Start Your Embedded System Design Journey Today..!

LPC2148 essential development features a plug and play design that makes it easy for connections and helps Students, hobbyists, enthusiasts, and professionals to focus more on Program / application development. LPC2148 Trainer Kit equipped with on board IO's, communication interfaces & peripherals. It is really easy to design, experiment with, and test circuits without soldering. It's used in many educational institutions and R&D LAB acrossthe world.

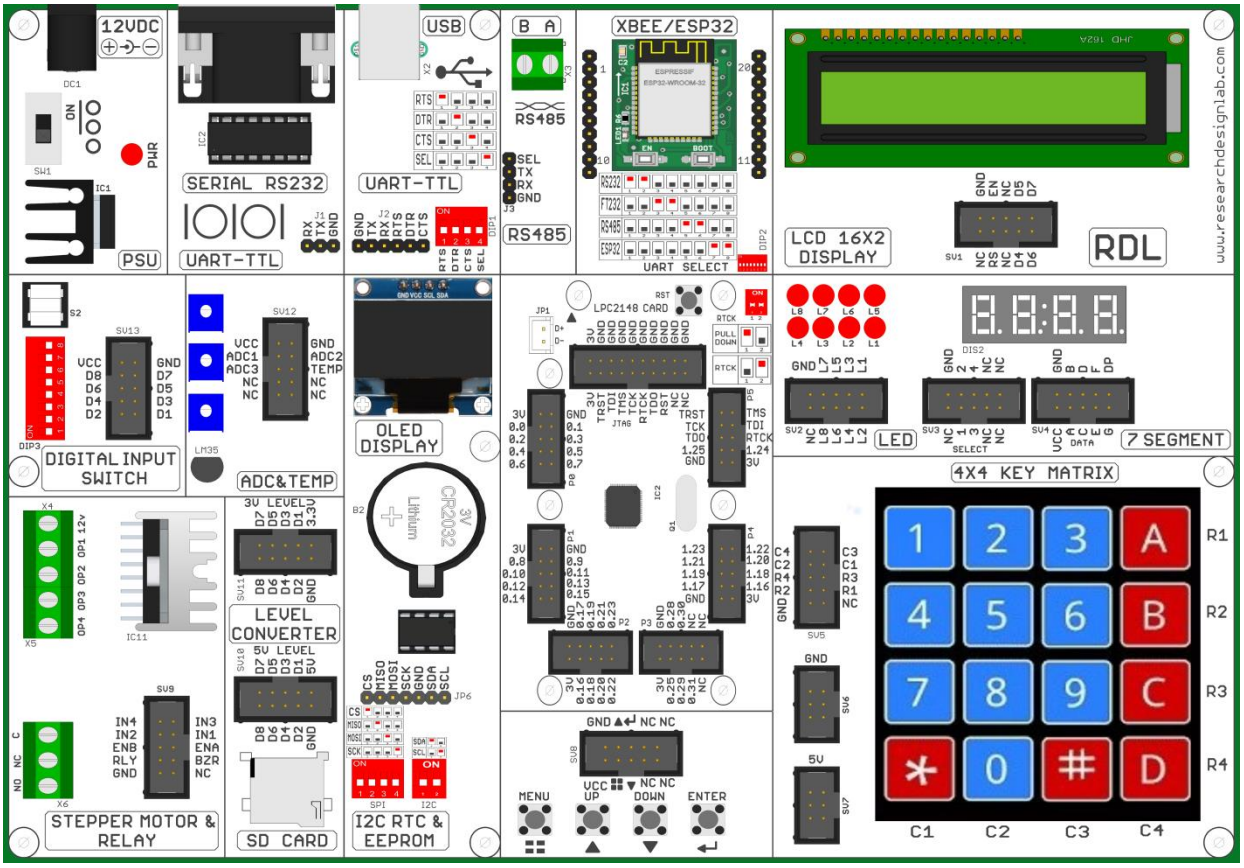
Features:

- Plug & Play Interface Connectivity
- Professional EMI/RFI Complaint PCB Layout Design
- Modular Block design makes Easy access & quick Prototyping
- FRC connectivity features minimize the connection Error.
- High Quality Grade PCB with wooden Enclosure.
- Stackable daughter board LPC2148.
- On board debugging JTAG Option
- USB onboard programming
- 8 interfacing LED's.
- 1 * 4 interfacing keys.
- 4* 4 interfacing keypad matrix.
- Two channel RS232 port for communication.
- 3 ADC potentiometer input interface.
- 16X2 LCD, OLED interface.
- 46 general purpose IO
- On Board Power supply 3.3V, 5V 12V,GND.
- 8 pin DIP switches.
- On board voltage level converter
- Board enabled with the header forstacking the WiFi module.
- Board enabled with the header forstacking the Arduino shields.
- ON board Micro SD card
- ON/OFF slide switch.
- RDL bus.
- External jumper nodes.
- Reset button.
- Power plug-in DC Socket.
- Power supply indicator LED.
- Test led for Tx, Rx.
- 7Seg Multiplexed Display.
- FT232RL USB communication.

Scope of Learning Experiments:

- LED blinking
- 8 bit LED Left shift, Right shiftand counting operation
- Keypad Interrupt Interface
- 16*2 LCD interface
- Matrix Keypad Interface
- ADC & DAC interface
- Traffic Light Signal Interface
- 8 bit DIP switch interface
- 7 Segment interface
- L298 Driver for DC Motor and Stepper motor interface
- Elevator Interface
- Buzzer, Relay interface
- RS485, RS232 serial communication
- UART Operation
- RTC DS1307 I2C protocol interface
- AT24C04 EEPROM I2C protocol interface
- RF/WiFi Communication
- SPI protocol interface
- Temperature Sensor Interface
- Automatic Number Plate Recognition

ARM Board Narration:



Index

1. Pre-Requisites 5
2.1. How to install Keil uVision 4 6
2.2 How to install FTDI driver 10
2.3 How to install Flash Magic Utility 14
3. Library Files (LibFiles) 15
4. How to Create a New Project in Keil µVision4 16
5. How to Enable Hex File Generation 26
6. How to Upload Hex file Using Flash Magic 31
7. How to Upload Hex file Using Philips Flash Utility 32
8. Lab Programs 33
8.1. Blinking an LED 33
8.2. Liquid Crystal Display. 35
8.3. ADC 38
8.4. UART. 42
8.5. Real Time Clock 44
8.6. Hex Keypad 49
8.7. Stepper Motor 56
8.8. PWM 59
8.9. EEPROM 63
8.10. Arithmetic operation 68
8.11. Logical operation 74
8.12. Seven Segment Display 80

1. Pre-Requisites

On your system you need to have the following Software installed before executing lab programs.

1. Keil uVision 4 software
2. FTDI driver
3. Flash Magic Utility

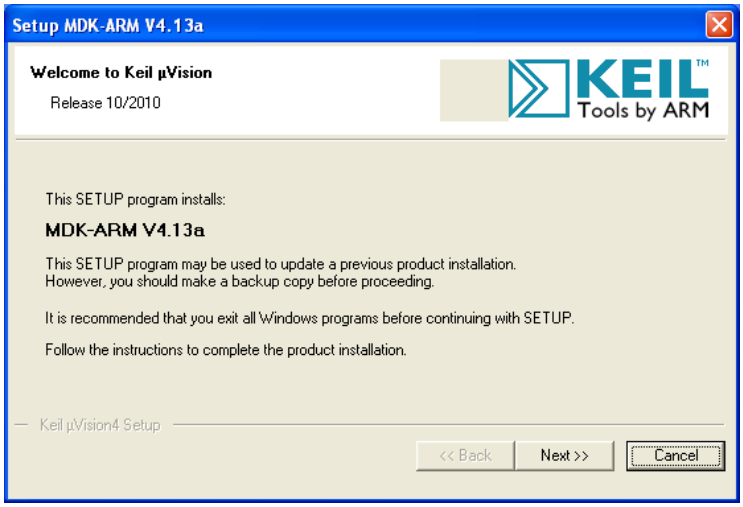
Follow the next section to install all these software.

2. Installation

2.1. How to install Keil uVision 4

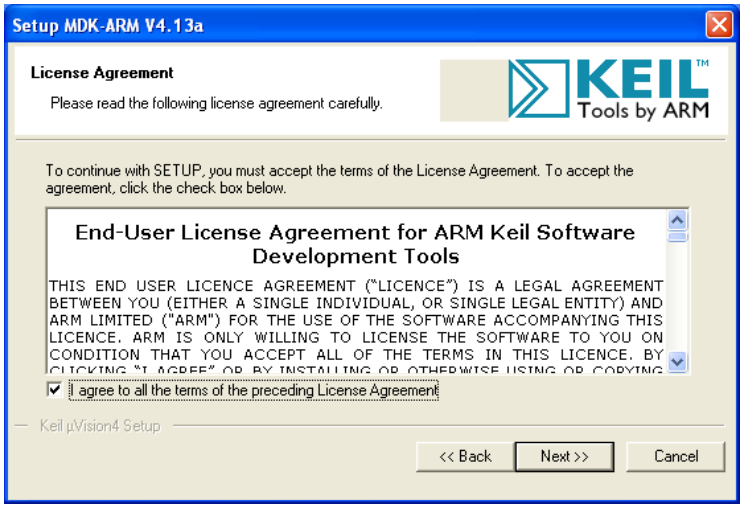
Follow the steps below to install Keil uVision 4 on your system.

1. Run setup file MDK-ARM V4.13a.exe (Double click the file icon).
2. Setup Window appears....

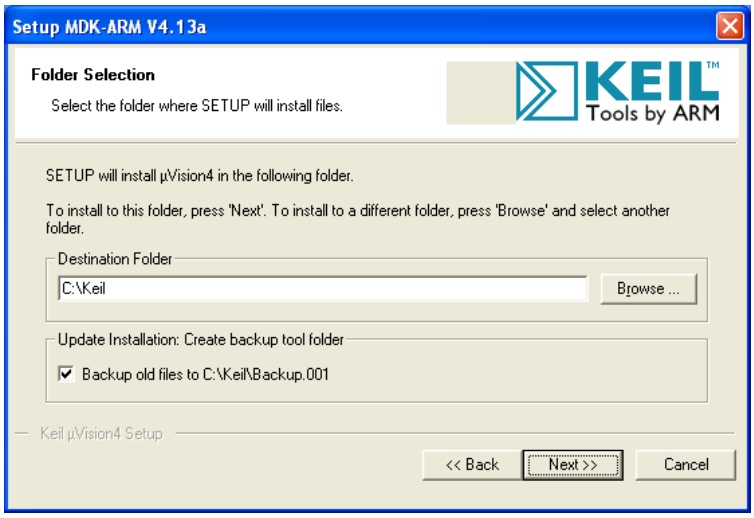


3. Click Next.

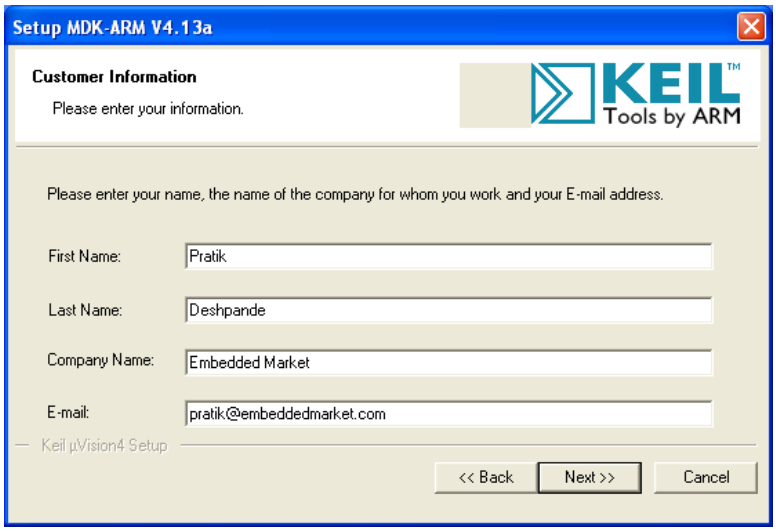
4. Click Checkbox of agreement in order to proceed and click Next



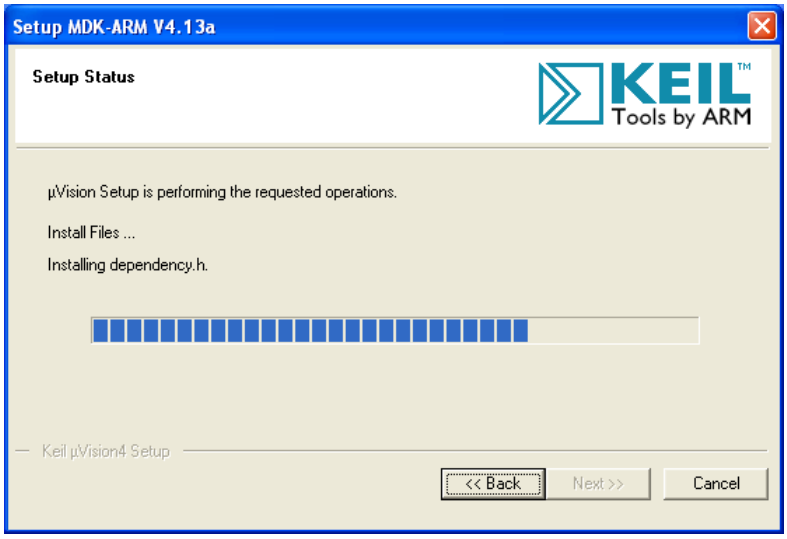
5. In folder selection window, select suitable directory for the installation and press next (default directory should work fine!)



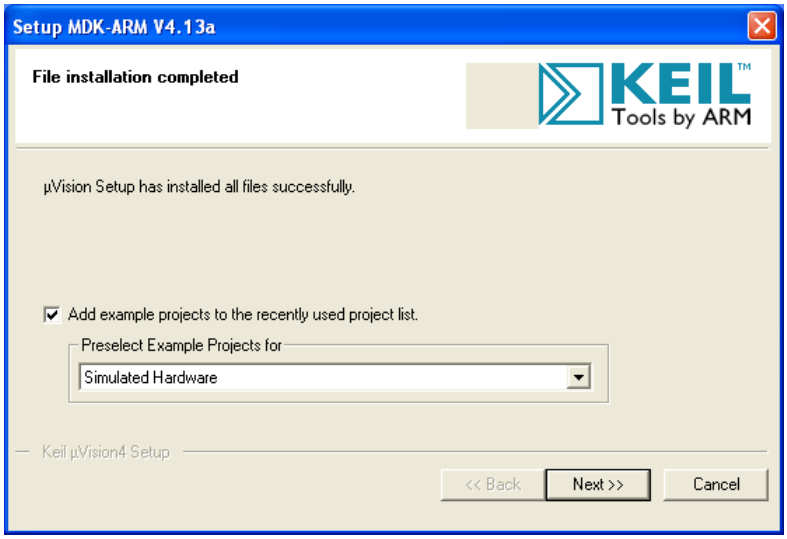
6. Enter customer information and click next



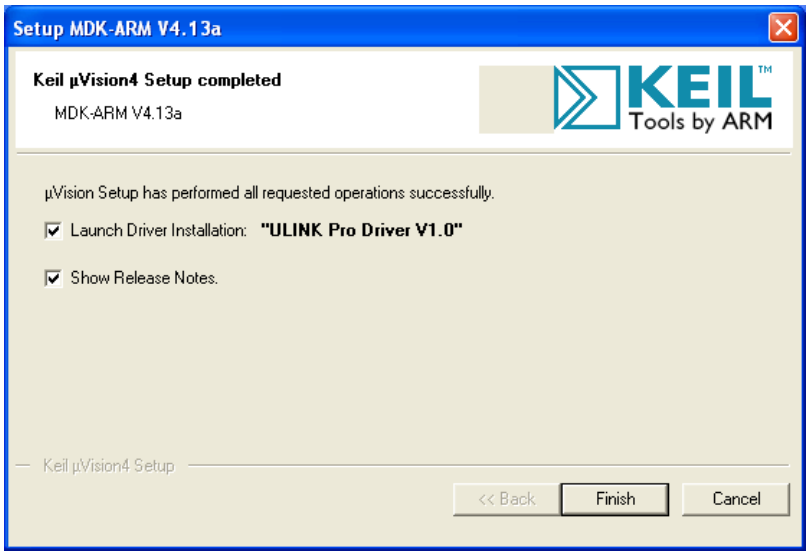
7. The setup should begin. Setup status window will show installation of different files. Wait until it gets installed completely.



8. Once the installation gets over, message should appear with text **"File installation Completed",** Click Next without making any changes anywhere in the window.



9. Next message will confirm **"Keil uVision 4 Setup completed".** Click Finish

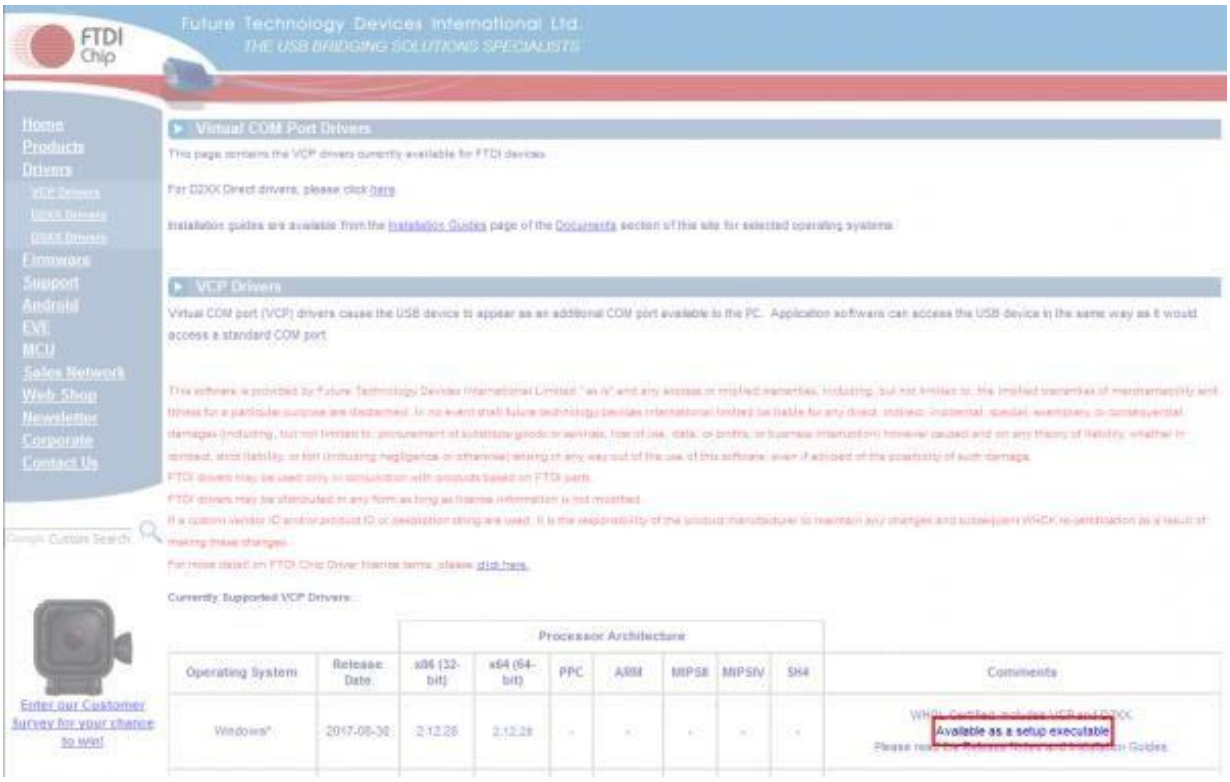


10. This will lead you to the webpage of Keil Development Suit for ARM. This will provide you release notes for Keil Development kit. This document gives you the brief idea of which microcontrollers are being added in the Keil uVision 4.

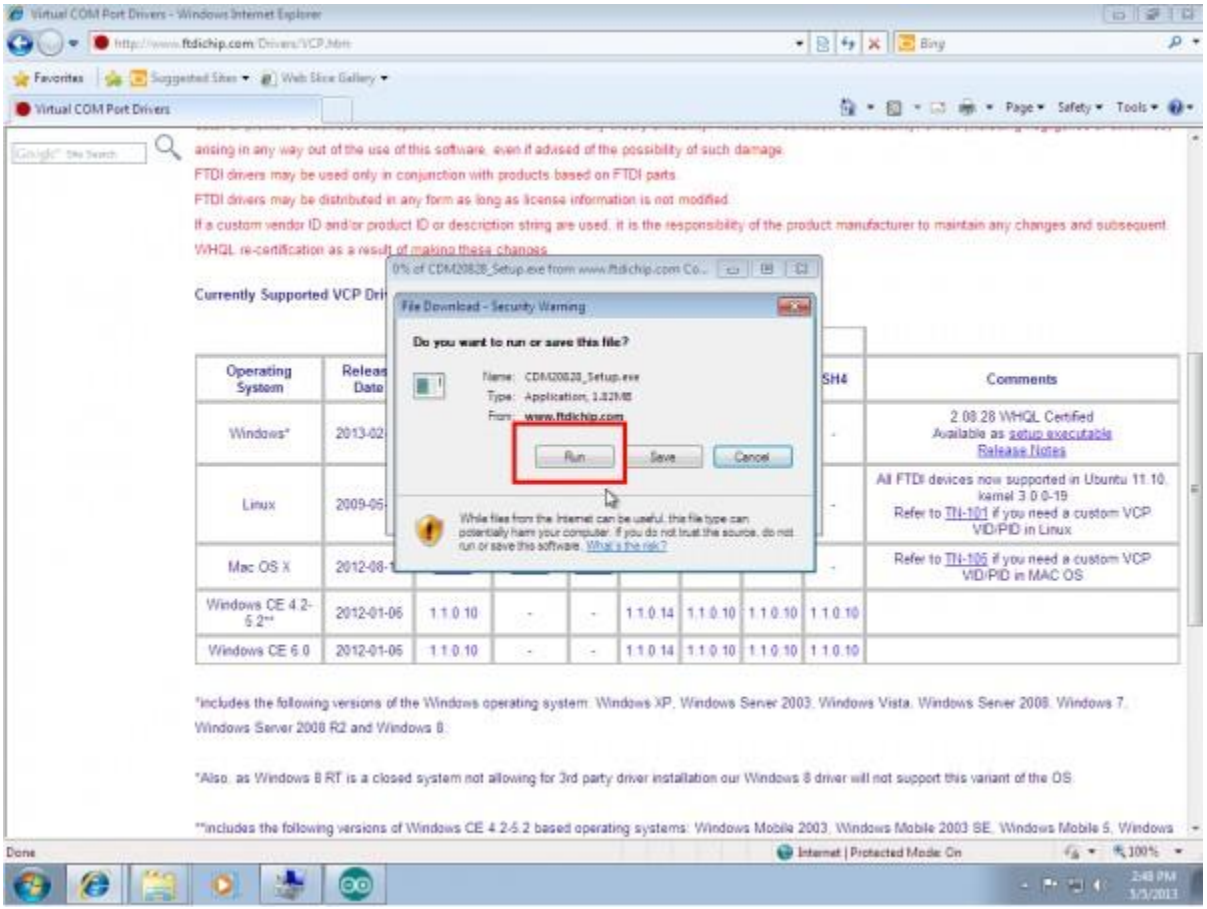
11. Run "Keil uVision 4" by double clicking the "Keil uVision 4" icon from the desktop. Same can be done from Start menu.

2.2 How to install FTDI driver

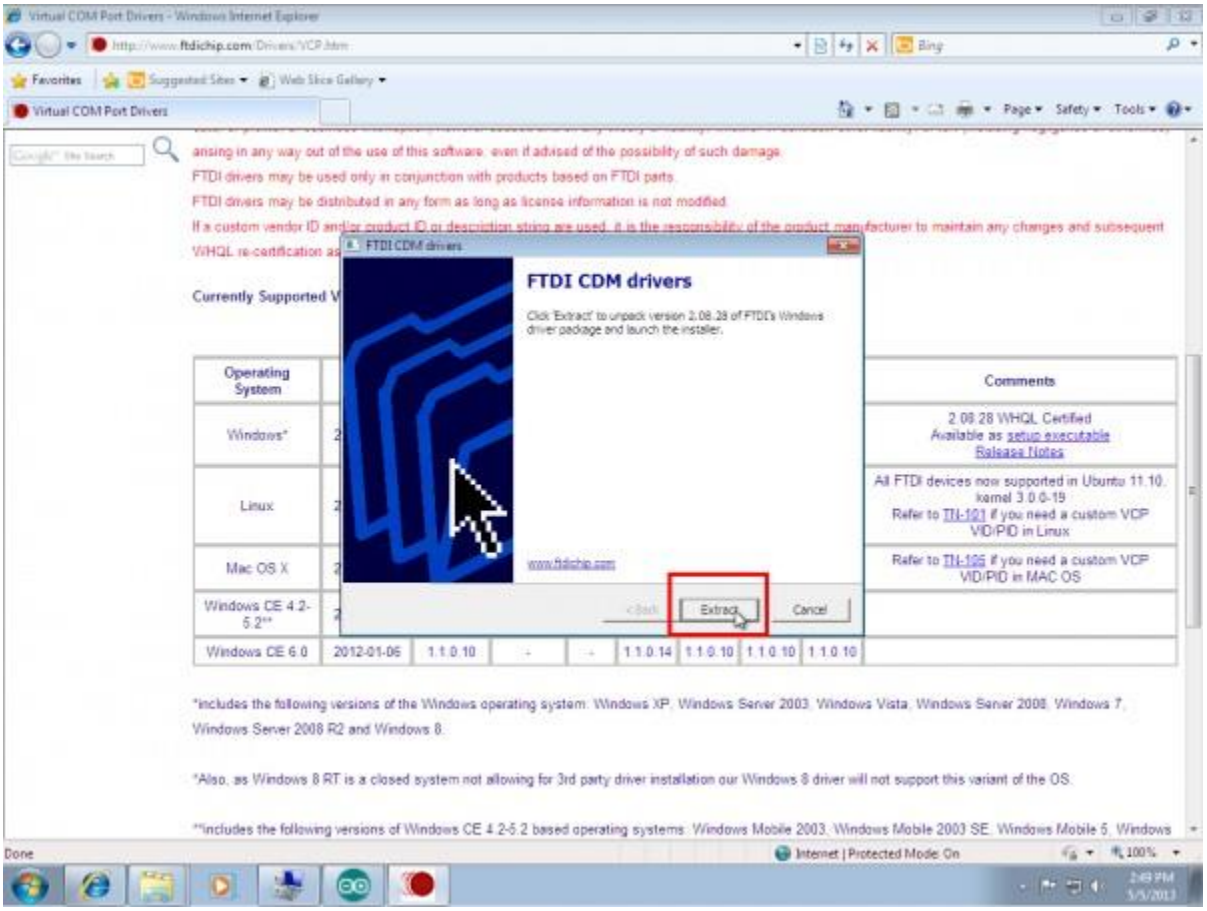
1. Visit **FTDI's VCP Drivers page** (http://www.ftdichip.com/Drivers/VCP.htm) for the latest download of the Windows FTDI Driver executable and clicking on the Window's **"Available as a setup executable"** link. Make sure to unzip the executable before proceeding to the next step.



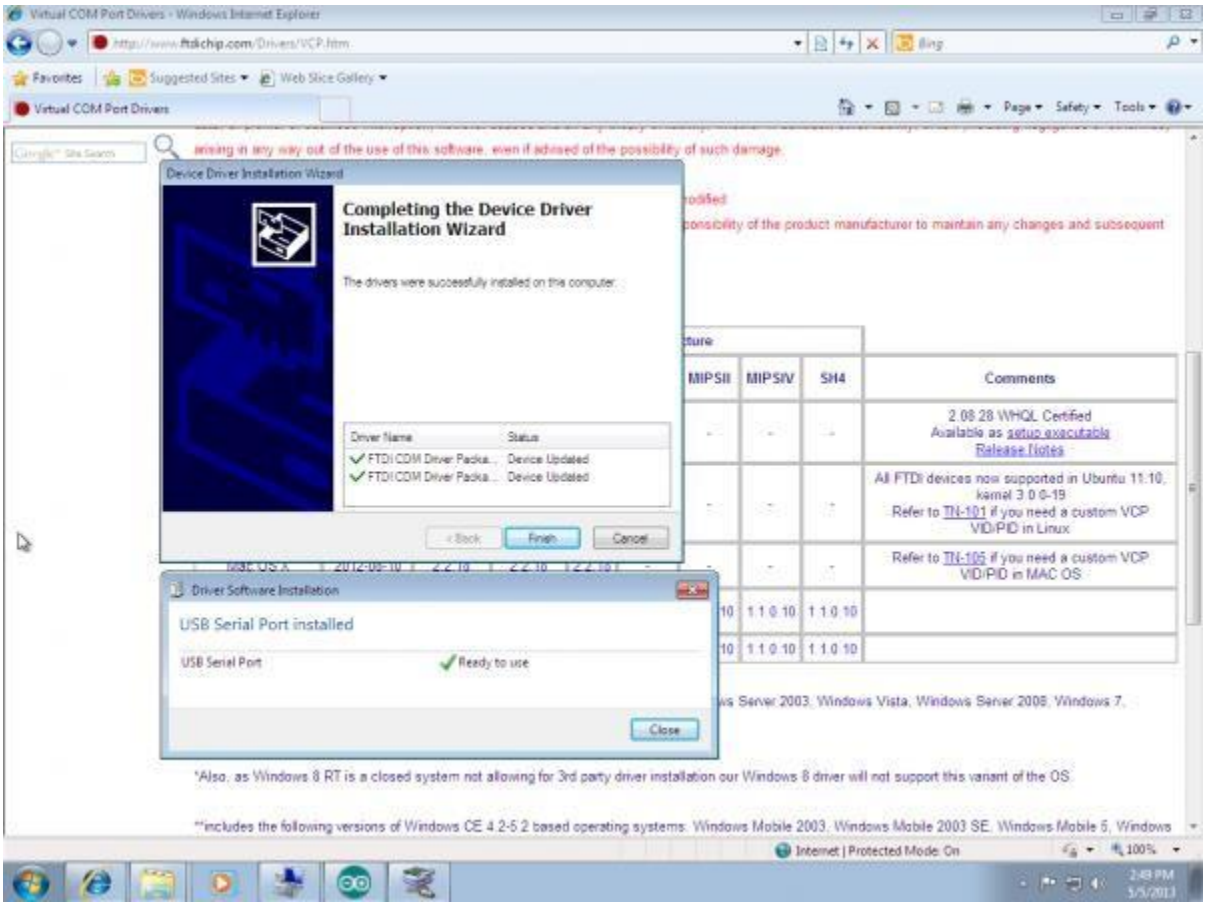
2. Choose *'Run'* once it is has finished downloading, or find the file you just downloaded ***"CDM21228_Setup.exe"*** and double-click it to run it.



3. Choose ***'Extract'*** and continue through the installation until it finishes.



4. If everything was successful, you should see some nice green check marks, indicating success!

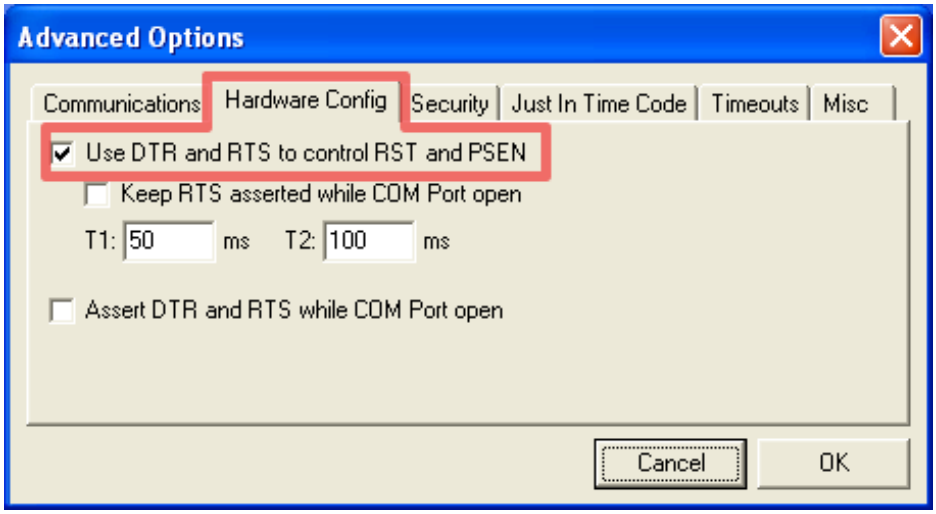


Note: You may need administrator privileges on your machine in order for this to run properly.

2.3 How to install Flash Magic Utility

1. Visit FlashMagic website http://www.flashmagictool.com/ and download the file FlashMagic.exe.
2. Execute the downloaded file FlashMagic.exe, and follow the instructions.
3. Start Flash Magic by selecting it from the Start Menu. In the Flash Magic windows select Options > Advanced Options ... menu item. In the window that appears enable the check box that says Use DTR and RTS to control RST and P0.14, and click on Ok.

   When this option is enabled, during code download, the flashing tool will automatically switch the device into ISP mode. For more information on this, see the board user manual.

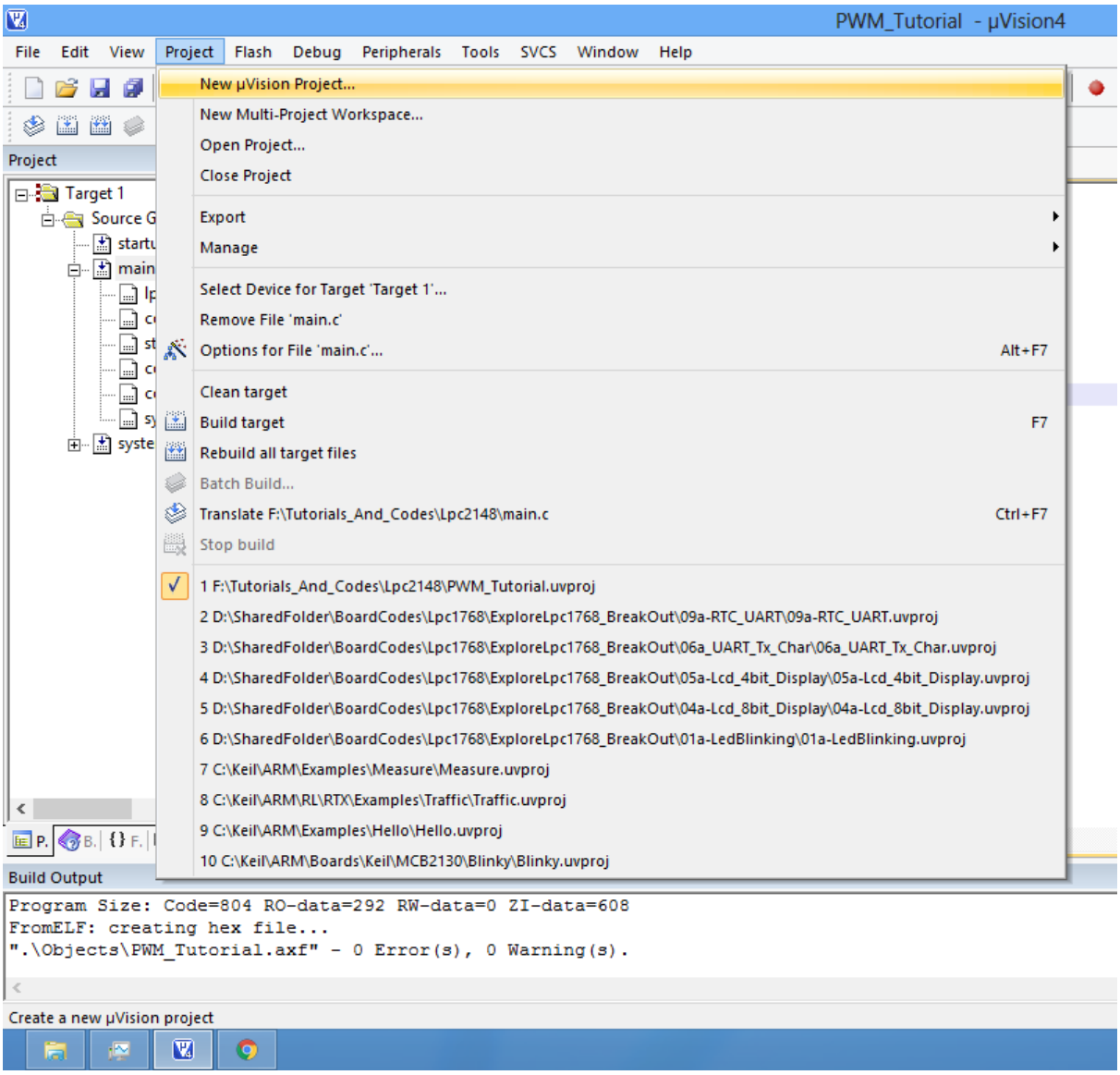


3. Library Files (LibFiles)

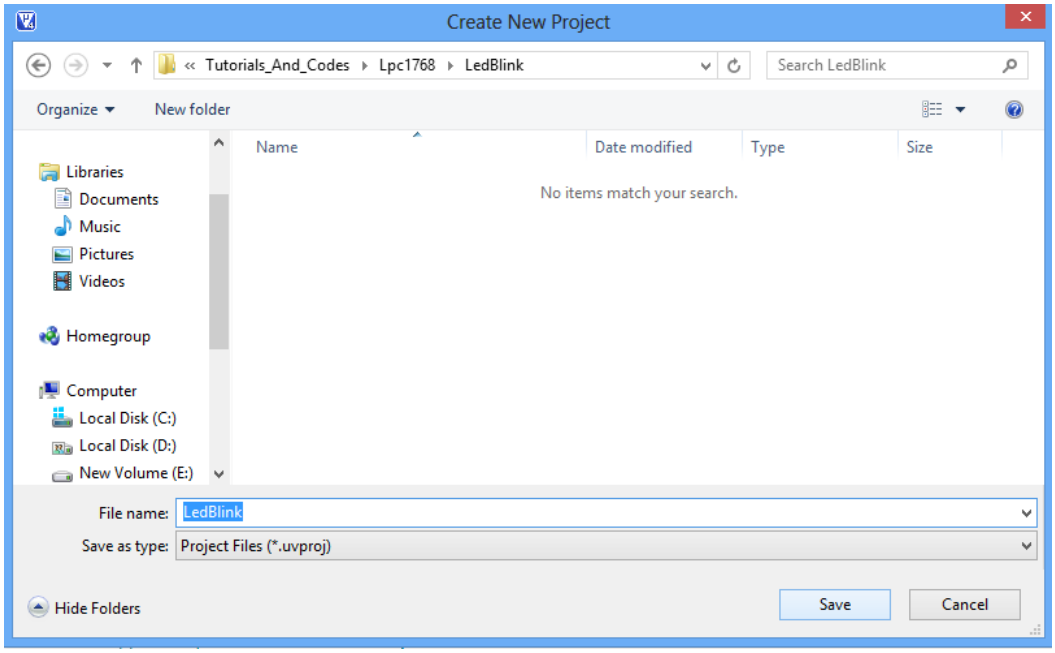
Basically "**LibFiles"** contains **"system_lpc21xx.c"** and **".h"** files**.**
Before executing lab programs it is important to have this library files in our system as we are going to link our programs to this folder.
Place **"LibFiles"** folder in a common path in all the computers. (ex: C->Documents->ARM->LibFiles)

4. How to Create a New Project in Keil µVision4

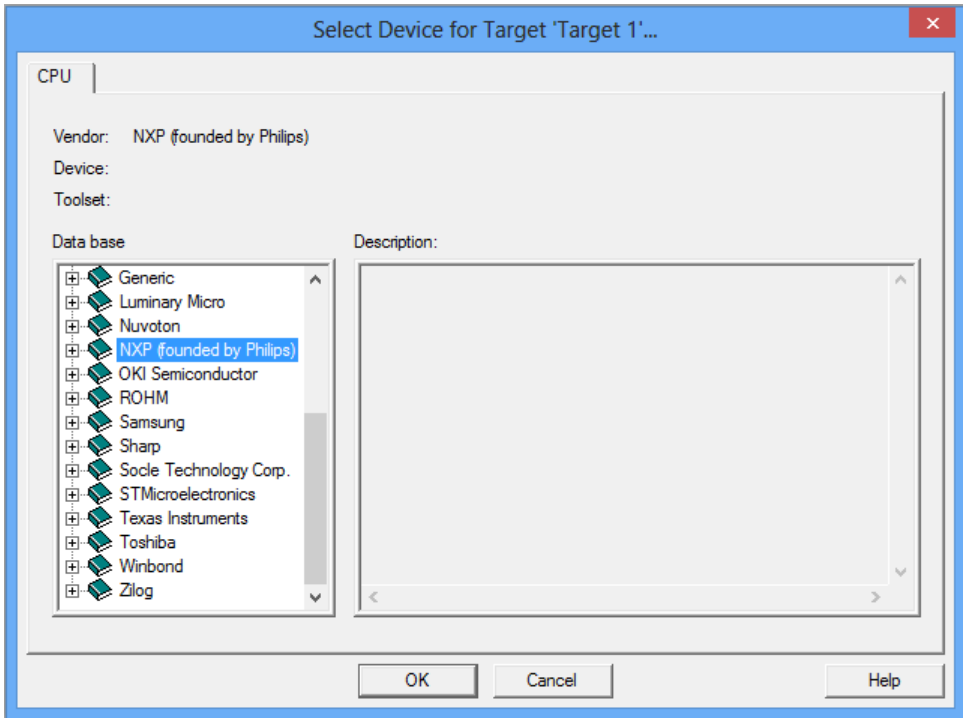
1. Open the Keil software and select **"Project-> New µvision project"** as shown below.



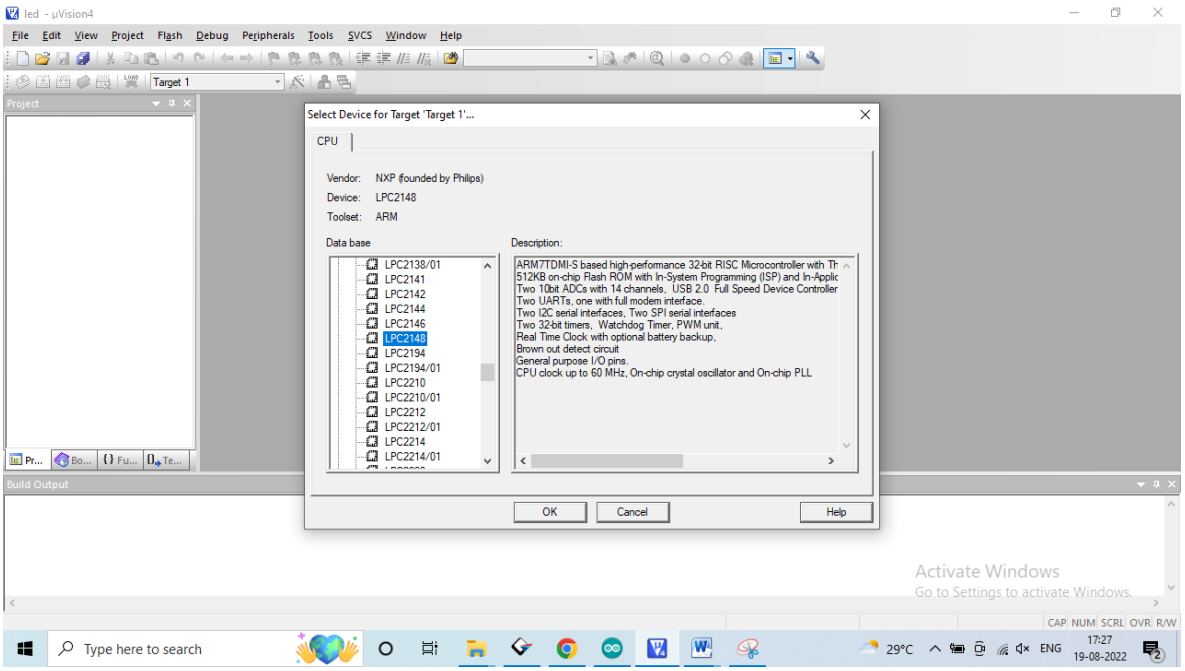
2. Browse to your project folder (/Create a folder) and provide the project name and click on save.



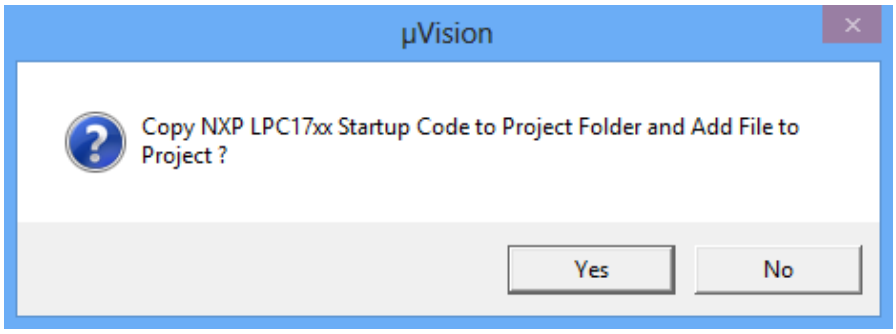
3. Once the project is saved a new pop up "Select Device for Target" opens, Select the controller (NXP:LPC2148) and click on OK.



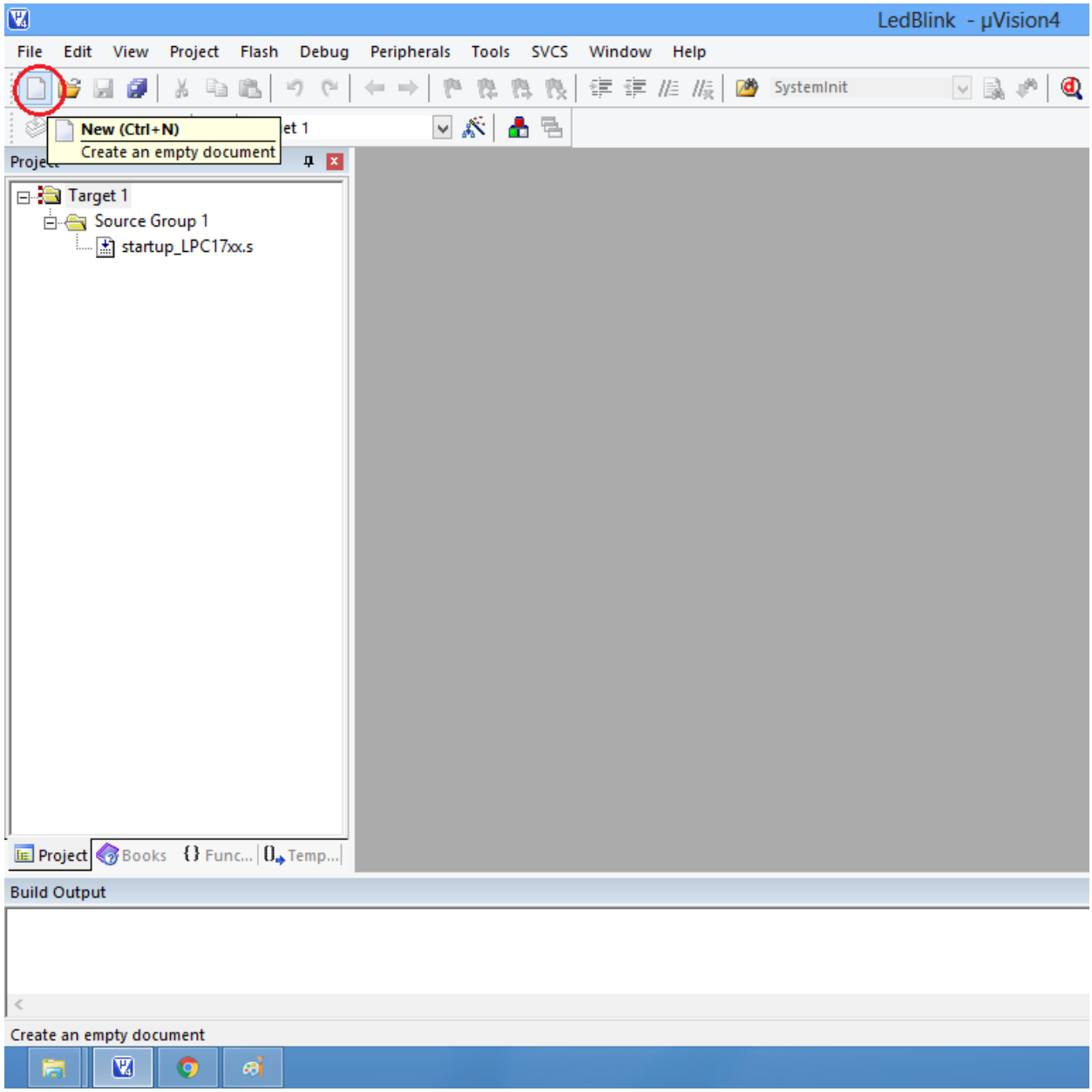
4. Select the controller (NXP:LPC2148) and click on OK.



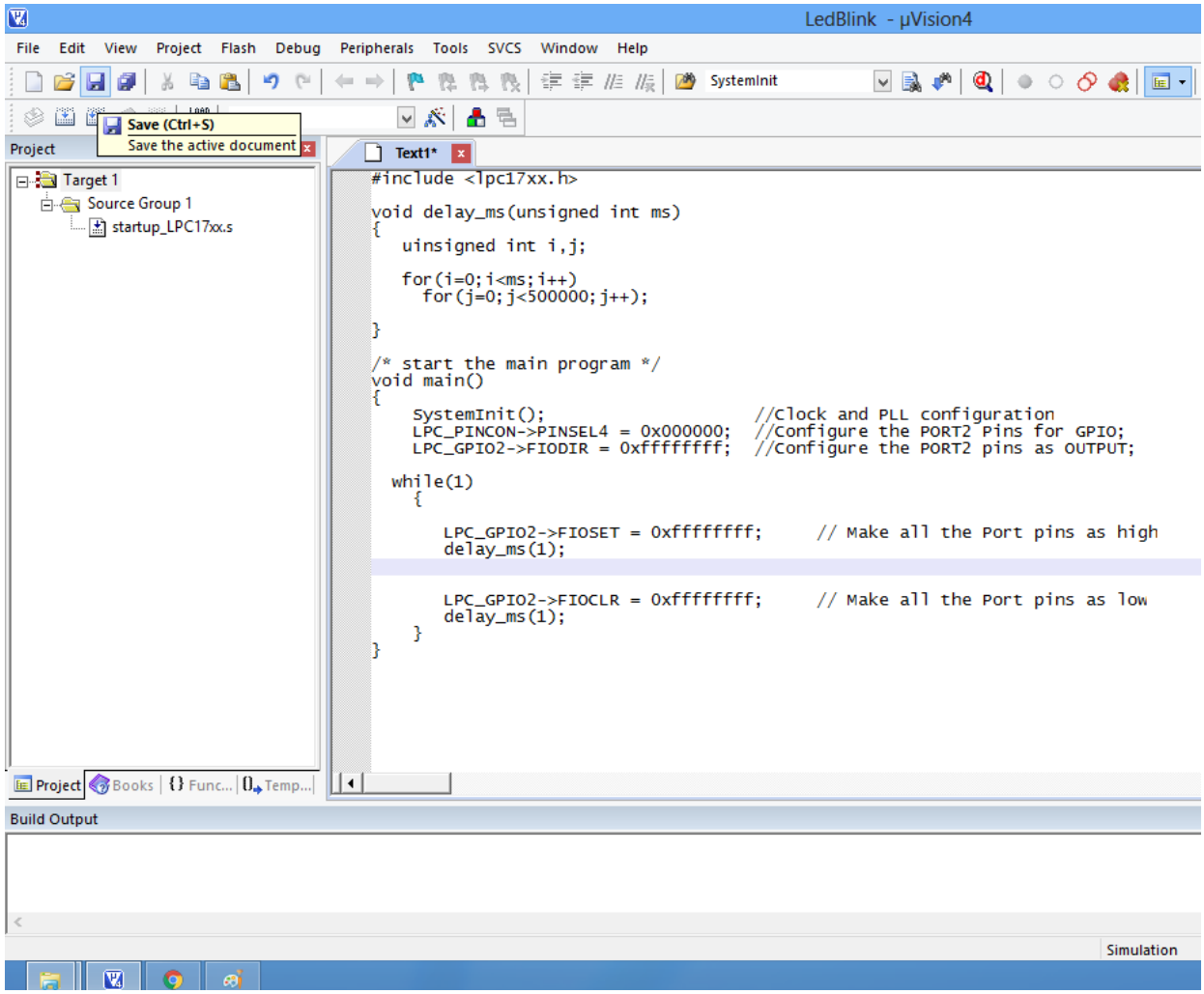
5. As LPC2148 needs the startup code, click on **Yes** option to include the **LPC21xx Startup** file.



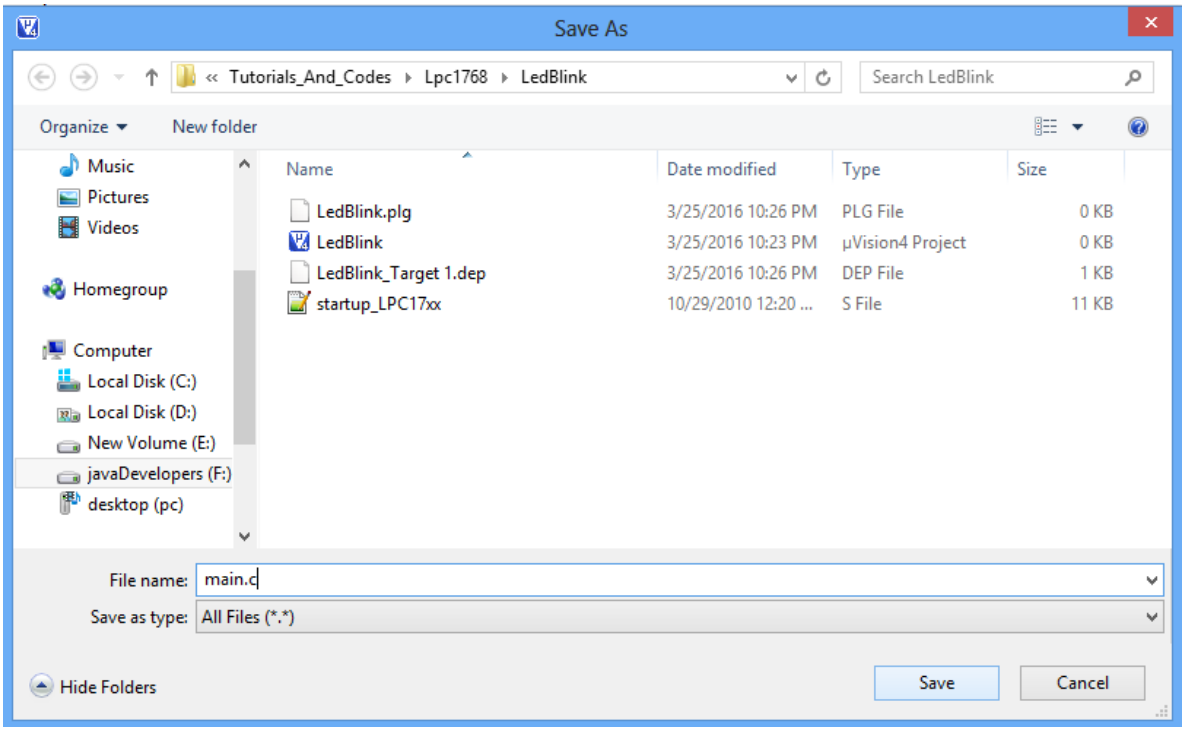
6. Create a new file to write the program.



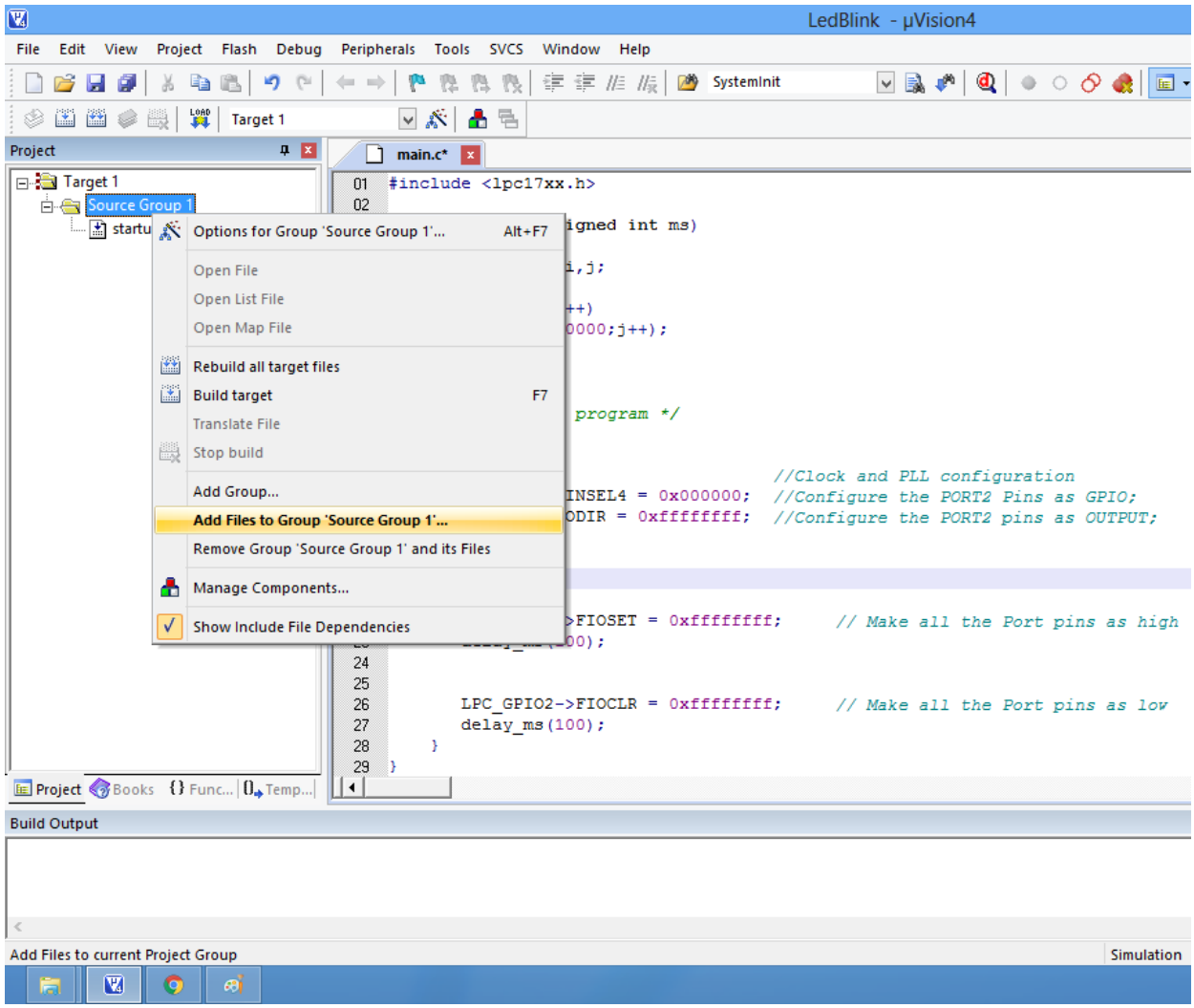
7. Type the code or Copy paste the below code snippet.



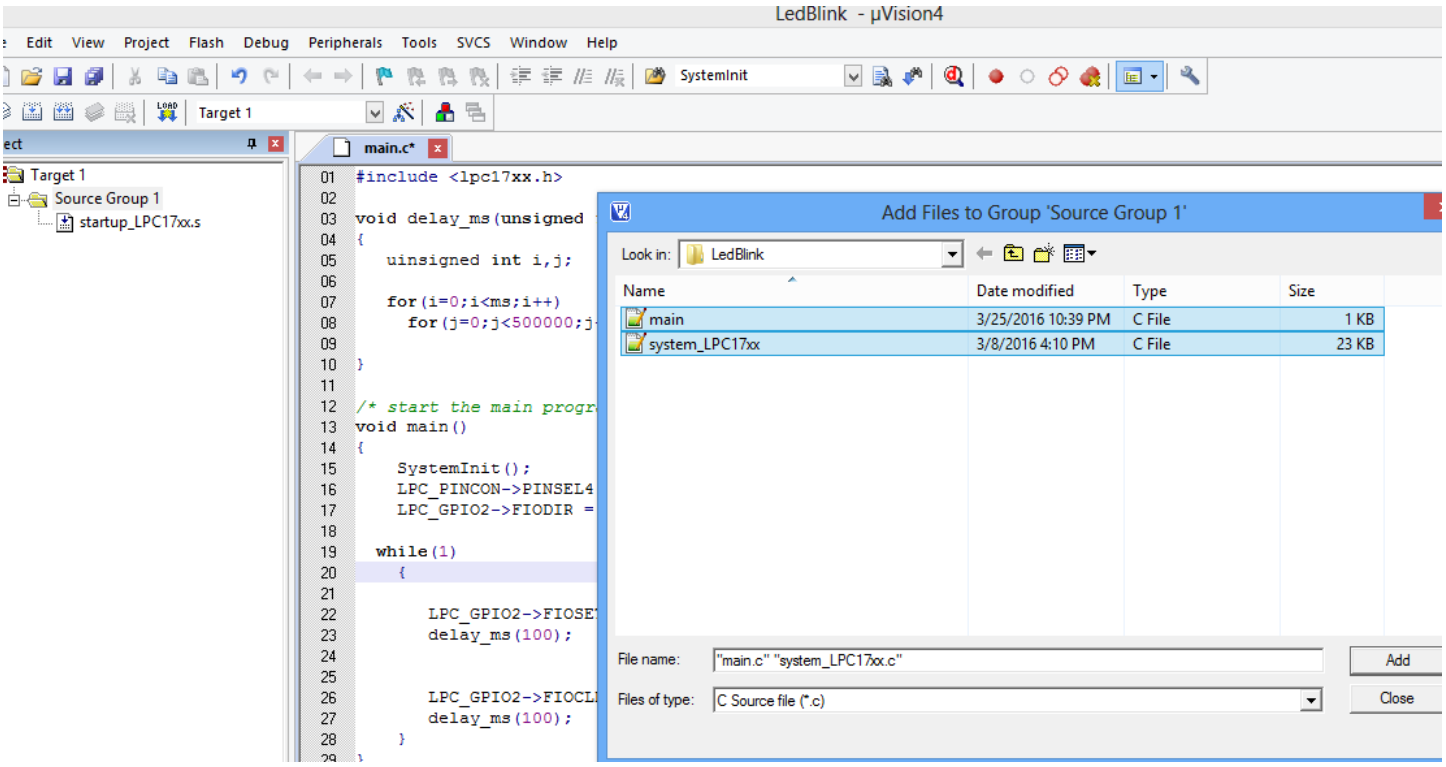
8. After typing the code save the file as **main.c**.



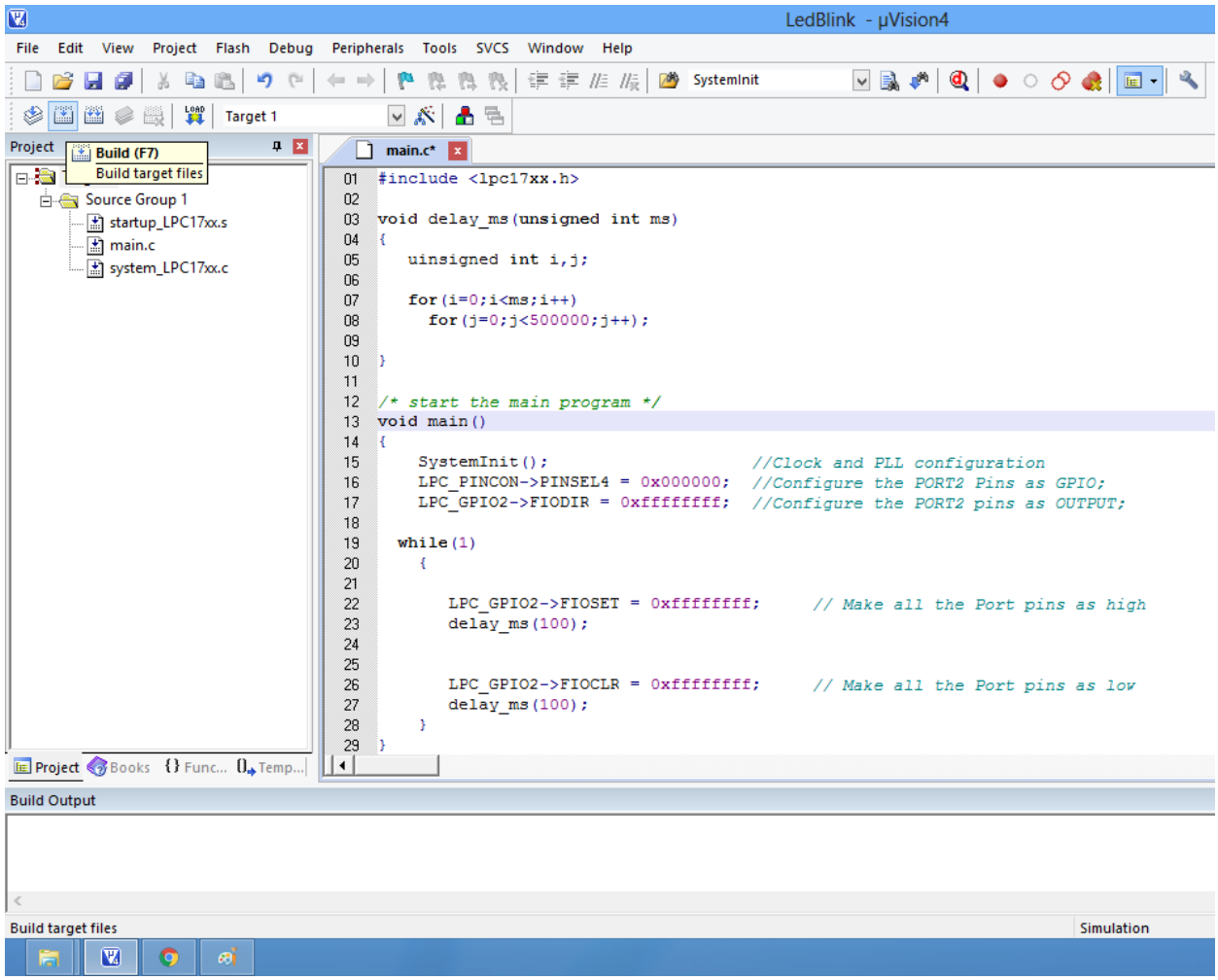
9. Add the recently saved file to the project.



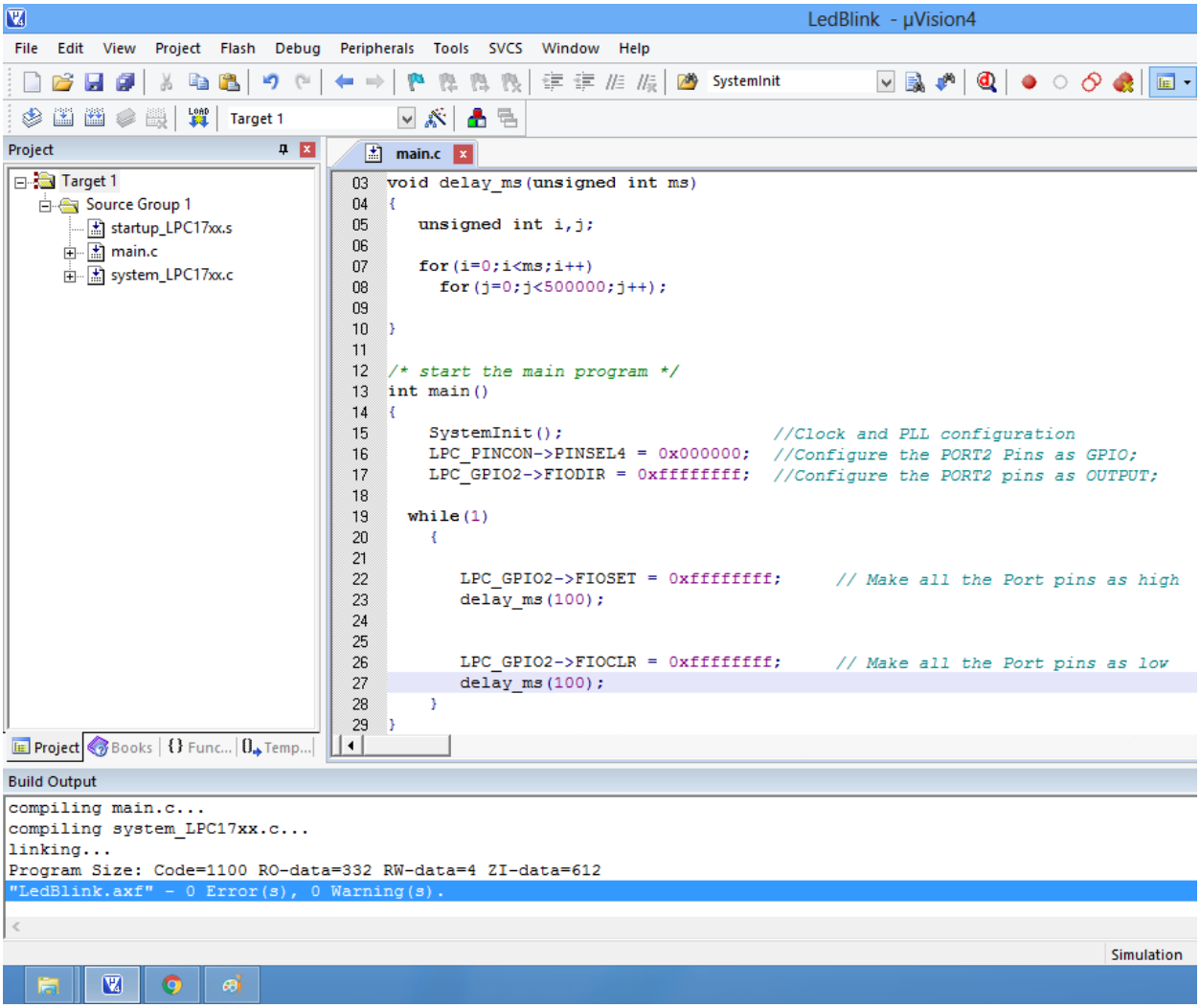
10. Add the **"main.c"** along with system_LPC21xx.c.



11. Build the project and fix the compiler errors/warnings if any.

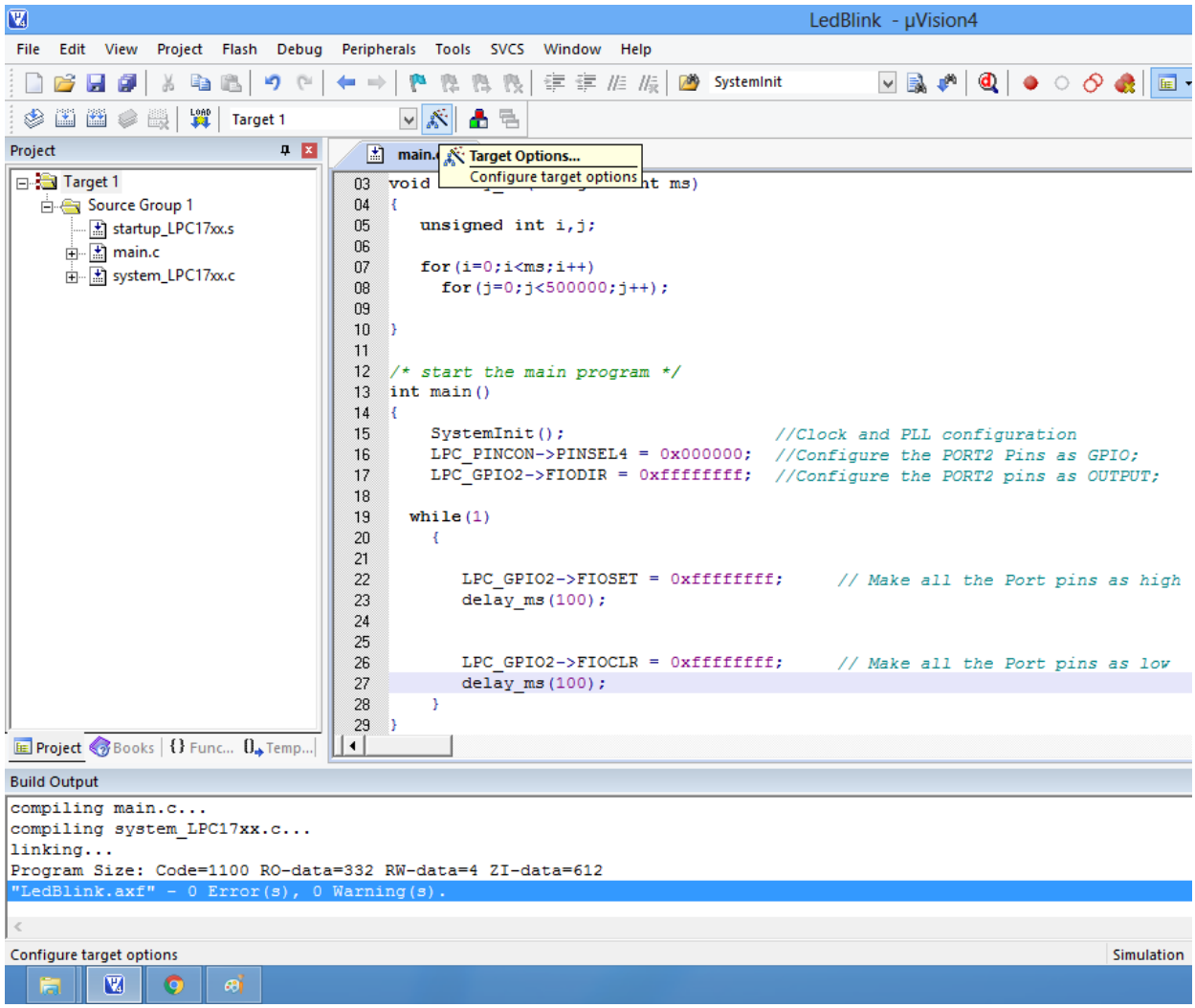


12. Code is compiled with no errors. The **.hex** file is still not generated. Follow section 4 ([how to enable Hex File Generation](#))

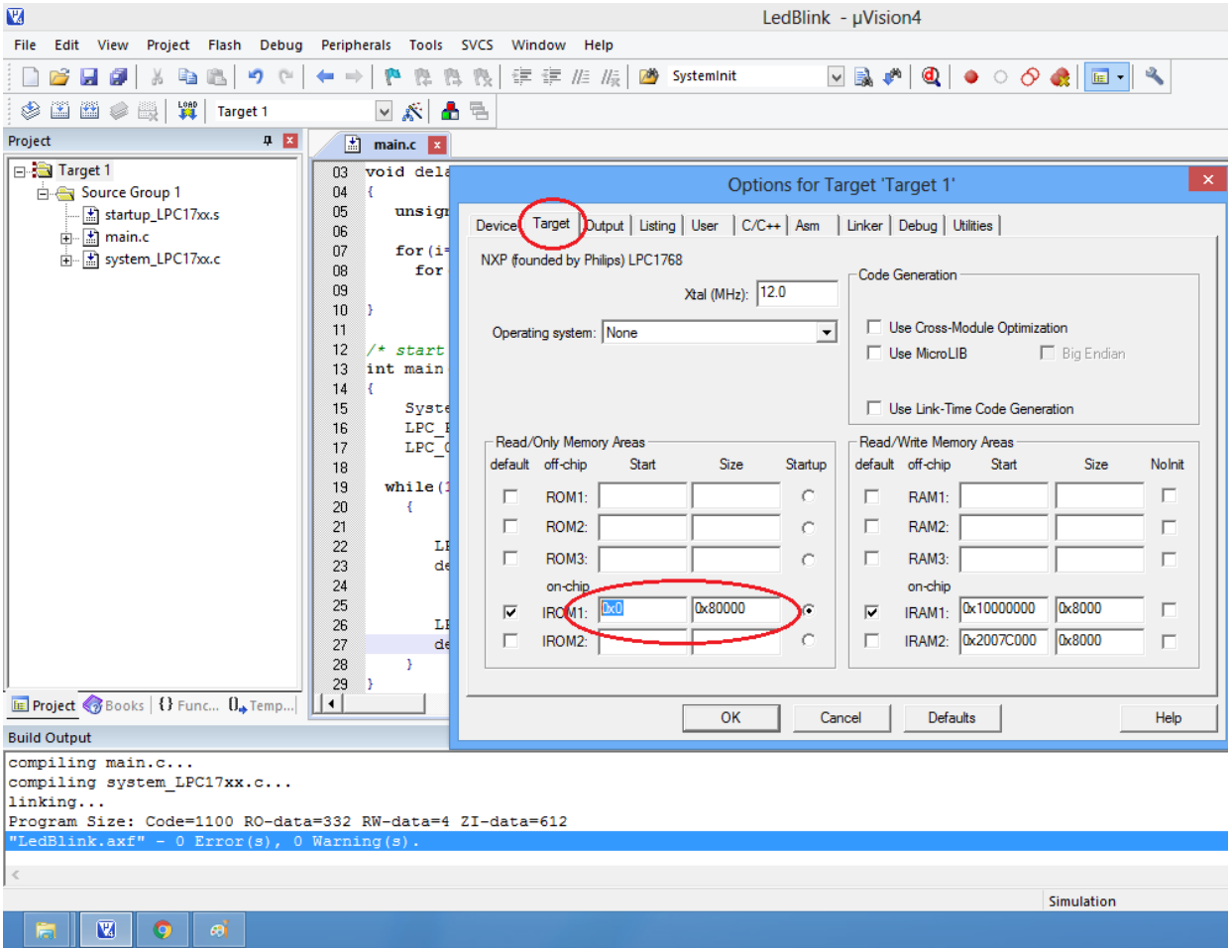


5. How to Enable Hex File Generation

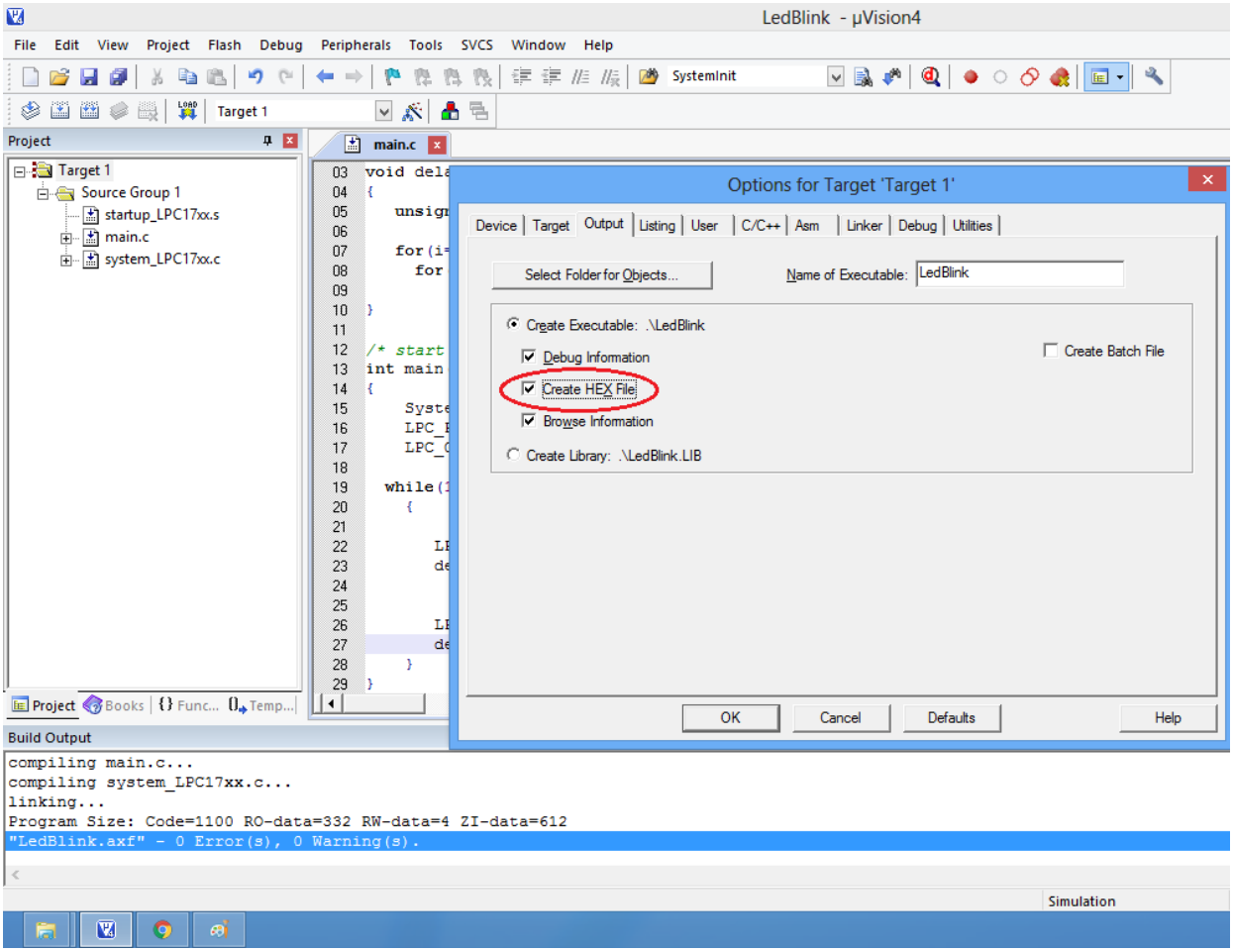
1. Click on **Target Options** (**or** right click on **"Target 1"** and select **"Options for Target 'Target 1'…)** to select the option for generating .hex file.



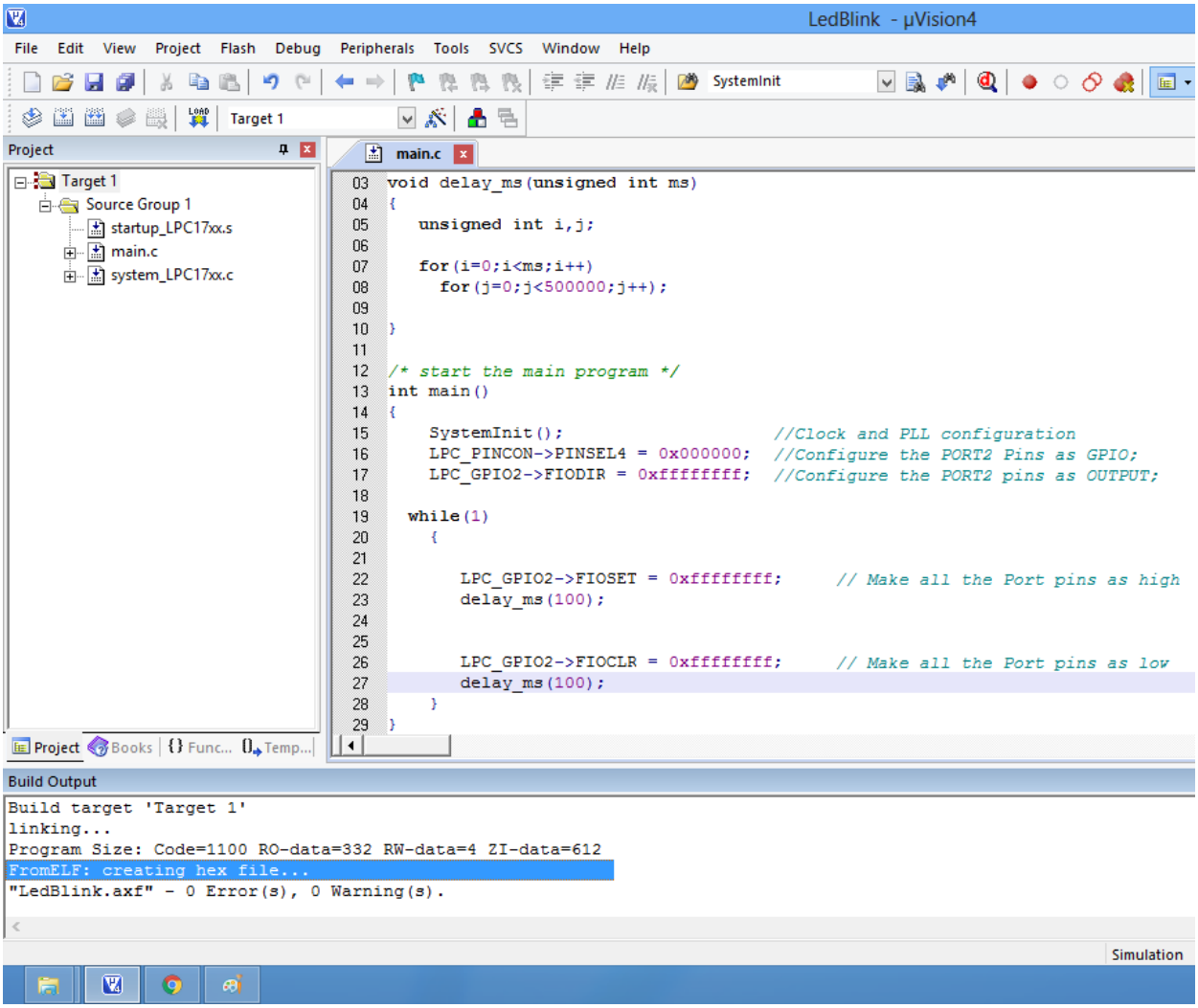
2. Set IROM1 start address as 0x0000.



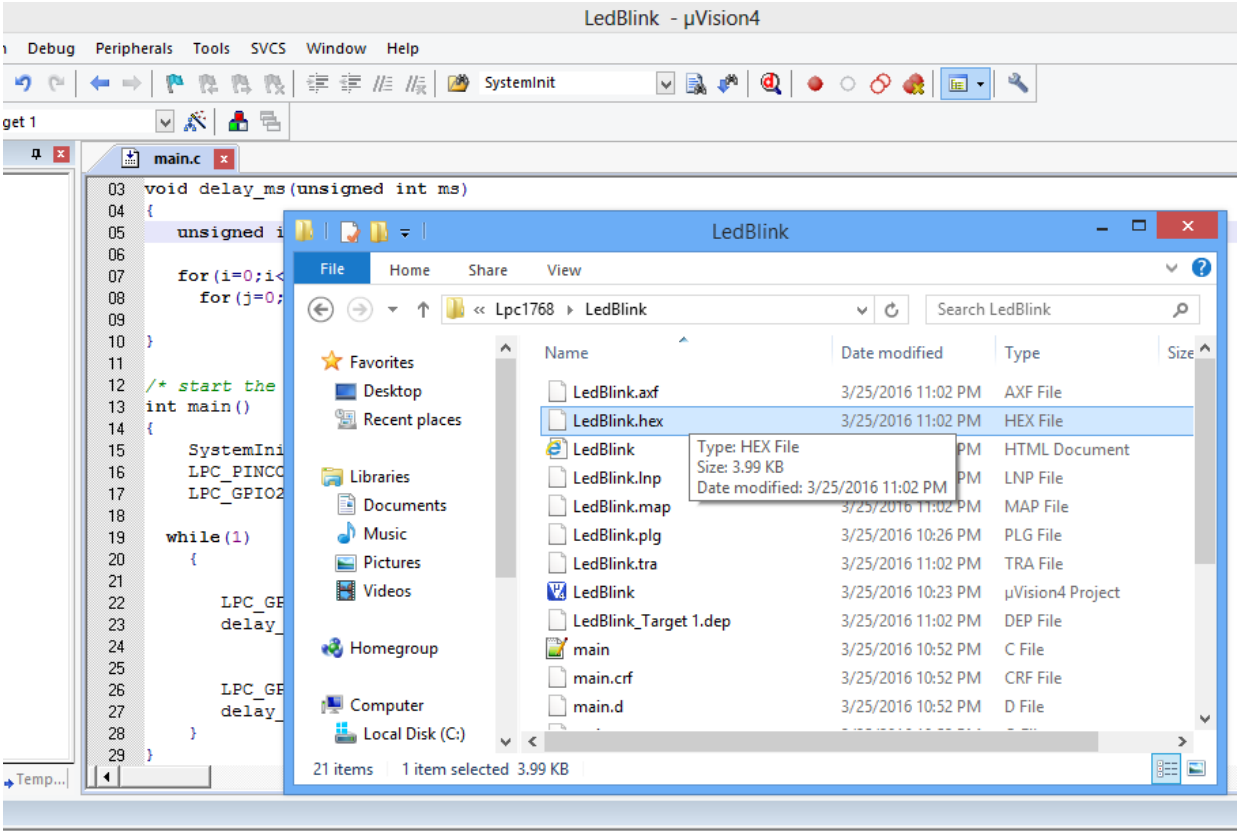
3. Enable the option to generate the .hex file



4. Hex file is generated after a rebuild.



5. Check the project folder for the generated .hex file.

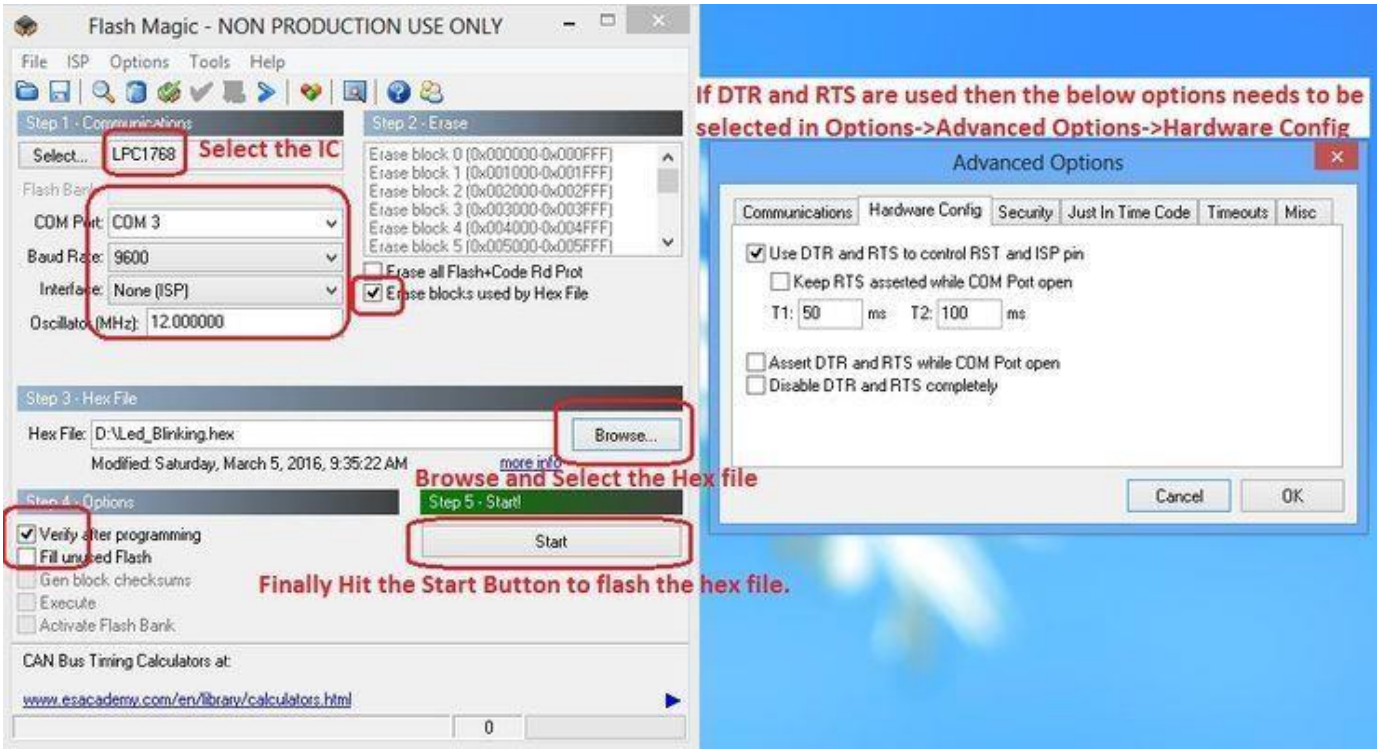


6. How to Upload Hex file Using Flash Magic

Once the **".hex"** file is generated in keil, we need to upload the hex file to the hardware. We use Flash Magic Tool for the same.

Open the flash magic software and follow the below steps.

1. Select the IC **"LPC2148"** from Select Menu.
2. Select the COM Port. **Check the device manager for detected Com port**.
3. Select Baud rate from 9600-115200
4. Select None [ISP] Option for Interface.
5. Oscillator Frequency 12.000000(12Mhz).
6. Check the Erase blocks used by Hex file option
7. Browse and Select the hex file.
8. Check the Verify After Programming Option.
9. If DTR and RTS are used then go to Options->Advanced Options-> Hardware Config and select the Use DTR and RTS Option.
10. Hit the Start Button to flash the hex file.
11. Once the hex file is flashed, reset the board. Now the controller should run your application code.

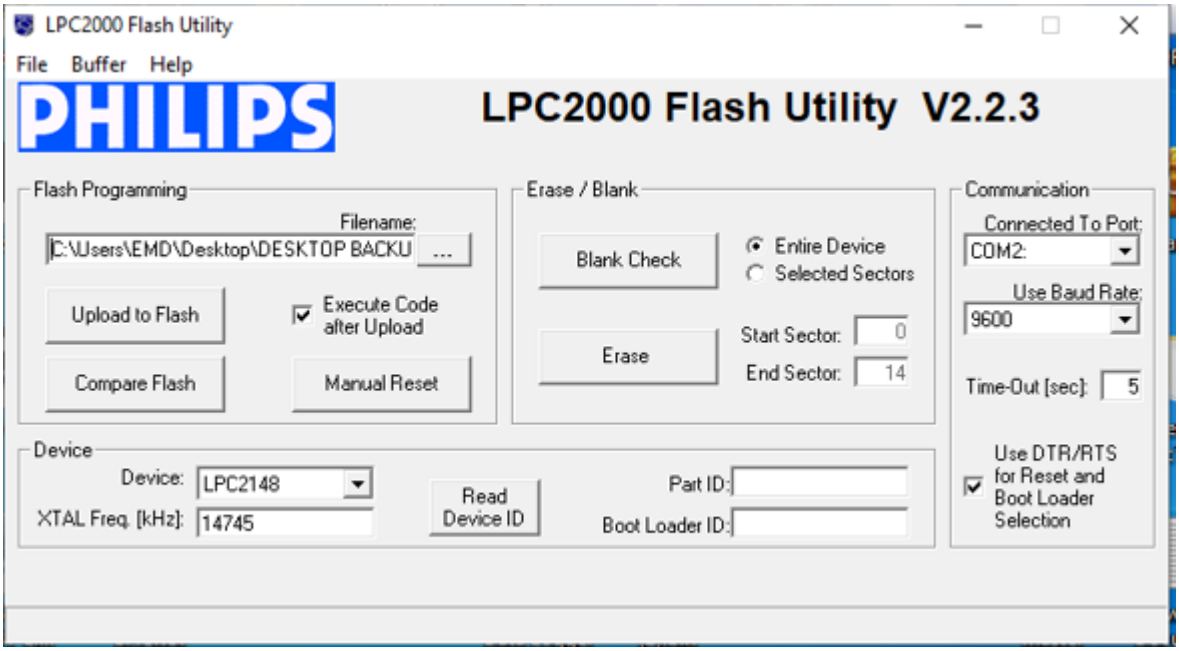


7. How to Upload Hex file Using Philips Flash Utility

Once the **".hex"** file is generated in keil, we need to upload the hex file to the hardware. We use Flash Magic Tool for the same.

Open the flash magic software and follow the below steps.

1. Press the **Read Device ID** option in the Software. Automatically detects the IC.
2. Select the COM Port. **Check the device manager for detected Com port**.
3. Select Baud rate from 9600-115200
4. Click **Use DTR/RTS for Reset and Boot Loader Selection**.
5. Check the Erase blocks used by Hex file option
6. Click **Execute Code after Upload.**
7. Browse and Select the hex file.
8. Click **Upload to Flash.**
9. Once the hex file is flashed, reset the board. Now the controller should run your application code.



8. Lab Programs

EXPERIMENT NO 1

Blinking an LED

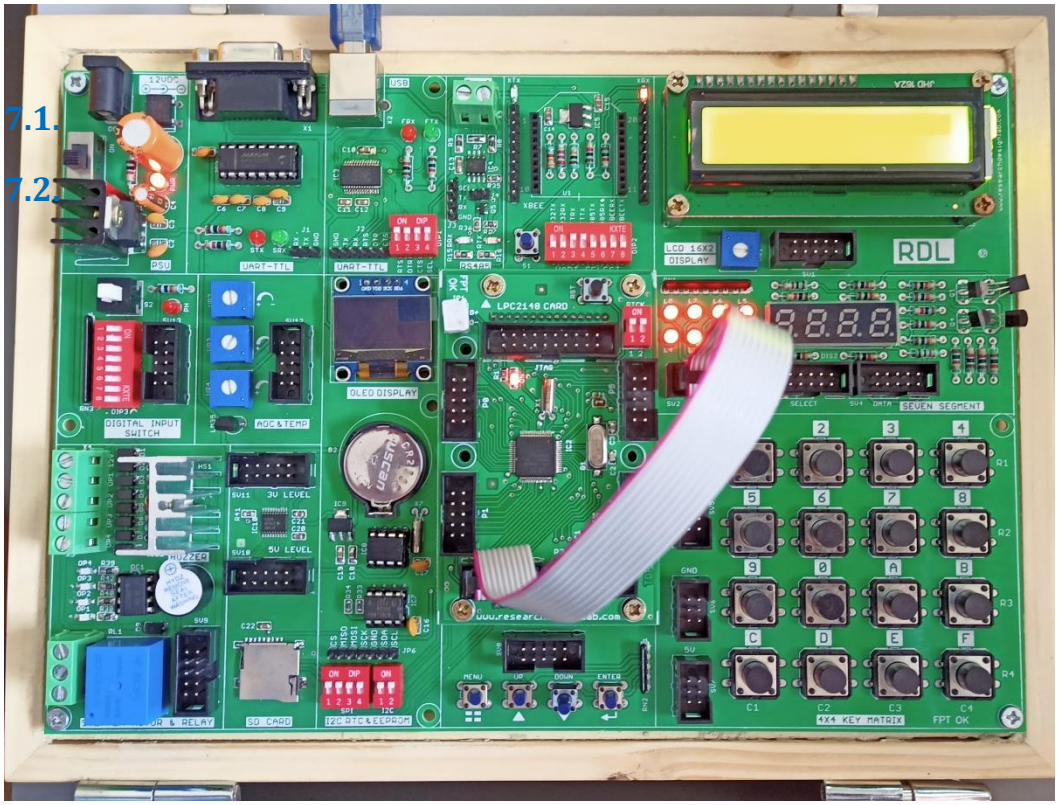
Aim:
Interfacing LED's with ARM LPC2148.

Description:

Turning ON and OFF an LED's after Particular delay.

Hardware Requirement:

ARM LPC2148 Trainer Kit, FRC cables, USB A to B cable and 12V 2A Adapter.



Program:

```
#include<LPC21xx.h>
void delay1(int);   //define delay function
int main()
{
PINSEL0=0x00000000;     //select pins for blinking led
IODIR0=0x00ffffff;      //select I/O pins as output
while(1)
{
 IOSET0=0x00ffffff;     //sets pins high
 delay1(1000);                          //gives 1 SEC delay
 IOCLR0=0x00ffffff;     //clears pins
 delay1(1000);                          //gives 1 SEC delay
}
}
/*  Delay routine; gives an approximate delay in milliseconds   */

void delay1(int d)      // delay routine
{
    int i;
   while(d)
   {
     for(i=0;i<7000;i++){;}
      d--;
   }
}
```

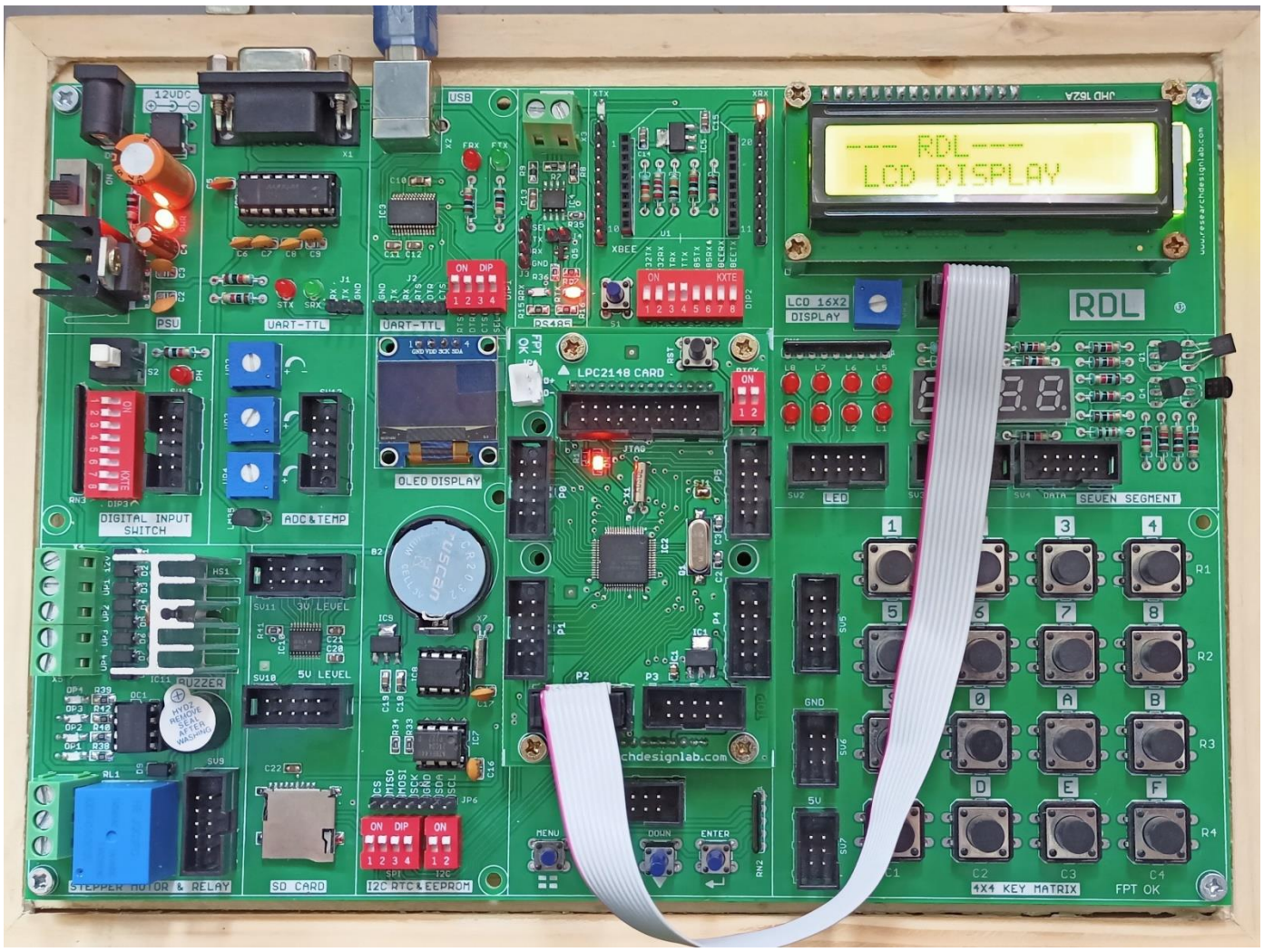
EXPERIMENT NO 2

Liquid Crystal Display

Aim:
Interfacing LCD Display with ARM LPC2148

Description:
To display the message on the LCD screen.

Hardware required:
ARM LPC2148 Trainer Kit,FRC cables,USB A to B cable and 12V 2A Power Adapter.



Program:

```
#include <lpc214x.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

void delay_ms(uint16_t j) /* Function for delay in milliseconds */
{
   uint16_t x,i;
    for(i=0;i<j;i++)
    {
   for(x=0; x<6000; x++);    /* loop to generate 1 millisecond delay with Cclk = 60MHz */
    }
}

void LCD_CMD(char command)
{
    IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((command & 0xF0)<<16) ); /* Upper nibble of command */
    IO0SET = 0x00020000; /* EN = 1 */
    IO0CLR = 0x00010000; /* RS = 0, RW = 0 */
    delay_ms(5);
    IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = RW = 0)   */
    delay_ms(5);
    IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((command & 0x0F)<<20) ); /* Lower nibble of command */
    IO0SET = 0x00020000; /* EN = 1 */
    IO0CLR = 0x00010000; /* RS = 0, RW = 0 */
    delay_ms(5);
    IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = RW = 0)   */
    delay_ms(5);

}

void LCD_INIT(void)
{
    IO0DIR = 0x00FFFFF0; /* P0.20 to P0.23 LCD Data. P0.16,,17 as RS RW and EN */
    delay_ms(20);
    LCD_CMD(0x02);  /* Initialize lcd in 4-bit mode */
    LCD_CMD(0x28);  /* 2 lines */
    LCD_CMD(0x0C);   /* Display on cursor off */
    LCD_CMD(0x06);  /* Auto increment cursor */
    LCD_CMD(0x01);   /* Display clear */
    LCD_CMD(0x80);  /* First line first position */
}
```

```
void LCD_STRING (char* msg)
{
    uint8_t i=0;
    while(msg[i]!=0)
    {
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg[i] & 0xF0)<<16) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg[i] & 0x0F)<<20) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            i++;
    }
}
void LCD_CHAR (char msg)
{
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg & 0xF0)<<16) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg & 0x0F)<<20) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
}
int main(void)
{
    LCD_INIT();
    LCD_STRING("--- RDL---");
    LCD_CMD(0xC0);
    LCD_STRING(" LCD DISPLAY");

    return 0;
}
```

EXPERIMENT NO 3

ADC

Aim:

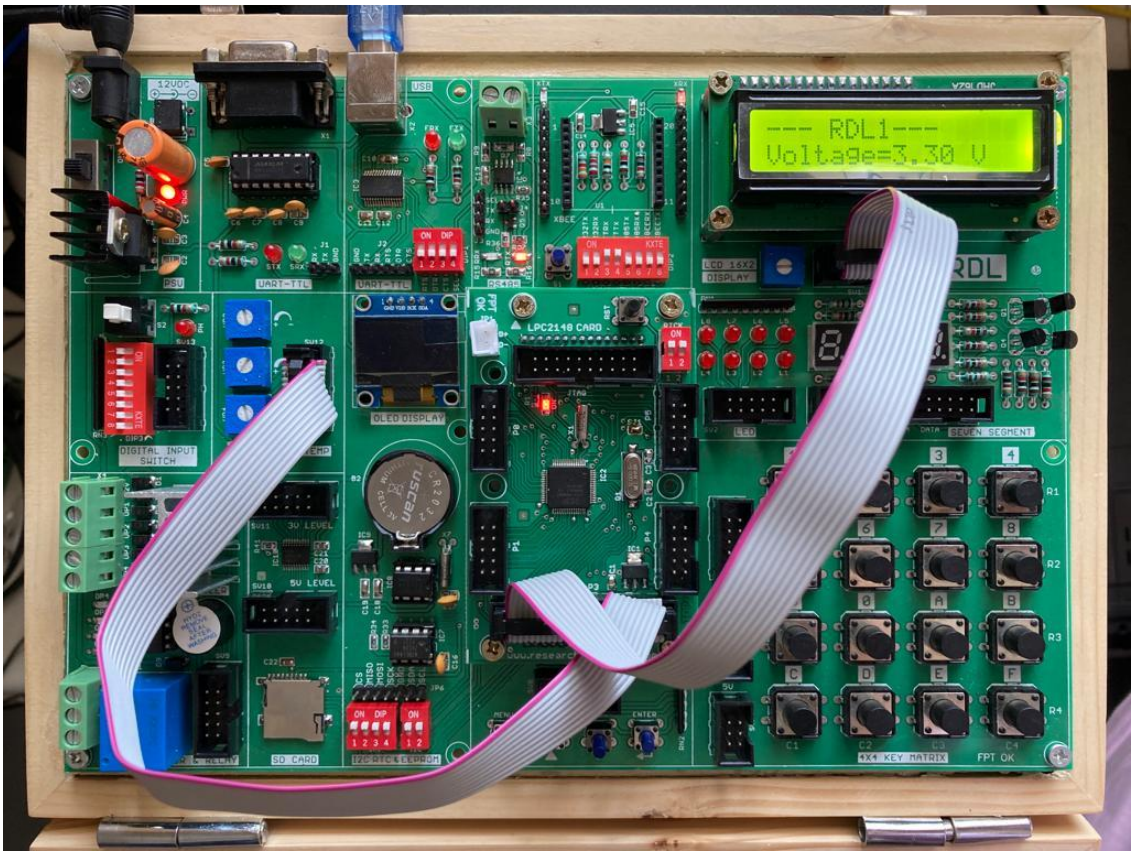
Interfacing ADC with ARM LPC2148.

Description:

To learn how to read ADC Values and display the values in the LCD.

Hardware required:

ARM LPC2148 Trainer Kit,FRC cables,USB A to B cable and 12V 2A Power Adapter.



Program

```
#include <lpc214x.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>

void delay_ms(uint16_t j) /* Function for delay in milliseconds */
{
   uint16_t x,i;
    for(i=0;i<j;i++)
    {
   for(x=0; x<6000; x++);    /* loop to generate 1 millisecond delay with Cclk = 60MHz */
    }
}

void LCD_CMD(char command)
{
    IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((command & 0xF0)<<16) ); /* Upper nibble of command */
    IO0SET = 0x00020000; /* EN = 1 */
    IO0CLR = 0x00010000; /* RS = 0, RW = 0 */
    delay_ms(5);
    IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = RW = 0)   */
    delay_ms(5);
    IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((command & 0x0F)<<20) ); /* Lower nibble of command */
    IO0SET = 0x00020000; /* EN = 1 */
    IO0CLR = 0x00010000; /* RS = 0, RW = 0 */
    delay_ms(5);
    IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = RW = 0)   */
    delay_ms(5);

}

void LCD_INIT(void)
{
    IO0DIR = 0x00FFFFF0; /* P0.12 to P0.15 LCD Data. P0.4,5,6 as RS RW and EN */
    delay_ms(20);
    LCD_CMD(0x02);  /* Initialize lcd in 4-bit mode */
    LCD_CMD(0x28);  /* 2 lines */
    LCD_CMD(0x0C);   /* Display on cursor off */
    LCD_CMD(0x06);  /* Auto increment cursor */
```

```
    LCD_CMD(0x01);   /* Display clear */
    LCD_CMD(0x80);  /* First line first position */
}

void LCD_STRING (char* msg)
{
    uint8_t i=0;
    while(msg[i]!=0)
    {
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg[i] & 0xF0)<<16) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg[i] & 0x0F)<<20) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            i++;
    }
}

void LCD_CHAR (char msg)
{
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg & 0xF0)<<16) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg & 0x0F)<<20) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
}
    /*
int main(void)
{
    LCD_INIT();
    LCD_STRING("--- RDL---");
```

```
    LCD_CMD(0xC0);
    LCD_STRING(" LCD DISPLAY");


    return 0;
} */
 int main(void)
{
 uint32_t result;
//  float voltage;
    char volt[18];

//  LCD_CMD(0xC0);
//  LCD_STRING(" LCD DISPLAY");
    LCD_INIT();
            PINSEL1 = 0x01000000; /* P0.28 as AD0.1 */
    AD0CR = 0x00200402; /* ADC operational, 10-bits, 11 clocks for conversion */

    LCD_STRING("--- RDL1---");
    while(1)
    {
            AD0CR = AD0CR | (1<<24); /* Start Conversion */
            while ( !(AD0DR1 & 0x80000000) ); /* Wait till DONE */
            result = AD0DR1;
            result = (result>>6);
            result = (result & 0x000003FF);
    //      voltage = ( (result/1023.0) * 3.3 ); /* Convert ADC value to equivalent voltage */
            LCD_CMD(0xc0);
    //      sprintf(volt, "Voltage=%.2f V  ", voltage);
            sprintf(volt, "ADC=%i  ", result);
            LCD_STRING(volt);
            memset(volt, 0, 18);
    }
    }
```

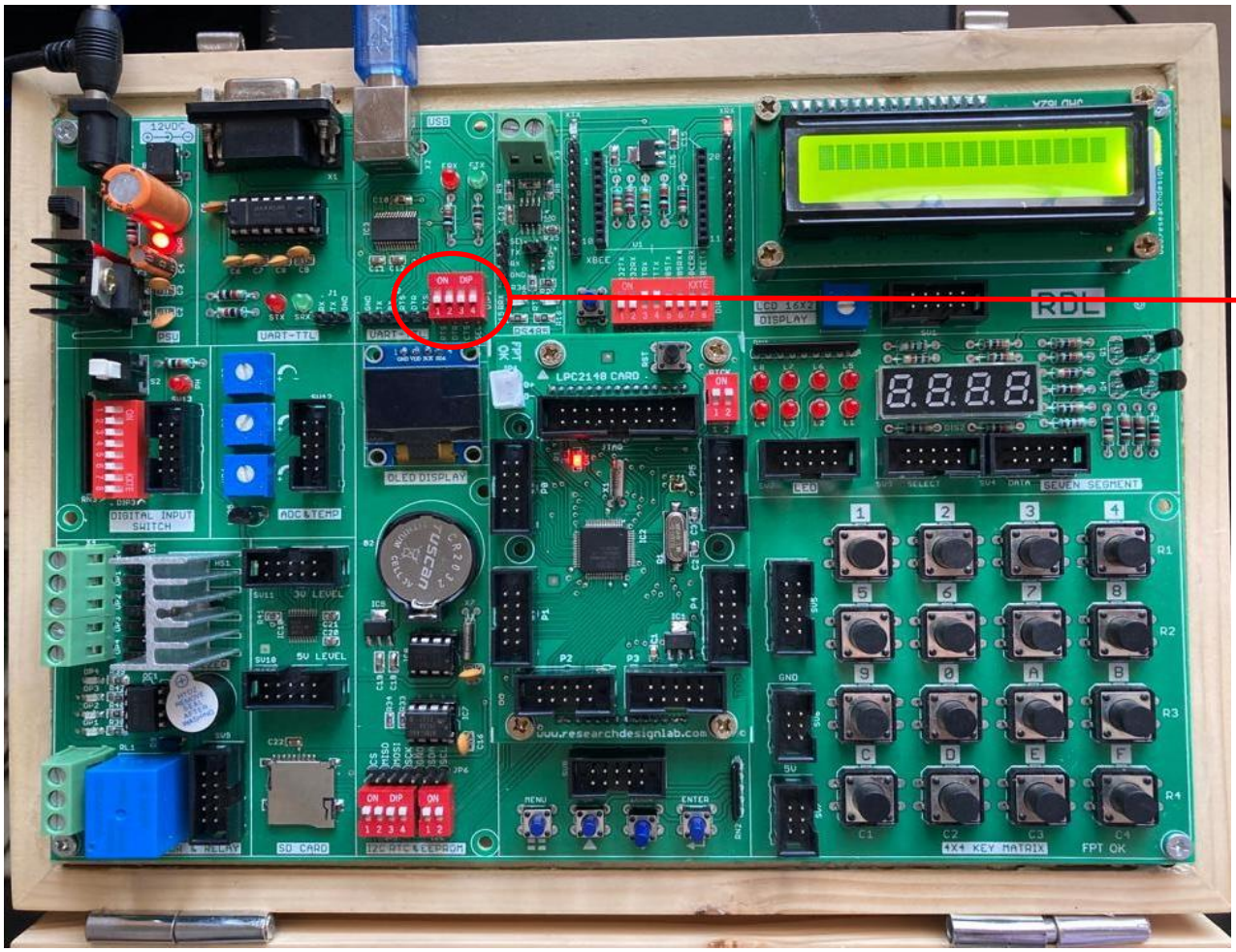
EXPERIMENT NO 4

UART

Aim:
Interfacing UART with ARM LPC2148

Description:
Transmit/Receive Data using UART and display the data's on the terminal Software.

Hardware required:
ARM LPC2148 Trainer Kit , USB A to B Cable and 12V 2A Power Adapter.



Program

```
#include <lpc214x.h>
#include <stdint.h>
#include "UART.h"

int main(void)
{
    char receive;
    UART0_init();
    while(1)
    {
            receive = UART0_RxChar();
            UART0_SendString("Received:");
            UART0_TxChar(receive);
            UART0_SendString("\r\n");
    }
}
```

EXPERIMENT NO 5

RTC (Real Time Clock)

Aim:

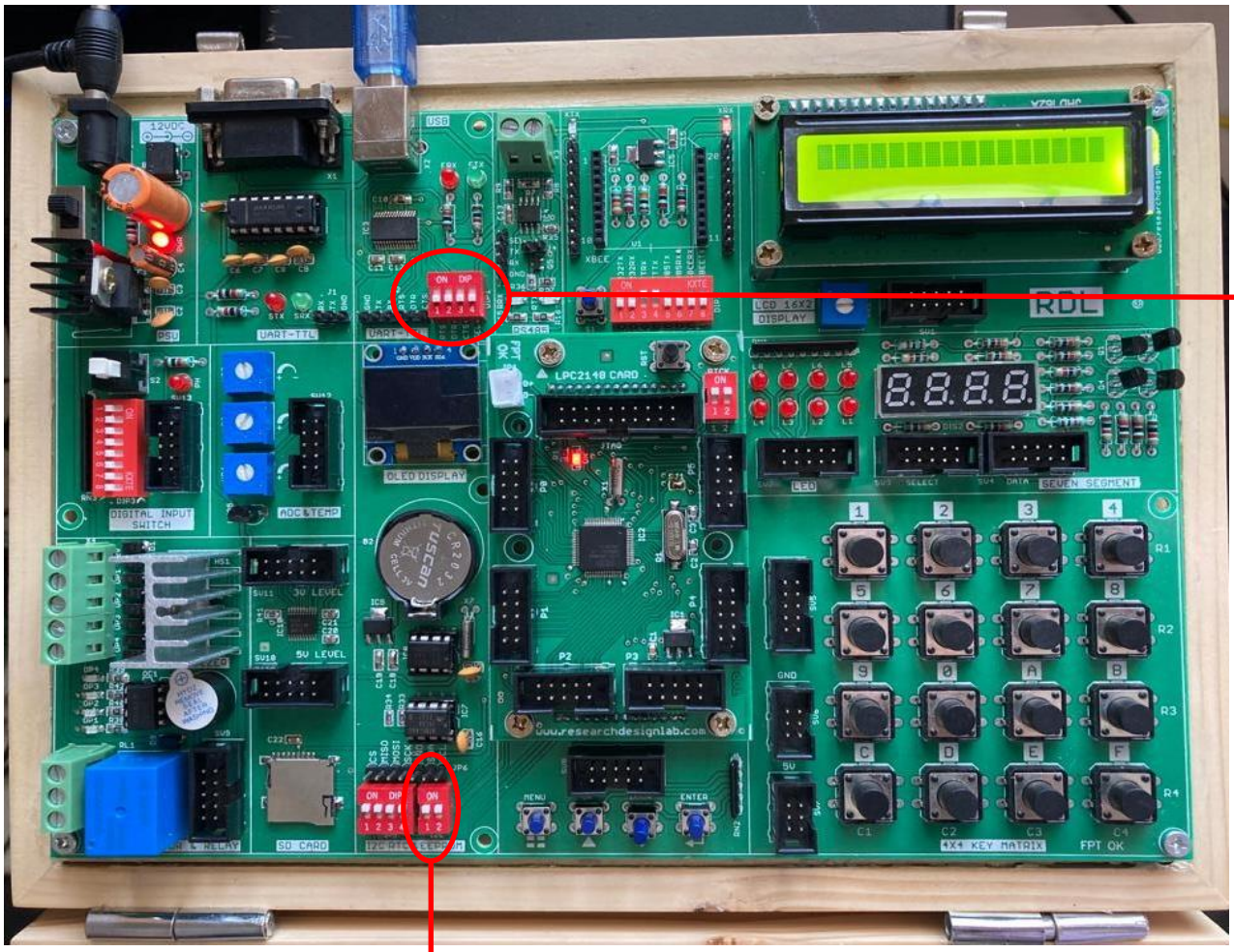
Interfacing Real Time Clock with ARM LPC2148

Description:

To transmit Date and Time using UART and display the data's on the terminal Software.

Hardware required:

ARM LPC2148 Trainer Kit, FRC Cables, USB A to B Cable and 12V 2A Power Adapter.



Program

```
#include <lpc214x.h>
#include <stdint.h>
#include <stdio.h>
#include "UART0.h"
uint8_t alarm, flag;
__irq void RTC_ISR(void)
{
    if (ILR & 0x01)
    {
            flag = 1;
            ILR = ILR | 0x01;
    }
    if (ILR & 0x02)
    {
            alarm = 1;
            ILR = ILR | 0x02;
    }
    VICVectAddr = 0;
}

typedef struct
{
    uint8_t seconds;
    uint8_t minutes;
    uint8_t hours;
    uint8_t day_of_month;
    uint8_t day_of_week;
    uint16_t day_of_year;
    uint8_t month;
    uint16_t year;
}
RTC_Time;
void RTC_Set_Time( RTC_Time Set_Time)
{
    SEC = Set_Time.seconds;
    MIN = Set_Time.minutes;
    HOUR = Set_Time.hours;
    DOM = Set_Time.day_of_month;
    DOW = Set_Time.day_of_week;
    DOY = Set_Time.day_of_year;
    MONTH = Set_Time.month;
    YEAR = Set_Time.year;
}
```

```
void RTC_Set_Alarm_Time( RTC_Time Alarm_Time)
{
    ALSEC = Alarm_Time.seconds;
    ALMIN = Alarm_Time.minutes;
    ALHOUR = Alarm_Time.hours;
    ALDOM = Alarm_Time.day_of_month;
    ALDOW = Alarm_Time.day_of_week;
    ALDOY = Alarm_Time.day_of_year;
    ALMON = Alarm_Time.month;
    ALYEAR = Alarm_Time.year;
}

RTC_Time RTC_Get_Time(void)
{
    RTC_Time time;
    time.seconds = SEC;
    time.minutes = MIN;
    time.hours = HOUR;
    time.day_of_month = DOM;
    time.day_of_week = DOW;
    time.day_of_year = DOY;
    time.month = MONTH;
    time.year = YEAR;
    return time;
}

int main(void)
{
    /* Setting Time + Alarm */
    RTC_Time set_time, alarm_time, current_time;
    char timestr[30], datestr[30];
    alarm = 0;
    flag = 0;
    IO0DIR = 0x00000010; /* P0.4 as output pin for LED */
    UART0_init();
    PCONP = (PCONP | (1<<9)); /* PCRTC = 1 */
    /* The RTC registers cannot be written to unless we make PCRTC = 1 */
    ILR = 0x0; /* No RTC interrupts */
    CCR = 0x12; /* 32.768kHz clock and Reset Clock Tick Counter */
    CCR = 0x10;
    CIIR = 0x00; /* No interrupts */
    AMR = 0x00; /* Alarm registers not masked */
    VICVectAddr0 = (unsigned) RTC_ISR;
    VICVectCntl0 = 0x0000002D;
```

```
    VICIntEnable = 0x00002000;
    VICIntSelect = 0x00000000;
    set_time.seconds = 00;
    set_time.minutes = 25;
    set_time.hours = 11;
    set_time.day_of_month = 6;
    set_time.day_of_week = 5;
    set_time.day_of_year = 279;
    set_time.month = 10;
    set_time.year = 2017;
    RTC_Set_Time(set_time);
    CIIR = 0x01; /* Seconds value increment interrupt */
    alarm_time.seconds = 15;
    alarm_time.minutes = 25;
    alarm_time.hours = 11;
    alarm_time.day_of_month = 6;
    alarm_time.day_of_week = 5;
    alarm_time.day_of_year = 279;
    alarm_time.month = 10;
    alarm_time.year = 2017;
    RTC_Set_Alarm_Time(alarm_time);
    CCR = 0x11; /* 32.768kHz clock and clock Enable */
    ILR = 0x03; /* RTC interrupts enabled */
    IO0CLR = 0x00000010;
    /* Set the Time and Alarm once using above code lines */
    /* Once the time and alarm is set, comment out the above code lines and uncomment the code
lines for "Only RTC Read" and program the device */
    /* If this is not done, the time will be set repeatedly to same value whenever the device is reset or
powered */


    /* Only RTC Read */
//  RTC_Time current_time;
//  char timestr[30], datestr[30];
//  alarm = 0;
//  flag = 0;
//  IO0DIR = 0x00000010; /* P0.4 as output pin for LED */
//  UART0_init();
//  AMR = 0x00; /* Alarm registers not masked */
//  CCR = 0x10;
//  VICVectAddr0 = (unsigned) RTC_ISR;
//  VICVectCntl0 = 0x0000002D;
//  VICIntEnable = 0x00002000;
//  VICIntSelect = 0x00000000;
//  CCR = 0x11; /* 32.768kHz clock and clock enable */
```

```
//  ILR = 0x03; /* RTC interrupts enabled */
//  IO0CLR = 0x00000010;

   /* Code lines below are common for "Setting time + Alarm" as well as "Only RTC Read" */
    while(1)
    {
            if(alarm == 1)
            {
                    current_time = RTC_Get_Time();
                    sprintf(timestr,"Alarm!!!: %d:%d:%d
\r\n",current_time.hours,current_time.minutes,current_time.seconds);
                    UART0_SendString(timestr);
                    uint8_t i;
                    for(i=0;i<10;i++)
                    {
                            IO0SET = 0x00000010;
                            delay_ms(300);
                            IO0CLR = 0x00000010;
                            delay_ms(300);
                    }
                    alarm = 0;
            }
            if (flag == 1)
            {
                    current_time = RTC_Get_Time();
                    sprintf(timestr,"Time: %d:%d:%d
",current_time.hours,current_time.minutes,current_time.seconds);
                    sprintf(datestr,"Date: %d/%d/%d
\r\n",current_time.day_of_month,current_time.month,current_time.year);
                    UART0_SendString(timestr);
                    UART0_SendString(datestr);
                    flag = 0;
            }
    }
}
```

EXPERIMENT NO 6

Hex Keypad

Aim:

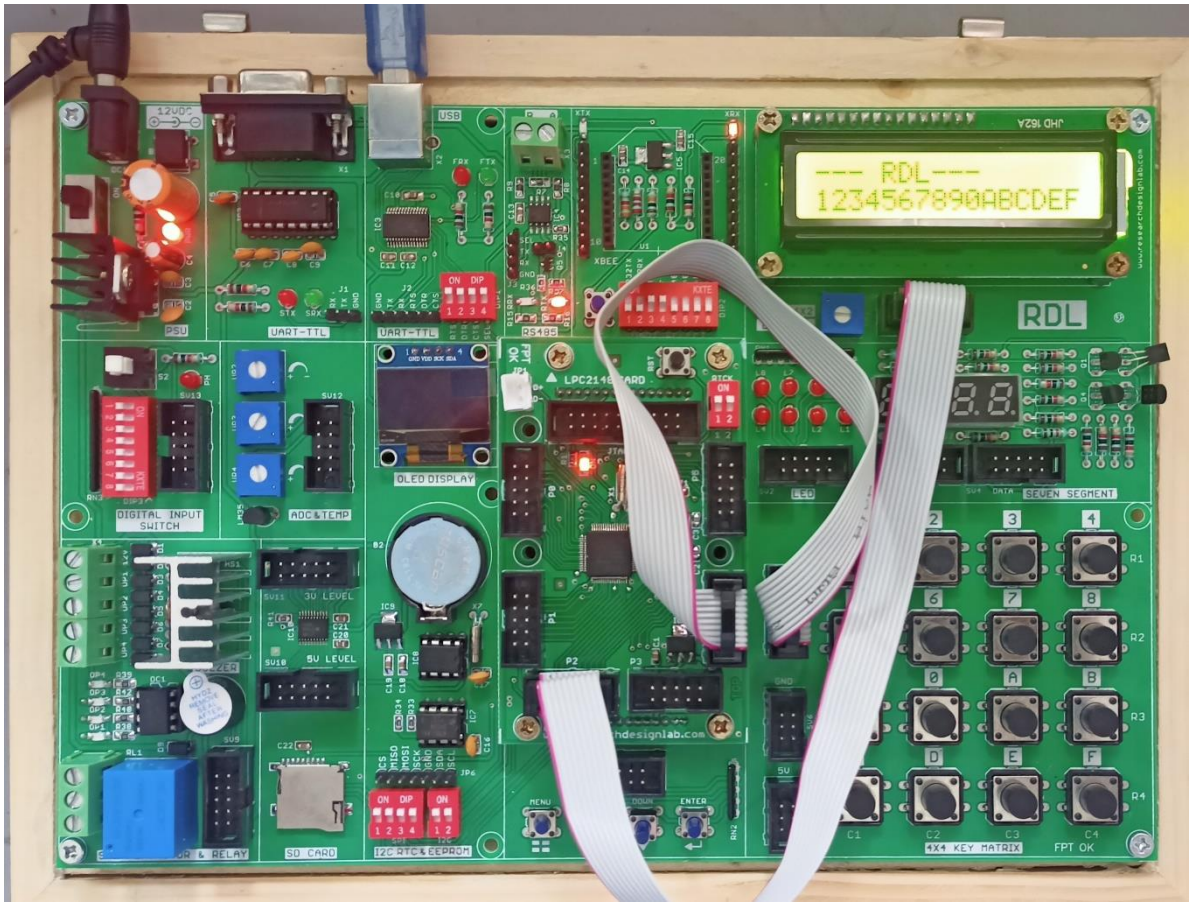
To interface 4x4 Hex keypad with ARM LPC2148.

Description:

To display the pressed key on the LCD Display.

Hardware Required:

ARM LPC2148 Trainer Kit, FRC Cables, USB A to B Cable and 12V 2A Power Adapter.



Program

```
#include <lpc214x.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include<string.h>
unsigned int adc_value1,adc_value2,C;
unsigned char buf[16] = {0};
char check_key(void);
 unsigned int KeyPort=0x00f00000;

#define sw1 0x00010000
#define sw2 0x00020000
#define sw3 0x00040000
#define sw4 0x00080000

void delay_ms(uint16_t j) /* Function for delay in milliseconds */
{
   uint16_t x,i;
    for(i=0;i<j;i++)
    {
   for(x=0; x<6000; x++);    /* loop to generate 1 millisecond delay with Cclk = 60MHz */
    }
}

void LCD_CMD(char command)
{
    IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((command & 0xF0)<<16) ); /* Upper nibble of command */
    IO0SET = 0x00020000; /* EN = 1 */
    IO0CLR = 0x00010000; /* RS = 0, RW = 0 */
    delay_ms(5);
    IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = RW = 0)   */
    delay_ms(5);
    IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((command & 0x0F)<<20) ); /* Lower nibble of command */
    IO0SET = 0x00020000; /* EN = 1 */
    IO0CLR = 0x00010000; /* RS = 0, RW = 0 */
    delay_ms(5);
    IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = RW = 0)   */
    delay_ms(5);

}
```

```
void LCD_INIT(void)
{
    IO0DIR = 0x00FFFFF0; /* P0.20 to P0.23 LCD Data. P0.16,,17 as RS RW and EN */
    delay_ms(20);
    LCD_CMD(0x02);  /* Initialize lcd in 4-bit mode */
    LCD_CMD(0x28);  /* 2 lines */
    LCD_CMD(0x0C);   /* Display on cursor off */
    LCD_CMD(0x06);  /* Auto increment cursor */
    LCD_CMD(0x01);   /* Display clear */
    LCD_CMD(0x80);  /* First line first position */
}

void LCD_STRING (char* msg)
{
    uint8_t i=0;
    while(msg[i]!=0)
    {
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg[i] & 0xF0)<<16) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg[i] & 0x0F)<<20) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            i++;
    }
}

void LCD_CHAR (char msg)
{
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg & 0xF0)<<16) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
            delay_ms(2);
            IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
            delay_ms(5);
            IO0PIN = ( (IO0PIN & 0xFF0FFFFF) | ((msg & 0x0F)<<20) );
            IO0SET = 0x00030000; /* RS = 1, EN = 1 */
            IO0CLR = 0x00000020; /* RW = 0 */
```

```
                delay_ms(2);
                IO0CLR = 0x00020000; /* EN = 0, RS and RW unchanged(i.e. RS = 1, RW = 0) */
                delay_ms(5);
}

int main(void)
{
   IODIR1=KeyPort;
     LCD_INIT();
     LCD_STRING("--- RDL---");
     LCD_CMD(0xC0);


     while(1)
{

delay_ms(5);
LCD_CHAR(check_key());
 }

}
     // IODIR0=0X0007ffCF0;

 char check_key(void)
     {

     while(1)
     {
       //

     IOSET1=0x00EF0000;
   delay_ms(5);
     if((IOPIN1&sw1)==0)
     {
     C='1';
     while((IOPIN1&sw1)==0);
     return(C);

     }
     if((IOPIN1&sw2)==0)
     {
     C='5';
     while((IOPIN1&sw2)==0);
     return(C);
     }
```

```
if((IOPIN1&sw3)==0)
{

C='9';
while((IOPIN1&sw3)==0);
return(C);

}
if((IOPIN1&sw4)==0)
{
C='C';
while((IOPIN1&sw4)==0);
return(C);
}
IOCLR1=0x00EF0000;
delay_ms(5);
IOSET1=0x00DF0000;
delay_ms(5);
if((IOPIN1&sw1)==0)
{
C='2';
while((IOPIN1&sw1)==0);
return(C);

}
if((IOPIN1&sw2)==0)
{
C='6';
while((IOPIN1&sw2)==0);
return(C);
}
if((IOPIN1&sw3)==0)
{
C='0';
while((IOPIN1&sw3)==0);
return(C);
}
if((IOPIN1&sw4)==0)
{
C='D';
while((IOPIN1&sw4)==0);
return(C);
```

```
    }

    IOCLR1=0x00DF0000;
delay_ms(5);
    IOSET1=0x00BF0000;
delay_ms(5);
    if((IOPIN1&sw1)==0)
    {
    C='3';
    while((IOPIN1&sw1)==0);
    return(C);
    }
    if((IOPIN1&sw2)==0)
    {
    C='7';
    while((IOPIN1&sw2)==0);
    return(C);
    }
    if((IOPIN1&sw3)==0)
    {
    C='A';
    while((IOPIN1&sw3)==0);
    return(C);
    }
    if((IOPIN1&sw4)==0)
    {
    C='E';
    while((IOPIN1&sw4)==0);
    return(C);
    }
    IOCLR1=0x00BF0000;
    delay_ms(5);
    IOSET1=0x007F0000;
    delay_ms(5);
    if((IOPIN1&sw1)==0)
    {
    C='4';
    while((IOPIN1&sw1)==0);
    return(C);

    }
    if((IOPIN1&sw2)==0)
    {
    C='8';
    while((IOPIN1&sw2)==0);
```

```
    return(C);
    }


    if((IOPIN1&sw3)==0)
    {
    C='B';
    while((IOPIN1&sw3)==0);
    return(C);
    }
    if((IOPIN1&sw4)==0)
    {
    C='F';
    while((IOPIN1&sw4)==0);
    return(C);
    }
    IOCLR1=0x007F0000;
delay_ms(5);

    }

    }
```

EXPERIMENT NO 7

Stepper Motor

Aim:

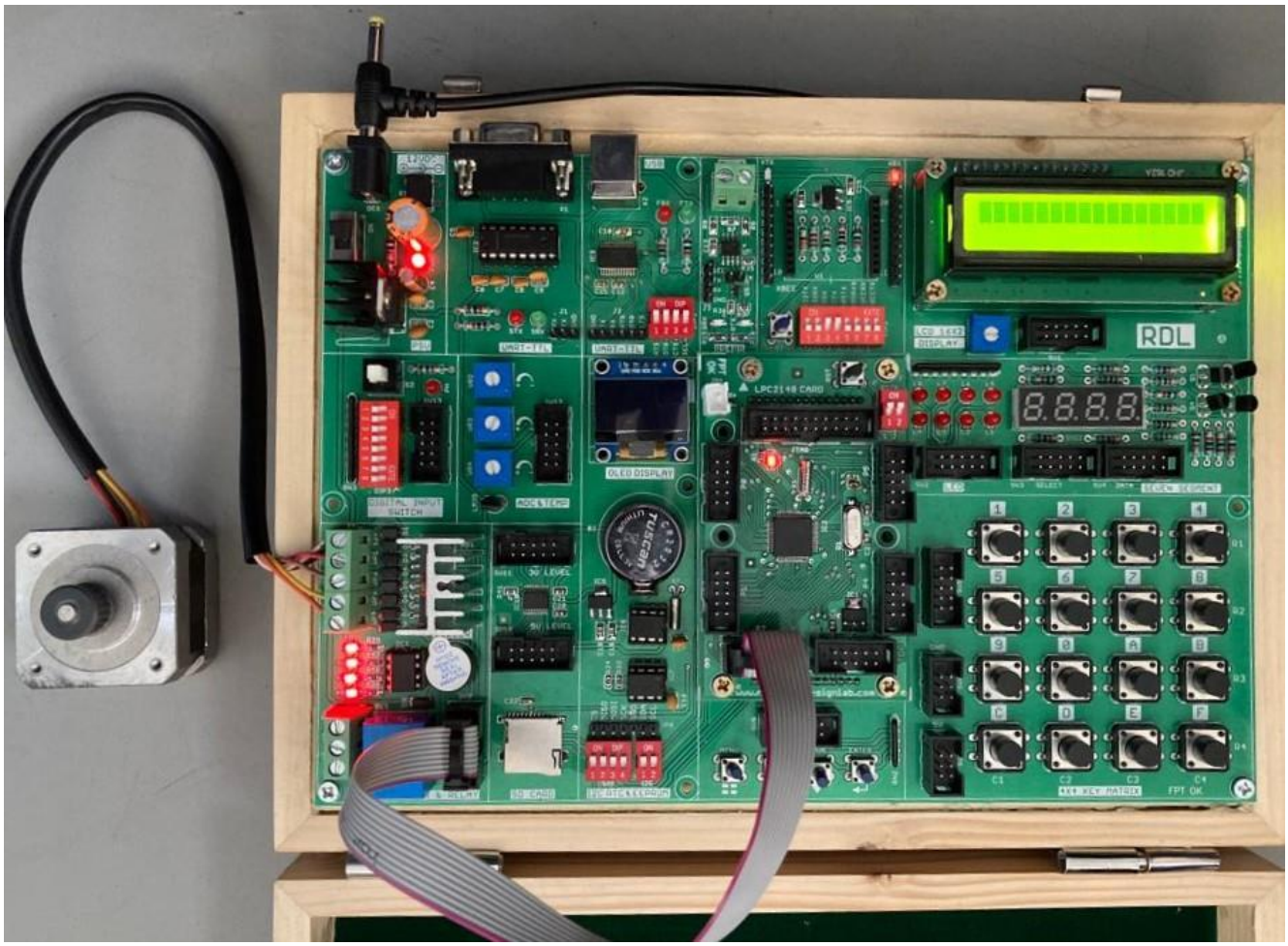
To interface Stepper Motor with ARM LPC2148.

Description:

To rotate Stepper Motor using ARM LPC2148.

Hardware Required:

ARM LPC2148 Trainer Kit, Stepper Motor, FRC Cables, USB A to B Cable and 12V 2A Power Adapter.



Program

```
#include<LPC21xx.h>

void _delay_ms(int j)

{

int x,l;

     for(l=0;l<j;l++)

     {

   for(x=0; x<6000; x++);    /* loop to generate 1 milisecond delay with Cclk = 60MHz */

     }

}

int main()

{

 int i=0;

PINSEL0=0x00000000;     //select pins for blinking led

IODIR0=0x00ff0000;        //select I/O pins as output

while(1)

{

  for(i=0;i<100;i++)

                {

                        IOSET0 = 0x00EC0000;

                        _delay_ms(4);

                        IOCLR0 = 0x00EC0000;

                        IOSET0 = 0x00DC0000;

                        _delay_ms(4);
```

```
                IOCLR0=0x00DC0000;
                IOSET0 = 0x00BC0000;
                _delay_ms(4);
                IOCLR0=0x00BC0000;
                IOSET0 = 0x007C0000;
                _delay_ms(4);
                IOCLR0=0x007C0000;
        }
        for(i=0;i<100;i++)
        {
                IOSET0 = 0x007C0000;
                _delay_ms(4);
                IOCLR0=0x007C0000;
                IOSET0 = 0x00BC0000;
                _delay_ms(4);
                IOCLR0=0x00BC0000;
                IOSET0 = 0x00DC0000;
                _delay_ms(4);
                  IOCLR0=0x00DC0000;
                IOSET0 = 0x00EC0000;
                _delay_ms(4);
                IOCLR0=0x00EC0000;
        }
        }
}

/*  Delay routine;gives an approximate delay in milliseconds    */
```

EXPERIMENT NO 8

PWM

Aim:

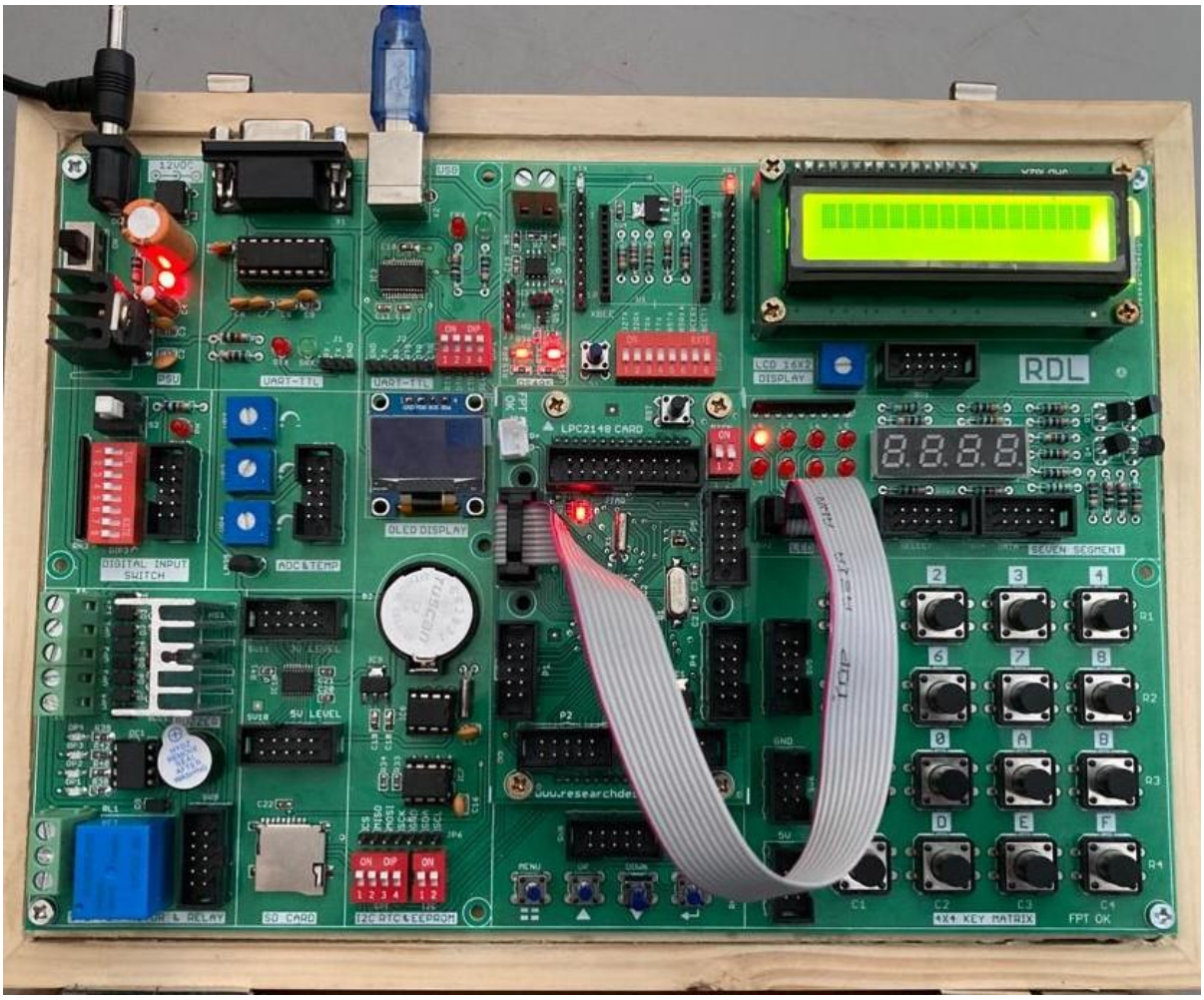
To interface PWM with ARM LPC2148.

Description:

To control the Brightness of the LED through PWM using ARM LPC2148.

Hardware Required:

ARM LPC2148 Trainer Kit, FRC Cables, USB A to B Cable and 12V 2A Power Adapter.



Program

```
#include <lpc214x.h>
#define PLOCK 0x00000400
#define PWMPRESCALE 60   //60 PCLK cycles to increment TC by 1 i.e 1 Micro-second
void initPWM(void);
void initClocks(void);
void setupPLL0(void);
void feedSeq(void);
void connectPLL0(void);
void delay_ms(int j) /* Function for delay in milliseconds */
{
   int x,i;
    for(i=0;i<j;i++)
    {
   for(x=0; x<6000; x++);    /* loop to generate 1 millisecond delay with Cclk = 60MHz */
    }
}

int DUTY=0;
int main(void)
{
    initClocks(); //Initialize CPU and Peripheral Clocks @ 60Mhz
    initPWM(); //Initialize PWM

 //IO0DIR = 0x1; This is not needed!
    //Also by default all pins are configured as Inputs after MCU Reset.
    IO0DIR = 0x01;

   while(1)
   {

                    for(DUTY=0;DUTY<10000;DUTY++)
                    {
                    PWMMR1 = DUTY; //25% Bright
                    PWMLER = (1<<1);
                    delay_ms(1);
                    }


            /*      if( ((IO0PIN) & (1<<2)) ) // Check P0.2
                    {
                            PWMMR1 = 2500; //25% Bright
                            PWMLER = (1<<1);
                    }
```

```
                    else if( ((IO0PIN) & (1<<3)) ) // Check P0.3
                    {
                            PWMMR1 = 5000; //50% Bright
                            PWMLER = (1<<1);
                    }
                    else if( ((IO0PIN) & (1<<4)) ) // Check P0.4
                    {
                            PWMMR1 = 7500; //75% Bright
                            PWMLER = (1<<1);
                    }
                    else if( ((IO0PIN) & (1<<5)) ) // Check P0.5
                    {
                            PWMMR1 = 10000; //T-ON=100% , Hence 25% Bright
                            PWMLER = (1<<1); //Update Latch Enable bit for PWMMR1
                    }  */
    }
    //return 0; //normally this wont execute ever
}
void initPWM(void)
{
    /*Assuming that PLL0 has been setup with CCLK = 60Mhz and PCLK also = 60Mhz.*/
    /*This is a per the Setup & Init Sequence given in the tutorial*/

    PINSEL0 = 0x00000002;                   //SELECT PIN Select 0 FOR PWM1(P0.0)
    //PINSEL0 = (1<<1); // Select PWM1 output for Pin0.0
    PWMPCR = 0x0; //Select Single Edge PWM - by default its single Edged so this line can be removed
    PWMPR = PWMPRESCALE-1; // 1 micro-second resolution
    PWMMR0 = 10000; // 10ms period duration
    PWMMR1 = 2500; // 2.5ms - pulse duration i.e width (Brigtness level)
    PWMMCR = (1<<1); // Reset PWMTC on PWMMR0 match
    PWMLER = (1<<1) | (1<<0); // update MR0 and MR1
    PWMPCR = (1<<9); // enable PWM output
    PWMTCR = (1<<1) ; //Reset PWM TC & PR

    //Now , the final moment - enable everything
    PWMTCR = (1<<0) | (1<<3); // enable counters and PWM Mode

    //PWM Generation goes active now
    //Now you can get the PWM output at Pin P0.0!
}

void initClocks(void)
{
    setupPLL0();
    feedSeq(); //sequence for locking PLL to desired freq.
```

```
    connectPLL0();
    feedSeq(); //sequence for connecting the PLL as system clock

    //SysClock is now ticking @ 60Mhz!

    VPBDIV = 0x01; // PCLK is same as CCLK i.e 60Mhz
    //PLL0 Now configured!
}
//---------PLL Related Functions :---------------

void setupPLL0(void)
{
    //Note : Assuming 12Mhz Xtal is connected to LPC2148.

    PLL0CON = 0x01; // PPLE=1 & PPLC=0 so it will be enabled
                    // but not connected after FEED sequence
    PLL0CFG = 0x24; // set the multipler to 5 (i.e actually 4)
                    // i.e 12x5 = 60 Mhz (M - 1 = 4)!!!
                    // Set P=2 since we want FCCO in range!!!
                    // So , Assign PSEL =01 in PLL0CFG as per the table.
}

void feedSeq(void)
{
    PLL0FEED = 0xAA;
    PLL0FEED = 0x55;
}

void connectPLL0(void)
{
    // check whether PLL has locked on to the  desired freq by reading the lock bit
    // in the PPL0STAT register

    while( !( PLL0STAT & PLOCK ));

    // now enable(again) and connect
    PLL0CON = 0x03;
}
```

EXPERIMENT NO 9

EEPROM

Aim:

To interface EEPROM with ARM LPC2148.

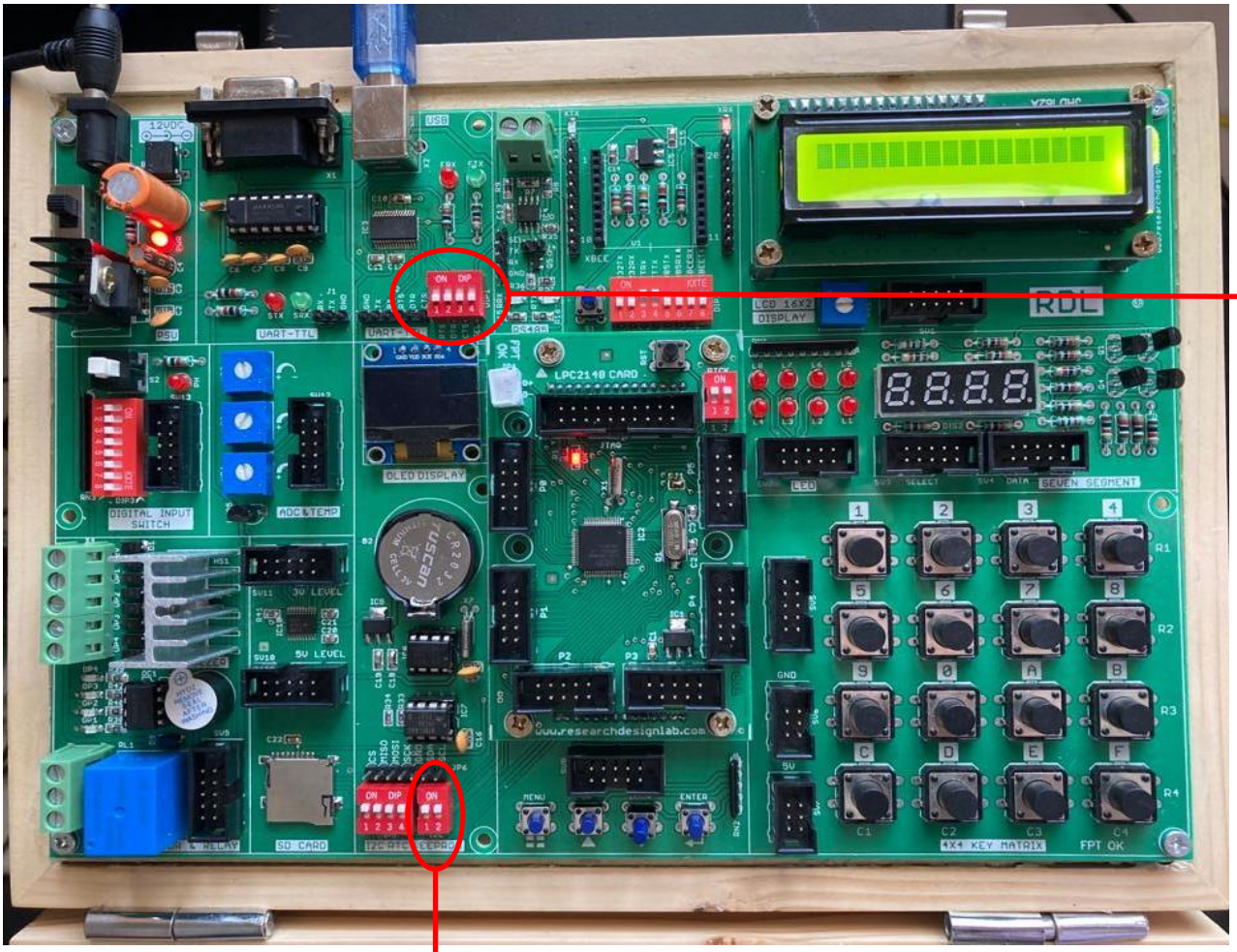
Description:

Transmit EEPROM Data using UART and display the data's on the terminal Software.

.

Hardware Required:

ARM LPC2148 Trainer Kit, FRC Cables, USB A to B Cable and 12V 2A Power Adapter.



Program

```
#include<LPC214x.h>
void I2C_init(void);
void byte_write(unsigned char address, unsigned char location, unsigned char data);
void send_start(void);
void send_stop(void);
unsigned char byte_read(unsigned char address,unsigned char location);
void msdelay(unsigned int time);

void uart0Init(void)
{
   // port 0 tx P0.1 and rx P0.0 selected
   U0LCR=0x83; //8bit data, no parity, 1 stop bit
   U0DLL=97;// 9600 baud rate @15Mhz Pclk
   U0LCR=0x03;// DLAB=0

}
void uart0Putch(unsigned char ch)
{
   U0THR=ch;    // Transmitter holding register
   while(!(U0LSR & 0x20));// wait still THR=0
}
void UART0_Txstring(unsigned char *Str)
{
int i=0;
while(Str[i]!='\0')
 {
  uart0Putch(Str[i]);
  i++;
 }
}

int main()
{
   int i;
   unsigned char read_data;
   unsigned char
write_data[10]="KANWAL";//{0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4A};
   PINSEL0=0x00000055;
   PINSEL1=0x00000000;
   PINSEL2=0x00000000;

 uart0Init();
   I2C_init();
//    LCD_writestring("Writing from I2C");
```

```
 UART0_Txstring("Writing from I2C");
   for(i=0;i<10;i++)
   {
      byte_write(0xA0,i,write_data[i]);
  uart0Putch(write_data[i]);

      msdelay(100);
      }
    //   LCD_cmd(0xC0);
  uart0Putch(0x0d);
   UART0_Txstring("reading from I2C  : ");
      for(i=0;i<6;i++)
      {
      read_data=byte_read(0xA0,i);
      uart0Putch(read_data);
     // LCD_data(read_data);
   }
}

void I2C_init(void)
{
   I2C0CONCLR=0xFF;
   I2C0CONSET=0x40;                //enable I2C
   I2C0SCLH=75;    //0x4B
   I2C0SCLL=75;    //0x4B
}

void byte_write(unsigned char address, unsigned char location, unsigned char data)
{
   I2C0CONCLR=0xFF;
   I2C0CONSET=0x40;

   send_start();                //send start condition
   while(!(I2C0CONSET&0x08));      //check SI flag
   I2C0CONCLR=0x28;              //clear SI flag and start
   I2C0DAT=address&0xFE;           //selecting address in
   while(!(I2C0CONSET&0x08));      //check SI flag
   I2C0CONCLR=0x28;              //clear SI flag and start
   I2C0DAT=location;             //sending memory location
   while(!(I2C0CONSET&0x08));      //check SI flag
   I2C0CONCLR=0x28;              //clear SI flag and start


   I2C0DAT=data;
   while(!(I2C0CONSET&0x08));      //check SI flag
```

```
   I2C0CONCLR=0x28;              //clear SI flag and start flag

   send_stop();                  //send stop bit
}

void send_start()
{
   I2C0CONSET=0x20;
}

void send_stop()
{
   I2C0CONSET=0x10;
}
unsigned char byte_read(unsigned char address,unsigned char location)
{
   unsigned char data;
   I2C0CONCLR=0xFF;
   I2C0CONSET=0x40;

   send_start();
   while(!(I2C0CONSET&0x08));       //check SI flag
   I2C0CONCLR=0x28;              //clear SI flag and start
   I2C0DAT=address&0xFE;           //selecting address in
   while(!(I2C0CONSET&0x08));       //check SI flag
   I2C0CONCLR=0x28;              //clear SI flag and start
   I2C0DAT=location;             //sending memory location
   while(!(I2C0CONSET&0x08));       //check SI flag
   I2C0CONCLR=0x28;              //clear SI flag and start
   send_start();                 //repeated start
   while(!(I2C0CONSET&0x08));       //check SI flag
   I2C0CONCLR=0x28;
   I2C0DAT=address|0x01;
   while(!(I2C0CONSET&0x08));       //check SI flag
   I2C0CONCLR=0x28;
   I2C0CONCLR=0x04;              //NACK

   while(!(I2C0CONSET&0x08));       //check SI flag
   I2C0CONCLR=0x28;
   data=I2C0DAT;
   send_stop();

   return data;
}
```

```
void msdelay(unsigned int time)
{
   int i,j;
   for(i=0;i<time;i++)
   {
      for(j=0;j<1008;j++)
      {
      }
```

EXPERIMENT NO 10

Arithmetic Operation

Aim:

To do arithmetic operations with ARM LPC2148.

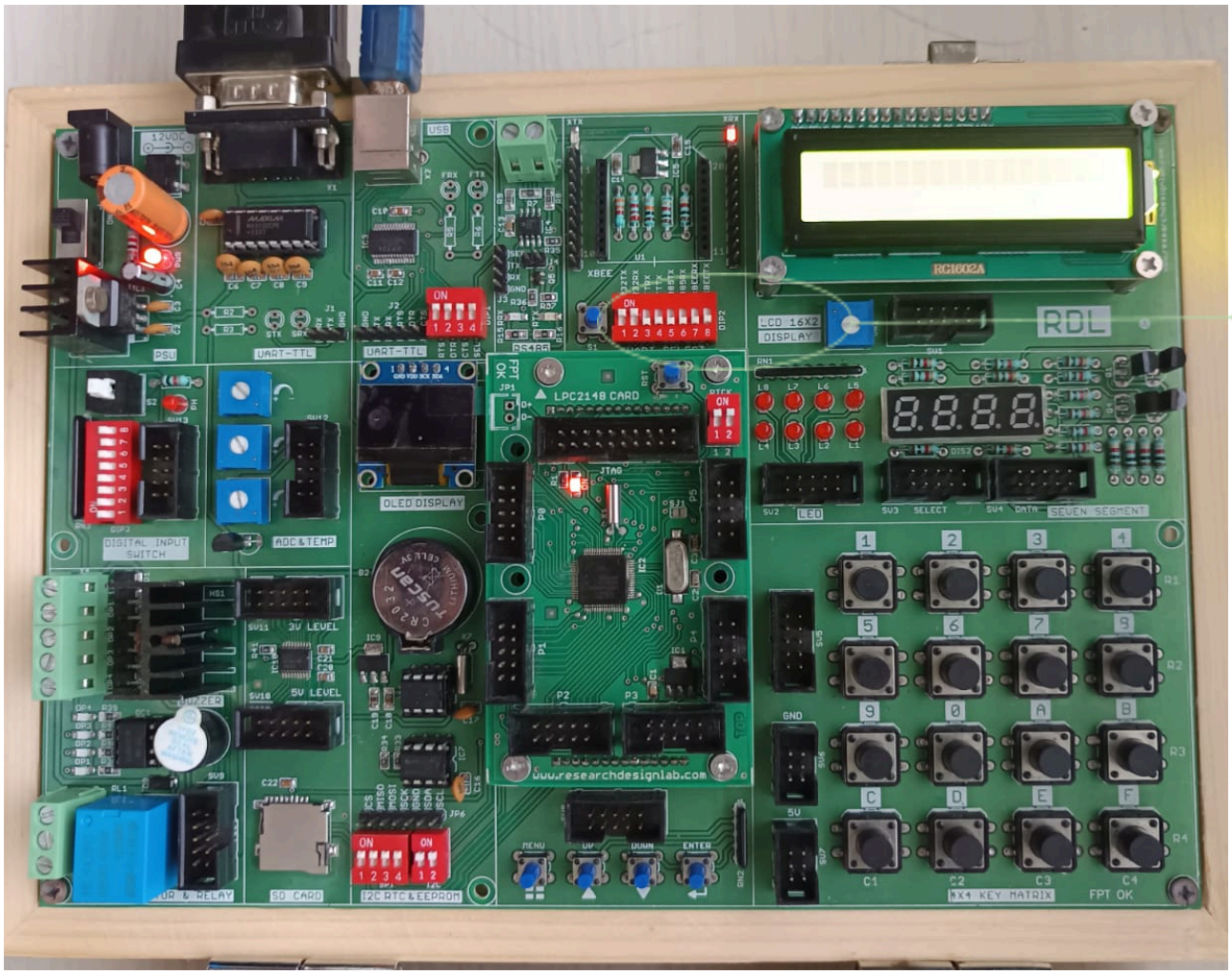
Description:

Send Arithmetic operator and numbers in tera term and its result will be displayed in teraterm.

.

Hardware Required:

ARM LPC2148 Trainer Kit, RS232 cable, USB A to B Cable and 12V 2A Power Adapter.



Make 232 TX and 232RX High after uploading the program

Program

In main.c

```
#include <stdio.h>              /* prototype declarations for I/O functions */
#include <LPC214x.H>            /* LPC21xx definitions                      */
#include "Serial.h"

#define UART0_TEXT "\n\r RDL TECHNOLOGIES PVT LTD ... Arithmetic Operations"

void delay(int count)
{
   int j=0,i=0;

   for(j=0;j<count;j++)
   {
      /* At 60Mhz, the below loop introduces
      delay of 10 us */
      for(i=0;i<35;i++);
   }
}

/* Function to convert a string to an integer */
int atoi(char *str) {
   int num = 0;
   while (*str) {
      num = num * 10 + (*str - '0');
      str++;
   }
   return num;
}

/* Function to perform arithmetic operations */
void perform_operation(char operation, int num1, int num2)
{
   int result;
   char buffer[100];

   switch(operation) {
```

```
        case '+':
            result = num1 + num2;
            sprintf(buffer, "Result: %d + %d = %d\n\r", num1, num2, result);
            break;
        case '-':
            result = num1 - num2;
            sprintf(buffer, "Result: %d - %d = %d\n\r", num1, num2, result);
            break;
        case '*':
            result = num1 * num2;
            sprintf(buffer, "Result: %d * %d = %d\n\r", num1, num2, result);
            break;
        case '/':
            if (num2 != 0) {
                result = num1 / num2;
                sprintf(buffer, "Result: %d / %d = %d\n\r", num1, num2, result);
            } else {
                sprintf(buffer, "Error: Division by zero\n\r");
            }
            break;
        default:
            sprintf(buffer, "Error: Invalid operation\n\r");
            break;
    }

    uart0_puts(buffer);
}

/* Function to read a string from UART */
void uart0_gets(char *buf, int max_len) {
    int i = 0;
    char ch;
    while (i < max_len - 1) {
        ch = uart0_getkey();  // Get character from UART
        if (ch == '\n' || ch == '\r') {
            buf[i] = '\0';
            return;
        }
        buf[i++] = ch;
    }
    buf[i] = '\0';  // Null-terminate the string
```

```
}

int main (void)  {              /* execution starts here                  */
   char operation;
   char num1_str[20], num2_str[20];
   int num1, num2;

   uart0_init();              // Initialize UART0

   while (1) {                        /* An embedded program does not stop */
      uart0_puts(UART0_TEXT);

      uart0_puts("\n\rEnter operation (+, -, *, /): ");
      operation = uart0_getkey();    // Get operation input

      uart0_puts("\n\rEnter first number: ");
      uart0_gets(num1_str, sizeof(num1_str));  // Get first number input
      num1 = atoi(num1_str);  // Convert string to integer

      uart0_puts("\n\rEnter second number: ");
      uart0_gets(num2_str, sizeof(num2_str));  // Get second number input
      num2 = atoi(num2_str);  // Convert string to integer

      perform_operation(operation, num1, num2);  // Perform arithmetic operation

      delay(100000);  // Delay
   }
}
```

In Serial.c

```
#include <LPC21xx.H>                    /* LPC21xx definitions              */
#include "Serial.h"

#define CR     0x0D

/* implementation of putchar (also used by printf function to output data) */
int sendchar (int ch)  {              /* Write character to Serial Port    */
   if (ch == '\n')  {
      while (!(U1LSR & 0x20));
      U1THR = CR;                        /* output CR */
```

```
    }
    while (!(U1LSR & 0x20));
    return (U1THR = ch);
}

int uart0_getkey (void)  {                  /* Read character from Serial Port   */
    while (!(U0LSR & 0x01));
    return (U0RBR);
}

int uart1_getkey (void)  {                  /* Read character from Serial Port   */
    while (!(U1LSR & 0x01));
    return (U1RBR);
}

void uart1_init()
{
    /* initialize the serial interface   */
    PINSEL0 = 0x00050000;          /* Enable RxD1 and TxD1                    */
    U1LCR = 0x83;                  /* 8 bits, no Parity, 1 Stop bit           */
    U1DLL = 97;                    /* 9600 Baud Rate @ 15MHz VPB Clock        */
    U1LCR = 0x03;                  /* DLAB = 0                                */
}

void uart1_putc(char c)
{
    while(!(U1LSR & 0x20)); // Wait until UART1 ready to send character
    U1THR = c;  // Send character
}

void uart1_puts(char *p)
{
    while(*p) // Point to character
    {
        uart1_putc(*p++);   // Send character then point to next character
    }
}

void uart0_init()
{
    /* initialize the serial interface   */
```

```
    PINSEL0 = 0x00000005;          /* Enable RxD0 and TxD0                    */
    U0LCR = 0x83;                  /* 8 bits, no Parity, 1 Stop bit           */
    U0DLL = 97;                    /* 9600 Baud Rate @ 15MHz VPB Clock        */
    U0LCR = 0x03;                  /* DLAB = 0                                */
}

void uart0_putc(char c)
{
    while(!(U0LSR & 0x20)); // Wait until UART0 ready to send character
    U0THR = c; // Send character
}

void uart0_puts(char *p)
{
    while(*p) // Point to character
    {
        uart0_putc(*p++);  // Send character then point to next character
    }
}
```

EXPERIMENT NO 11

Logical Operation

Aim:

To do logical operations with ARM LPC2148.

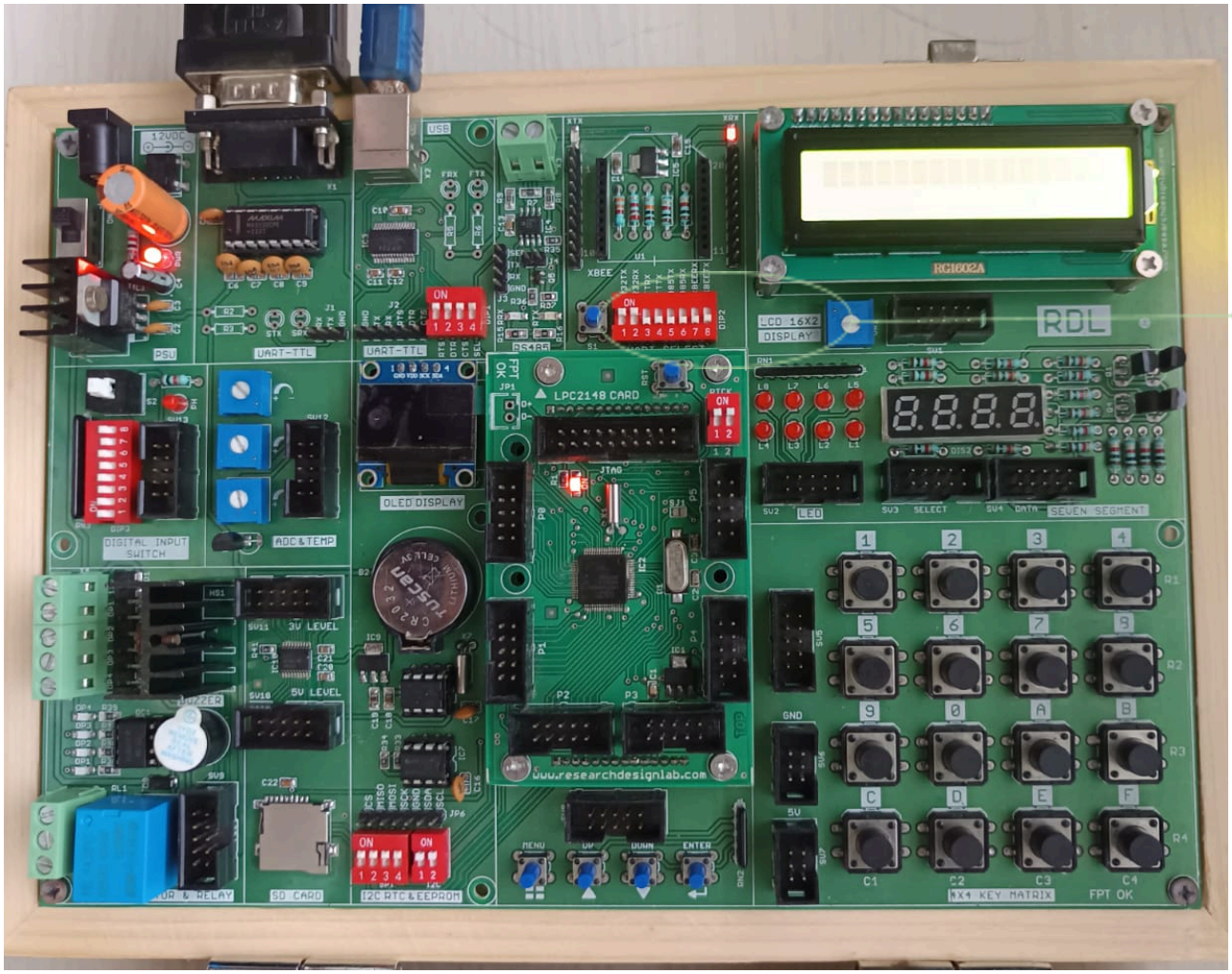
Description:

Send logical operator and numbers in tera term and its result will be displayed in teraterm.

.

Hardware Required:

ARM LPC2148 Trainer Kit, RS232 cable, USB A to B Cable and 12V 2A Power Adapter.



Make 232 TX and 232RX High after uploading the program

Program

In main.c

```
#include <stdio.h>              /* prototype declarations for I/O functions */
#include <LPC214x.H>            /* LPC21xx definitions                      */
#include "Serial.h"

#define UART0_TEXT "\n\r RDL TECHNOLOGIES PVT LTD ... Logical Operations"

void delay(int count)
{
   int j=0,i=0;

   for(j=0;j<count;j++)
   {
      /* At 60Mhz, the below loop introduces
      delay of 10 us */
      for(i=0;i<35;i++);
   }
}

/* Function to convert a string to an integer */
int atoi(char *str) {
   int num = 0;
   while (*str) {
      num = num * 10 + (*str - '0');
      str++;
   }
   return num;
}

/* Function to perform logical operations */
void perform_operation(char operation, int num1, int num2)
{
   int result;
```

```
    char buffer[100];

    switch(operation) {
        case '&':  // AND
            result = num1 & num2;
            sprintf(buffer, "Result: %d & %d = %d\n\r", num1, num2, result);
            break;
        case '|':  // OR
            result = num1 | num2;
            sprintf(buffer, "Result: %d | %d = %d\n\r", num1, num2, result);
            break;
        case '~':  // NOT (only for num1)
            result = ~num1;
            sprintf(buffer, "Result: ~%d = %d\n\r", num1, result);
            break;
        case 'n':  // NAND
            result = ~(num1 & num2);
            sprintf(buffer, "Result: ~( %d & %d ) = %d\n\r", num1, num2, result);
            break;
        case 'o':  // NOR
            result = ~(num1 | num2);
            sprintf(buffer, "Result: ~( %d | %d ) = %d\n\r", num1, num2, result);
            break;
        default:
            sprintf(buffer, "Error: Invalid operation\n\r");
            break;
    }

    uart0_puts(buffer);
}

/* Function to read a string from UART */
void uart0_gets(char *buf, int max_len) {
    int i = 0;
    char ch;
    while (i < max_len - 1) {
        ch = uart0_getkey();  // Get character from UART
        if (ch == '\n' || ch == '\r') {
```

```
            buf[i] = '\0';
            return;
        }
        buf[i++] = ch;
    }
    buf[i] = '\0';  // Null-terminate the string
}

int main (void)  {              /* execution starts here                    */
    char operation;
    char num1_str[20], num2_str[20];
    int num1, num2;

    uart0_init();               // Initialize UART0

    while (1) {                         /* An embedded program does not stop */
        uart0_puts(UART0_TEXT);

        uart0_puts("\n\rEnter operation (& (AND), | (OR), ~ (NOT), n (NAND), o (NOR)): ");
        operation = uart0_getkey();    // Get operation input

        uart0_puts("\n\rEnter first number: ");
        uart0_gets(num1_str, sizeof(num1_str));  // Get first number input
        num1 = atoi(num1_str);  // Convert string to integer

        if (operation != '~') {  // NOT operation requires only one operand
            uart0_puts("\n\rEnter second number: ");
            uart0_gets(num2_str, sizeof(num2_str));  // Get second number input
            num2 = atoi(num2_str);  // Convert string to integer
        } else {
            num2 = 0;  // No second operand for NOT operation
        }

        perform_operation(operation, num1, num2);  // Perform logical operation

        delay(100000);  // Delay
    }
}
```

In Serial.c

```
#include <LPC21xx.H>                    /* LPC21xx definitions              */
#include "Serial.h"

#define CR     0x0D

/* implementation of putchar (also used by printf function to output data) */
int sendchar (int ch)  {                /* Write character to Serial Port    */
   if (ch == '\n')  {
      while (!(U1LSR & 0x20));
      U1THR = CR;                       /* output CR */
   }
   while (!(U1LSR & 0x20));
   return (U1THR = ch);
}

int uart0_getkey (void)  {                  /* Read character from Serial Port   */
   while (!(U0LSR & 0x01));
   return (U0RBR);
}

int uart1_getkey (void)  {                  /* Read character from Serial Port   */
   while (!(U1LSR & 0x01));
   return (U1RBR);
}

void uart1_init()
{
   /* initialize the serial interface   */
   PINSEL0 = 0x00050000;          /* Enable RxD1 and TxD1                    */
   U1LCR = 0x83;                  /* 8 bits, no Parity, 1 Stop bit           */
   U1DLL = 97;                    /* 9600 Baud Rate @ 15MHz VPB Clock        */
   U1LCR = 0x03;                  /* DLAB = 0                                */
}
```

```
void uart1_putc(char c)
{
   while(!(U1LSR & 0x20)); // Wait until UART1 ready to send character
   U1THR = c;  // Send character
}

void uart1_puts(char *p)
{
   while(*p) // Point to character
   {
      uart1_putc(*p++);   // Send character then point to next character
   }
}

void uart0_init()
{
   /* initialize the serial interface   */
   PINSEL0 = 0x00000005;          /* Enable RxD0 and TxD0                    */
   U0LCR = 0x83;                  /* 8 bits, no Parity, 1 Stop bit           */
   U0DLL = 97;                    /* 9600 Baud Rate @ 15MHz VPB Clock        */
   U0LCR = 0x03;                  /* DLAB = 0                                */
}

void uart0_putc(char c)
{
   while(!(U0LSR & 0x20)); // Wait until UART0 ready to send character
   U0THR = c; // Send character
}

void uart0_puts(char *p)
{
   while(*p) // Point to character
   {
      uart0_putc(*p++);  // Send character then point to next character
   }
}
```

EXPERIMENT NO 12

Seven Segment Display

Aim:

Display 0-9 digit in seven segment display with ARM LPC2148.

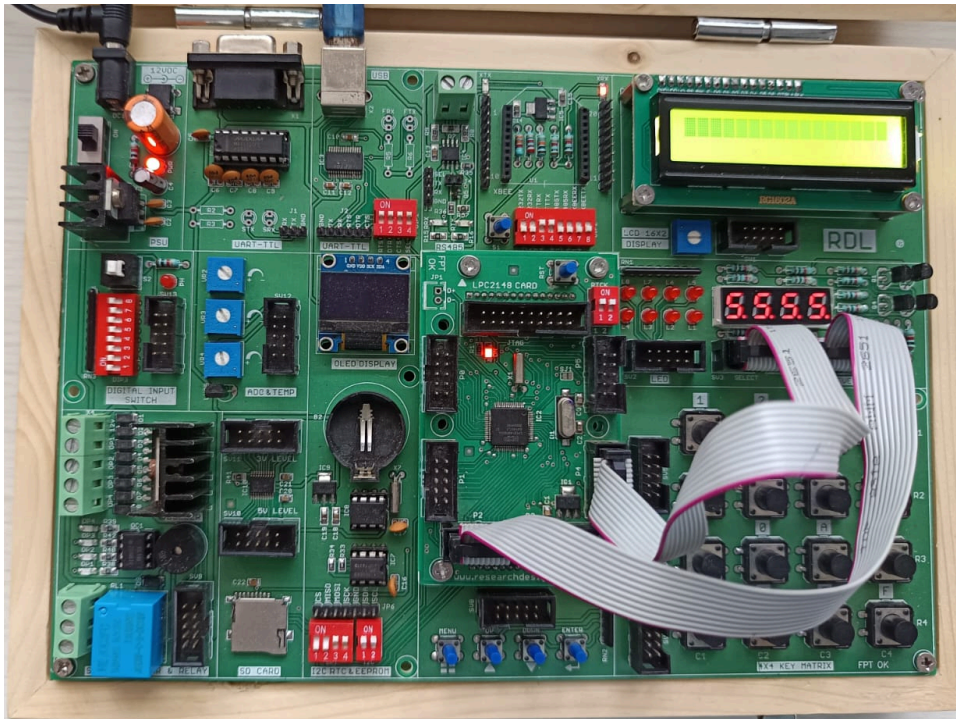
Description:

Display numbers from 0 to 9 in Seven segment display..

.

Hardware Required:

ARM LPC2148 Trainer Kit, FRC cable, USB A to B Cable and 12V 2A Power Adapter.



- Create HEX file of the code.
- Upload to the device.
- Connect P2 port to DATA port and P4 port to SELECT port .
- Connect power supply and USB cable.

Program

```
#include <LPC21xx.h>

// Define delay function
void delay1(int);

int main() {
   // Select pins for blinking LED
   PINSEL1 = 0x00000000;  // Configure P0.16 to P0.31 as GPIO
   PINSEL2 = 0x00000000;  // Configure P1.16 to P1.31 as GPIO
   IODIR0 = 0x00FF0000;   // Configure P0.16 to P0.23 as output
   IODIR1 = 0x000F0000;   // Configure P1.16 to P1.19 as output

   while (1) {
                                        IOCLR1 = 0x000F0000;
      // Turn on each LED one by one

//print 0
      IOSET0 = (1 << 22);
      IOCLR0 = (1 << 16) | (1 << 17) | (1 << 18) | (1 << 19) | (1 << 20) | (1 << 21) | (1 << 23);
      delay1(2000);        // 2000 ms delay


//print 1
         IOSET0 = (1 << 16) | (1 << 21) | (1 << 22) | (1 << 20) | (1 << 19);
         IOCLR0 = (1 << 18) | (1 << 17) | (1 << 23);
         delay1(2000);        // 2000 ms delay

//print 2
         IOSET0 = (1 << 21) | (1 << 18);
         IOCLR0 = (1 << 16) | (1 << 19) | (1 << 17) | (1 << 22) | (1 << 20) | (1 << 23);
         delay1(2000);        // 2000 ms delay

//print 3
         IOSET0 = (1 << 20) | (1 << 21);
         IOCLR0 = (1 << 16) | (1 << 19) | (1 << 18) | (1 << 22) | (1 << 17) | (1 << 23);
         delay1(2000);        // 2000 ms delay
```

```
//print 4
        IOSET0 = (1 << 16) | (1 << 20) | (1 << 19);
        IOCLR0 = (1 << 22) | (1 << 18) | (1 << 17) | (1 << 21) | (1 << 23);
        delay1(2000);       // 2000 ms delay


//print 5
        IOSET0 = (1 << 17) | (1 << 20);
        IOCLR0 = (1 << 16) | (1 << 19) | (1 << 18) | (1 << 22) | (1 << 21) | (1 << 23);
        delay1(2000);       // 2000 ms delay


//print 6
        IOSET0 = (1 << 17);
        IOCLR0 = (1 << 16) | (1 << 19) | (1 << 18) | (1 << 22) | (1 << 20) | (1 << 21) | (1 << 23);
        delay1(2000);       // 2000 ms delay


//print 7
        IOSET0 = (1 << 21) | (1 << 22) | (1 << 20) | (1 << 19);
        IOCLR0 = (1 << 16) | (1 << 18) | (1 << 17) | (1 << 23);
        delay1(2000);       // 2000 ms delay

//print 8
        IOCLR0 = (1 << 16) | (1 << 17) | (1 << 18) | (1 << 19) | (1 << 20) | (1 << 21) | (1 << 22) |
(1 << 23);
        delay1(2000);       // 2000 ms delay


//print 9
        IOSET0 = (1 << 20);
        IOCLR0 = (1 << 16) | (1 << 17) | (1 << 18) | (1 << 19) | (1 << 21) | (1 << 22) | (1 << 23);
        delay1(2000);       // 2000 ms delay
   }
}
```

```
/* Delay routine; gives an approximate delay in milliseconds */
void delay1(int d) {
    int i;
    while (d) {
        for (i = 0; i < 7000; i++) {
            // NOP (no operation) to introduce a delay
            __asm__("nop");
        }
        d--;
    }
}
```