MALLA REDDY ENGINEERING COLLEGE

(AUTONOMOUS)

An UGC Autonomous Institution, affiliated to JNTUH,
Accredited by NAAC with 'A' Grade & NBA and
Recipient of World Bank Assistance under TEQIP–II S.C.1.1

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL

54310 – ADVANCED EMBEDDED SYSTEM LABORATORY

I M.Tech II Semester

(MR-15 Regulations)

Prepared By
Thota Ravi Theja,
Assistant Professor,
ECE Department,
MREC (A).

Verified By

Approved By

LIST OF EXPERIMENTS

Note: The following programs are to be implement on ARM based Processors using Keil IDE and Flash magic.

List of experiments:

1. Introduction to ARM Development Board & Software
2. Simple Assembly Program for
    a. Addition | Subtraction | Multiplication | Division
3. Simple Assembly Program for
    a. Operating Modes, System Calls and Interrupts
    b. Loops, Branches
4. Write an Assembly programs to configure and control General Purpose Input/Output (GPIO) port pins.
5. Write an Assembly programs to read digital values from external peripherals and execute them with the Target board.
6. Program to demonstrate Time delay program using built in Timer / Counter feature on IDE environment
7. Program to demonstrates a simple interrupt handler and setting up a timer
8. Program demonstrates setting up interrupt handlers. Press button to generate an interrupt and trace the program flow with debug terminal.
9. Program to Interface 8 Bit LED and Switch Interface
10. Program to implement Buzzer Interface on IDE environment
11. Program to Displaying a message in a 2 line x 16 Characters LCD display and verify the result in debug terminal.
12. Demonstration of Serial communication. Transmission from Kit and reception from PC using Serial Port on IDE environment use debug terminal to trace the program.

INDEX

Experiment: 1

Introduction to ARM Board (LPC2148):

This section of the document introduces LPC2148 microcontroller board based on a 16-bit/32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine microcontrollers with embedded high-speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30% with minimal performance penalty. The meaning of LPC is Low Power Low Cost microcontroller. This is 32 bit microcontroller manufactured by Philips semiconductors (NXP).

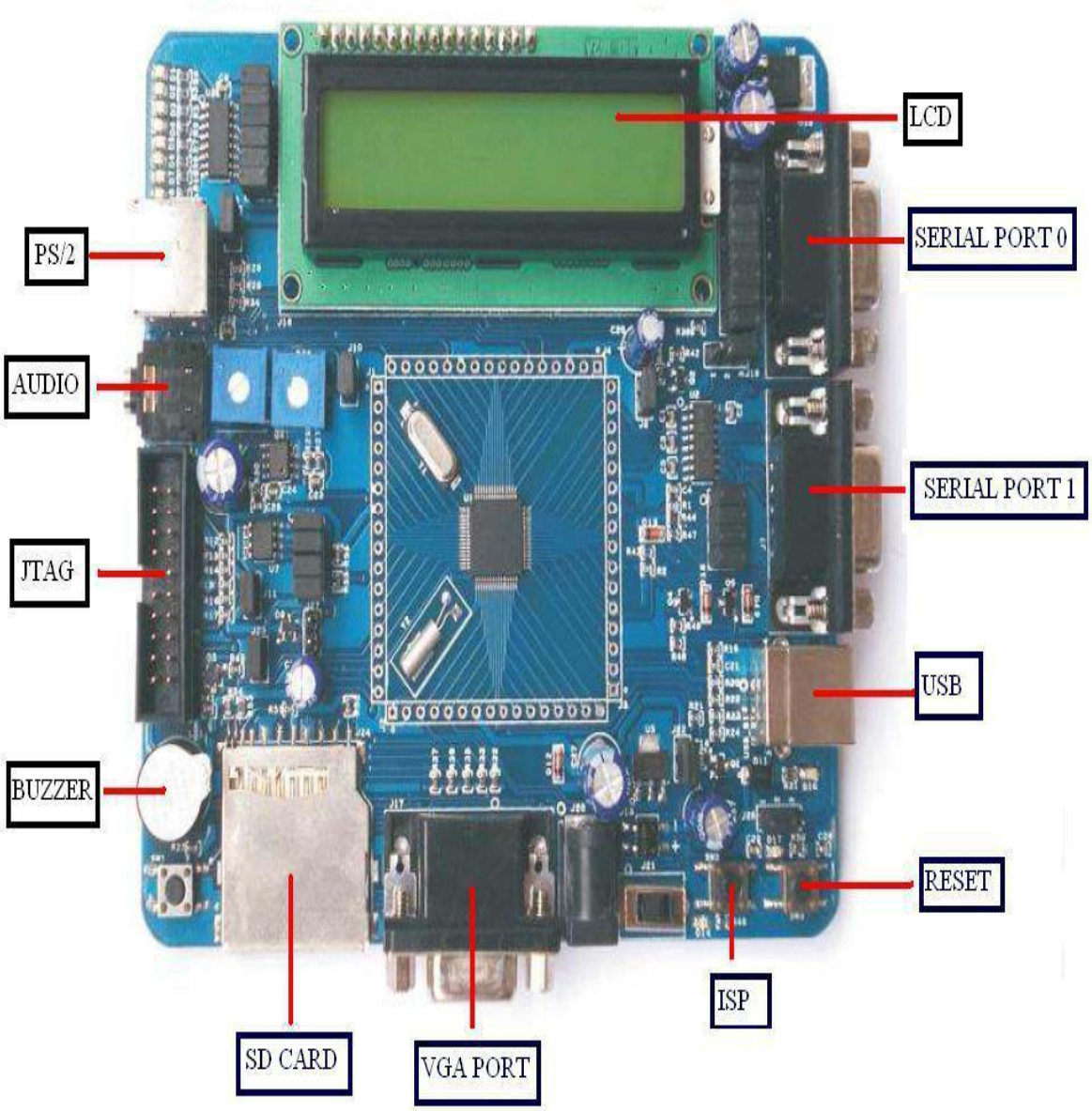
Due to their tiny size and low power consumption, LPC2148 is ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale.

Features of ARM Microcontroller:

16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory; 128-bit wide interface/accelerator enables high-speed 60 MHz operation. In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software, single flash sector or full chip erase in 400 ms and programming of 256 Bytes in 1 ms Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high-speed tracing of instruction execution.

USB 2.0 Full-speed compliant device controller with 2kB of endpoint RAM. In addition, the LPC2148 provides 8 kB of on-chip RAM accessible to USB by DMA. One or two (LPC2141/42 vs, LPC2144/46/48) 10-bit ADCs provide a total of 6/14 analog inputs, with conversion times as low as 2.44 ms per channel.Single 10-bit DAC provides variable analog output (LPC2148 only) Two 32-bit timers/external event counters (with four captures and four compare Channels each), PWM unit (six outputs) and watchdog. Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input.

LPC2148 (ARM7)



Modules and Jumpers Relationship

Jumper	Related	Usage
J6	UART0 & UART1	Connecting all pins enables both UART0 and UART1 and pins 5 and 7 enable UART0.
J8	VREF voltage	Connecting this will set the VTEF voltage to .3V
J9	Test LED's	Connecting all pins enables test LED's Pins 3 to 9 are connected to SPI0 lines of LPC2148.
J10	ADC	This will enable the ADC interface
J11	JTAG	This will enable the debug mode on the microcontroller.
J12	Keyboard(PS/2)	This will enable the PS/2 connector.
J13	Keyboard(PS/2)	This will provide 5V supply to PS/2
J18	LCD	Connecting all pins enabled LCD. Pins 1 to 7 are data lines, 9 to 13 are control lines and pin 15 is 5V power pin.
J19	LCD Backlight	If pins 1 are 2 are connected the LCD back light will always stay PN and if pins 2 and 3 are connected the back can be controlled by firmware.
J22	Power supply to board	Connecting this will provide 3.3V supply to board.
J25	I2C	By connecting all pins it enables I2C interface and its status is displayed on LCD.
J26	Bootloader select	If pins 1 and 2 are connected, manual bootloader mode is selected and If pins 2 and 3 are connected auto bootloader mode is selected. UART0 to be used foe this purpose.
J27	RTC	Connect a battery to use RTC.

Introduction to KEIL software:

Introduction:

1. Keil was found in 1982 by Gunler and Reinhard keil.
2. Keil implemented the first c compiler design.
3. Keil provides a board range of development tools, ANSK, macro assembler, debugger and linkers.
4. Keil micro vision debuggers accurately simulate on-chip peripherals (input/output, A/D converters, D/A converters).

Features:

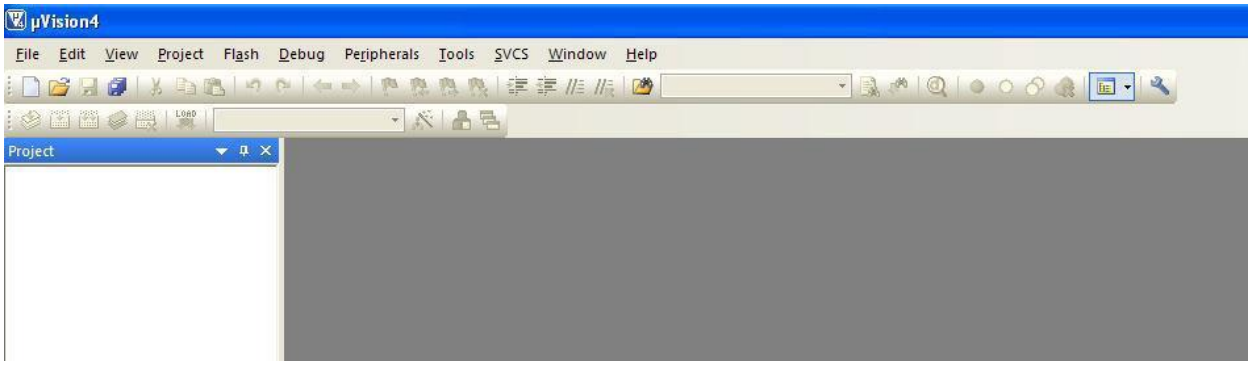
1. Nine basic data types including 32-bit IEEE floating point.
2. Flexible, variable allocation with bit, data, bdata, idata, xdata and pdata of the memory types.
3. Interrupt functions may be written in C.
4. Fall use of the 8051 register banks.
5. Complete symbol and type information for source level debugger.
6. Use of AIMP and ACAII instructions.
7. Bit addressable data objects.
8. Built in interface for the RTX51 real time kernel.
9. Support for dual data pointers at the ATMEL, AED, EMPRESS, and DELL as semiconductor in line on Philips and transcend microcontroller.

KEIL µVision 4 Tool

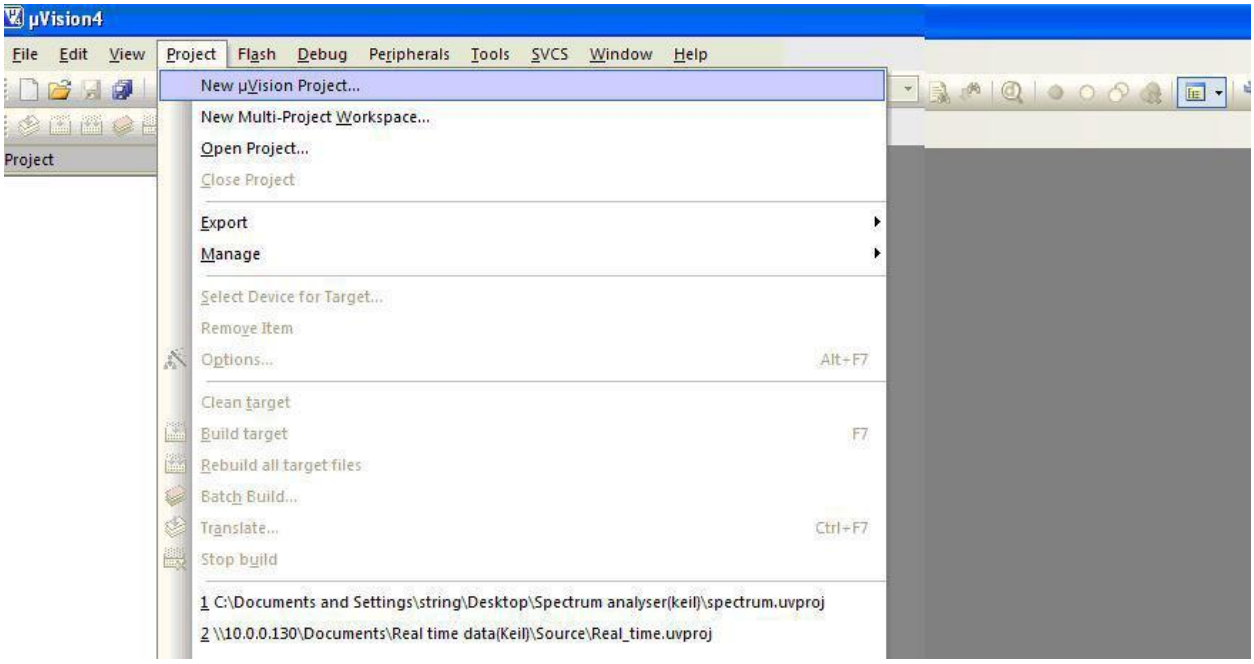
How to work with keil?

How to create a new µProject?

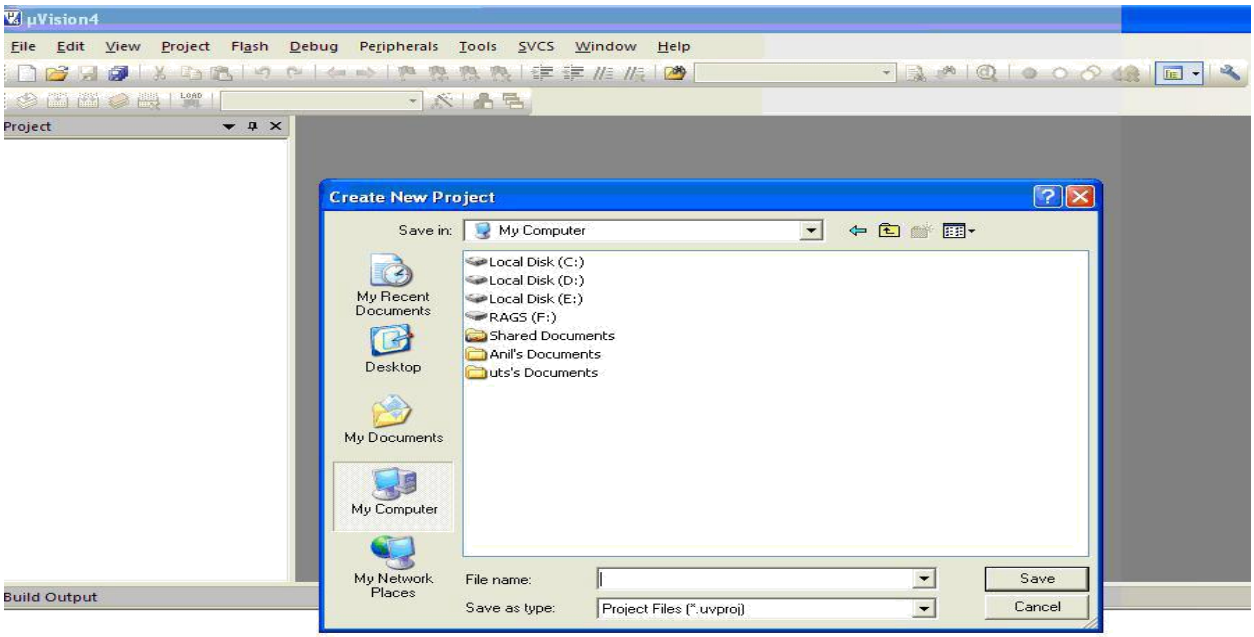
Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below.



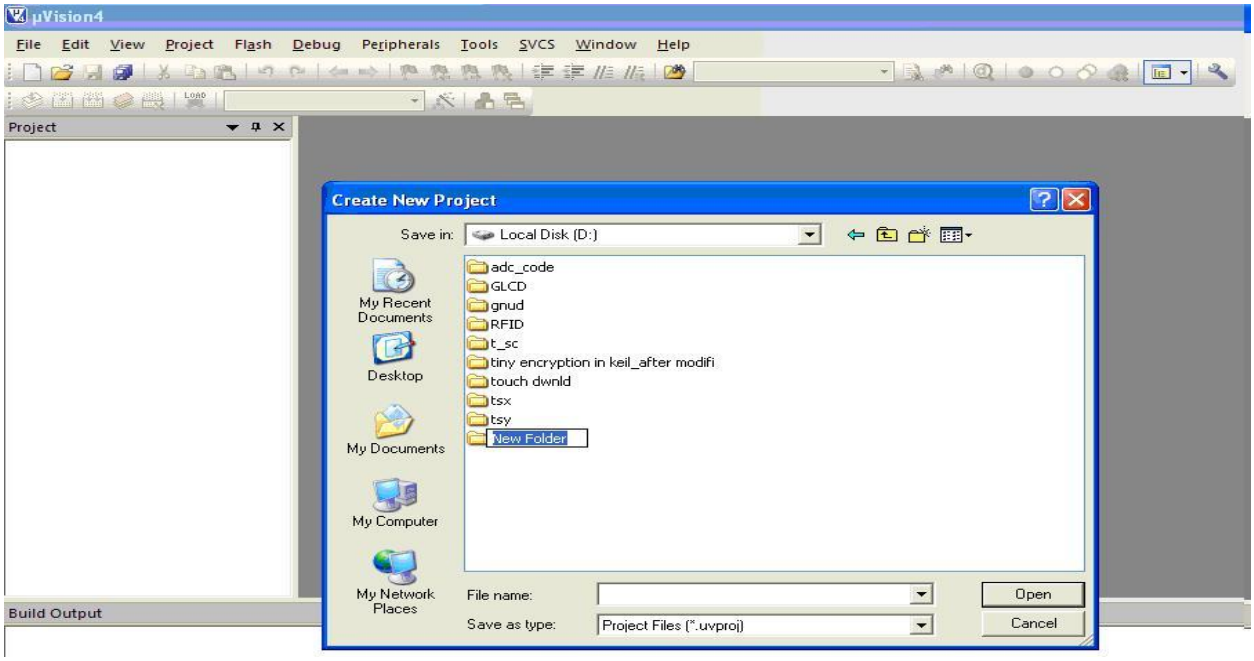
Step 2: To create new project go to project select new micro vision project.



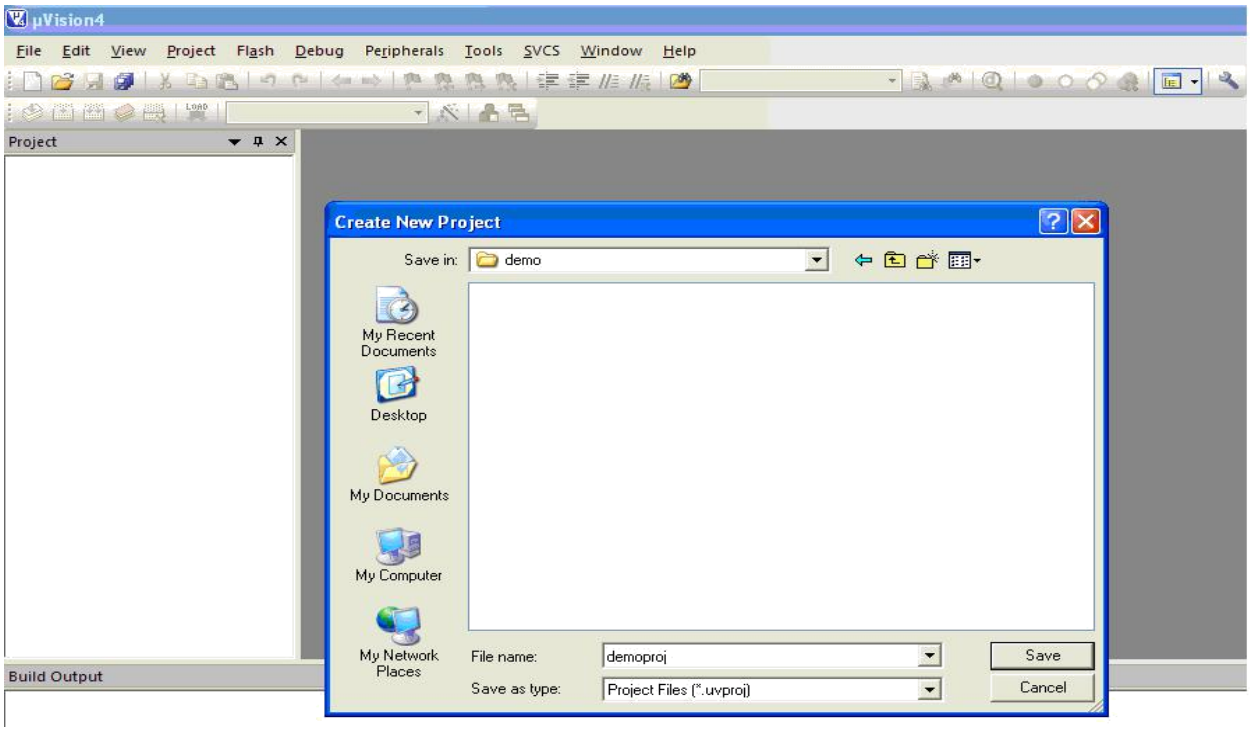
Step 3: select a drive where you would like to create your project.



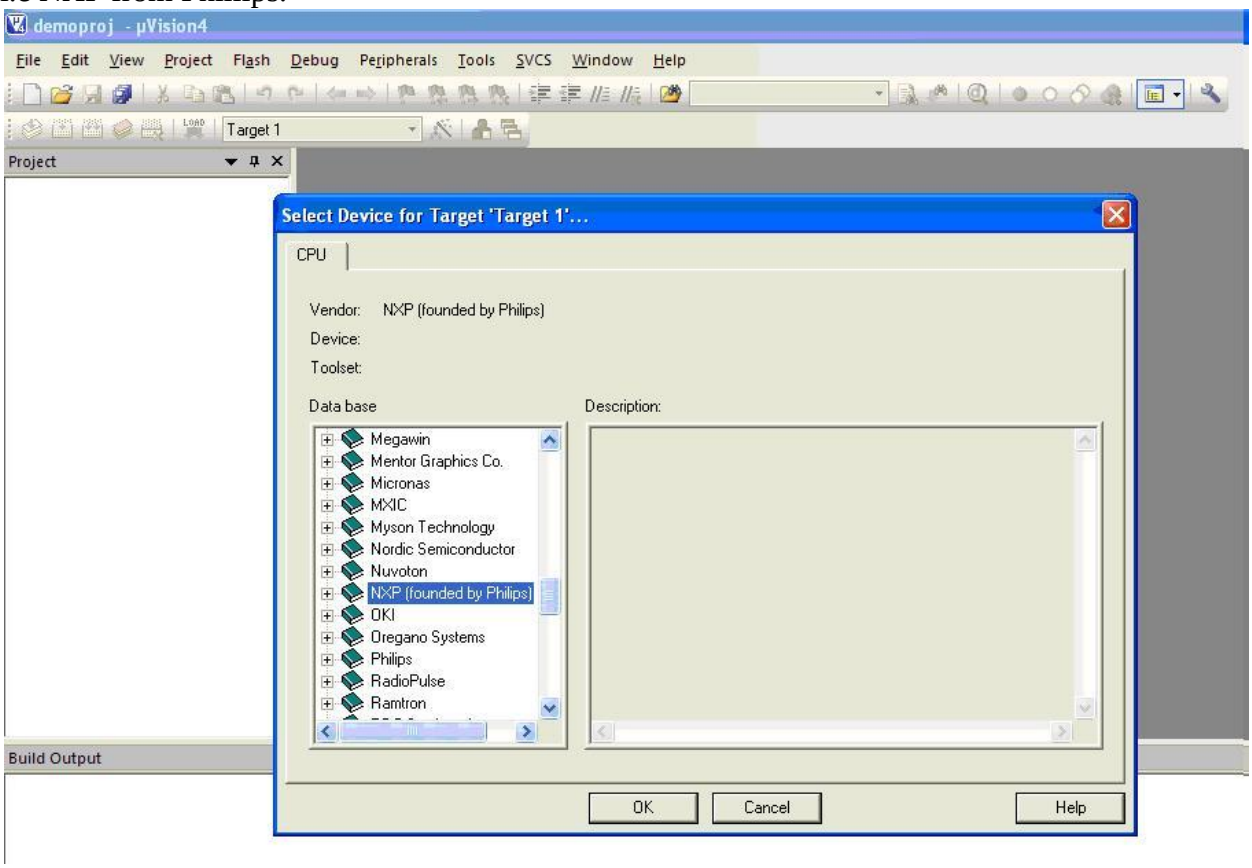
Step 4: Create a new folder and name it with your project name.



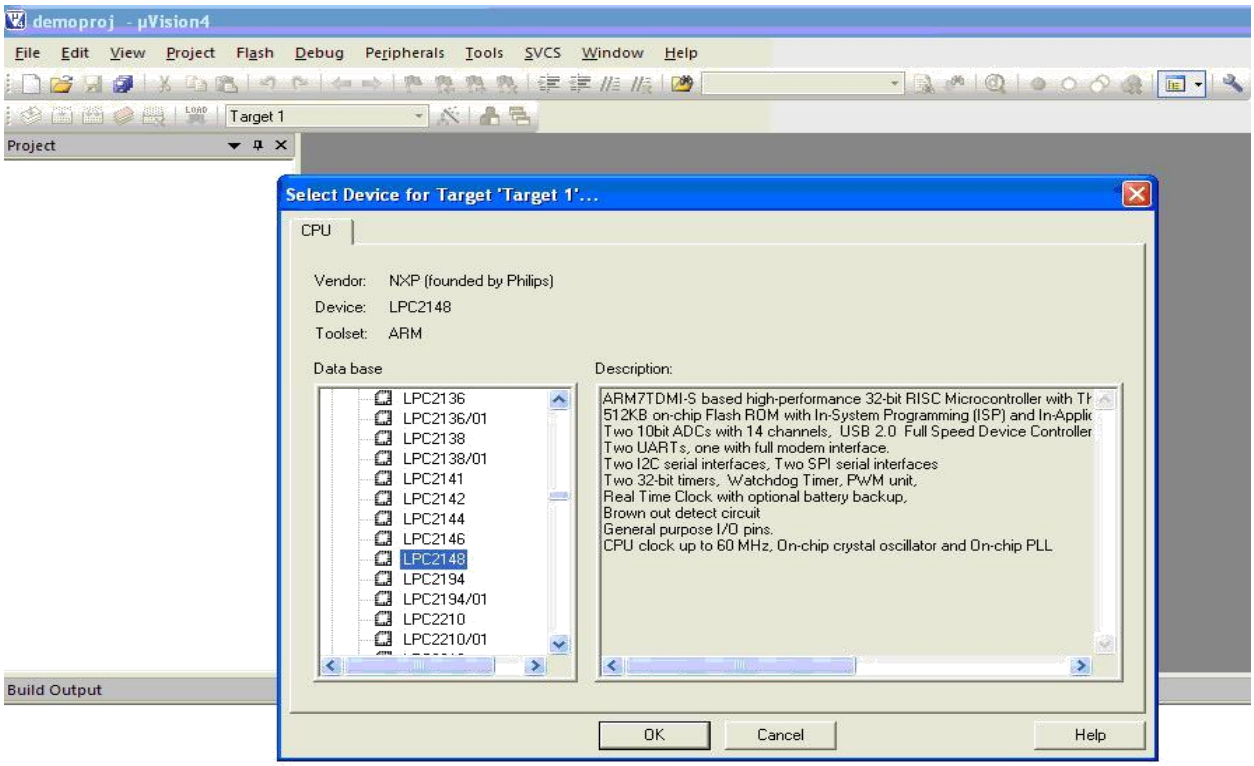
Step 5: Open that project folder and give a name of your project executable file and save it.



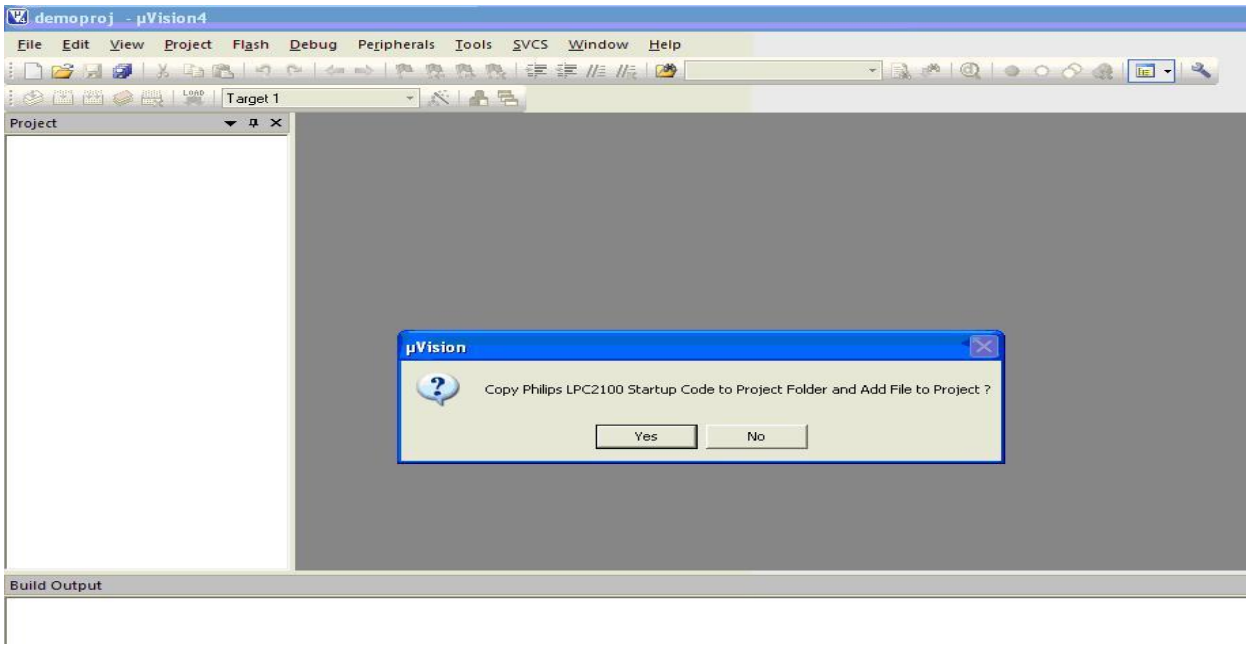
Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips.



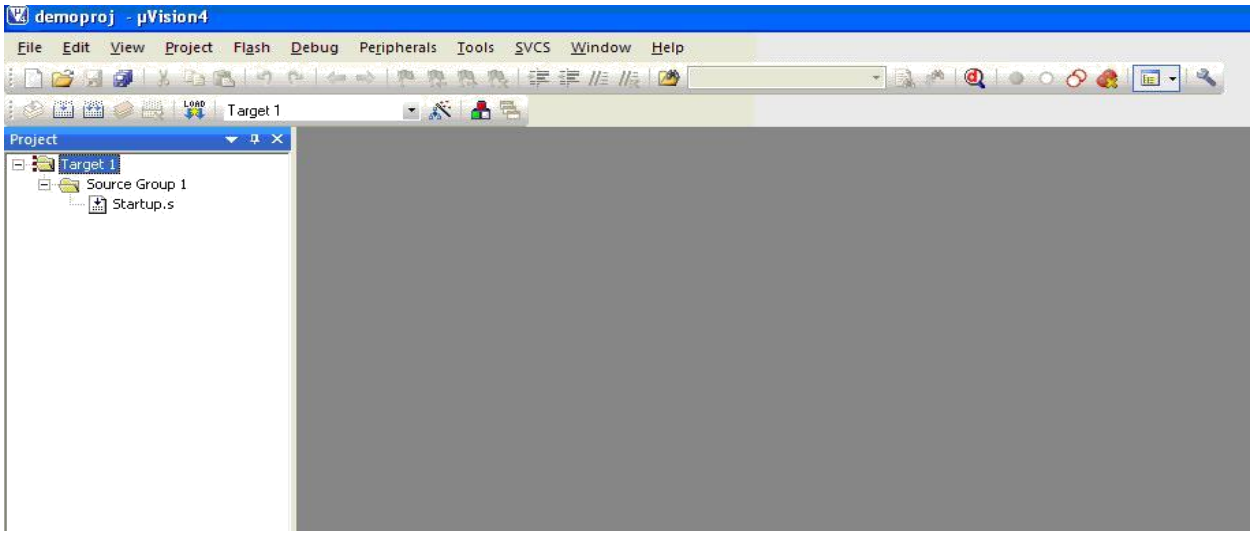
Step 7: Select your chip as LPC2148.



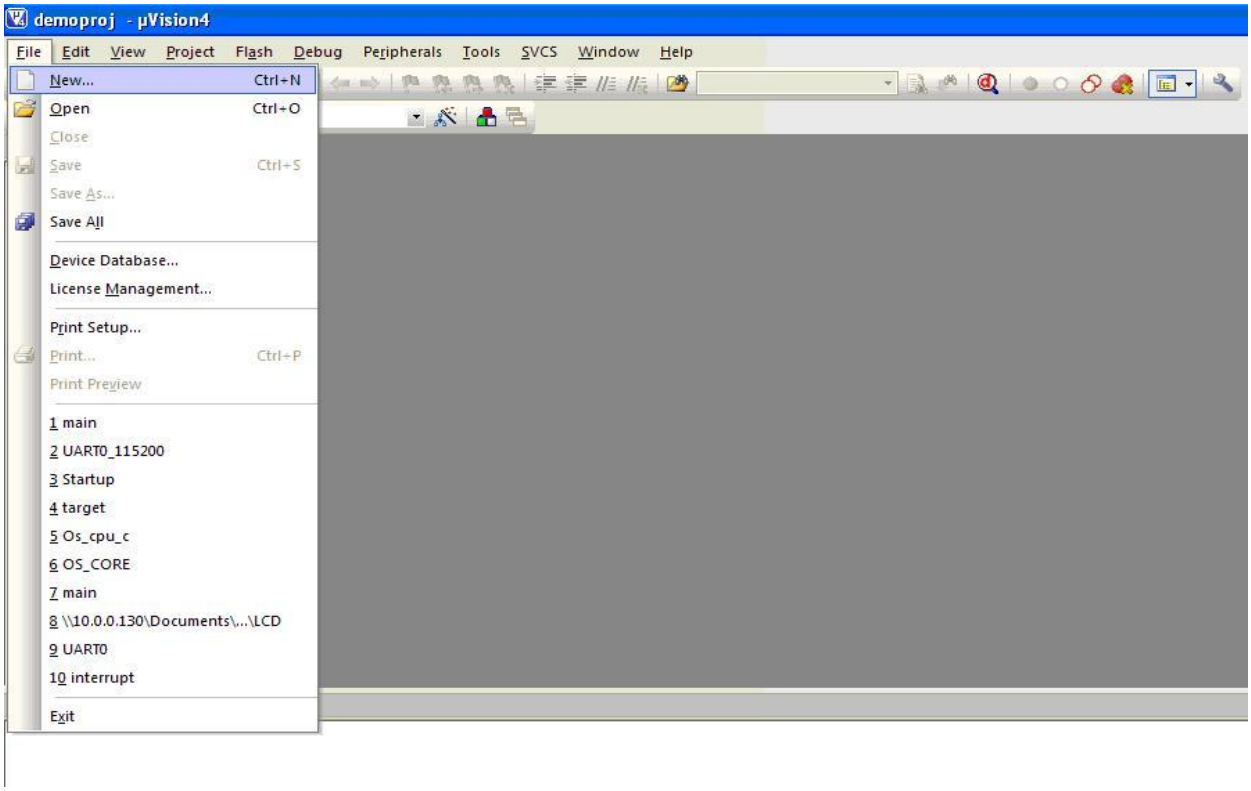
Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP file. Select YES.



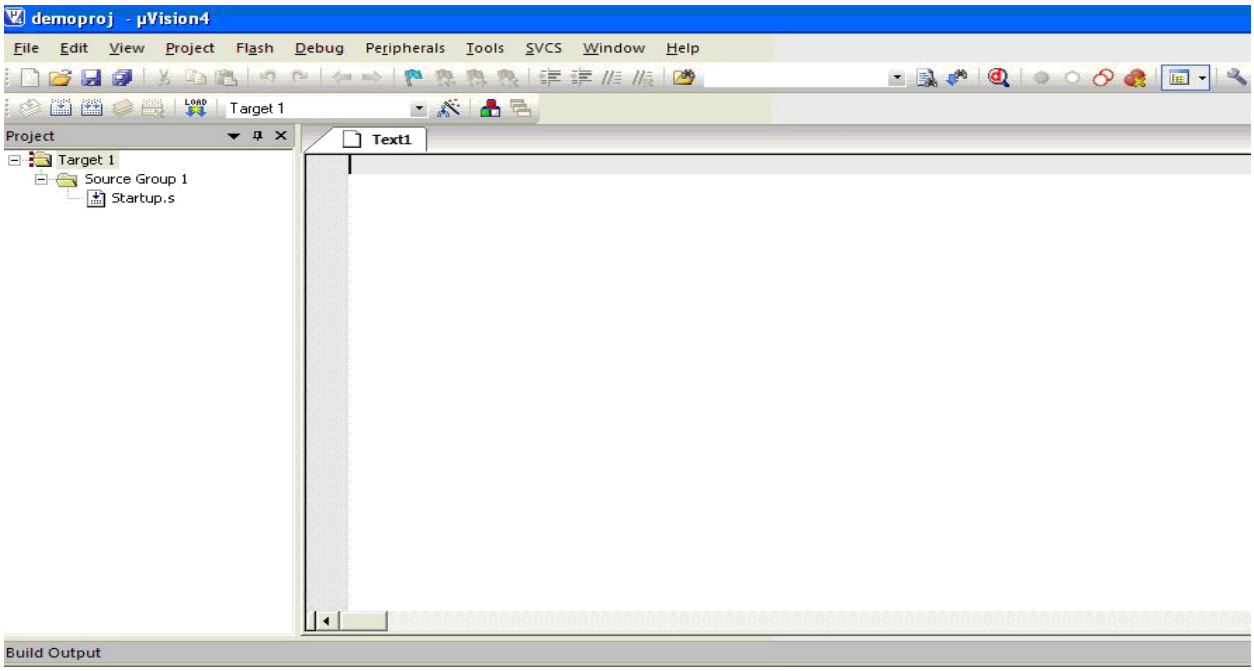
Step 9: A target is created and startup file is added to your project target and is shown below.



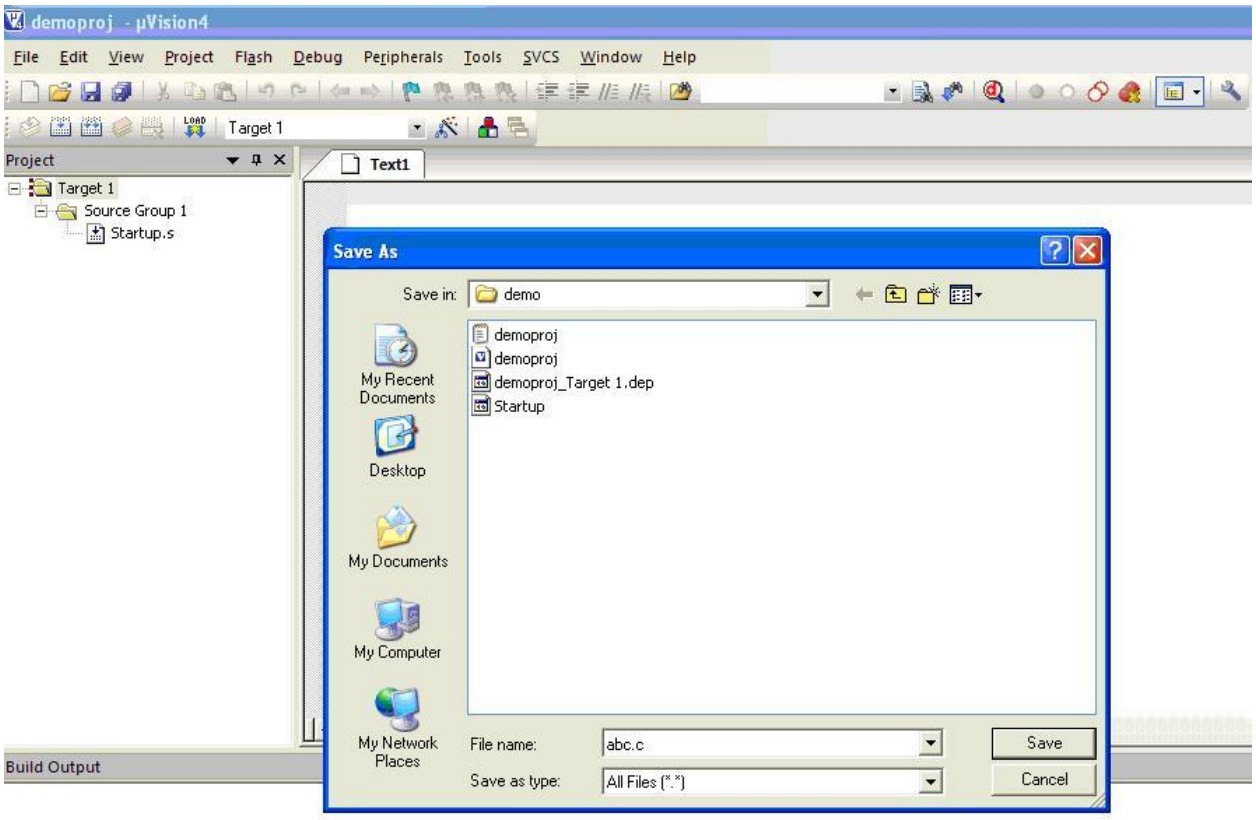
Step 10: To write your project code select a new file from FILE menu bar or click on  icon.



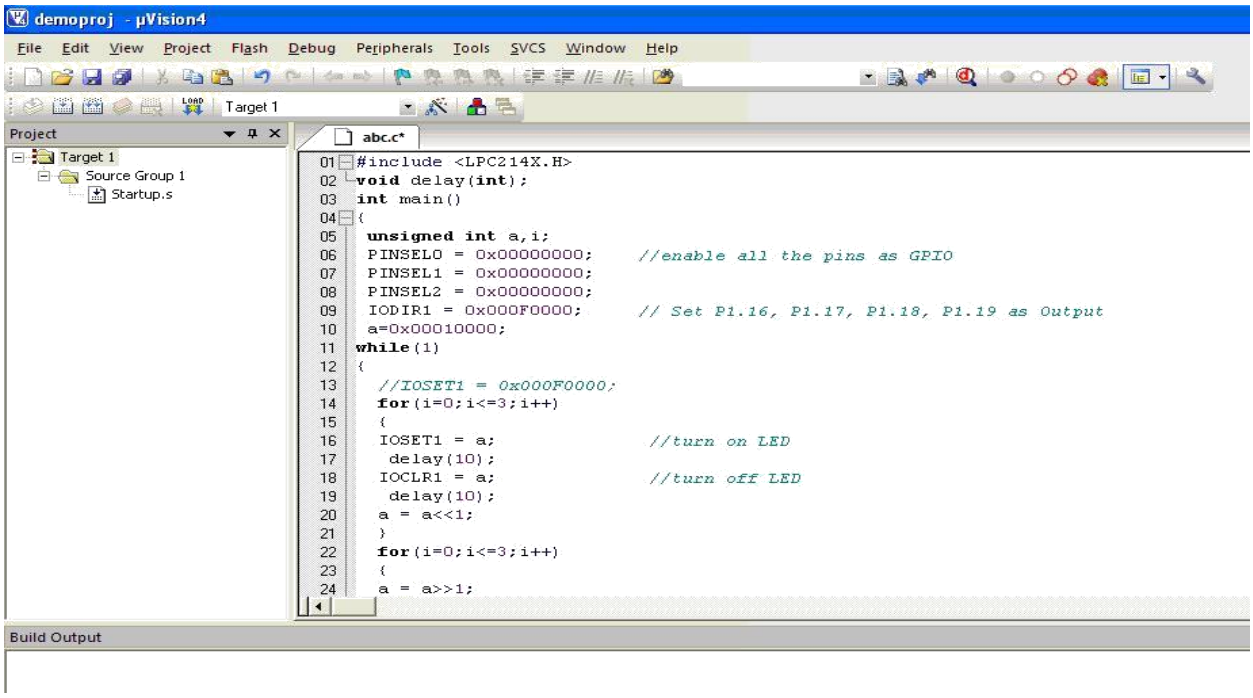
Step 11: It will display some text editor, to save that file select SAVE option from FILE menu bar.



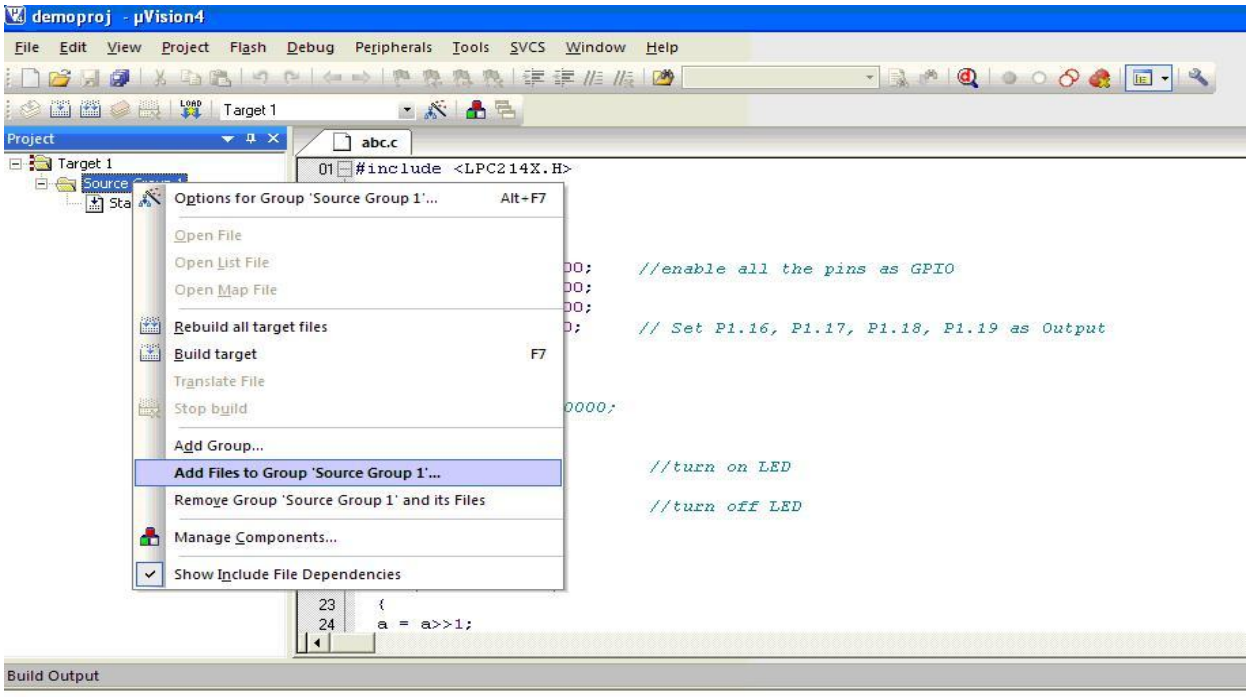
Step 12: By giving a file name with extension .C for c files and save it.



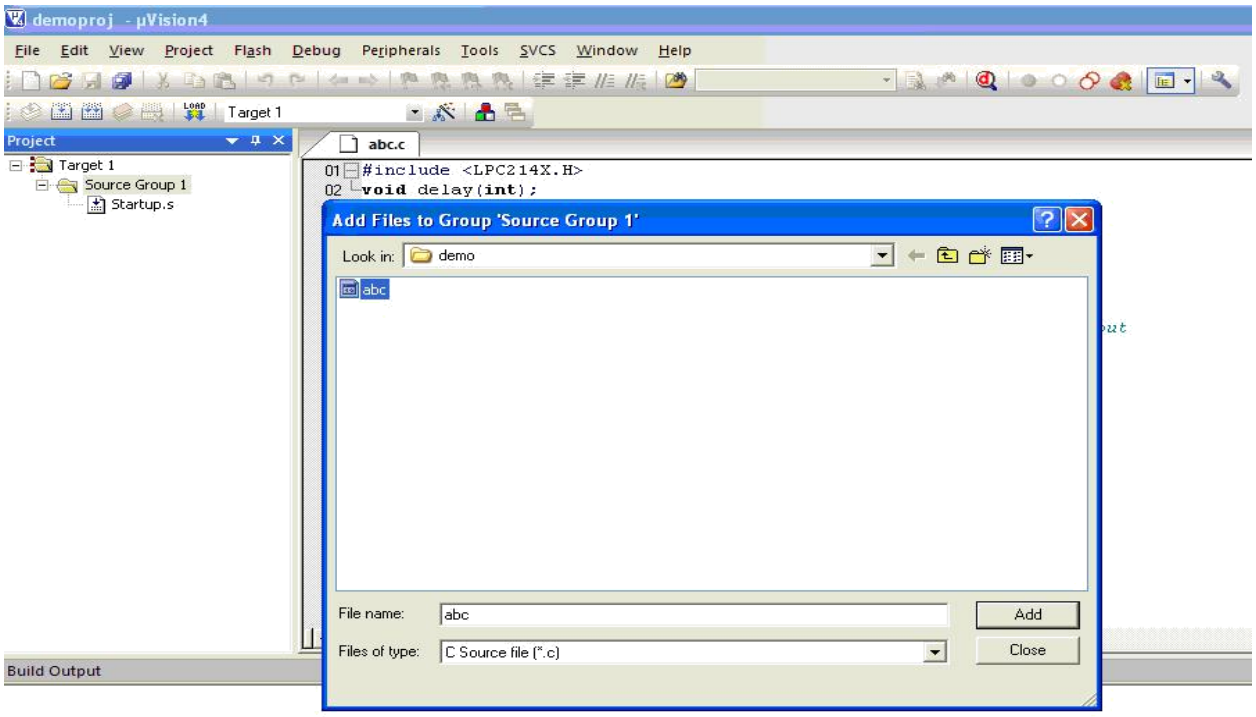
Step 13: Write the code of your project and save it.



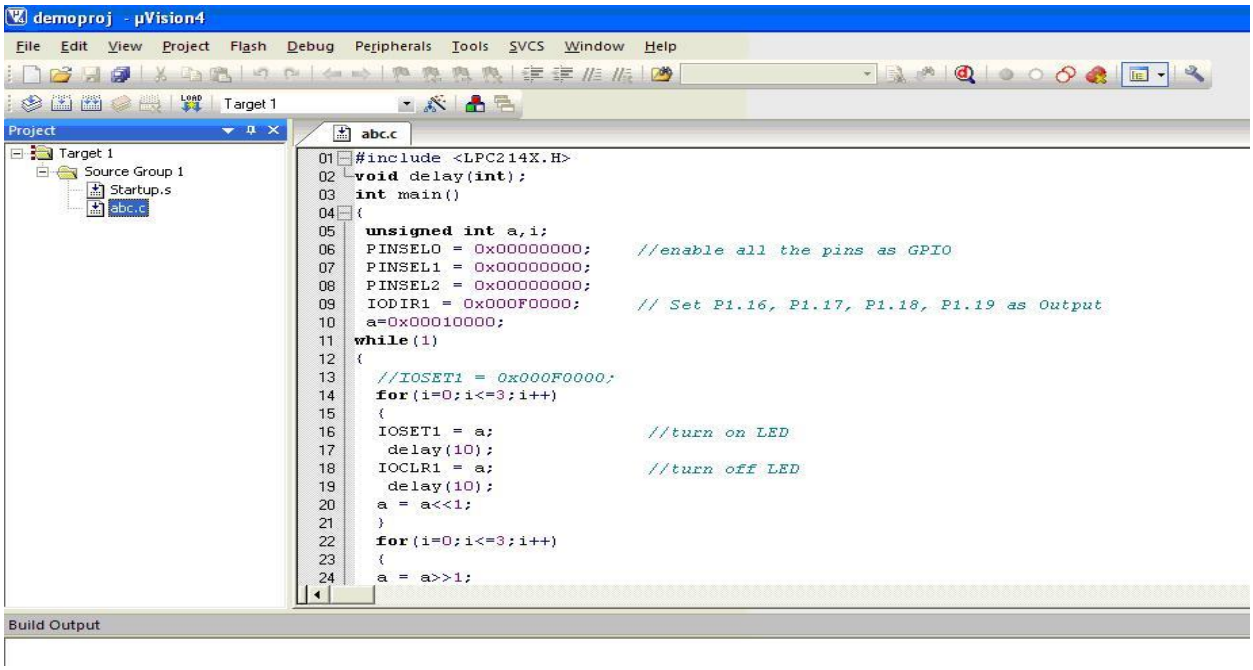
Step 14: To add our c file to target give a right click on Source Group, choose "ADD files to Group "option.



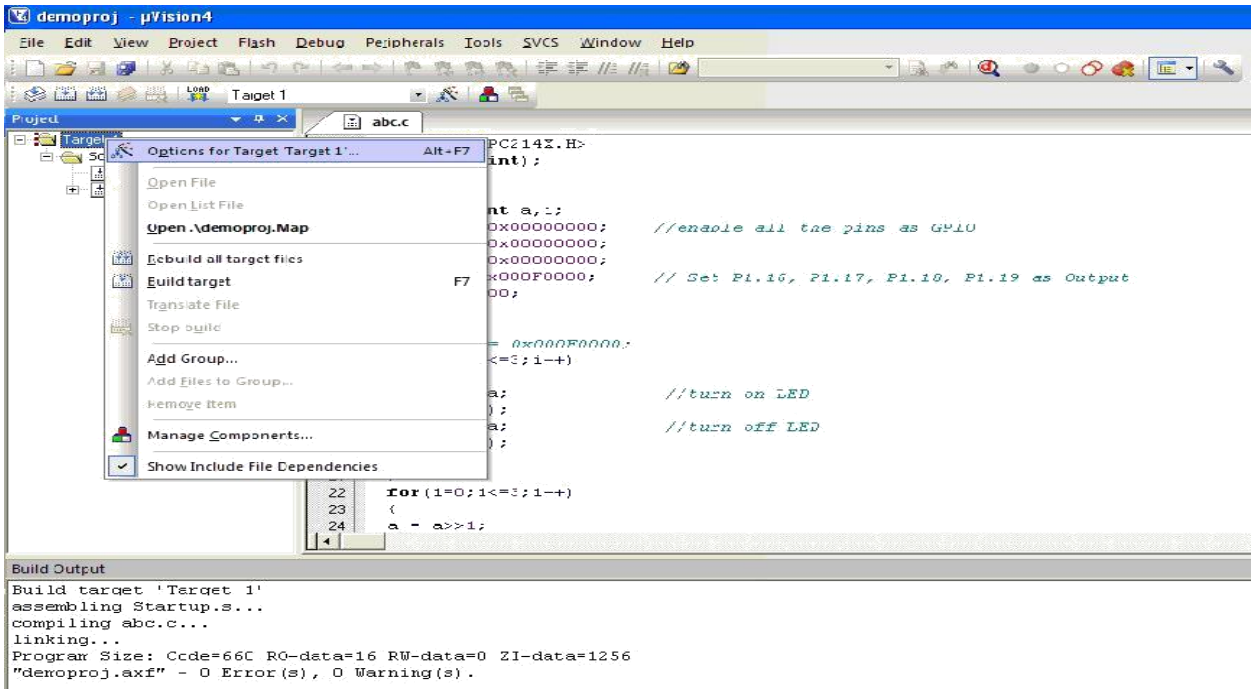
Step 15: It displays some window there select the file you have to add and click on ADD option.



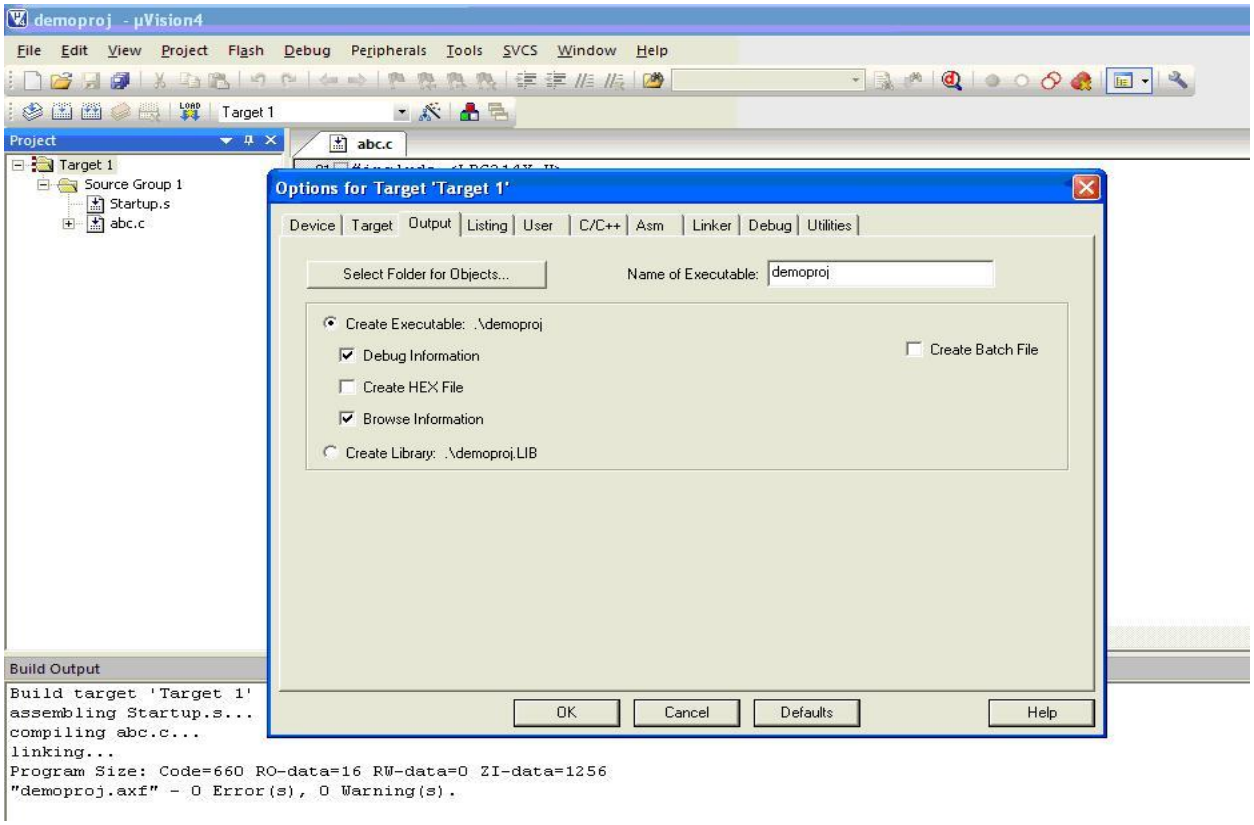
Step 16: The file will be added to our target and it shows in the project window.



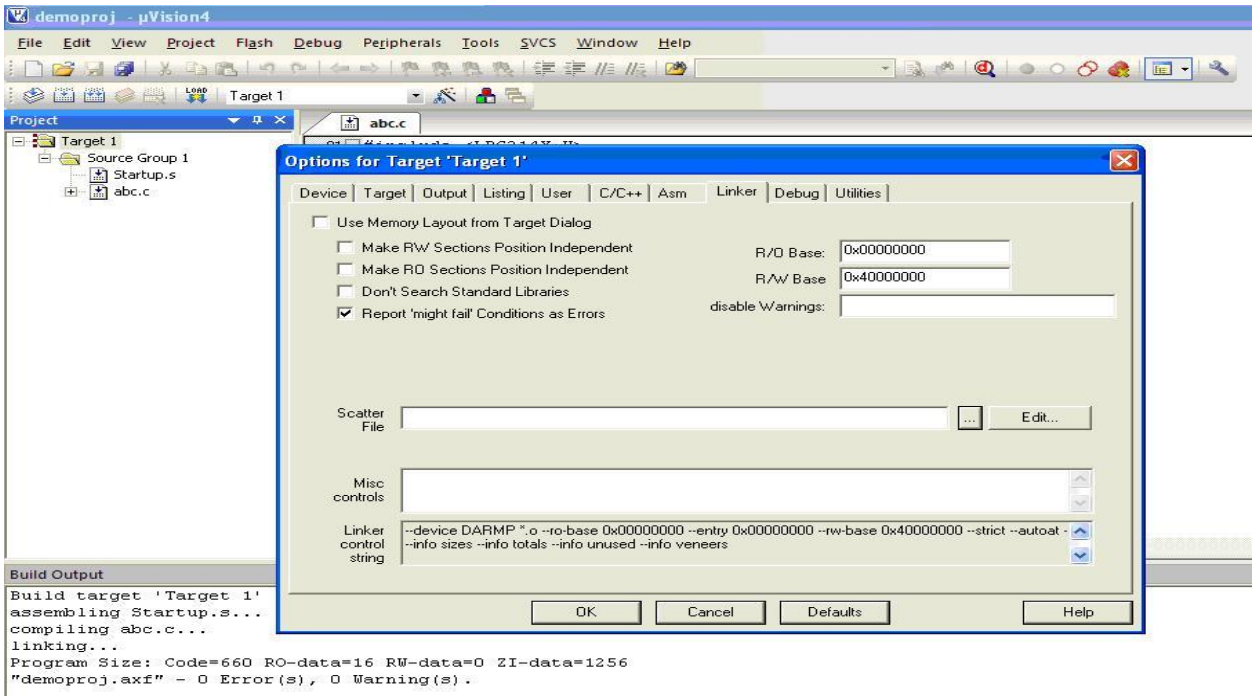
Step 17: Now give a right click on target in the project window and select "Options for Target".



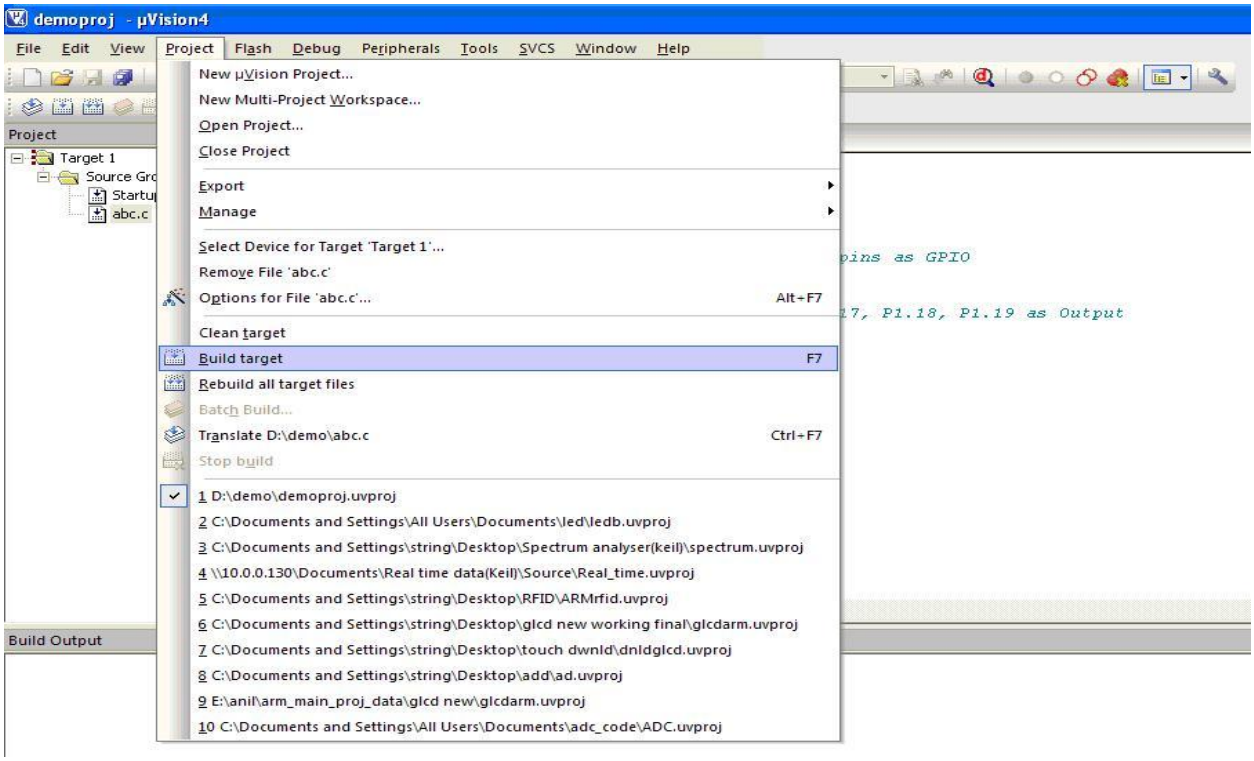
Step 18: It will show some window, in that go to output option and choose Create Hex file option by selecting that box.



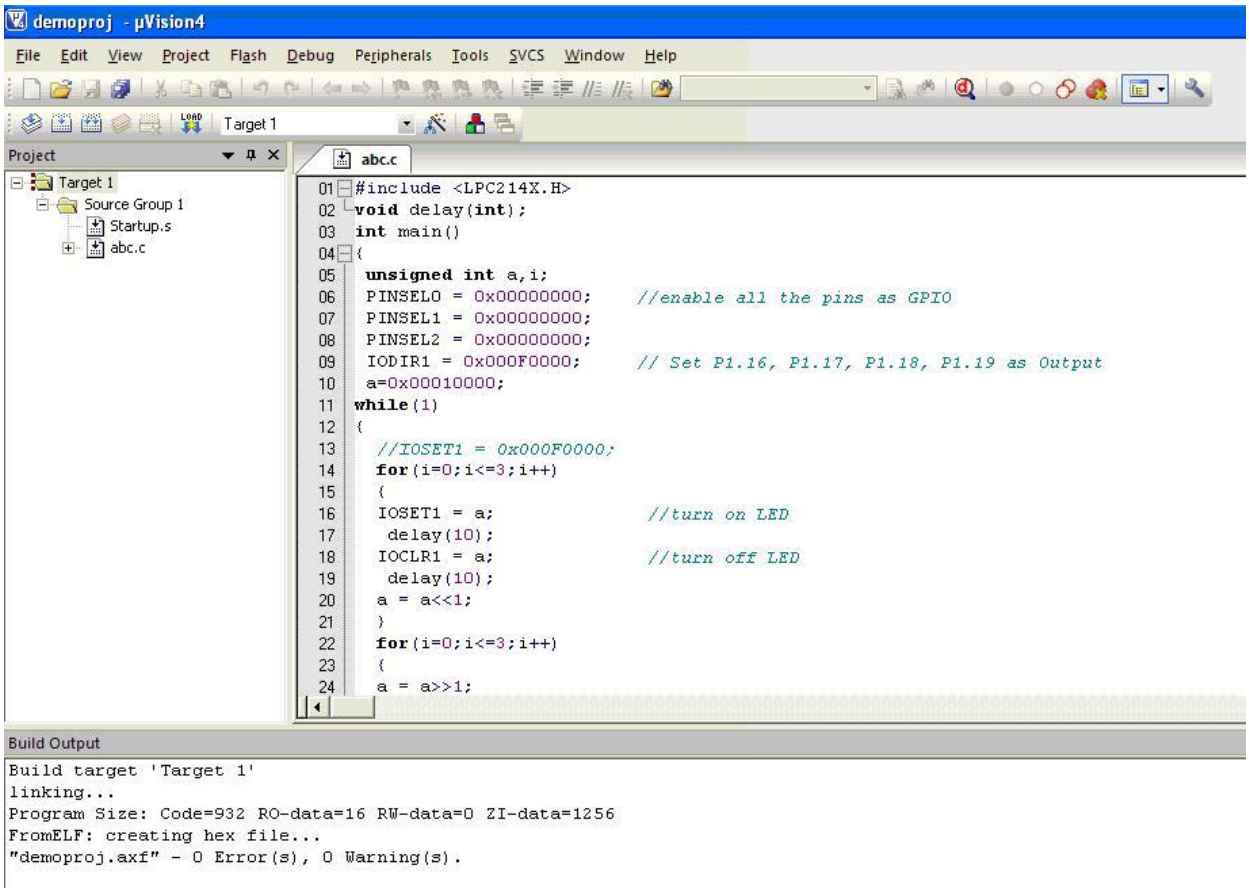
Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.



Step 20: Now to Compile your project go to Project select Build Target option or press F7 or click on (Build) icon or (Build All) icon.



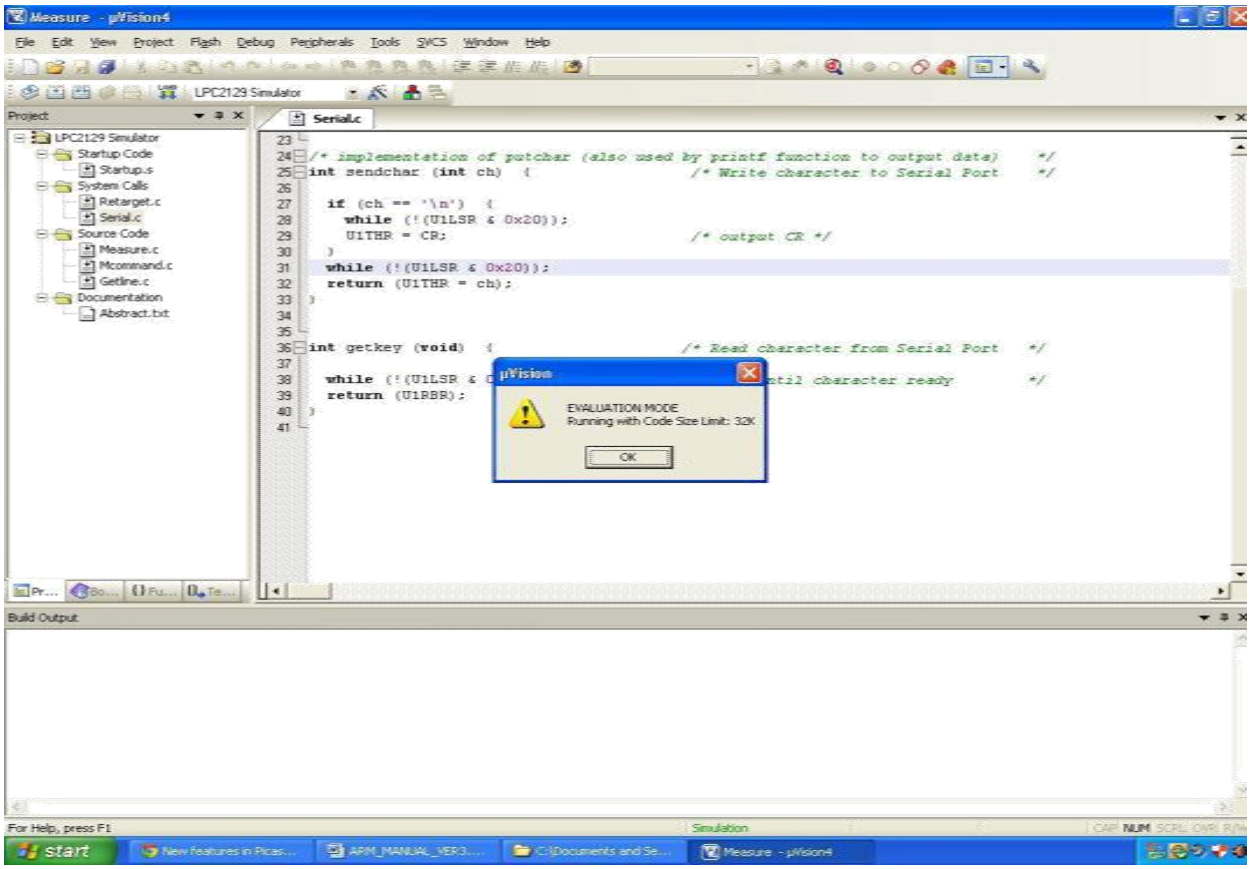
Step 21: In the build OUT PUT window you can see the errors and warnings if there exits. And here Your project Hex file will be created.



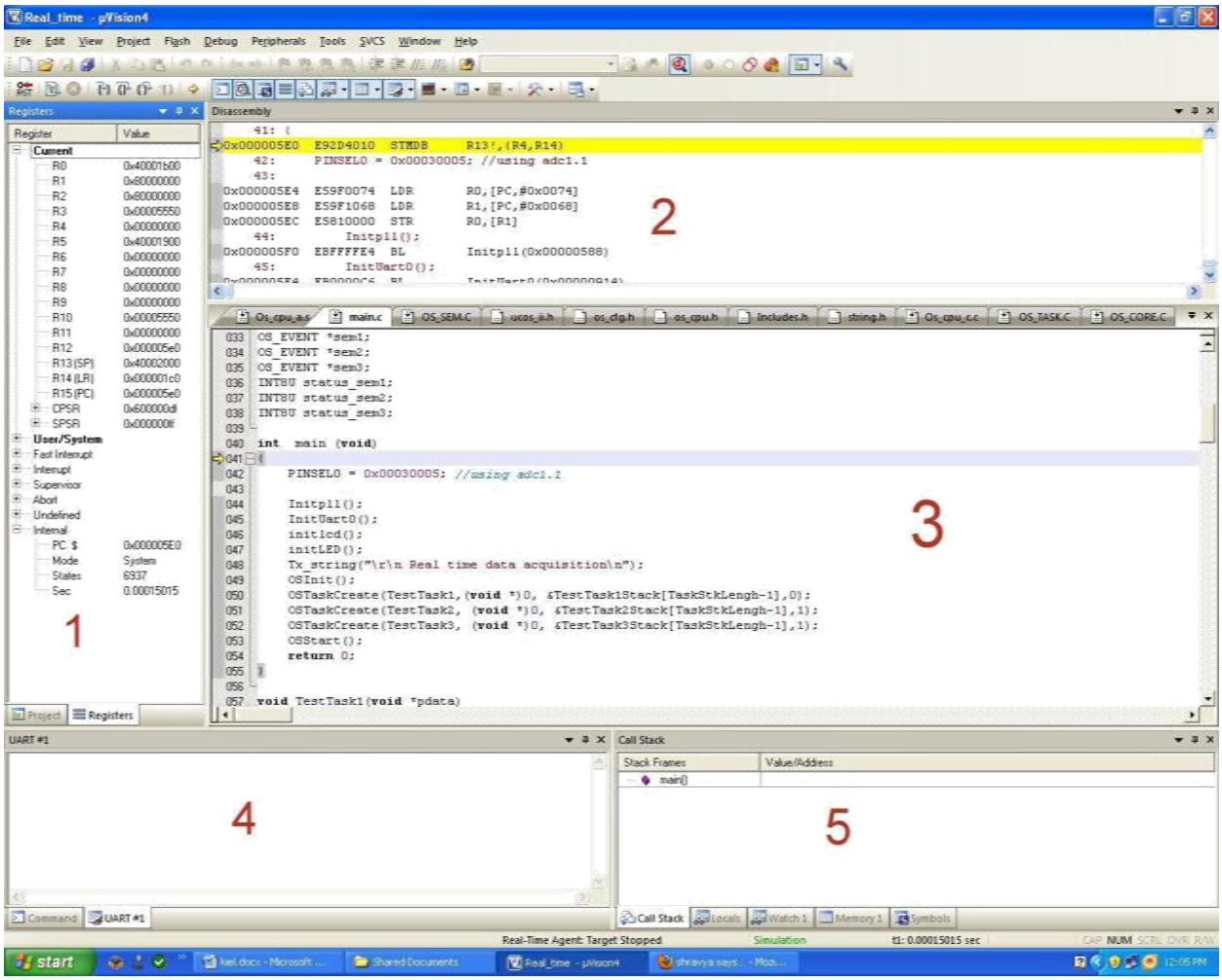
DEBUGGER:

We can debug and also simulate the program using debugger option, just click on icon or can also select fron menu bar go to Debug option and in that the first one "Start/Stop debug".

When you click on that it will show you one window as shown below, click ok.



After clicking OK it will enter simulation screen as show below,



The above figure consists of 5 windows as shown in figure

Window 1: This window is called as REGISTER window.

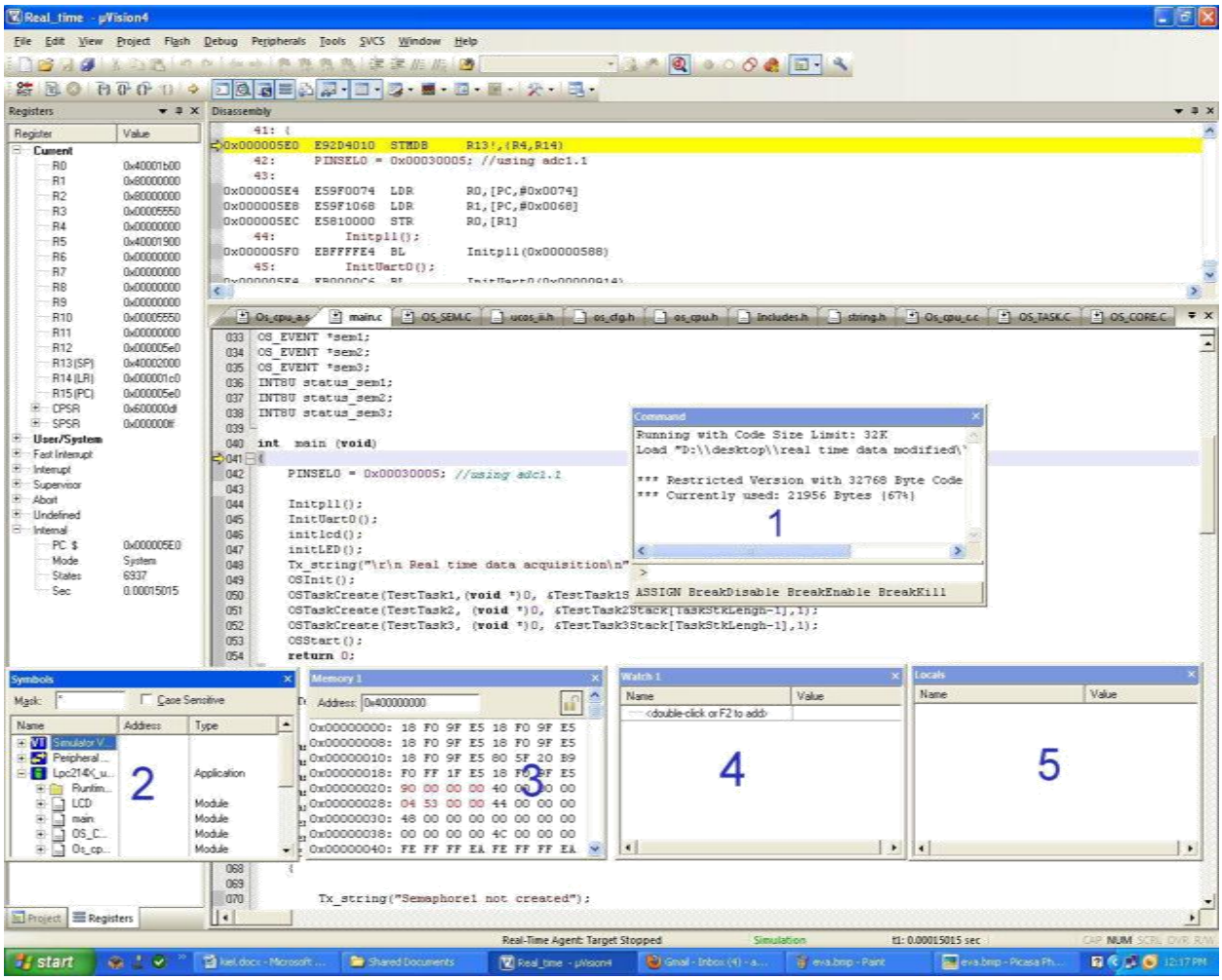
Window 2: This is ASSEMBLY window, where we can see assembly code for our corresponding C code.

Window 3: This is Editor Window where we can write our C code or Edit it.

Window 4: This is serial communication output window where we can see any output from ARM on serial terminal.

Window 5: This window is Call Stack where we can see the local variables updating which is store in stack memory.

These are the default windows we can see other windows which is present in debugger are shown in figure below



Window 1 is Command window which will display any errors or warning when we build the file.

Window 2 is Symbol window where we can see each and every variables of .c files like lcd.c, main.c etc

Window 3 is a memory window where we can see values stored in a particular memory.

Window 4 is Watch window where we can see run time changes in the variables similar to symbol window but in symbol window we cannot see Global variables, in this window we can all variables.

Window 5 is a local window in this we can see function's local variables which are having local scope.

Execution of program:

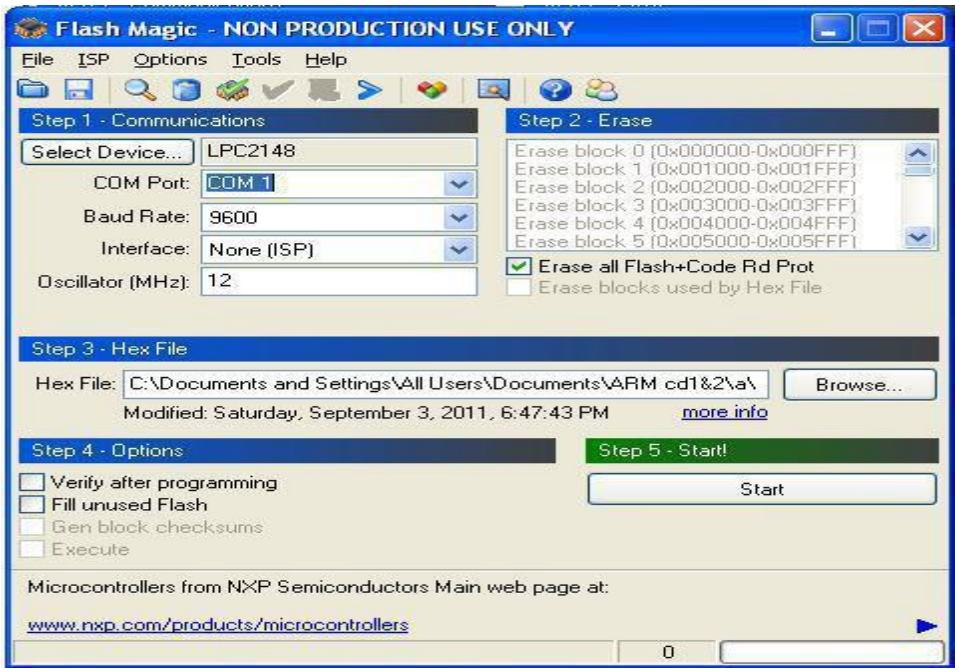
For executing the program we need to press this Icon or we can directly press F5 function key from keyboard. This execution will be done all at a time we can also execute the program step wise by pressing F11 or clicking on this icon is for "step one line", for step over the current line click on icon or press F10 function key, to step out of the current function press ctrl+F11 or click on icon.

Downloading Hex file onto ARM microcontroller:

Flash Magic Tool

To program the Microcontroller, Flash Magic tool is used. Generally, the microcontroller is in one of the two modes. One is RUN mode and the other is PROGRAMMING mode. In RUN mode microcontroller executes the application present in the microcontroller flash memory. In PROGRAMMING mode, microcontroller programs its flash memory in synchronization with Flash Magic.

To enter in to the programming mode, Hold down SW2(isp) and SW3(reset), then release SW3 first and finally SW2 . To enter in to Run Mode, press the SW3(reset) after programming is over.



Downloading Hex file onto microcontroller:

To program the flash memory, first keep the microcontroller in PROGRAMMING mode. Launch the Flash Magic Tool. Select the COM1, Baud rate as 19200, device as LPC2148; Oscillator Freq (MHz) as 12, in Communication block. Select the box erase all Flash + Code Rd Prot in Erase block. Select the box Verify after programming in Options Block. Select the hex file in Hex File block. Hold down SW2 (isp) and SW3 (reset), then release SW3 first and finally SW2 .Then click Start Button in Start Block.

Experiments: 2 & 3

Arithmetic and Branching Operations

Aim:

To verify arithmetic and branching operations by writing an assembly language program.

Tools:

(1)PC

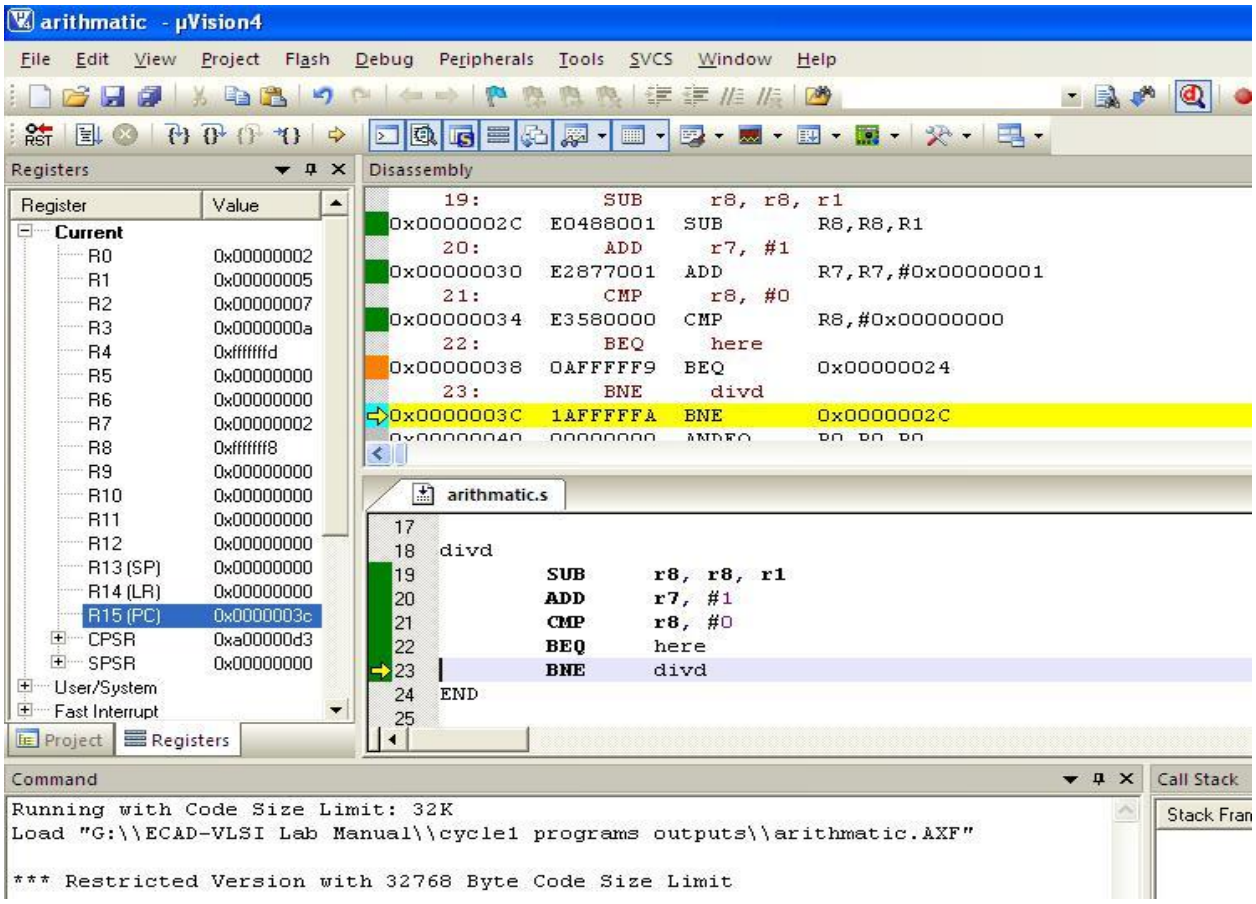
(2)Keil Microvision4

Program:

```
AREA   Arithmetic, CODE, READONLY
ENTRY
start
      MOV      r0, #2
      MOV      r1, #5
      ADD      r2, r0, r1
      MUL      r3, r0, r1
      SUBGT    r4, r0, r1
      SUBLT    r5, r1, r0
      CMP      r1, #0
      MOV      r8, r0
      BNE      divd

here   MOV       r8, r7

stop B stop

divd
      SUB    r8, r8, r1
```

```
        ADD     r7, #1
        CMP     r8, #0
        BEQ     here
        BNE     divd
END
```

Output:



Result:

Experiment: 4

Configuration and Controlling Of GPIO Ports

Aim:

To write an assembly level program for Configuring and Controlling of GPIO ports

Tools Used:

(1) PC

(2)Keil microvision4

Program:

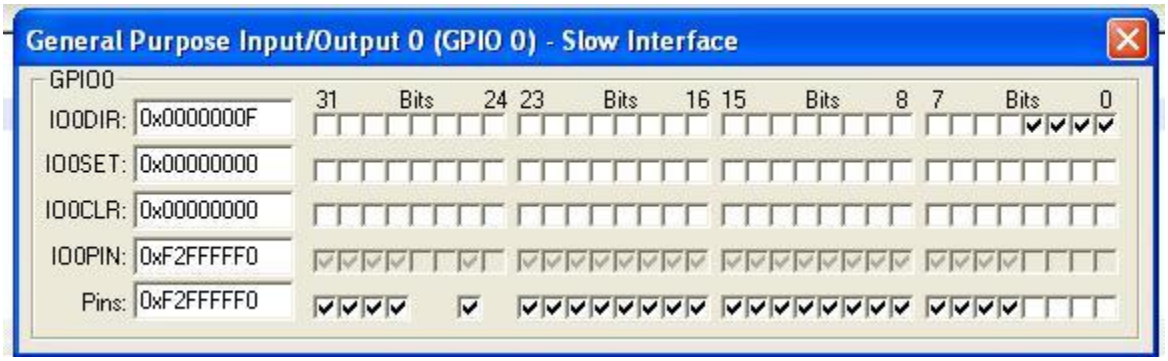
```
AREA myprogram, CODE, READONLY
ENTRY
PINSEL0 EQU 0xE002C000
PINSEL1 EQU 0xE002C004
PINSEL2 EQU 0xE002C014
IODIR0 EQU 0xE0028008
IOSET0 EQU 0xE0028004
IOCLR0 EQU 0xE002800C
IODIR1 EQU 0xE0028018
IOSET1 EQU 0xE0028014
IOCLR1 EQU 0xE002801c

MOV r0, #0
LDR r1, =PINSEL0
STR r0, [r1]

MOV r0, #0
LDR r1, =PINSEL1
STR r0, [r1]
```

```
MOV r0, #0
LDR r1, =PINSEL2
STR r0, [r1]

MOV r0, #0xFFFFFFFF
LDR r1, =IODIR0
STR r0, [r1]

MOV r0, #0xFFFFFFFF
LDR r1, =IODIR1
STR r0, [r1]
stop

MOV r0, #0xFFFFFFFF
LDR r1, =IOSET0
STR r0, [r1]

MOV r0, #0XFFFFFFFF
LDR r1, =IOSET1
STR r0, [r1]

MOV r3, #0
BL DELAY
MOV r3, #0
MOV r0, #0xFFFFFFFF
LDR r1, =IOCLR0
STR r0, [r1]

MOV r0, #0XFFFFFFFF
LDR r1, =IOCLR1
STR r0, [r1]
```

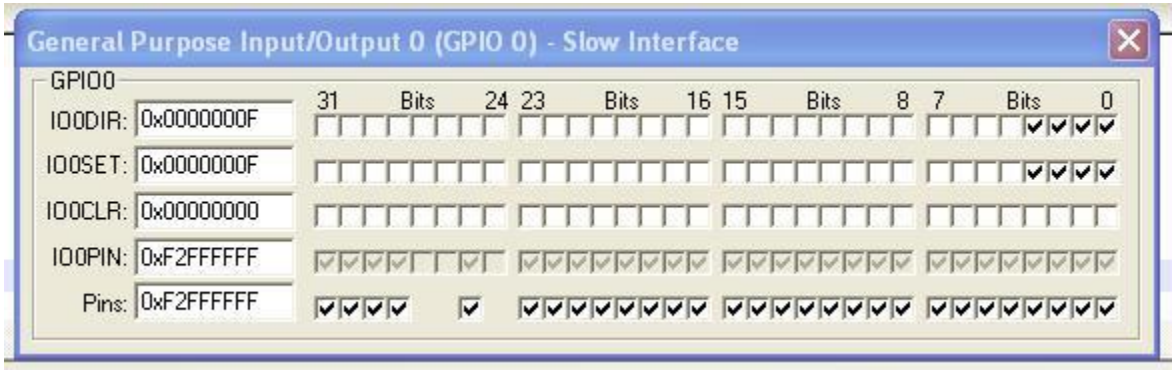
```
MOV r3, #0
BL DELAY

MOV r3, #0
B stop

DELAY
        LDR r5, =0xFFFFFF
        ;MOV r4, #255
HERE
        CMP R3, R5
;HERE_1
        ;CMP r4, r6
        ;ADD r6, #1
;BNE HERE_1
ADD R3, #1
BNE HERE
BX LR
END
```

Output:

Set:



Clear:



Results:

Experiment: 5 & 6

Timer Subroutine and External Value input in ARM7

Aim:

To write program to demonstrate time delay using built in timer/counter features on IDE environment.

Tools:

(1)PC

(2)keil microvision4

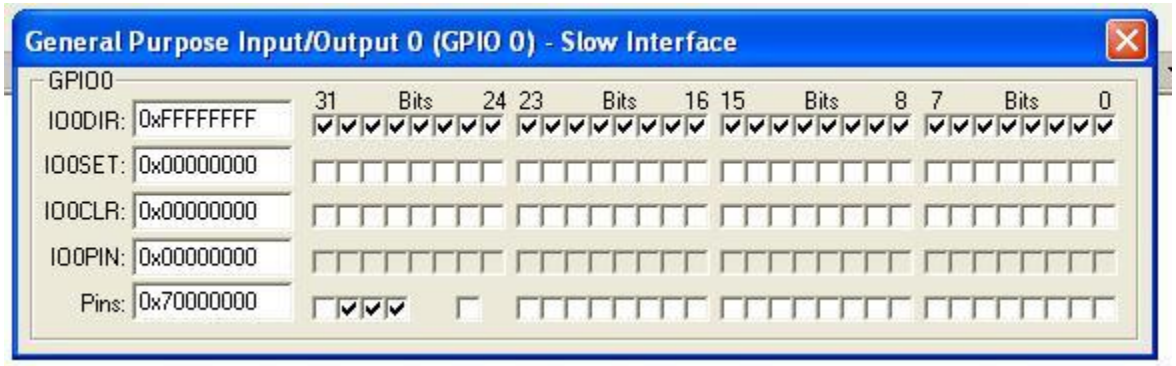
Program:

```
#include <LPC214X.H>
void timer_delay(void);
int main()
{
        PINSEL0 = 0x00000000;
        IODIR0 = 0xFFFFFFFF;
        while(1)
        {
                IO0SET = 0x55;
                timer_delay();
                IO0CLR = 0x55;
                timer_delay();
                IO0SET = 0xAA;
                timer_delay();
                IO0CLR = 0xAA;
                timer_delay();
        }
}
```

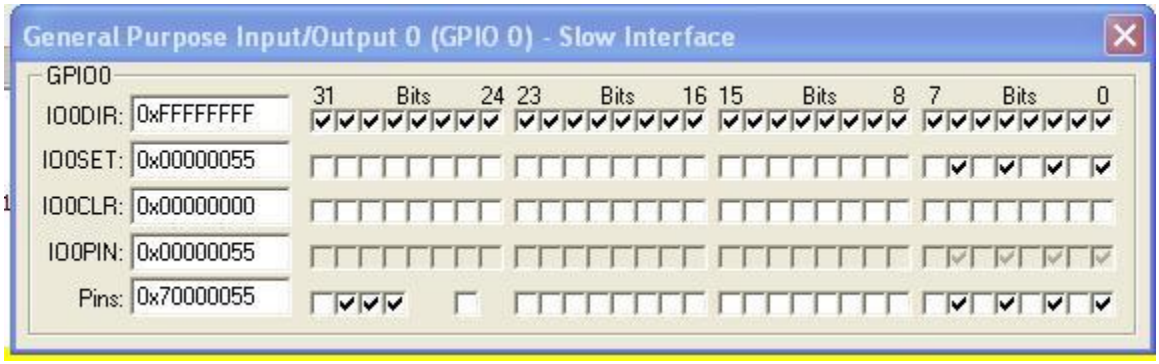
```
void timer_delay()
{
        T0TCR = 0x02;
        T0TC = 0;
        T0PC = 0;
        T0PR = 15000;
        T0MR0 = 1000;
        T0MCR = 0x02;
        T0TCR = 1;
while(T0TC != T0MR0);
}
```

Output:

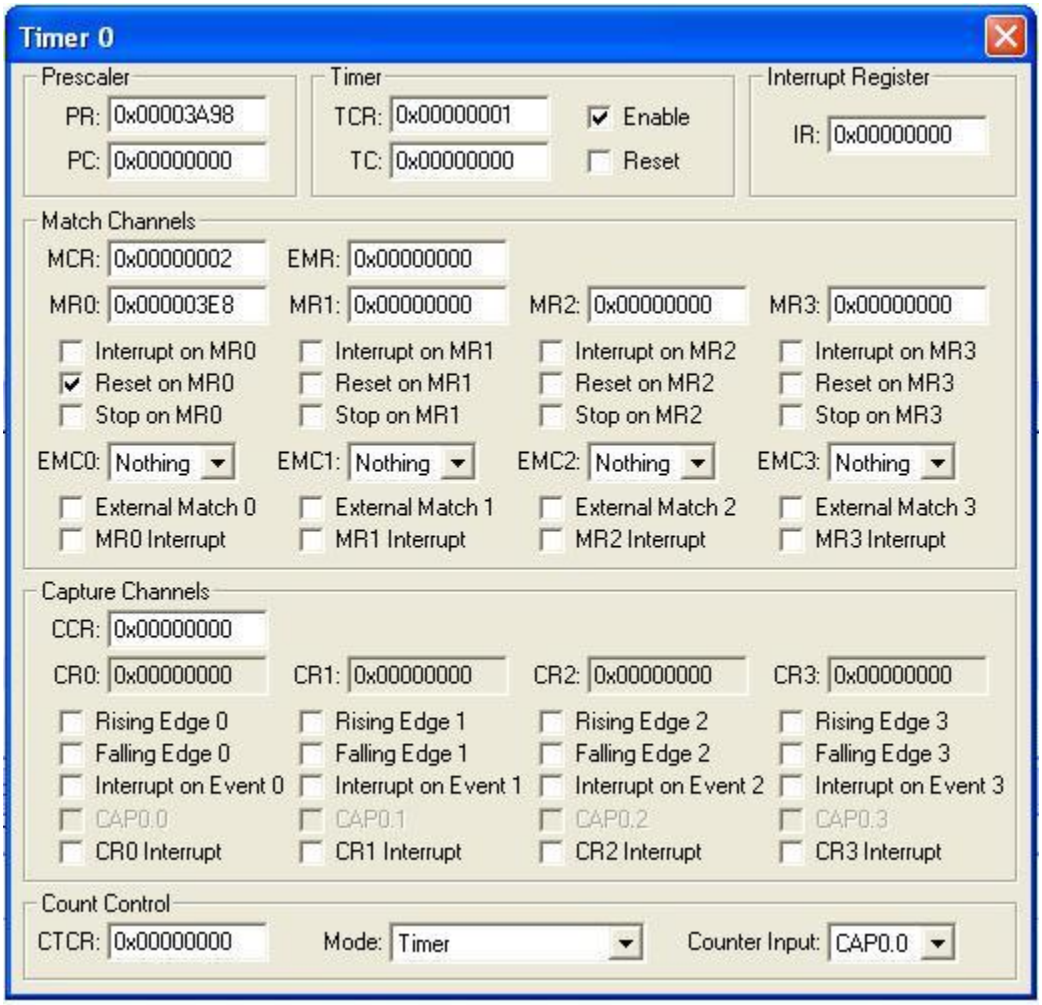
Set:



Clear:



Timer0:



Result:

Experiment: 7

Simple Interrupt Handler with Time Delay

Aim:

To write a program to demonstrate a simple interrupt handler and setting up timer features on IDE Environment.

Tools:

(1)PC

(2)Keil microvision 4

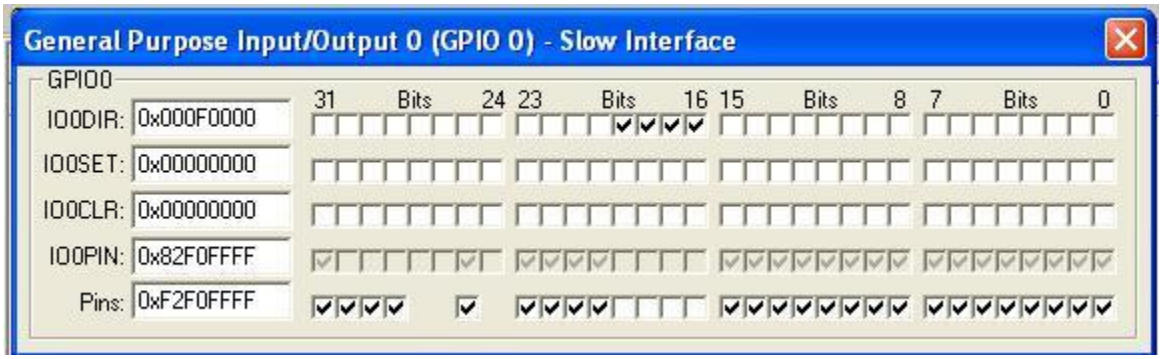
Program:

```
#include <LPC214X.H>

void Timer_0_Init(void);
void Timer_0_ISR(void) __irq;
unsigned char g_flag = 0;
int main()
{
        PINSEL0 = 0x0000000;
        PINSEL1 = 0x0000000;
        PINSEL2 = 0x0000000;
        IO0DIR = 0x000F0000;
        Timer_0_Init();
        T0TCR = 0x01;
   while(1);
 }
void Timer_0_Init(void)
{
        T0TCR = 0x02;
        T0TC = 0x00;
        T0PR = 15000;
        T0PC = 0;
        T0MR0 = 1000;
        T0MCR = 0x03;
```

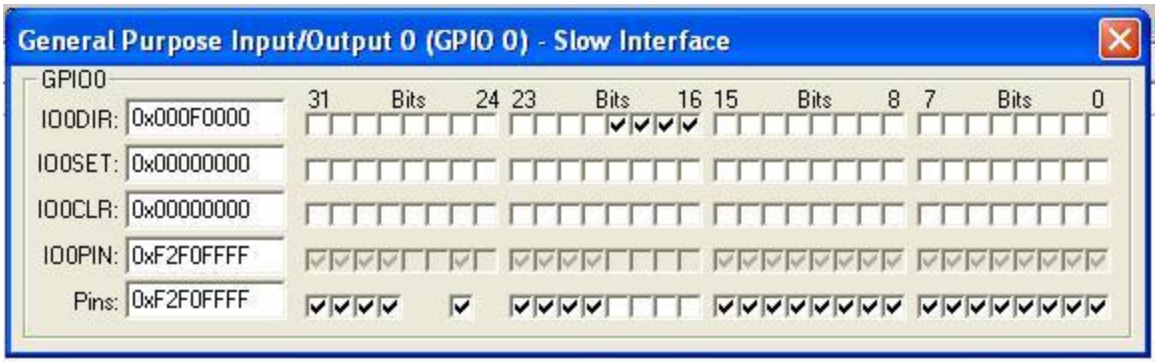
```
        VICIntSelect |= 0x00000000;
        VICVectCntl0 |= 0x20|4;
VICVectAddr0 = (unsigned)Timer_0_ISR;
VICIntEnable |= (1<<4);
 }

void Timer_0_ISR(void) __irq
{
        if(VICIRQStatus & 0x00000010)
        {
if(T0IR == 0x01)
                {
                        T0IR = 0x01;
        if(g_flag == 0)
        {
           IO0SET = 0x00003F00;
           g_flag = 1;
        }
        else
        {
           IO0CLR = 0x000003F00;
           g_flag = 0;
        }
      }

   }
VICVectAddr = 0x00000000;
}
```

Output:

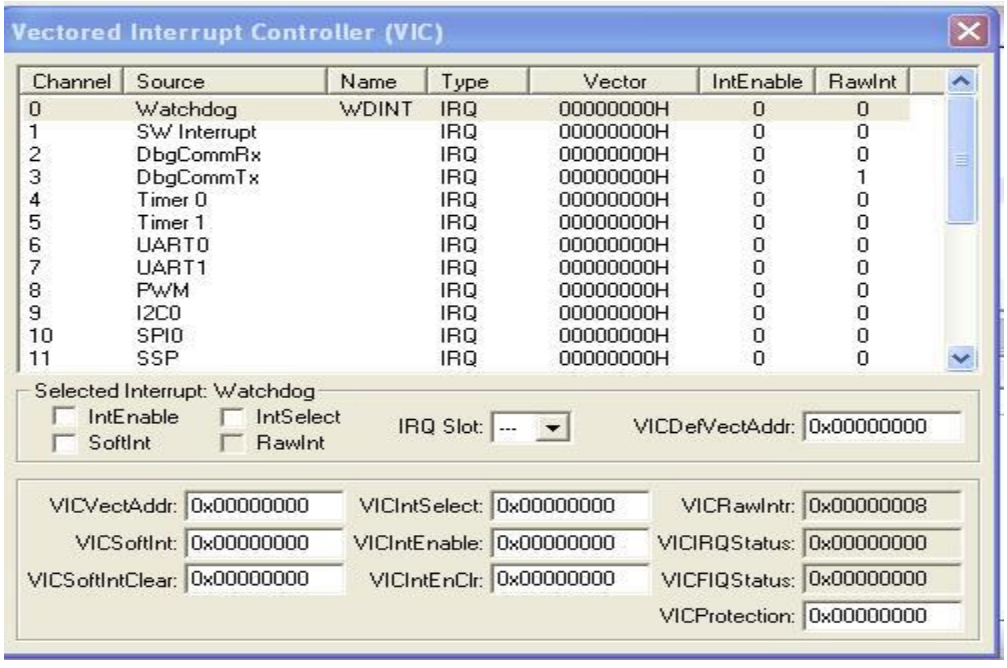
Reset:



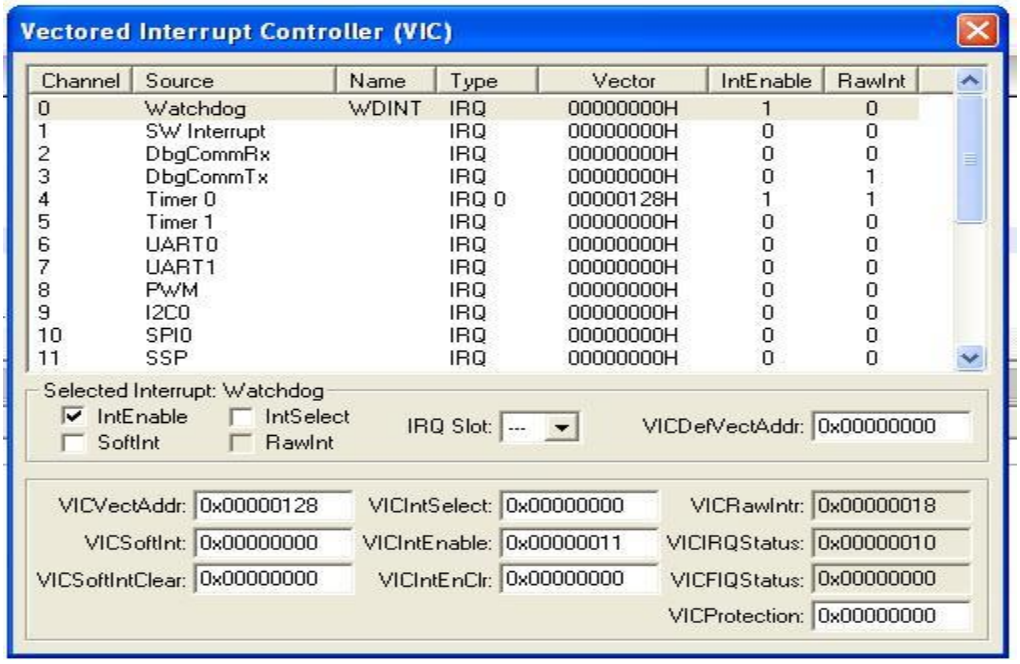
Enable:



VIC reset



VIC enable



Result:

Experiment: 8

Interrupt Handler
Aim:

To Write a Program to demonstrate Setting up interrupt handlers and generate an interrupt when button is pressed.

Tools:

(1) PC

(2) Keil microvison 4

Program:

(a)extinterrupthandler.c

```
#include <LPC214X.H>

#include "Exp 7.h"

void ExtInt_Init(void);

void ExtInt_Service(void)__irq

{

  UART_0_Send_String("External Interrupt has arrived \r\n");

  EXTINT |= 4;

  VICVectAddr = 0;

}

void ExtInt_Init(void)

{

  EXTMODE  |= 4;

  EXTPOLAR = 4;

  VICVectCntl0 = 0x20 | 16;

  VICVectAddr0 = (unsigned long) ExtInt_Service;
  VICIntEnable |= 1<<16;
```

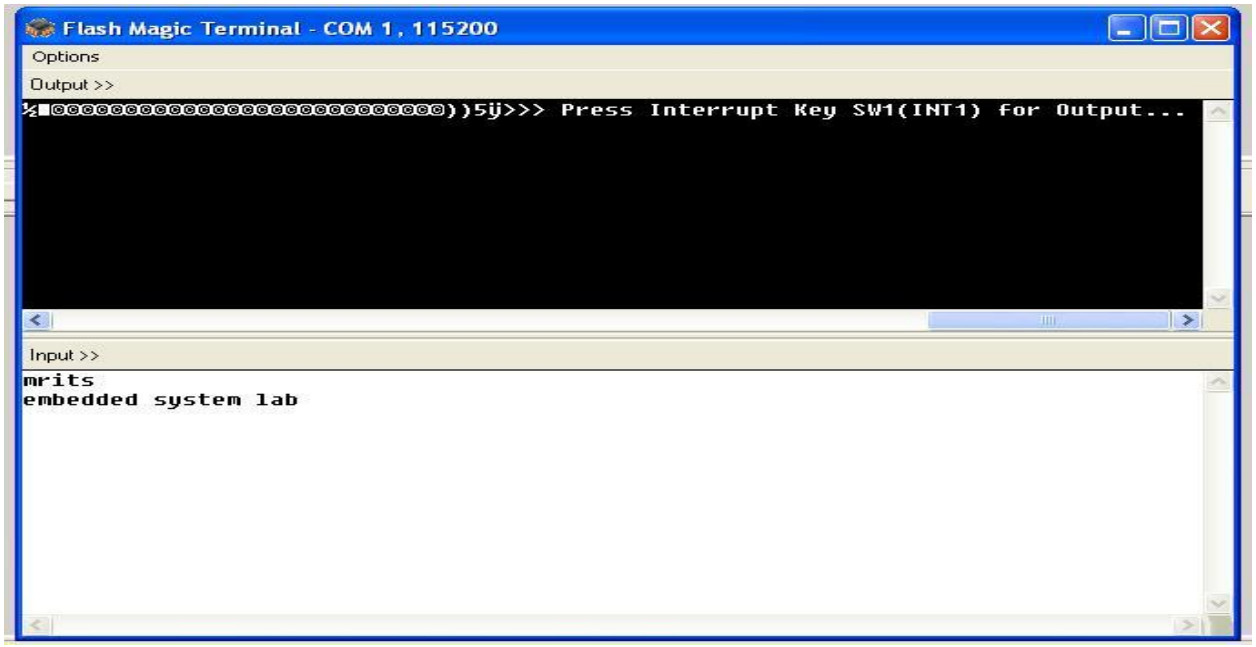
```
}

int main()

{
   PINSEL0  |= 0x80000005;

   PINSEL1 = 0x00000000;

   ExtInt_Init();

   UART_0_Init();

   UART_0_Send_String("* External Interrupt Demo

   *\n\n\r"); DelayMs(100);

   UART_0_Send_String(">>> Press Interrupt Key SW1(INT1) for Output... \n\n\n\r");

   DelayMs(100);

   while(1);
}
```

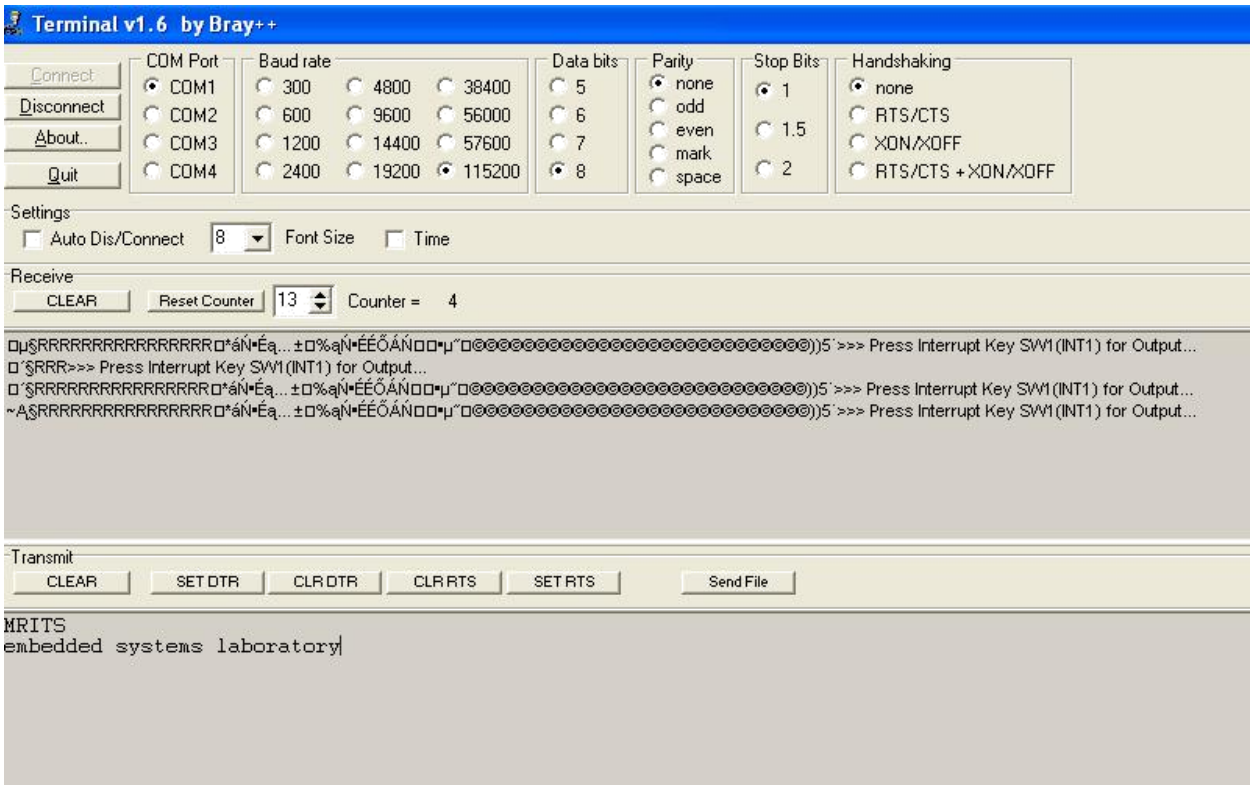
(b)Exp7.h

```
#define UART_0_BPS 115200

#define Fpclk 15000000

void UART_0_Init(void)

{

        unsigned int Baud16;

U0LCR = 0x83;

Baud16 = ((Fpclk / 16)/ UART_0_BPS);

   U0DLL = Baud16 % 256;

   U0DLM = Baud16 / 256;

   U0LCR = 0x03;
```

```
}
void UART_0_Send_Char(unsigned char value)

{
        U0THR = value;

while((U0LSR & 0x40 ) == 0);
}
void UART_0_Send_String(unsigned char *data)
{
        while(*data != '\0')

                UART_0_Send_Char(*data++);
}
void DelayMs(unsigned int count)
{
  unsigned int i,j;
     for(i=0;i<count;i++)
     {
        for(j=0;j<1000;j++);
      }
}
```

Output:




Terminal v1.6 by Bray++
Connect
Disconnect
About..
Quit
COM Port
COM1
COM2
COM3
COM4
Baud rate
300
600
1200
2400
4800
9600
14400
19200
38400
56000
57600
115200
Data bits
5
6
7
8
Parity
none
odd
even
mark
space
Stop Bits
1
1.5
2
Handshaking
none
RTS/CTS
XON/XOFF
RTS/CTS + XON/XOFF
Settings
Auto Dis/Connect
8
Font Size
Time
Receive
CLEAR
Reset Counter
13
Counter = 4
Press Interrupt Key SW1(INT1) for Output...
Transmit
CLEAR
SET DTR
CLR DTR
CLR RTS
SET RTS
Send File
MRITS
embedded systems laboratory

Result:

Experiment: 9

Interface 8-Bit LED and Switch

Aim:
To Write a program to Interface 8-bit LED and Switch Interface

Tools:

(1)PC

(2)Keil Microvision 4

Program:

(a)LED.c

```
#include <LPC214X.H>

#include "Exp8.h"

 unsigned int count =0;

int main()

{

 PINSEL0 = 0x00000000;

 PINSEL1 = 0x00000000;

 IODIR0 = 0x00003F00;

 IODIR1 = 0x00C00000;
 while(1)
 {
  if(SW1)
  {
   count = count+1;
   while(SW1);
//pattern_1_on;
   pattern_off;
  }
  if(count == 1)
   {
          IO0SET=0x00003F00;
          IO1SET=0x00C00000;
```

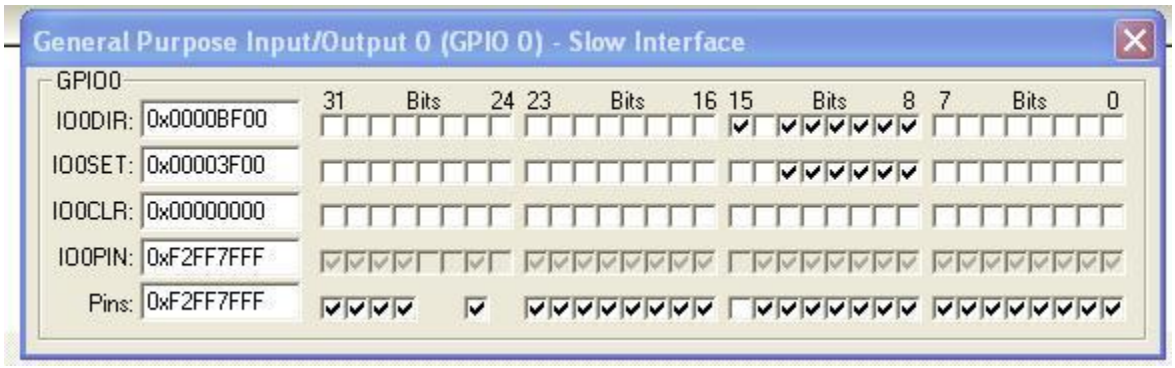
```
  }
 else if(count == 2)
 {
        IO0SET=0x00001200;
        IO1SET=0x00C00000;
 }
 else if(count == 3)
 {
        IO0SET=0x00002D00;

        count=0;
 }
 }
}
```

(b)Exp8.h

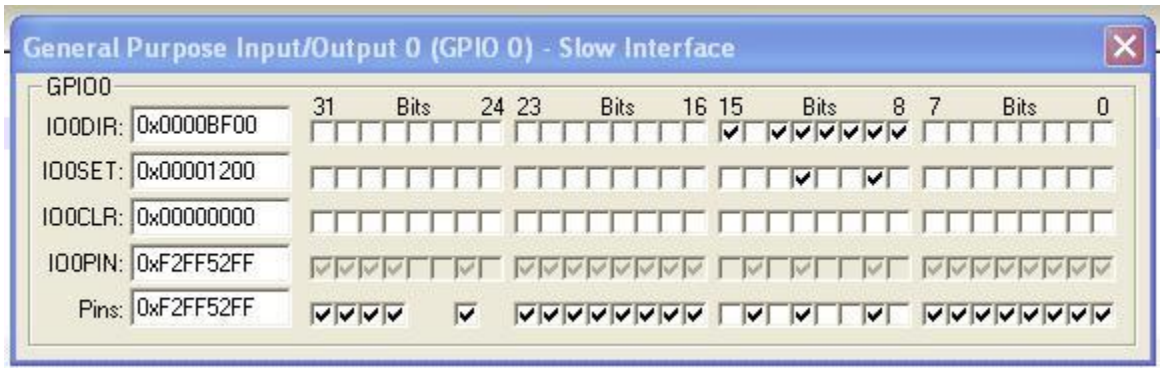
```
#define pattern_1_on IO0SET=0x00003F00;IO1SET=0x00C00000;

#define pattern_1_off

IO0CLR=0x00003F00;IO1CLR=0x00C00000; #define pattern_2_on

IO0SET=0x00001200;IO1SET=0x00C00000; #define pattern_2_off

IO0CLR=0x00001200;IO1CLR=0x00C00000; #define pattern_3_on

IO0SET=0x00002D00; #define pattern_3_off

IO0CLR=0x00002D00;

#define pattern_off IO1CLR = 0x00C00000;IO0CLR=0x00003F00;

#define SW1 (IOPIN0&(1<<15))
```

Output:

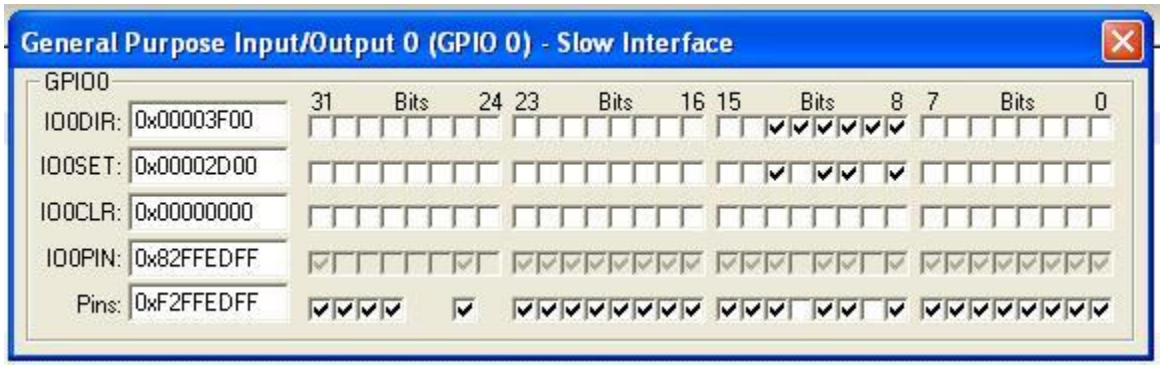
Pattern: 1



Pattern: 2



Pattern: 3



Result:

Experiment: 10

Buzzer Interfacing

Aim:
To write a program to implement Buzzer Interface on IDE environment

Tools:
(1) PC
(2) Keil Microvision 4

Program:

(a)buzzer.c

```
#include <LPC214X.H>

#include "Exp7.h"

int main()

{

PINSEL0 = 0x00000000;

PINSEL2 = 0x00000000;

IODIR0 = 0x00000800;

IODIR1 = 0x000F0000;

while(1)

{

if(SW1==0)

while(SW1 == 0)

BuZZer_ON;

else

BuZZer_OFF;

}

}
```

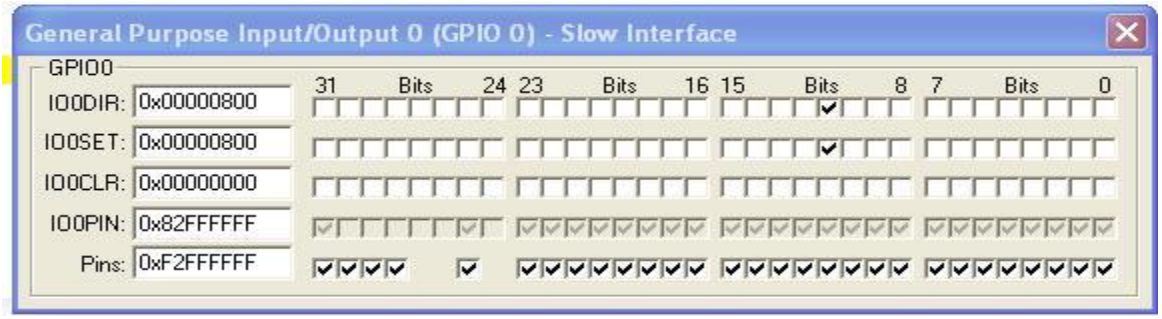
(b)Exp7.h

```
#define BuZZer_OFF IO0SET=(1<<11)

#define BuZZer_ON IO0CLR=(1<<11)

#define SW1 (IO0PIN&(1<<15))
```

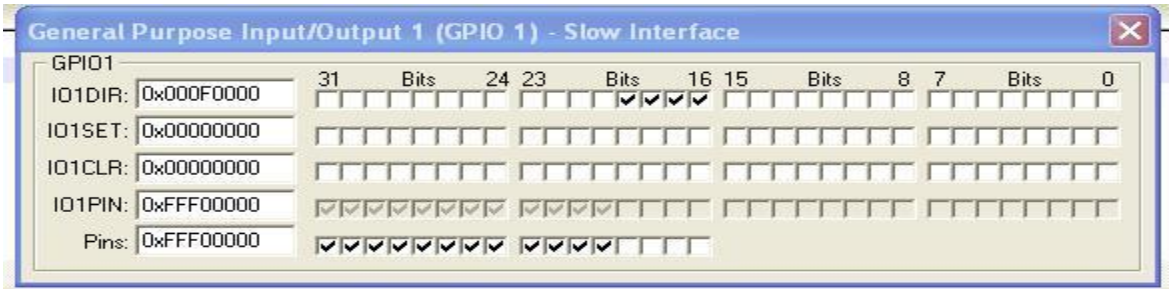
Output:

Buzzer-off:

GPIO0

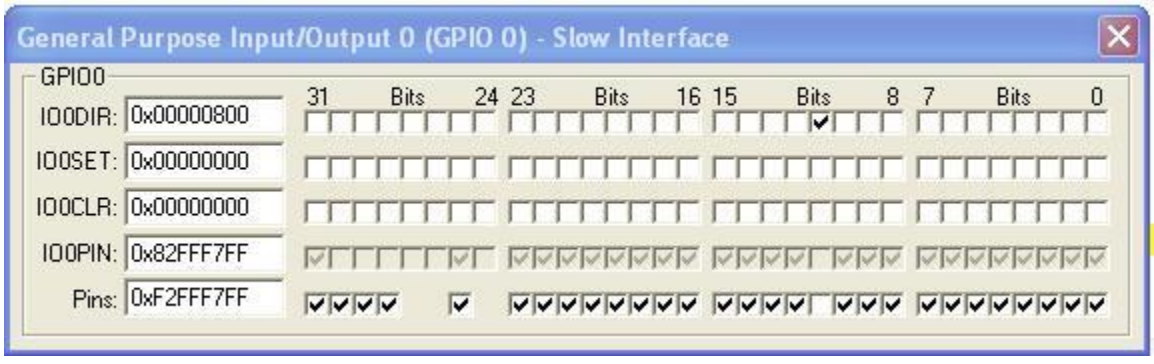


GPIO1

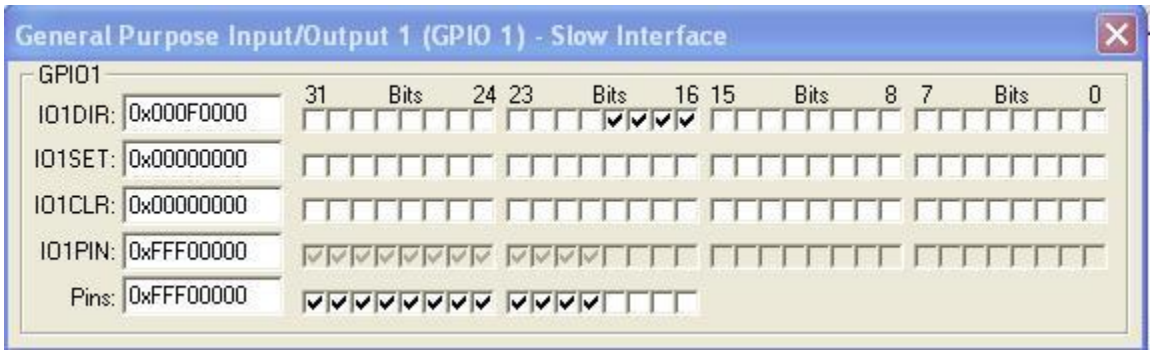


Buzzer-on

GPIO0



GPIO1



Result:

Experiment: 11

LCD Display Interfacing

Aim:

To write a program for Displaying message in a 2 line * 16 characters LCD display and verify the result in debug terminal.

Tools:

(1)PC
(2)Keil Microvision 4

Program:

(a)LCD.c:

```
#include <LPC214X.H>

#include "Exp6.h"

int main(void)

{
        PINSEL0 = 0x00000000;

        PINSEL1 = 0x00000000;

        PINSEL2 = 0x00000000;

         IO1DIR = 0x01C00000;

        IO0DIR = 0x00003C00;

   LCD_init();

        LCD_Command(0x80);

        LCD_Send_String(" UNISTRING TECH ");

        LCD_Delay(10);

        LCD_Command(0xC0);

        LCD_Send_String("SOLUTION PVT LTD");

        while(1);

}
```

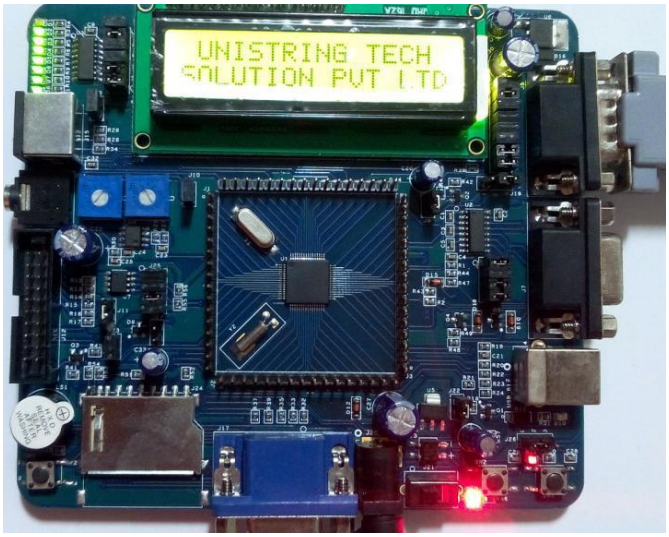
(b)Exp6.h

```
#define LCD_RS_DATA IO1SET=(1<<24)

#define LCD_RS_CMD IO1CLR=(1<<24)

#defineLCD_WR IO1CLR=(1<<23)

#defineLCD_RD IO1SET=(1<<23)

#define LCD_EN_HI IO1SET=(1<<22)

#defineLCD_EN_LOW IO1CLR=(1<<22)


void LCD_Delay(unsigned int s)

{

        unsigned int  i = 0;

        while(s-->0)

        {
                i = 0;

                while(i++<60000);

        }

}

void LCD_Data(unsigned int data)

{

        unsigned char temp;

        LCD_RS_DATA;

        LCD_WR;

        temp = data;
```

```
        IO0SET = (temp & 0xF0)<<6;
        LCD_EN_HI;
        LCD_EN_LOW;
        IO0CLR = (temp & 0xF0)<<6;
        IO0SET = (data & 0x0F)<<10;
        LCD_EN_HI;
        LCD_EN_LOW;
        IO0CLR = (data & 0x0F)<<10;
        LCD_Delay(10);
}
void LCD_Send_String(unsigned char *data)
{
        while(*data)
                LCD_Data(*data++);
}
void LCD_Command(unsigned int data)
{
        unsigned char temp;
        LCD_RS_CMD;
        LCD_WR;
        temp = data;
IO0SET = (temp & 0xF0)<<6;
        LCD_EN_HI;
        LCD_EN_LOW;
```

```
        IO0CLR = (temp & 0xF0)<<6;
        IO0SET = (data & 0x0F)<<10;
        LCD_EN_HI;
        LCD_EN_LOW;
        IO0CLR = (data & 0x0F)<<10;
        LCD_Delay(10);
}
void LCD_init(void)
{
        LCD_Command(0x28);

        LCD_Command(0x20);

        LCD_Command(0x0C);

        LCD_Command(0x06);

 }
```

Output:



Result:

Experiment: 12

Serial Communication Interfacing

Aim:

To write a program to demonstrate serial driver's Transmission from kit and Reception From PC using Serial Port on IDE environment.

Tools:

(1)PC
(2) Keil Microvision4

Program:

(a)serialdriver.c

```
#include <LPC214X.H>

void Tx_string(unsigned char *);

void  InitUart0(void);

 void main()

{

PINSEL0=0x05;

InitUart0();

Tx_string("Hello world\n");

while(1);

}
```

(b)serialdriver.c

```
//#include "LPC2148.h"

#include <LPC214X.H>

#define DESIRED_BAUDRATE 19200
```

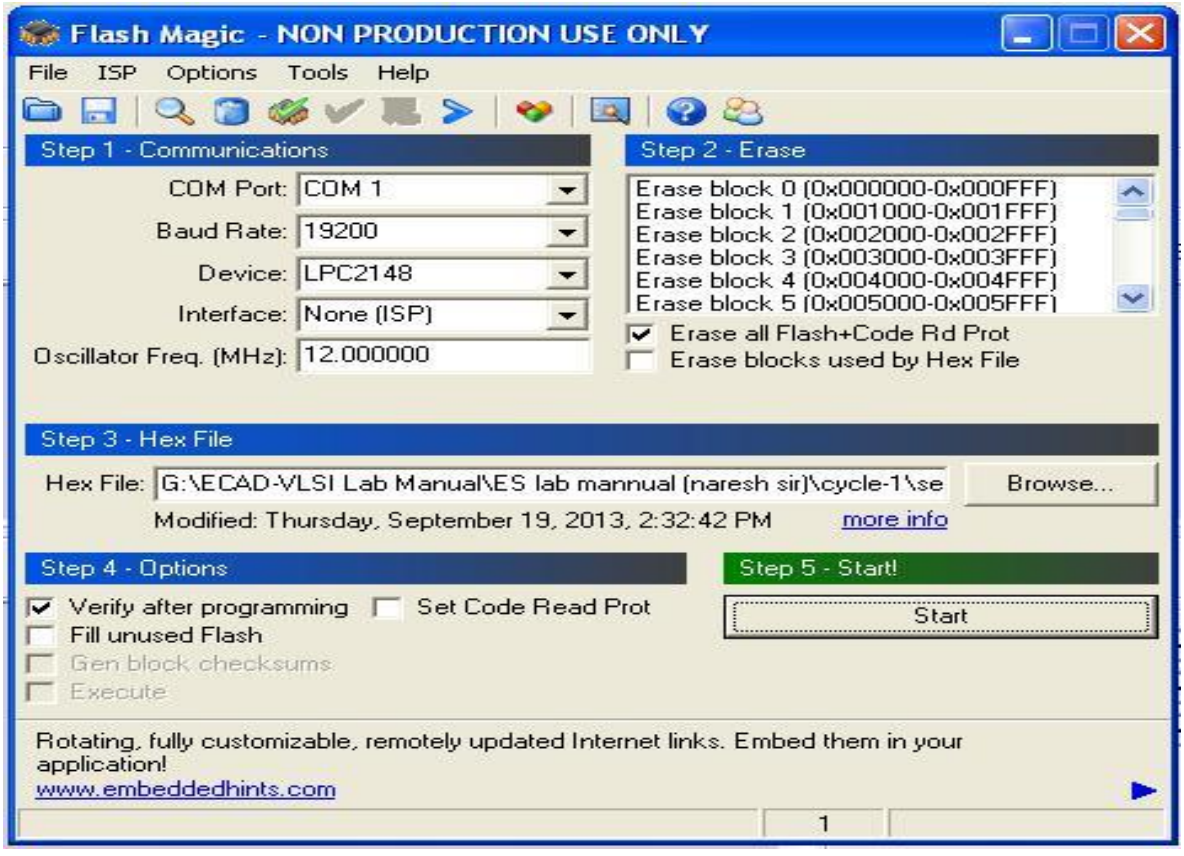
```
#define CRYSTAL_FREQUENCY_IN_HZ 12000000

#define MAX_PCLK (CRYSTAL_FREQUENCY_IN_HZ*5)

#define PCLK (MAX_PCLK/4)

#define DIVISOR (PCLK/(16*DESIRED_BAUDRATE))
void InitUart0(void)

{

        U0LCR=0x83;

        VPBDIV=0x00;

        U0DLL=DIVISOR&0xFF;

        U0DLM=DIVISOR>>8;

        U0LCR=0x03 ;

        U0FCR=0x05 ;

}

void Tx_char(char ch)

{
if (ch=='\n')
        {
                while (!(U0LSR&0x20)) {}

                U0THR='\r';
        }
        while (!(U0LSR&0x20)) {}
                U0THR=ch;
}
unsigned char Rx_char(void)
{
        char ch;
        while (!(U0LSR&0x01)) {}
        ch=U0RBR;
```

```
        return ch;
}
int Tx_string(char *s)

{
        int i=0;
        while(s[i]!='\0')
        {
                Tx_char(s[i]);
                i++;
        }
        return(i);
}
void Tx_num(unsigned int num)
{
        char str[9];
        int i=0,temp;
  for(i=0;i<8;i++)
  {
        temp=num%16;
    if(temp>9)
          str[7-i]=(0x37)+temp;
    else
           str[7-i]=(0x30)+temp;
    num=num/16;
  }
  str[8]='\0';
  Tx_string(str);
}
void Tx_num_dec(int num)
{
        char str[12];
        int i=0,temp,length;
  if(num<0)
  {
        num=-num;
    str[0]='-';
    i=1;
    length=11;
  }
  else
  {
        i=0;
    length=10;
  }
  for(;i<length;i++)
```

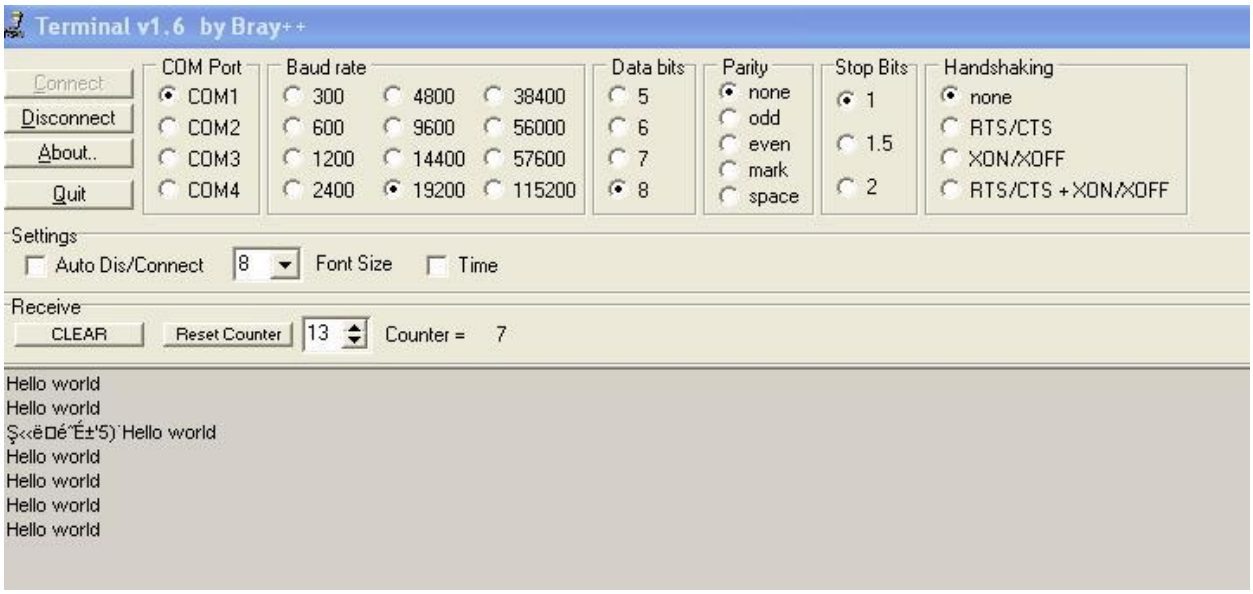
```
  {
        temp=num%10;
    if(temp>9)
          str[length-1-i]=(0x37)+temp;
    else
           str[length-1-i]=(0x30)+temp;
    num=num/10;

  }
  str[length]='\0';
  Tx_string(str);
}
```

Output:

Flash magic tool



Terminal



Result: