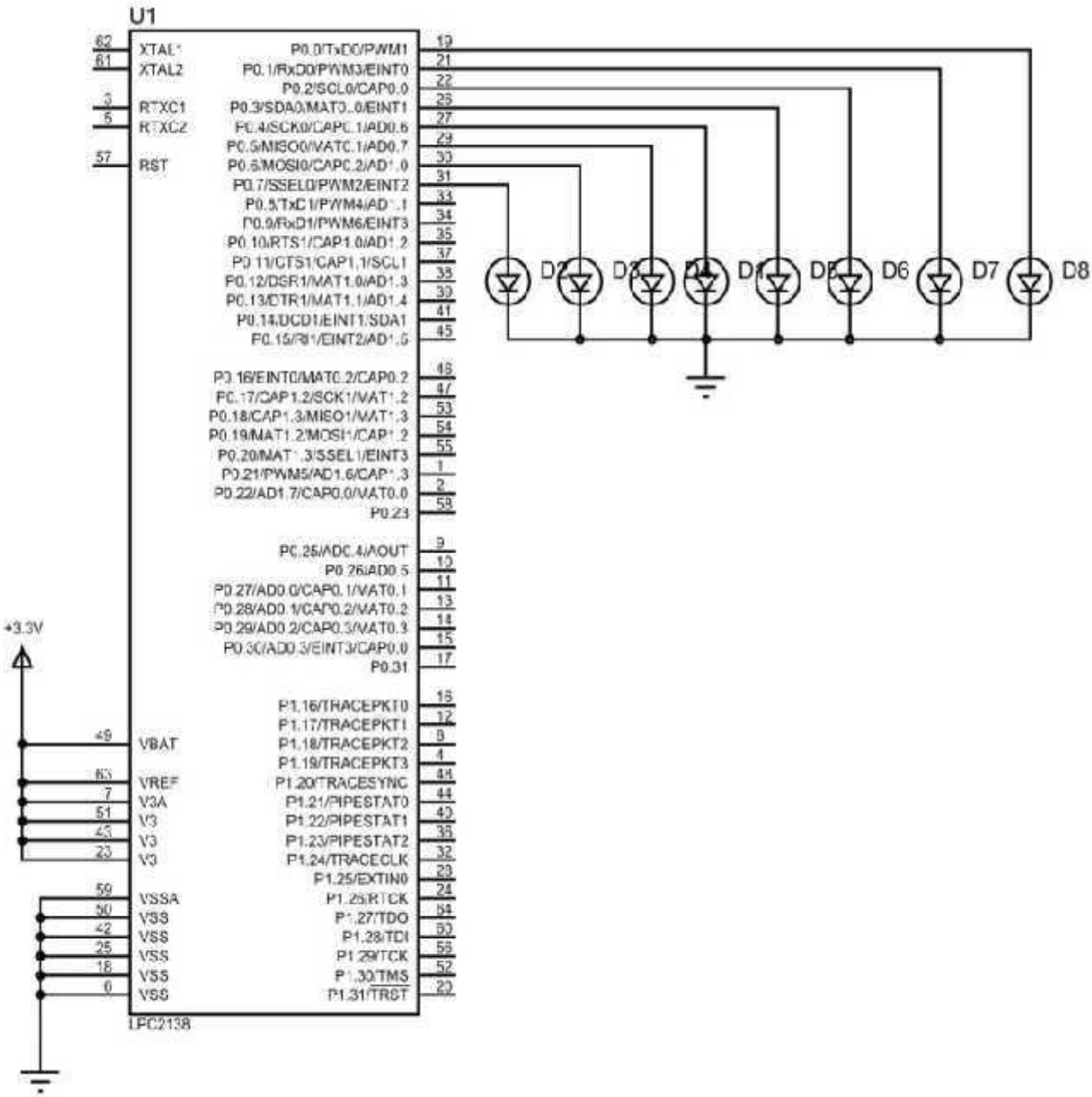


PROGRAM:

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

FLASHING OF LED:

CIRCUIT DIAGRAM:



PROGRAM:

TYPE-I:

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

TYPE-II:

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

TYPE-III

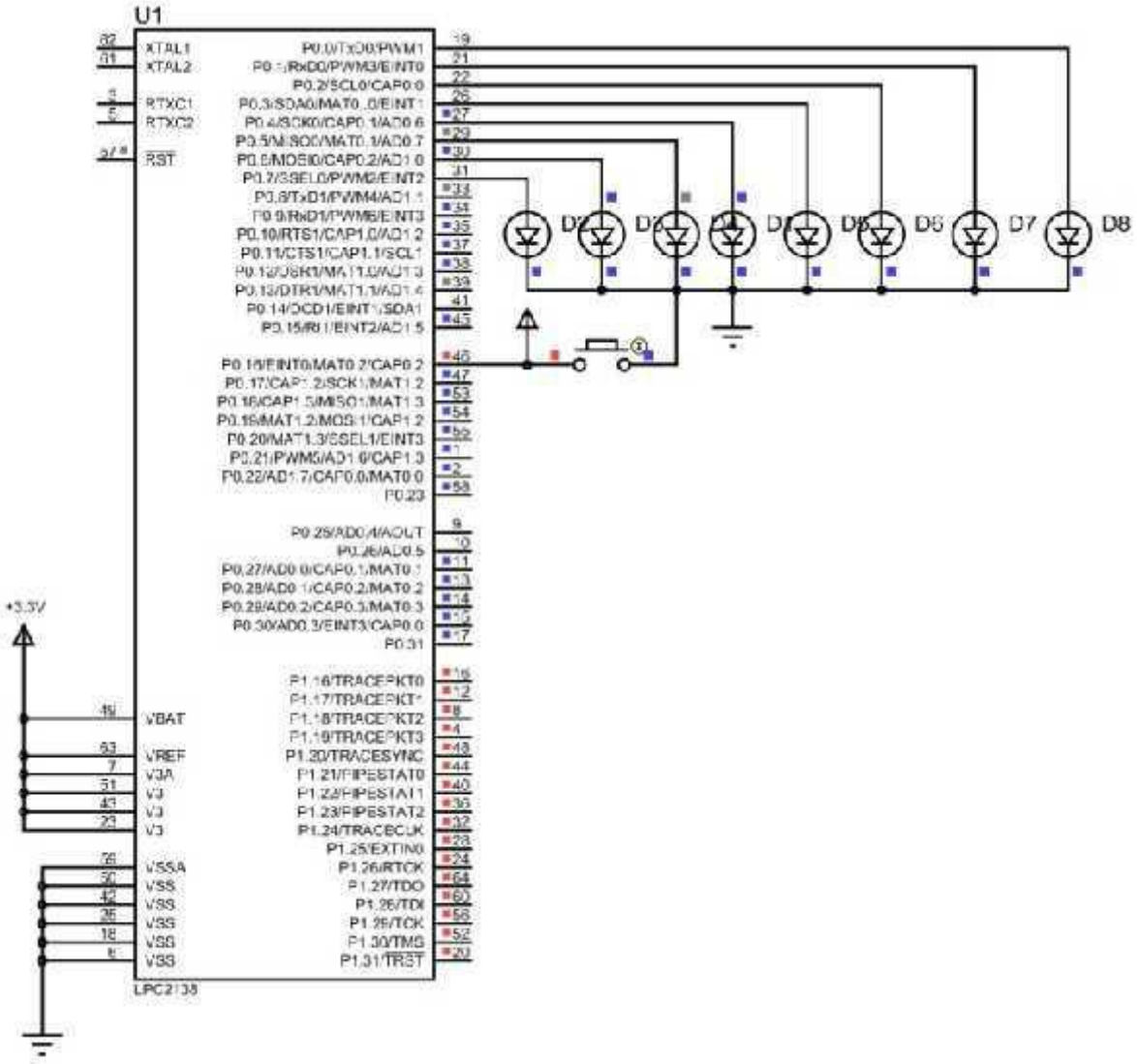
```
#include <lpc214x.h>
int i;
int main()
{
IODIR0=(1<<0)|(1<<1)|(1<<2)|(1<<3)|(1<<4)|(1<<5)|(1<<6)|(1<<7);
while(1)
{
IOSET0=(1<<0);
for(i=0;i<120000;i++);
IOCLR0=(1<<0);
for(i=0;i<120000;i++);
IOSET0=(1<<1);
for(i=0;i<120000;i++);
IOCLR0=(1<<1);
for(i=0;i<120000;i++);
IOSET0=(1<<2);
for(i=0;i<120000;i++);
IOCLR0=(1<<2);
for(i=0;i<120000;i++);
IOSET0=(1<<3);
for(i=0;i<120000;i++);
IOCLR0=(1<<3);
for(i=0;i<120000;i++);
IOSET0=(1<<4);
for(i=0;i<120000;i++);
IOCLR0=(1<<4);
for(i=0;i<120000;i++);
IOSET0=(1<<5);
for(i=0;i<120000;i++);
IOCLR0=(1<<5);
for(i=0;i<120000;i++);
IOSET0=(1<<6);
for(i=0;i<120000;i++);
IOCLR0=(1<<6);
for(i=0;i<120000;i++);
IOSET0=(1<<7);
for(i=0;i<120000;i++);
IOCLR0=(1<<7);
for(i=0;i<120000;i++);
}
}
```

FRONT AND BACK LED:

```
#include <lpc214x.h>
int b,i;
int main()
{
IODIR0=0x000000FF;
while(1)
{
for(b=0;b<8;b++)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
for(b=7;b>=0;b--)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
}
```

SWITCH CONTROLLED LED (1-SWITCH):

CIRCUIT DIAGRAM:

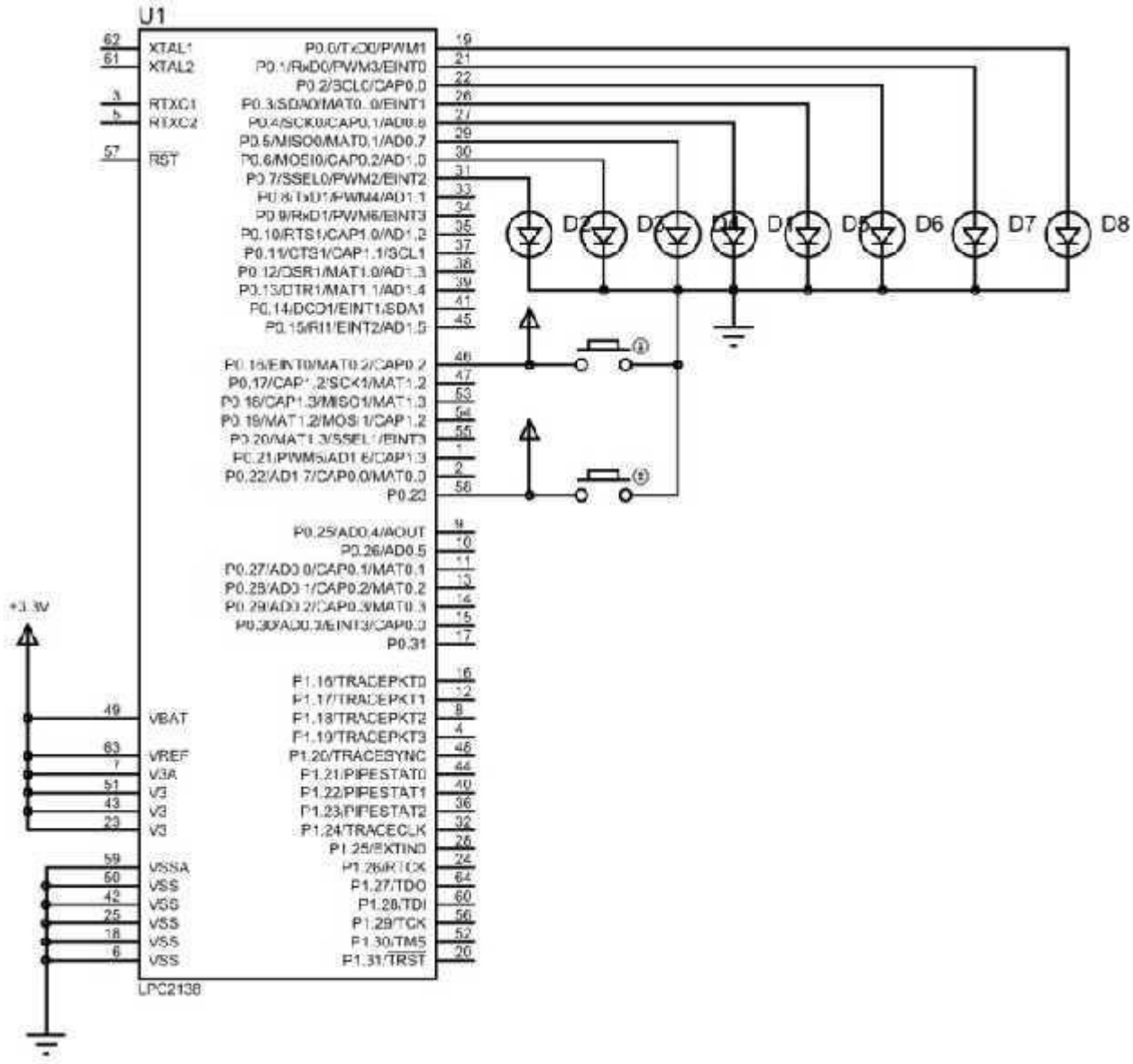


PROGRAM:

```
#include <lpc214x.h>
int i,b;
int main()
{
IODIR0=0x000000FF;
IODIR0=~(1<<16);
while(1)
{
if((IOPIN0&(1<<16))==0)
{
for(b=0;b<8;b++)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
else
{
IOCLR0=0x000000FF;
}
}
}
```

SWITCH CONTROLLED LED (2-SWITCH):

CIRCUIT DIAGRAM:



PROGRAM:

```
#include <lpc214x.h>
int i,b;
int main()
{
IODIR0=0x000000FF;
IODIR0=~(1<<16)&~(1<<23);
while(1)
{
if((IOPIN0&(1<<16))==0)
{
for(b=0;b<8;b++)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
else if((IOPIN0&(1<<23))==0)
{
for(b=7;b>=0;b--)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
else
{
IOCLR0=0x000000FF;
}
}
}
```

RESULT:

EXP NO:	INTERFACING OF LCD
DATE	

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

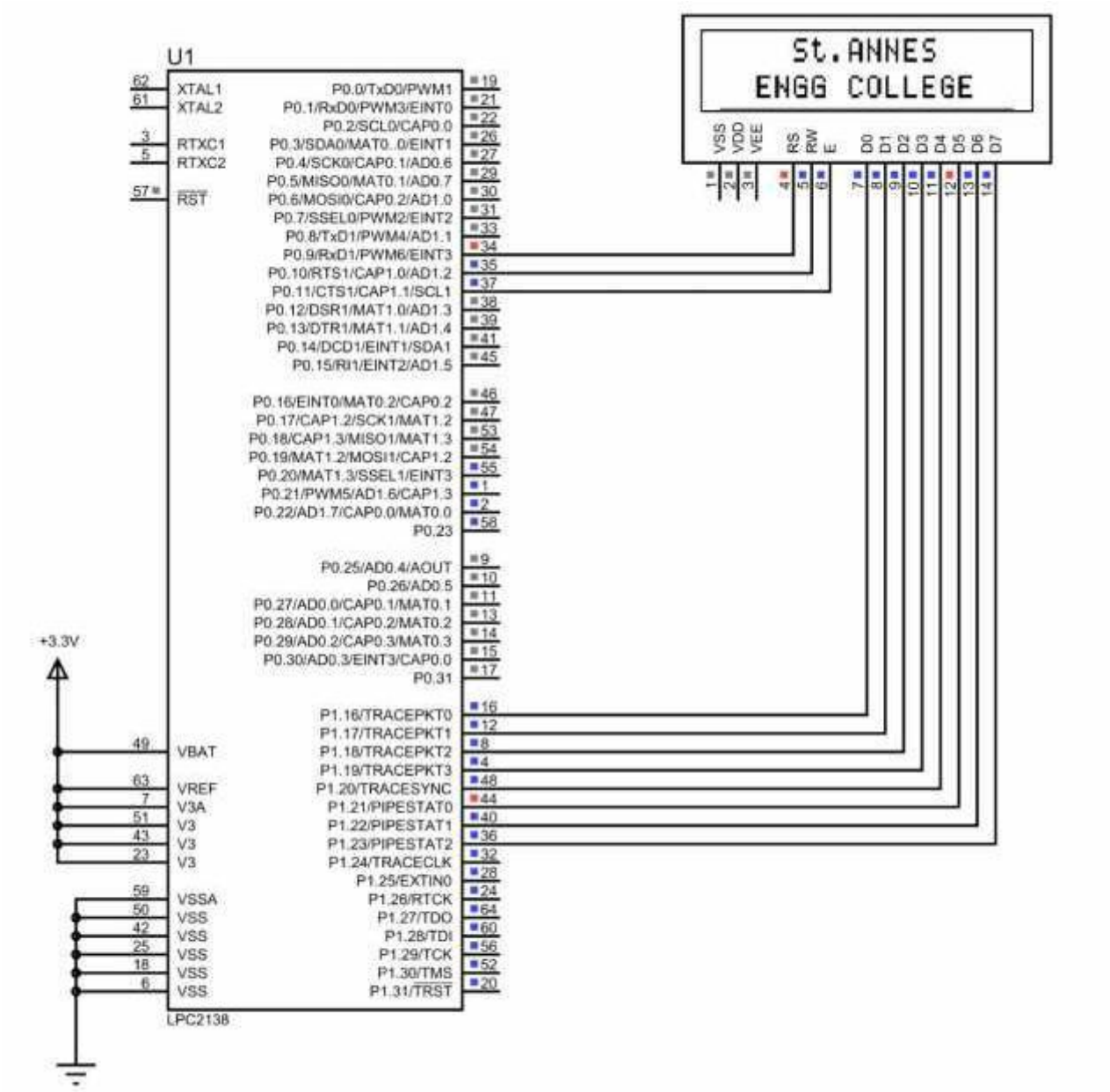
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LED Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

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

INTERFACING LCD:

CIRCUIT DIAGRAM:



PROGRAM:

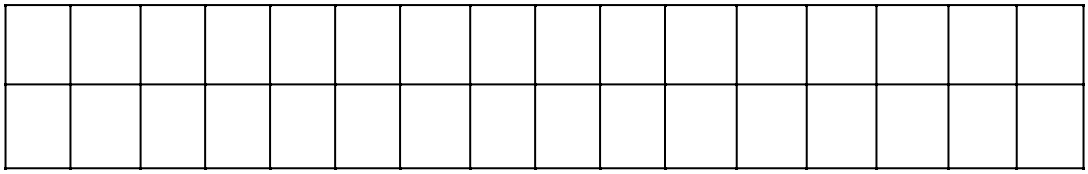
```
#include <lpc214x.h>
#include <lcd.h>

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

LCD LAYOUT:

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

OUTPUT:



RESULT:

EXP NO:	INTERFACING OF MATRIX KEYBOARD
DATE	

AIM:

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

HARDWARE & SOFTWARE TOOLS REQUIRED:

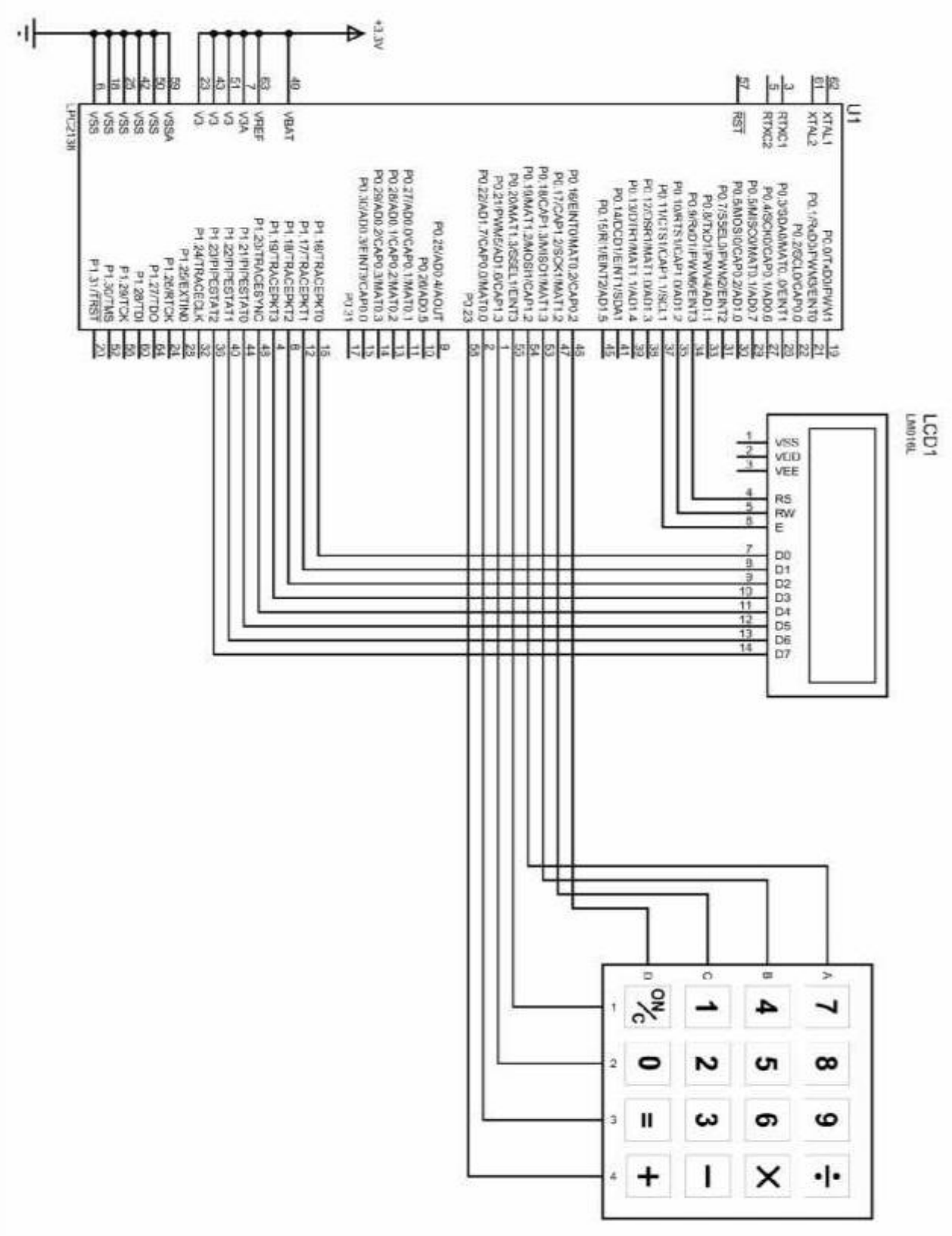
S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LCD Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1
6	Matrix Keypad Module	1

PROCEDURE

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

KEYBOARD INTERFACING:

CIRCUIT DIAGRAM:



PROGRAM:

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

int main()
{
 LCD_INIT();
 LCDSTR(0x00000080,"Matrix Keypad");
  LCDSTR(0x000000C0,"Key Pressed:   ");
while(1)
   {
        IO0CLR = CLR;
        IO0SET = C1;
        delay_ms(10);
        if(scan(R1))LCDSTR(0x000000CC,"0");  //K1
        if(scan(R2))LCDSTR(0x000000CC,"4");  //K5
        if(scan(R3))LCDSTR(0x000000CC,"8");  //K9
        if(scan(R4))LCDSTR(0x000000CC,"C");  //K13
        IO0CLR = CLR;
        IO0SET = C2;
        if(scan(R1))LCDSTR(0x000000CC,"1");  //K2
        if(scan(R2))LCDSTR(0x000000CC,"5");  //K6
        if(scan(R3))LCDSTR(0x000000CC,"9");  //K10
        if(scan(R4))LCDSTR(0x000000CC,"D");  //K14
        IO0CLR = CLR;
        IO0SET = C3;
        if(scan(R1))LCDSTR(0x000000CC,"2");  //K3
        if(scan(R2))LCDSTR(0x000000CC,"6");  //K7
        if(scan(R3))LCDSTR(0x000000CC,"A");  //K11
        if(scan(R4))LCDSTR(0x000000CC,"E");  //K15
        IO0CLR = CLR;
        IO0SET = C4;
        if(scan(R1))LCDSTR(0x000000CC,"3");  //K4
        if(scan(R2))LCDSTR(0x000000CC,"7");  //K8
        if(scan(R3))LCDSTR(0x000000CC,"B");  //K12
        if(scan(R4))LCDSTR(0x000000CC,"F");  //K16
      }
 }
```

RESULT:

EXP NO:	INTERFACING OF STEPPER MOTOR
DATE	

AIM:

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

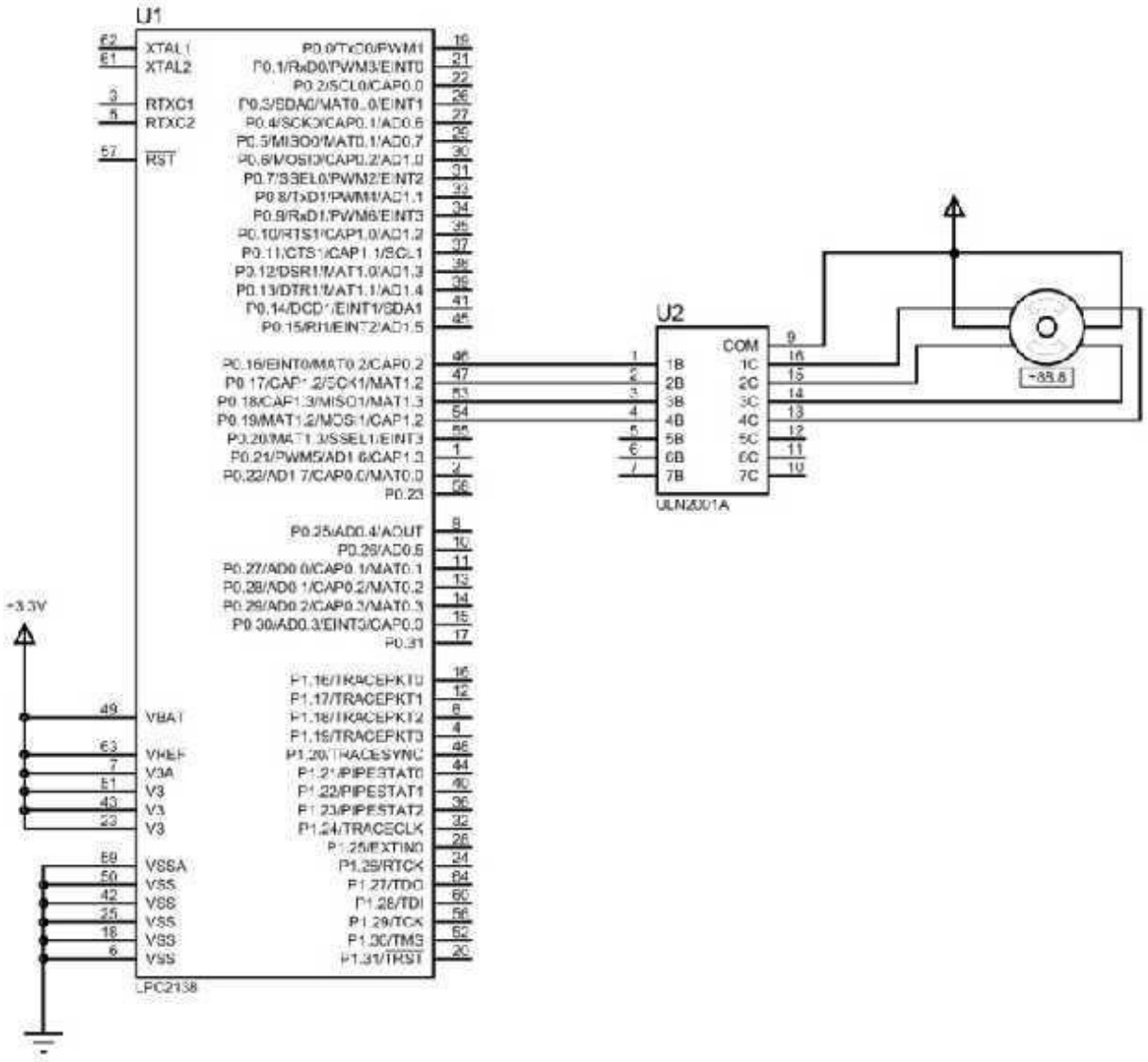
HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	Stepper Motor Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

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

STEPPER MOTOR: (FORWARD ROTATION)
CIRCUIT DIAGRAM:



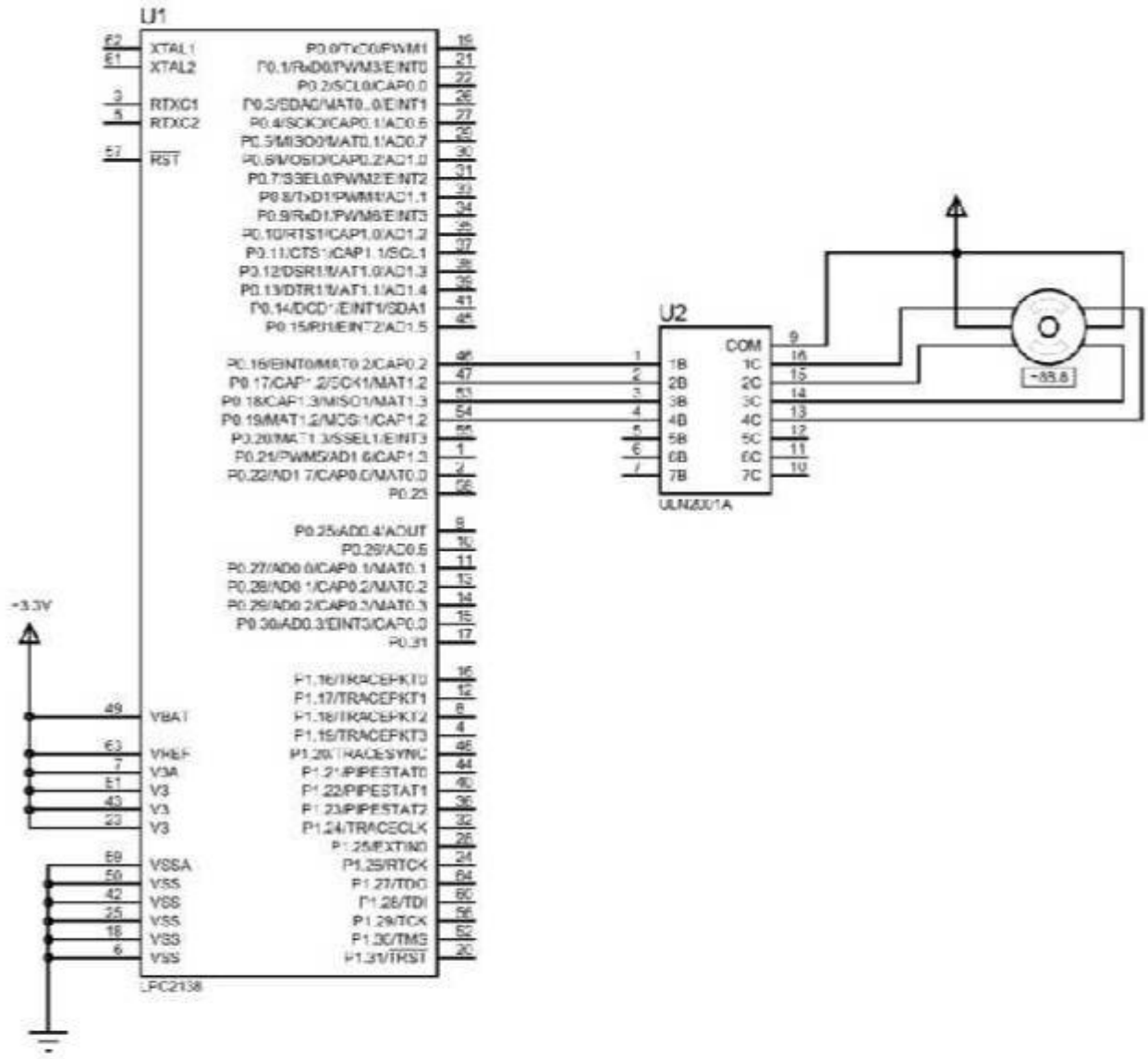
Pin No / Clock	0.16	0.17	0.18	0.19

PROGRAM:

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

int main()
{
IODIR0=(1<<16)|(1<<17)|(1<<18)|(1<<19);
          while(1)
          {
            //FORWARD DIRECTION
           IOCLR0=(1<<16);
           IOCLR0=(1<<17);
           IOSET0=(1<<18);
           IOSET0=(1<<19);
           delay_ms(10);
           IOCLR0=(1<<16);
           IOSET0=(1<<17);
           IOSET0=(1<<18);
           IOCLR0=(1<<19);
           delay_ms(10);
           IOSET0=(1<<16);
           IOSET0=(1<<17);
           IOCLR0=(1<<18);
           IOCLR0=(1<<19);
           delay_ms(10);
           IOSET0=(1<<16);
           IOCLR0=(1<<17);
           IOCLR0=(1<<18);
           IOSET0=(1<<19);
           delay_ms(10);
          }
}
```

STEPPER MOTOR: (REVERSE ROTATION)
CIRCUIT DIAGRAM:



Pin No / Clock	0.16	0.17	0.18	0.19

PROGRAM:

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

int main()
{
IODIR0=(1<<16)|(1<<17)|(1<<18)|(1<<19);
        while(1)
        {
         //REVERSE DIRECTION
         IOSET0=(1<<16);
         IOCLR0=(1<<17);
         IOCLR0=(1<<18);
         IOSET0=(1<<19);
         delay_ms(10);
         IOSET0=(1<<16);
         IOSET0=(1<<17);
         IOCLR0=(1<<18);
         IOCLR0=(1<<19);
         delay_ms(10);
         IOCLR0=(1<<16);
         IOSET0=(1<<17);
         IOSET0=(1<<18);
         IOCLR0=(1<<19);
         delay_ms(10);
         IOCLR0=(1<<16);
         IOCLR0=(1<<17);
         IOSET0=(1<<18);
         IOSET0=(1<<19);
         delay_ms(10);
        }
}
```

RESULT:

Miniprojects for IOT:

1. Garbage Segregator and Bin Level Indicator
2. Colour based Product Sorting
3. Image Processing based Fire Detection
4. Vehicle Number Plate Detection
5. Smart Lock System

Objective:

To apply the knowledge, they gained in doing the experiments.

Team constitution:

A team size may be from 3 to 4 students.

Guidelines:

1.Students shall from a group and can do their mini project.

2.Student must buy their own hardware setup for doing Miniprojects.

3.If they are utilizing the college resource, they should get approval from HoD.

4.At the end, a report along with hardware must be submitted to college.

5.If required students need to present their work as presentation.

Garbage Segregator and Bin Level Indicator

IDEA:

With progress in human technology we have seen a substantial progress in the amount of waste generated. Recycling is the only way to manage this huge amount of waste. But recycling requires garbage to be segregated. Without segregation garbage cannot be recycled because different type of garbage requires different recycling processes.

Also It is important to educate users and instruct them every time they come near the dustbin about instructions about throwing the trash. For this purpose we design a garbage disposal system that uses multiple dustbins with a voice based system that speaks to the user each time he she stands before the dustbin.

The system makes use of a camera to detect presence if any person in front of the dustbin. If a person is detected, the system issues voice instructions to the user about throwing right garbage in the right bin. In case the dustbin is full it instructs the user to find another dustbin to throw garbage in.

To develop this system we make use of a raspberry Pi controller. The controller is interfaced with a camera and a voice speaker for detection and communication. The controller gets dustbin level input using ultrasonic level sensors each having LED indicators interfaced to it. The level sensors are used to constantly feed the raspberry pi with bin levels.

The raspberry pi is also interfaced with a Wifi module to transmit the level data over the internet. The Level sensor panels are made to be easily mounted over any dustbin. This allows the system to be easily screwed over any dustbin for instant installation.

The data is transmitted over IOT to IOT gecko platform which displays the bin level data over internet. This indication can be used to alert the authorities that the garbage bins need to be emptied. Thus the system automates garbage segregation and level monitoring to help counter the garbage crisis using IOT.

Note: The Dustbins are not included in this kit. The sensors can be mounted over any open dustbins.

Components

Raspberry Pi

Wifi Module

Ultrasonic Level Sensors

LED Indicators

Camera

Speaker

Wiring

Supporting Frame

Buttons & Switches

Screws & Bolts

Resistors

Capacitors

Diodes

IC’s

Transistors

Connectors

PCB

Colour based Product Sorting

IDEA:

Color Based Object Sorting has a wide usage in fruit sorting as well as candy sorting industries. This system puts forward a mechanism to detect color and sort items through image processing. Once identified a mechanism is used to sort the candies into particular bins baskets. We here demonstrate this mechanism using a camera with electronic circuitry along with sorting mechanism using 3 bins. The system uses raspberry pi connected to a controller circuit to achieve this task. The controller circuit consists of a camera attached to it that detects color of a small object in front of it. A motor is used to feed an object to the camera chamber. As soon is the color is detected a signal is sent to the sorter mechanism which uses a motor to position the sorting tube towards respective section. A feeder is then used to push the object towards the tubs so that it gets sorted and next object is pulled in by the feeder. The action details are sent to the IOT server using iotgecko platform to keep track of the number of objects sorted in each section. Thus, we achieve a completely automated IOT based sorting system.

Hardware Specifications

Raspberry Pi 3

Camera

Servo Motor

LCD Display

Resistors

Capacitors

Transistors

Cables and Connectors

Diodes

PCB and Breadboards

LED

Transformer/Adapter

Push Buttons

Switch

IC

IC Sockets

Connector Shaft

Bed Frame

Tubes

Screws & Joints

Supporting Frame

Software Specifications

Programming Language:

Python

IOTGecko

BLOCK DIAGRAM:

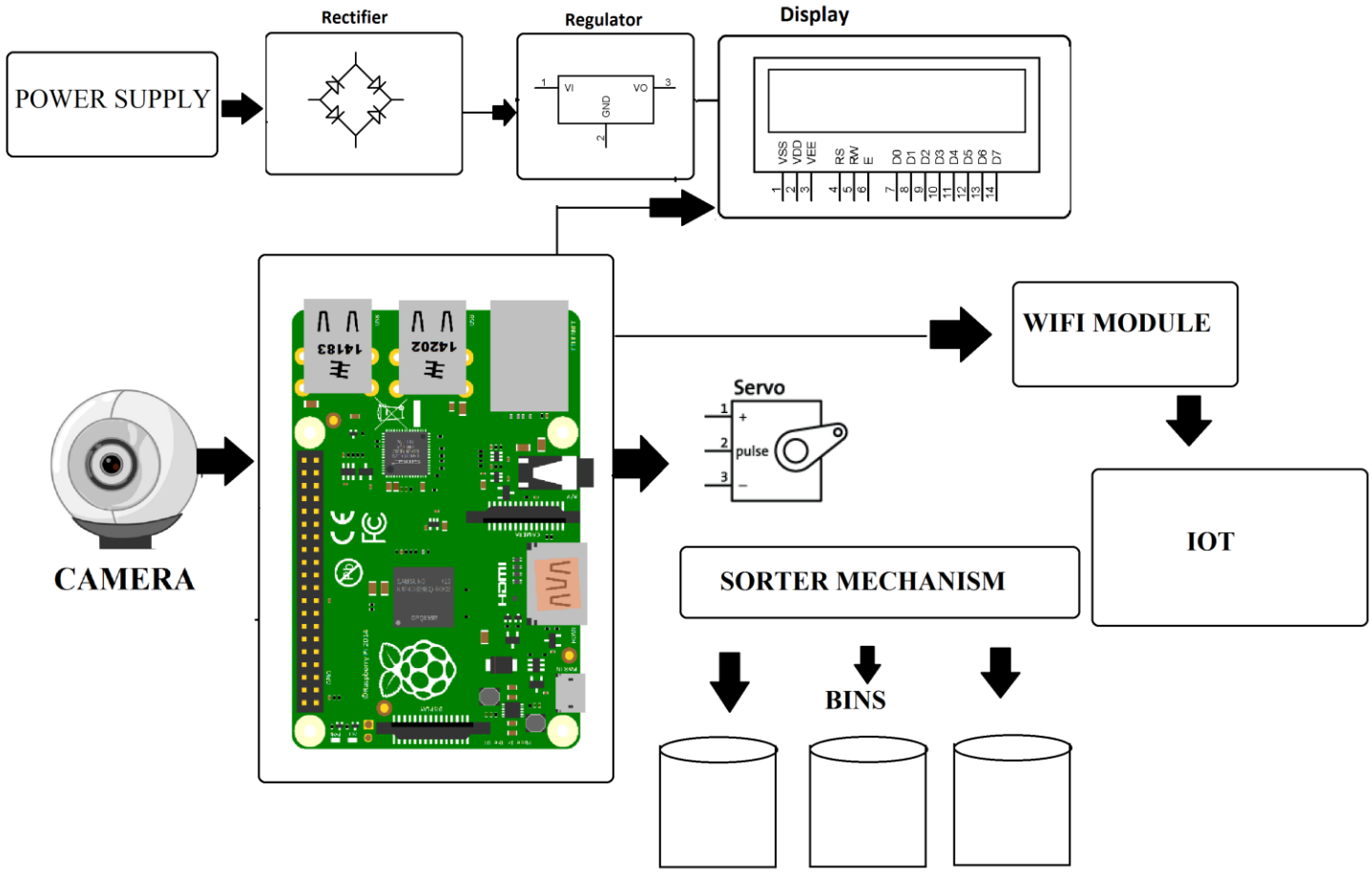


Image Processing based Fire Detection

IDEA:

The main advantage of Image Processing Based Fire Detection System is the early warning benefit. This system can be installed just about anywhere in a commercial building, malls and at many more public places for fire detection. This system uses camera for detecting fires. So we do not need any other sensors to detect fire. System processes the camera input and then processor processes it to detect fires. The heat signatures and fire illumination patterns are detected in images to determine if it is a fire and take action accordingly. On detecting fire system goes into emergency mode and sounds an alarm. Also displays the status on the LCD display informing about the system.

Hardware Specifications

Raspberry Pi 3

Camera

Buzzer

LCD Display

Resistors

Capacitors

Transistors

Cables and Connectors

Diodes

PCB and Breadboards

LED

Transformer/Adapter

Push Buttons

Switch

IC

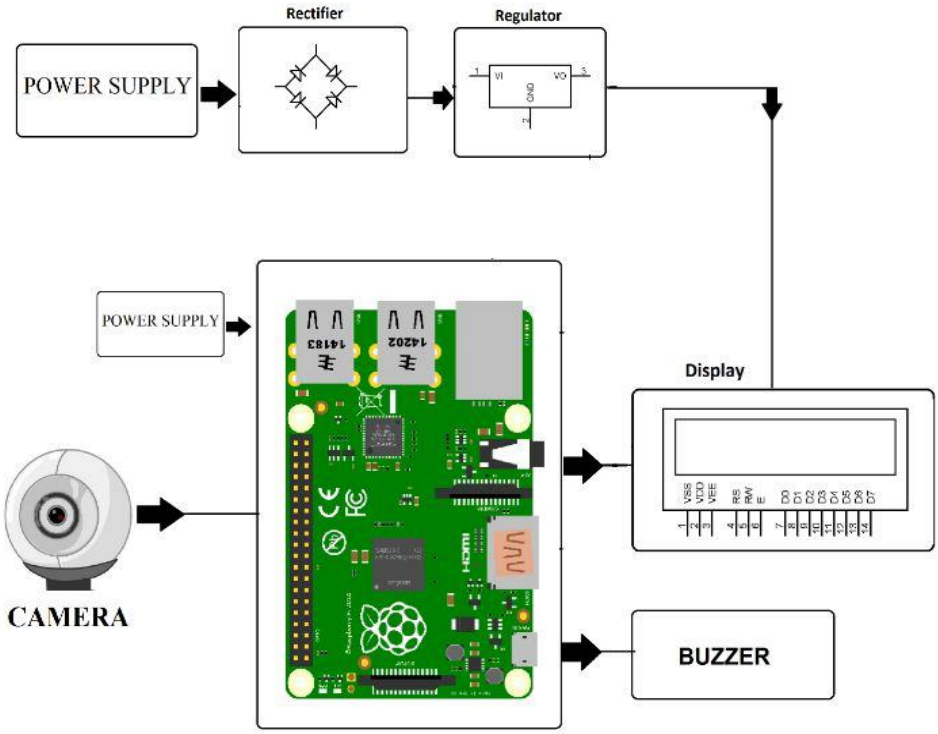
IC Sockets

Software Specifications

Linux

Programming Language: Python

BLOCK DIAGRAM:



Vehicle Number Plate Detection

IDEA:

OBJECTIVE

Project Code :TEMBMA3101

The main objective of this project is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate.

ABSTRACT

The basic idea of this project is to build a **number plate recognition system using python.** Real-Time license plate detection and recognition can be very useful for automating toll booths, finding out traffic rule breakers, and for addressing other vehicle-related security and safety issues.

The system uses a camera interfaced to a Raspberry Pi. The system constantly processes incoming camera footage to detect any trace of number plates. On sensing a number plate in front of the camera, it processes the camera input, extracts the number plate part from the image. Processes the extracted image using OCR and extracts the number plate number from it. The system then displays the extracted number on the monitor.

Keywords: Raspberry Pi, Ultrasonic sensor, web camera, power supply

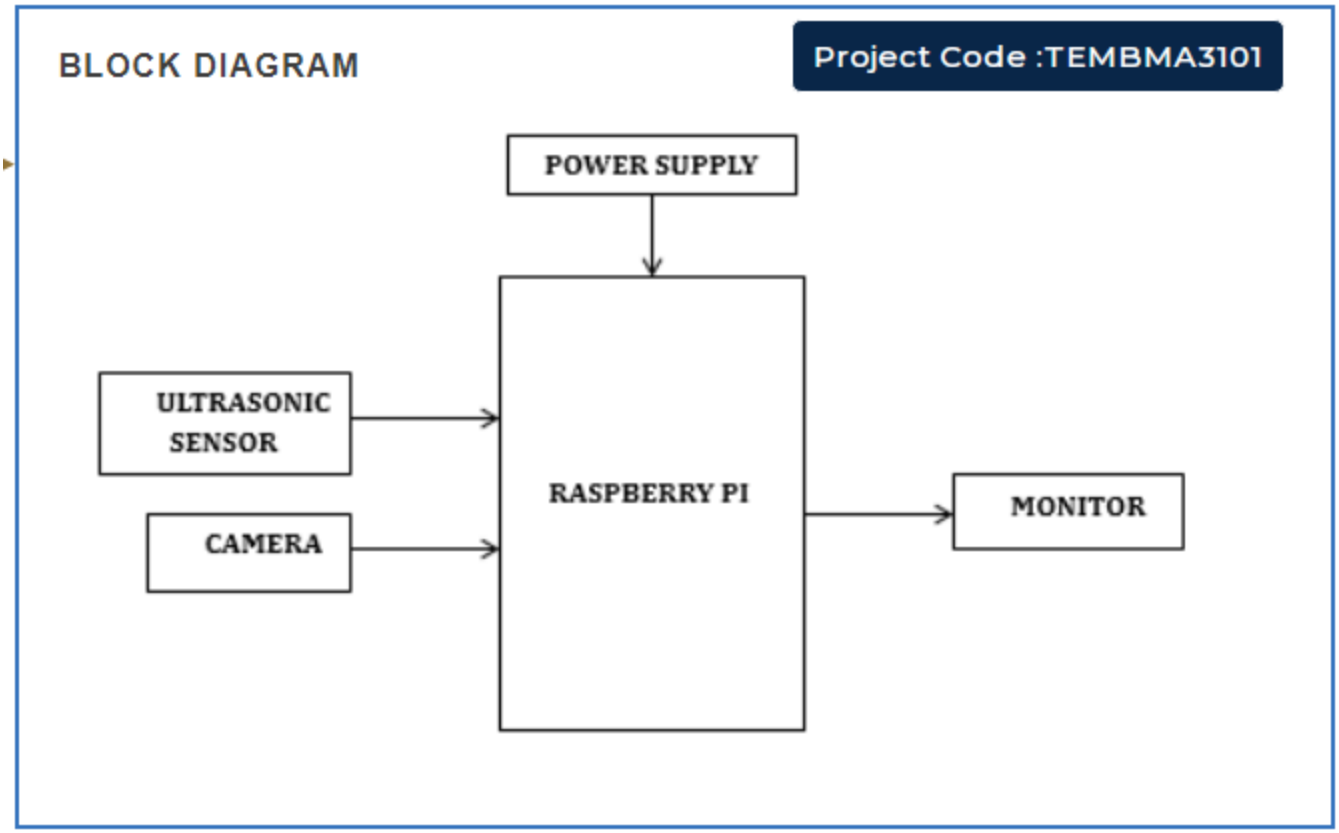
SPECIFICATIONS

Hardware components:

- Raspberry pi
- Memory Card
- Ultrasonic sensor
- Web camera
- Monitor
- 5V 2A Adapter

Software requirements:

- Python3 IDLE
- NOOBS OS
- VNC Viewer
- Advanced IP Scanner



SMART LOCKSYSTEM

IDEA:

From connected cars to connected wearables to home security, the Internet of Things is rapidly marking its presence in every field. Now we have IoT enabled home automation and security devices that can be controlled from anywhere in the world using the Internet of Things. There are many kinds of Wi-Fi door lock available in the market which makes your home more secure and saves time in finding the keys. Here we are also building a similar Wi-Fi door lock which can be controlled from the Smartphone

So in this project, we are going to make an **IOT based Door Lock System using NodeMCU**, Solenoid Lock, and Adafruit IO. Here NodeMCU will act as the main controller and connect the user to the door lock system using the Internet. This allows the user to lock/unlock his Home's door lock by using a smartphone from anywhere in the world.

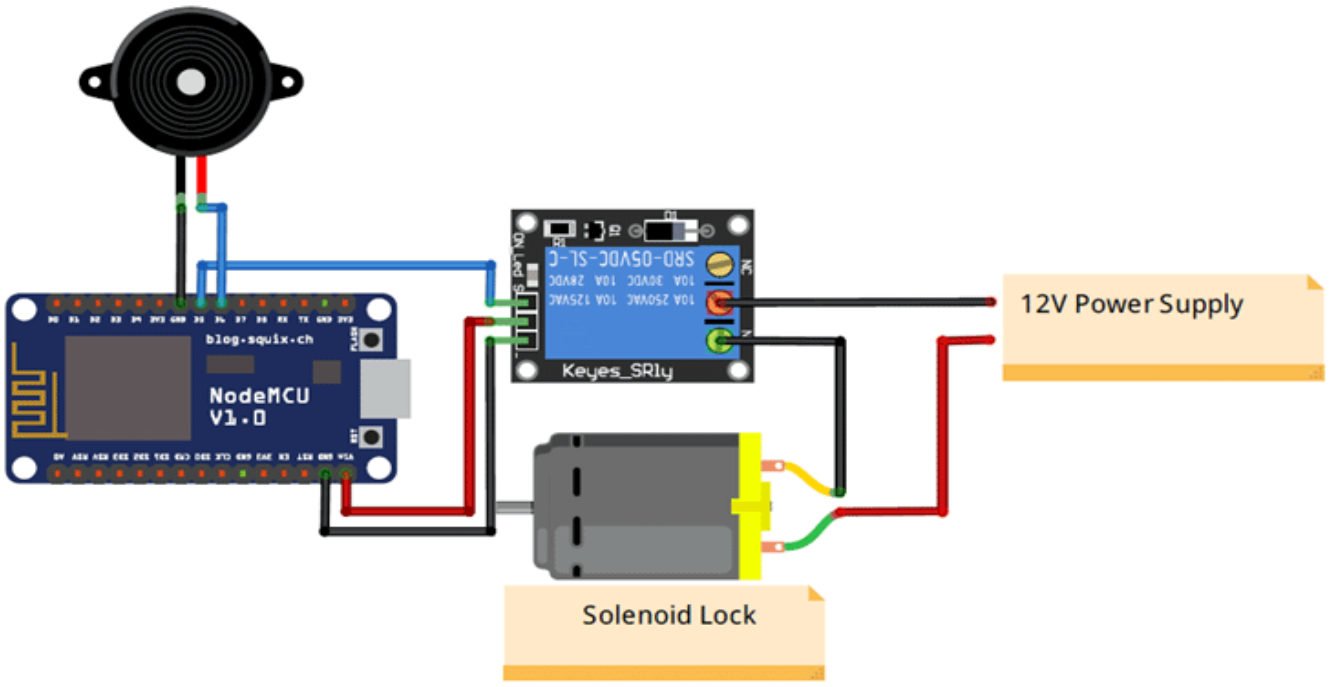
Components Required

NodeMCU ESP8266

Solenoid Lock

Relay Module

Buzzer



Reference: https://www.researchgate.net/publication/367087623_Smart_Door_Locking_System_Using_IoT