ARM-CORTEX M4 LP4088

Learn-Practice-Grow

www.pantechsolutions.net

8/19/2021

Contents

CHAPTER 1: Introduction

1.1 WELCOME

Thank you for purchasing the ARM CORTEX Tyro Board from Pantech ProLabs India Pvt Ltd. The ARM CORTEX Tyro is a development board which demonstrates the capabilities of the 64-pin LPC4088 devices.

The ARM CORTEX Tyro Board can be used as a stand-alone board built with an in-circuit USB programmer. Sample programs are provided to demonstrate the unique features of the supported devices.

The ARM CORTEX Tyro Board Kit comes with the following:

1. ARM CORTEX Tyro Development Board
2. Sample devices (LPC4088)
3. CD-ROM, which contains:
   a) Sample programs
   b) ARM CORTEX Tyro Development Board User's Guide
4. 5V Adapter
5. Straight Female to Female serial cable
6. One Hook wire
7. USB Cable

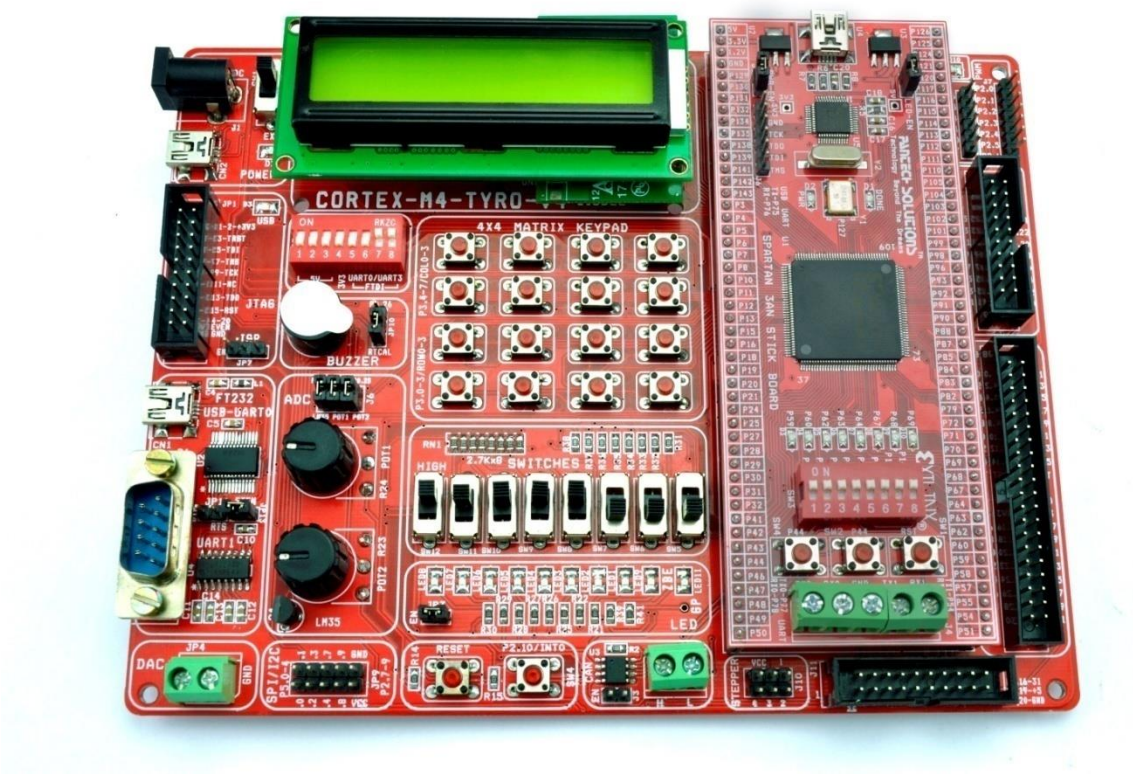
Note: If you are missing any part of the kit, please contact our support executive

1.2 ARM CORTEX TYRO DEVELOPMENT BOARD

The ARM CORTEX TYRO development board has the following hardware features:

- 8 Nos. Point LEDs (Logic Output)
- 8 Nos. Digital Input(SLIDE Switch)
- 2 Nos. Analog Input (Potentiometer)
- 2x16 Char LCD Interface
- Temperature Sensor(LM35)
- Internal RTC with Battery-Backup
- 1 No. UART(RS232)
- 1 No. USB UART
- USB 2.0 device (Virtual Port)
- DAC Output
- Interrupts Study, Reset Button
- 4x4 Matrix Keyboard
- 40-Pin Expansion Connector
- JTAG (Program/Debug) |ISP Programming
- 2 Nos. 20pin- I/O Expansion Connector
- PWM Terminations
- Stepper Interface
- Optional Onboard ZIGBEE Interface
- Onboard Buzzer
- Optional SPARTAN3AN FPGA Stick Interface
- SPI/I2C Expansion Connector
- SD card Interface
- CAN Interface (optional)

1.3 ARM CORTEX TYRO HARDWARE



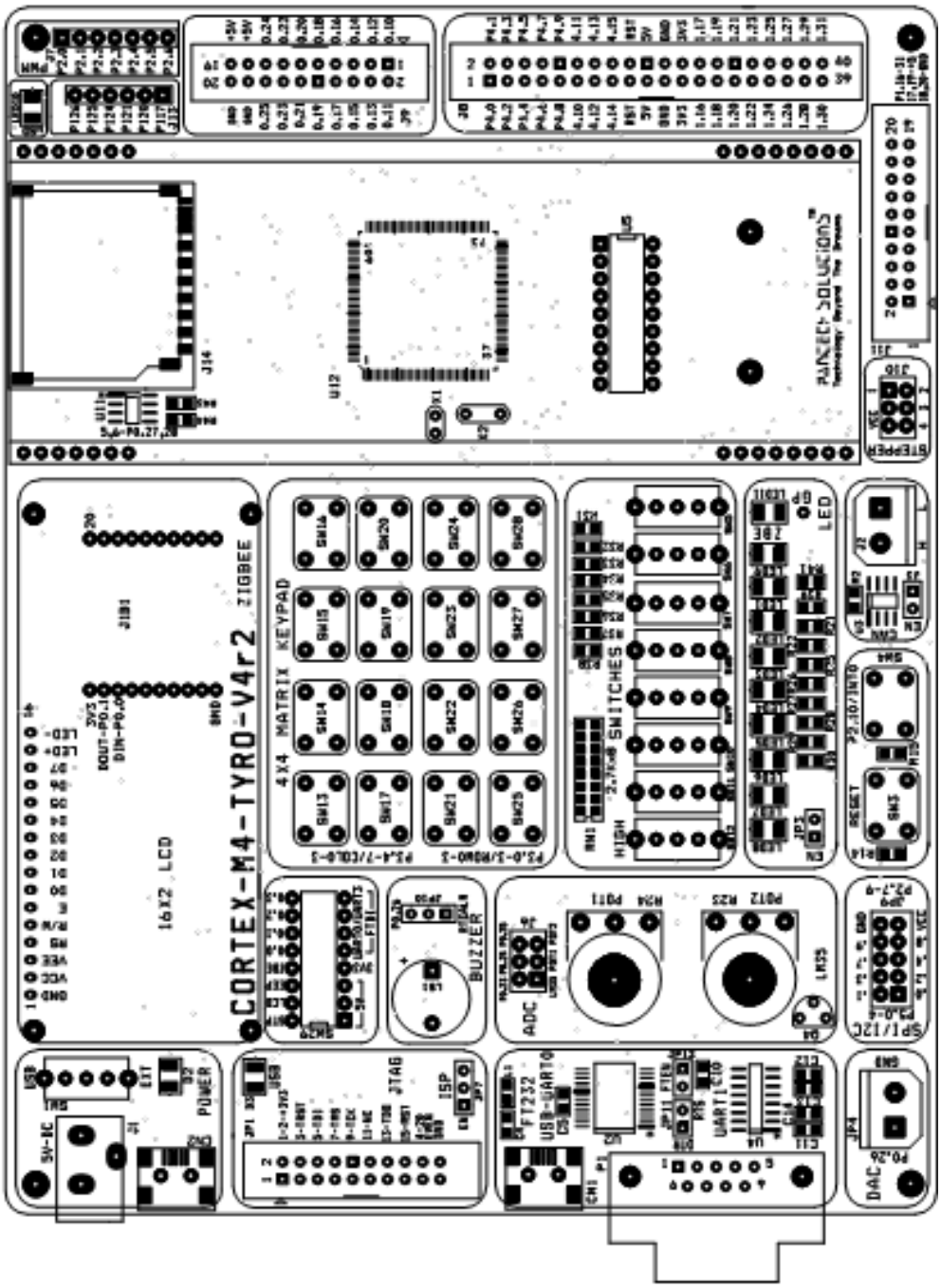
1.4 SAMPLE DEVICES

One FLASH device (LPC4088) is included on the board itself.

1.5 SAMPLE PROGRAMS

The ARM CORTEX TYRO Kit includes a CD-ROM with sample programs. These programs may be used with the included sample devices. Demo source code with compiled Hex file is provided.

1.6 ARM CORTEX TYRO BOARD LAYOUT



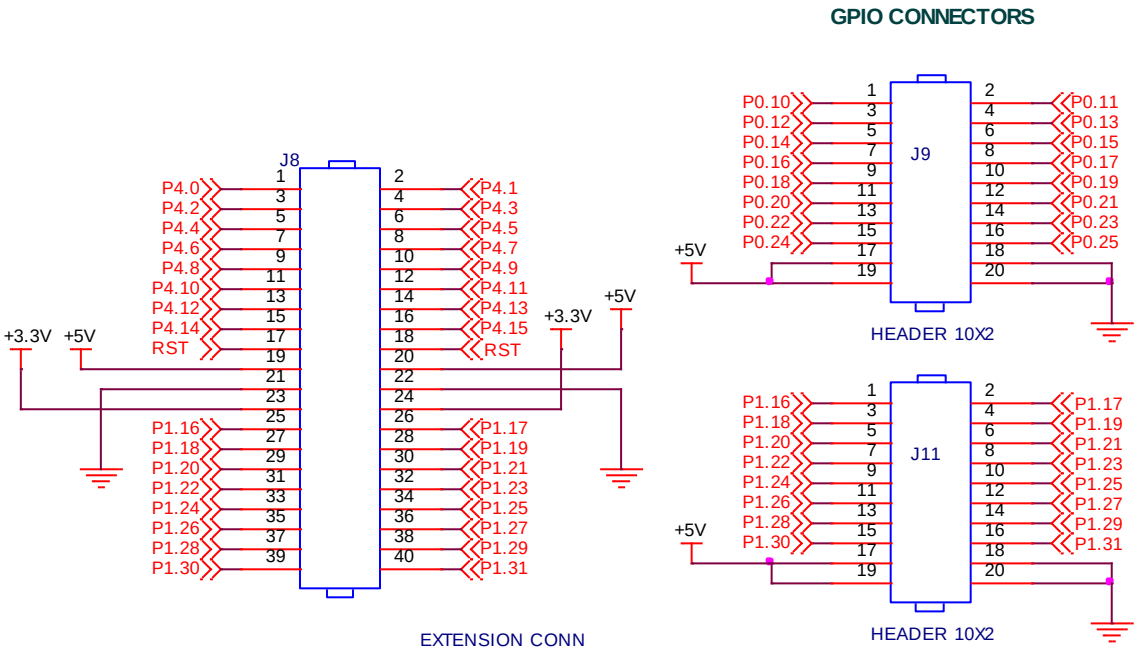
CHAPTER 2: Hardware Details

2.1 CONNECTORS

40 pin FRC box type connector: Instead of terminating each port separately, this connector has all the port pins. So More IO lines can be taken by using single cable.

The Ports are arranged as shown in the following figures

Similarly, two no. of 20x2 Connector gives access to various port lines.



2.2 JUMPER SETTINGS

JUMPER	DESCRIPTION
JP3	LED ENABLE: Place a Jumper
J6	ADC Selection: Place a jumper for the required channel; this will connect LPC4088 pins with POT or LM35.
JP7	ISP Enable
JP10	Connects the Buzzer with P0.26 or RTC alarm output pin
JP11	RTS and DTR pin Termination, Jumper not needed
JP13	FT232 Enable: Powers the FT232 IC

2.3 CONFIG DIP SWITCH (SW29)

DIP SWITCH PIN	PIN	DESCRIPTION
5V	1-STEP	Power Supply for Stepper Motor
5V	2-LCD	Power Supply for LCD
5V	3-EEP	Power Supply for I2C EEPROM
ZIGBEE	4-ZBE	Power Supply for ZIGBEE
FT232	5 and 6	Connects FT232with P0.0 and P0.1
FT232	7 and 8	Connects FT232with P0.2 and P0.3 (default)

2.5 POWER SUPPLY

The external power should be DC5V, 1A. The ARM board produces +3.3V using an onboard voltage regulator, which provides supply to the ARM controller.
Power supply is controlled through slide switch SW1. The 5V volt from USB or DC JACK is used for Peripherals directly

POWER SWITCH	 EXT USB	EXT Supply Turned ON
	 EXT USB	USB Supply Turned ON

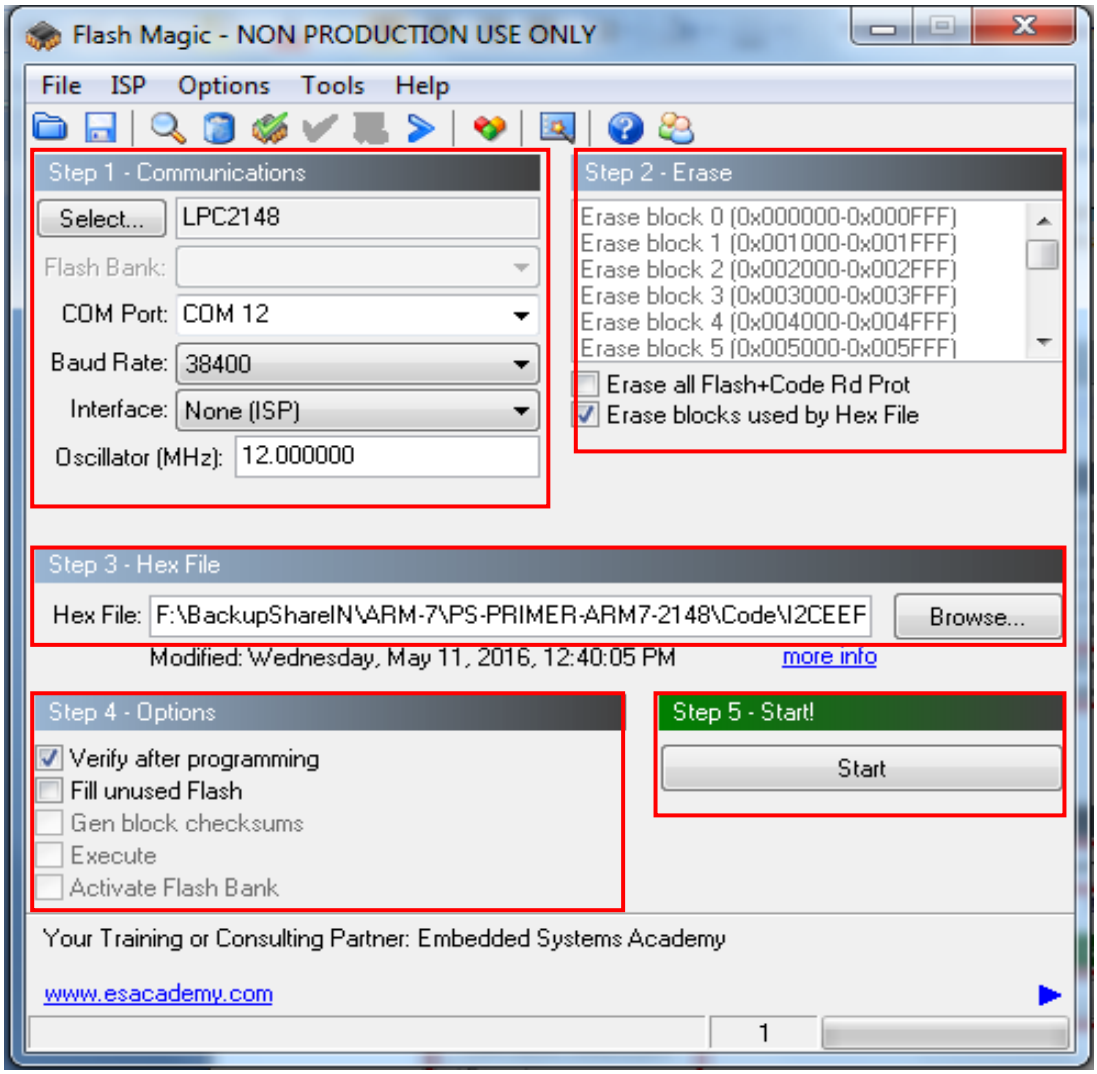
2.5.1 POWER SUPPLY TO THE PERIPHERALS

PERIPHERAL	SWITCH	PIN	DESCRIPTION
LPC4088	SW1	-	Turn ON the switch SW1 towards USB or EXT
FT232	JP13	-	Place a Jumper at JP13
STEPPER	SW29	1	Turn ON the DIP switch pin
LCD	SW29	2	Turn ON the DIP switch pin
I2C EEPROM	SW29	3	Turn ON the DIP switch pin
ZIGBEE	SW29	4	Turn ON the DIP switch pin

CHAPTER 3: Programming in Flash Magic

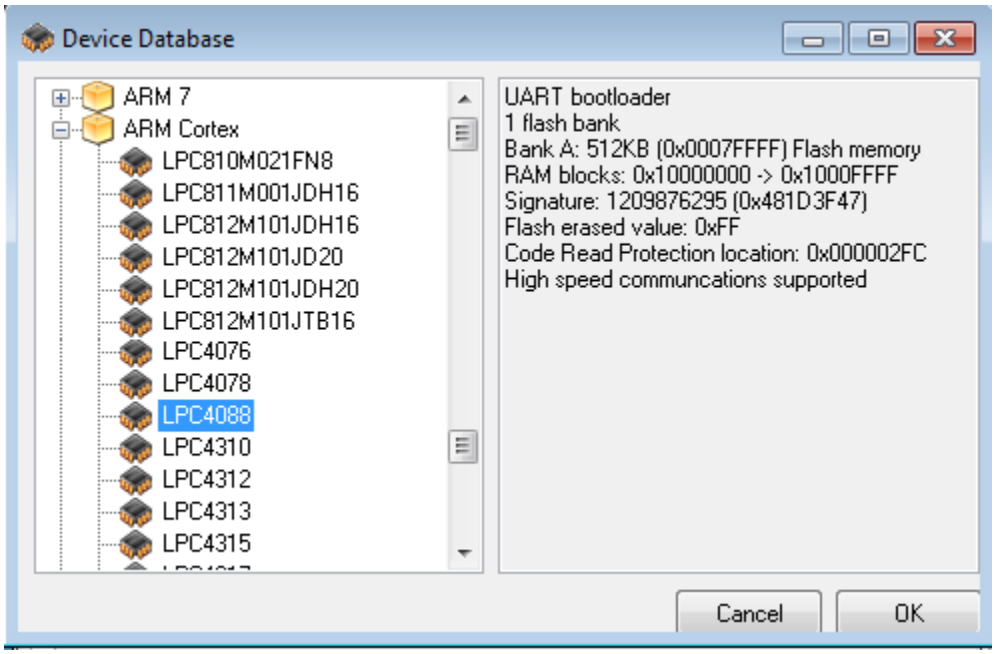
HARDWARE CONNECTION	**TURN ON DIP SWITCH SW29 PINS 7 AND 8. SUPPLY POWER TO THE BOARD USING SW1. PRESS AND HOLD SW4 (EINT0) AND PRESS RESET TO ENTER INTO BOOT-LOADER MODE**

3.1 FIVE STEPS PROGRAMMING



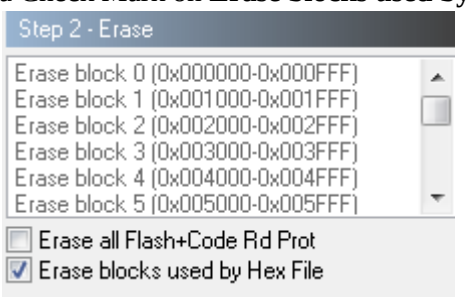
3.2 STEP 1 – COMMUNICATIONS:

1. Click **Select…,**
2. Expand **ARM Cortex from device database**
3. Scroll down and Select your IC.
4. Click OK.



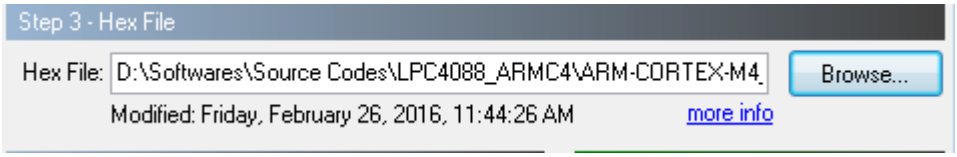
3.3 STEP 2 – ERASE

Put a Check Mark on **Erase blocks used by Hex File** checkbox

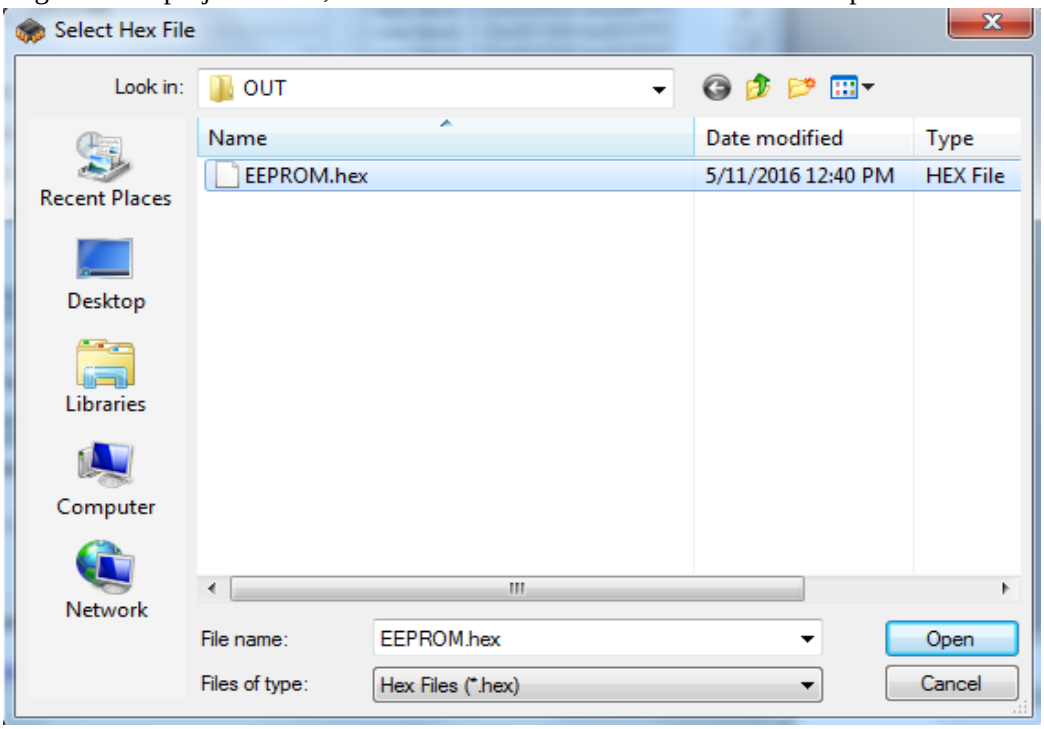


3.4 STEP 3 – HEX FILE

1.Click **Browse…**

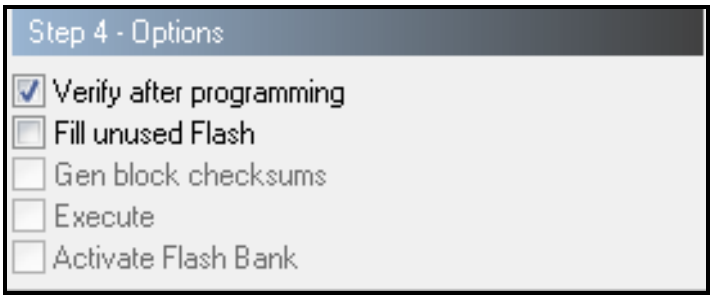


2.Navigate to the project folder; Select the hex file to be loaded and Click Open.



3.5 STEP 4 – OPTIONS

1. Put a check mark on each options that is required for the project



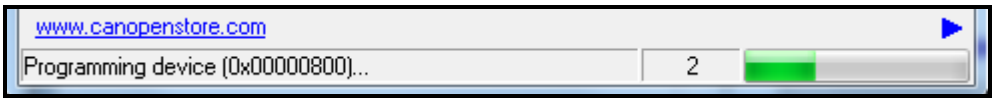
Generally **Verify after programming** is required to cross check the burned hex file with the loaded hex file and the remaining options left unchecked.

3.6 STEP 5 – START

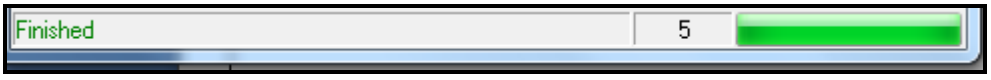
1. Click Start. This will start programming the chip

2. See the Status bar for current status

Programming:



Programming Finished:

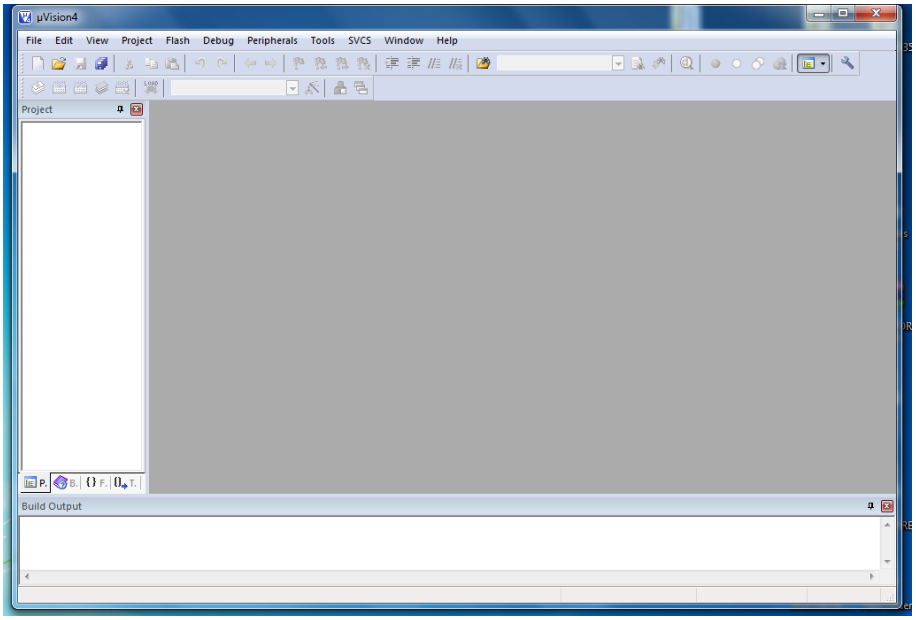


3.7 EXECUTION

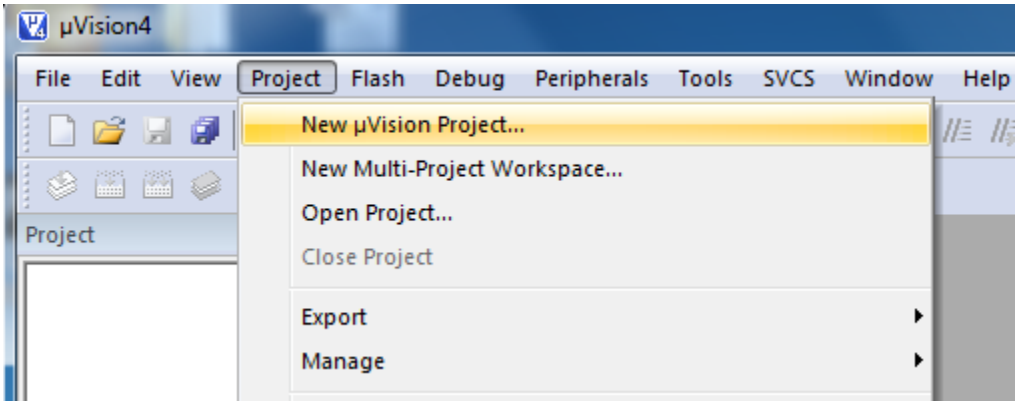
Press RESET after programming

CHAPTER 4: Creating a Project in Keil

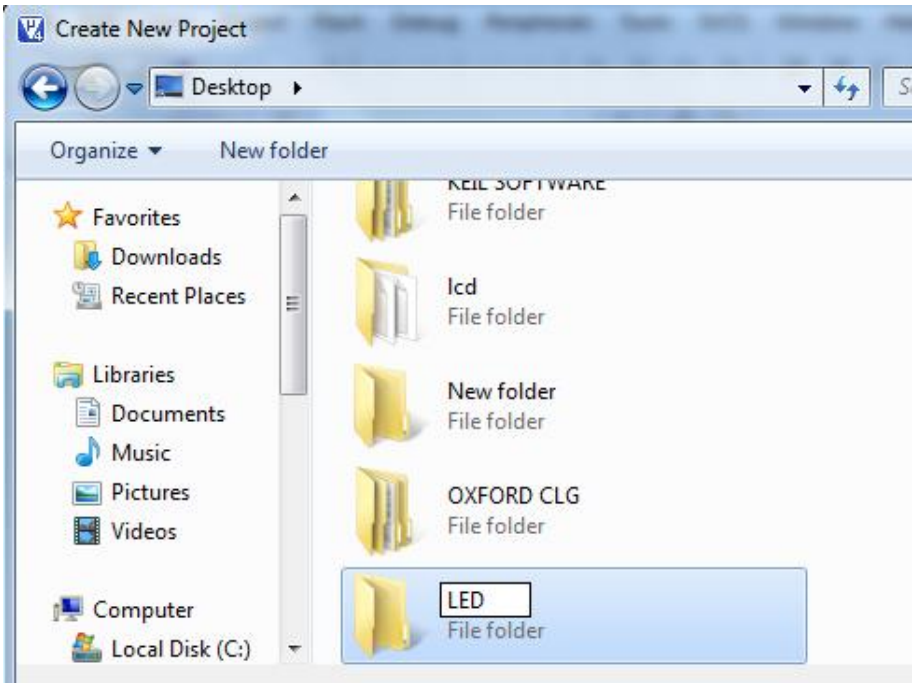
STEP1: Open Keil and the environment will be as following



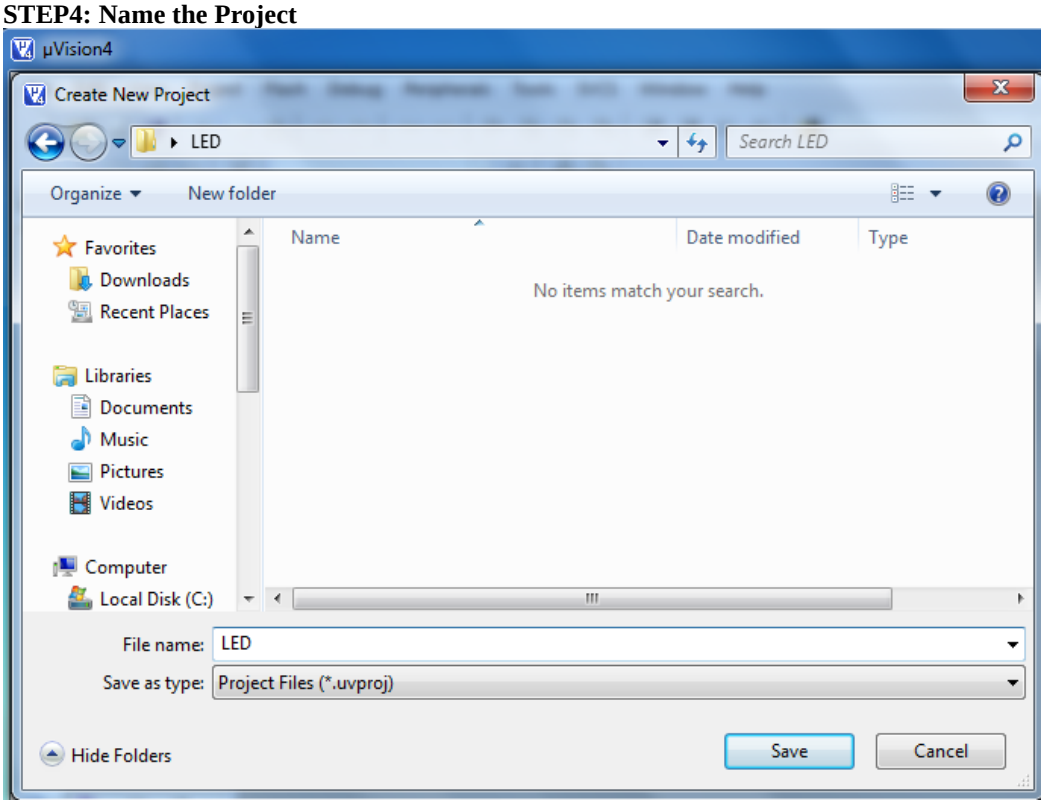
STEP2: Select Project→New uvision Project



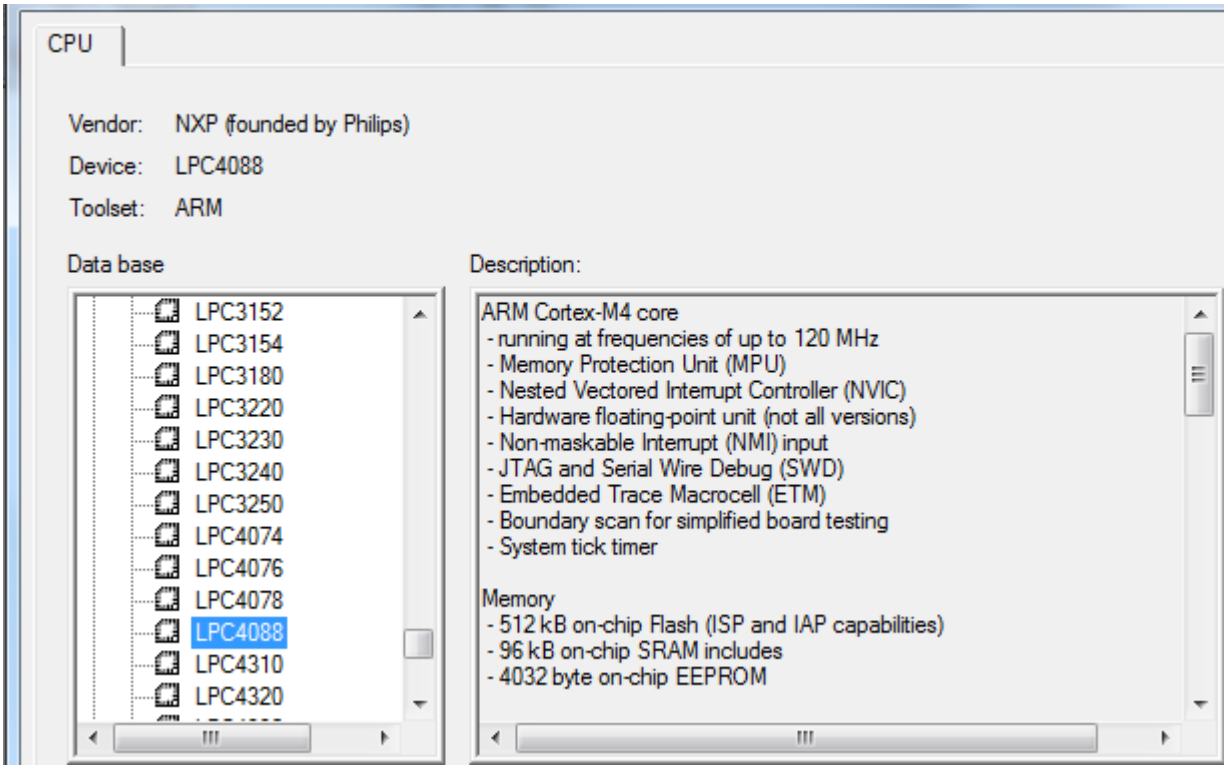
STEP3: Create a Project folder



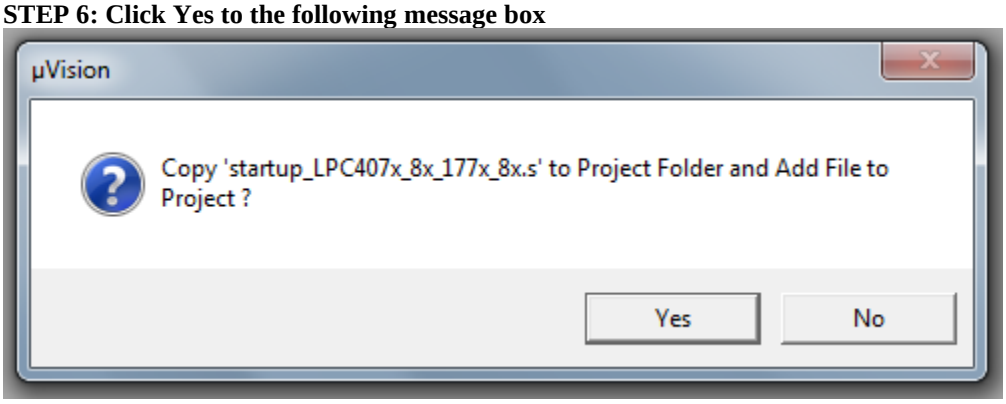
STEP4: Name the Project



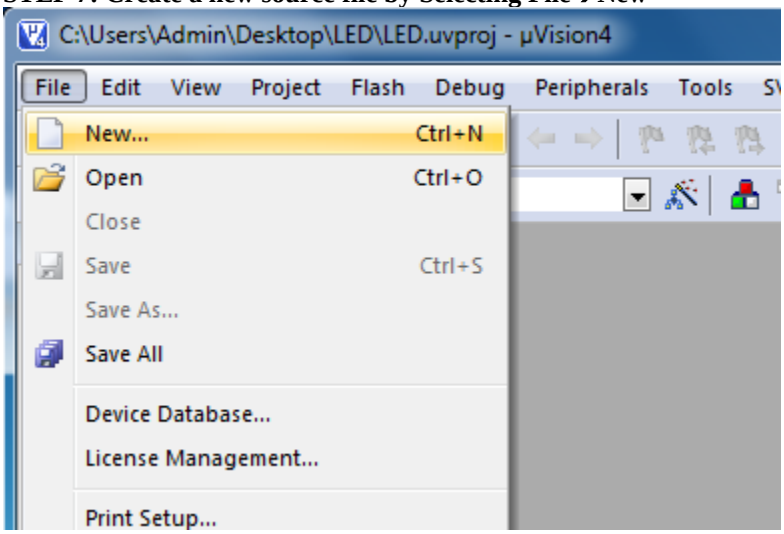
STEP5: Select the Processor and Click Ok



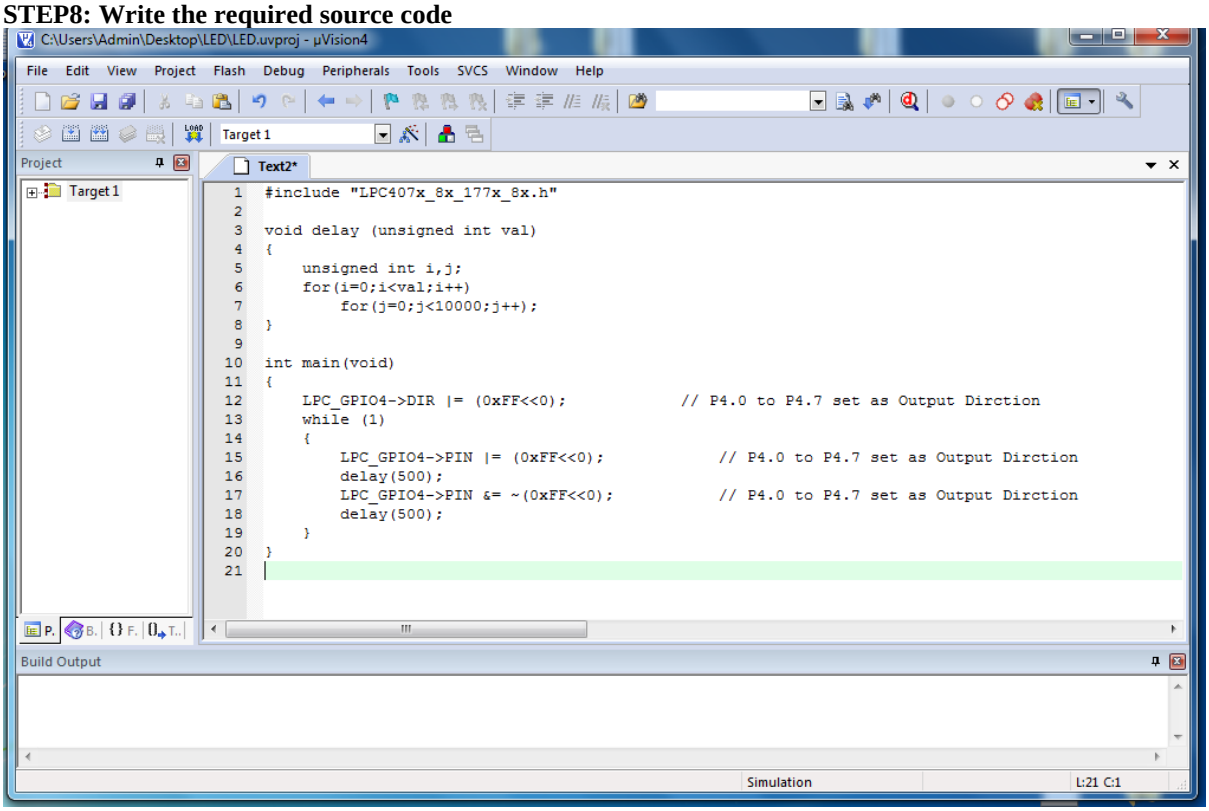
STEP 6: Click Yes to the following message box



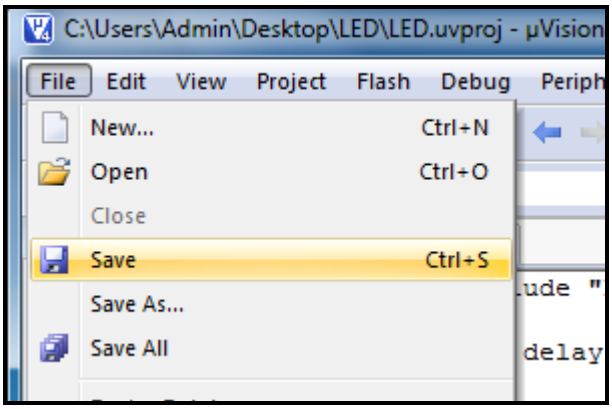
STEP 7: Create a new source file by Selecting File→New

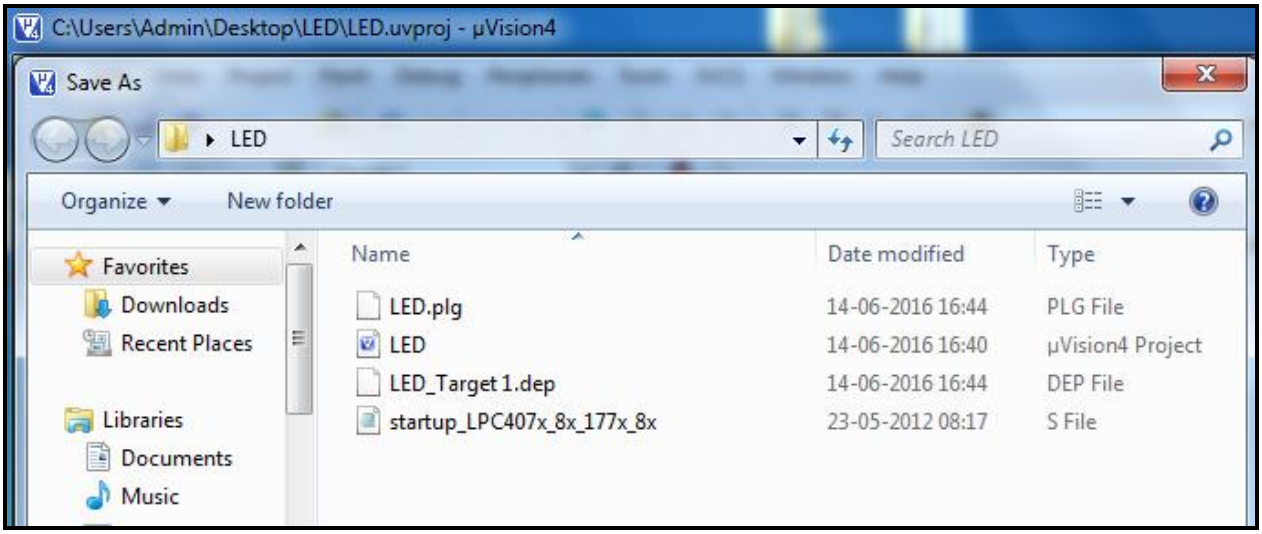


STEP8: Write the required source code

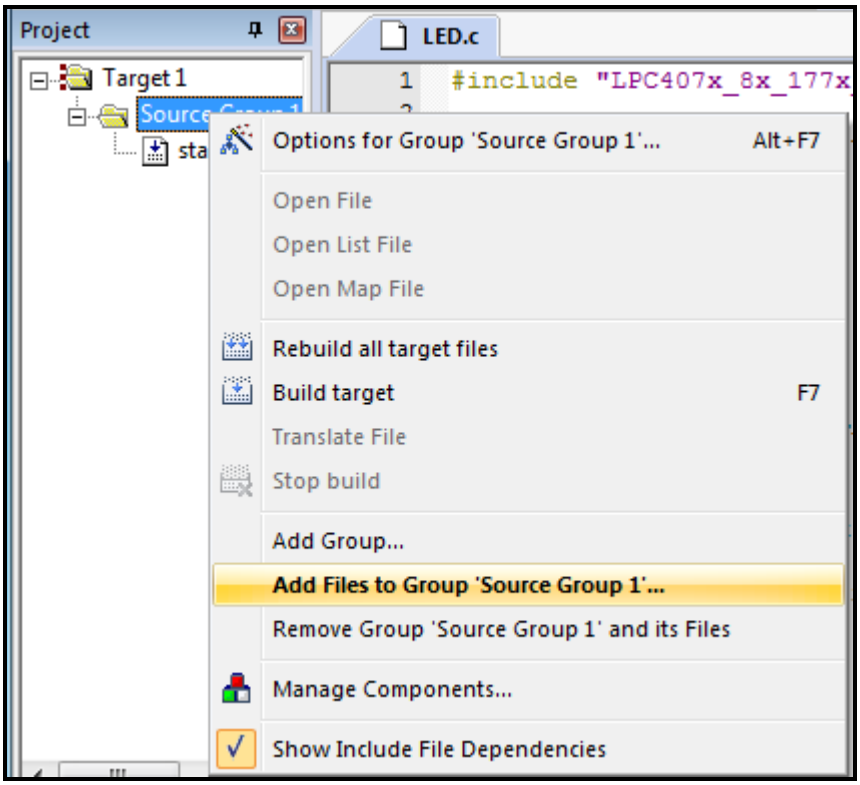


STEP 9: Save this file with ".c" Extension

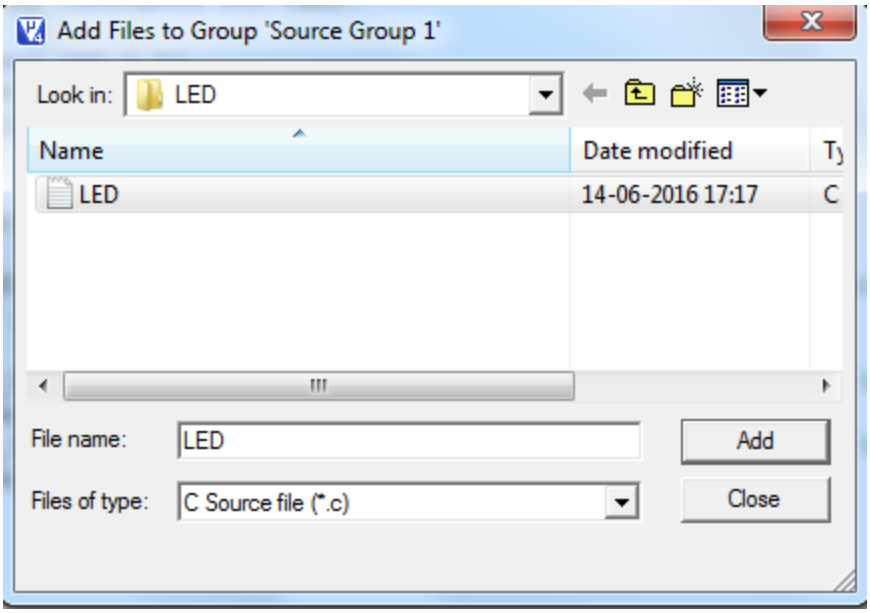




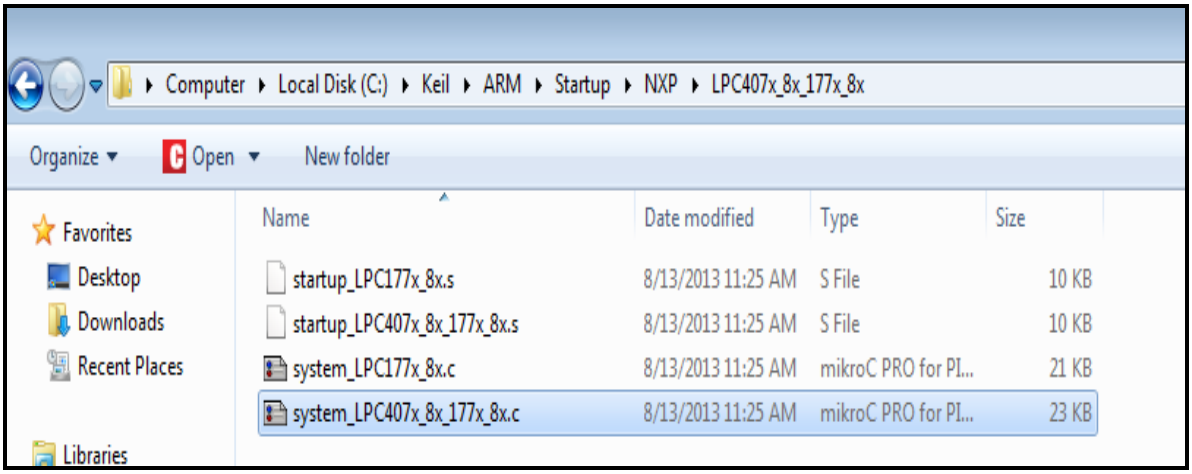
STEP 11: Right Click on Source group and select add files to group



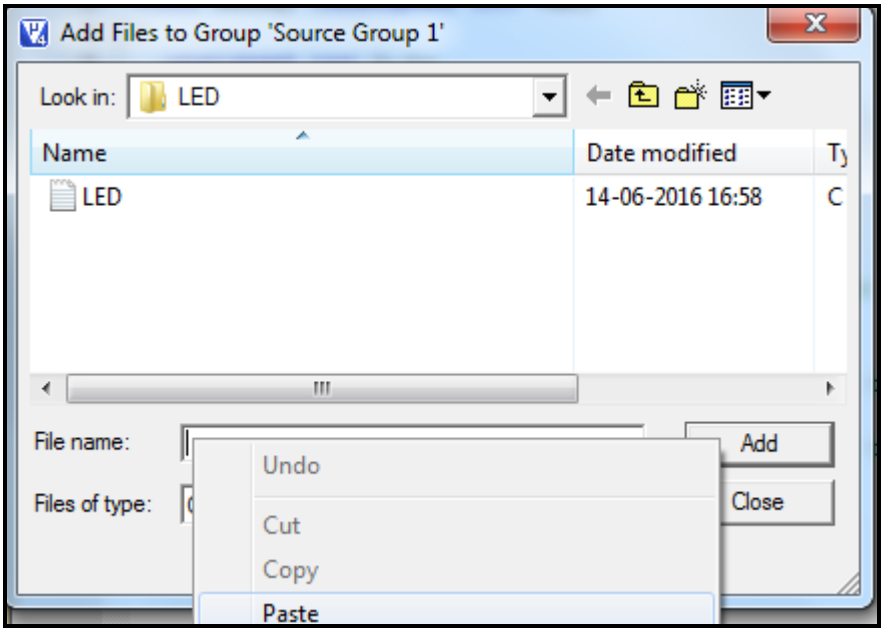
STEP 12: Select the source code file, click Add and then click close



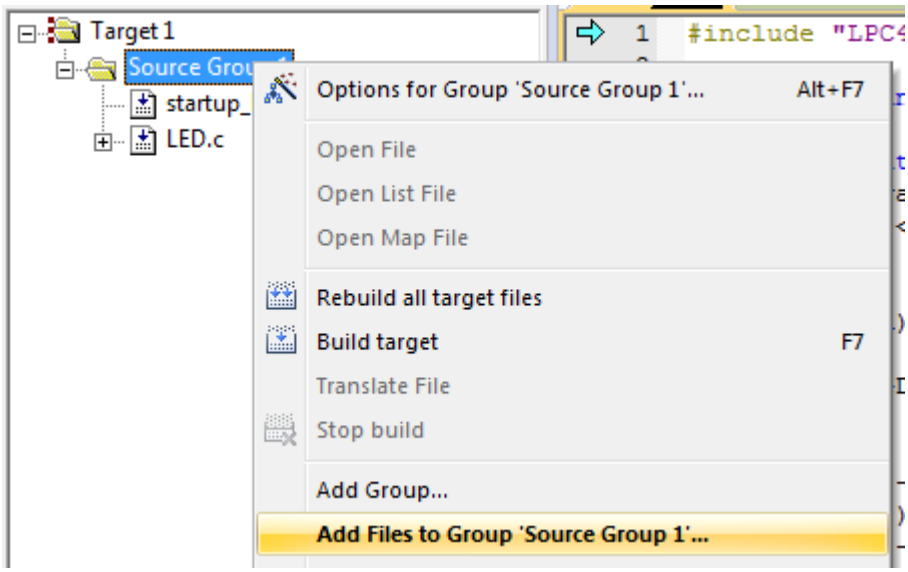
STEP 13: Navigate to the folder "C:\Keil\ARM\Startup\NXP\LPC407x_8x_177x_8x" and copy the file "system_LPC407x_8x_177x_8x.c"



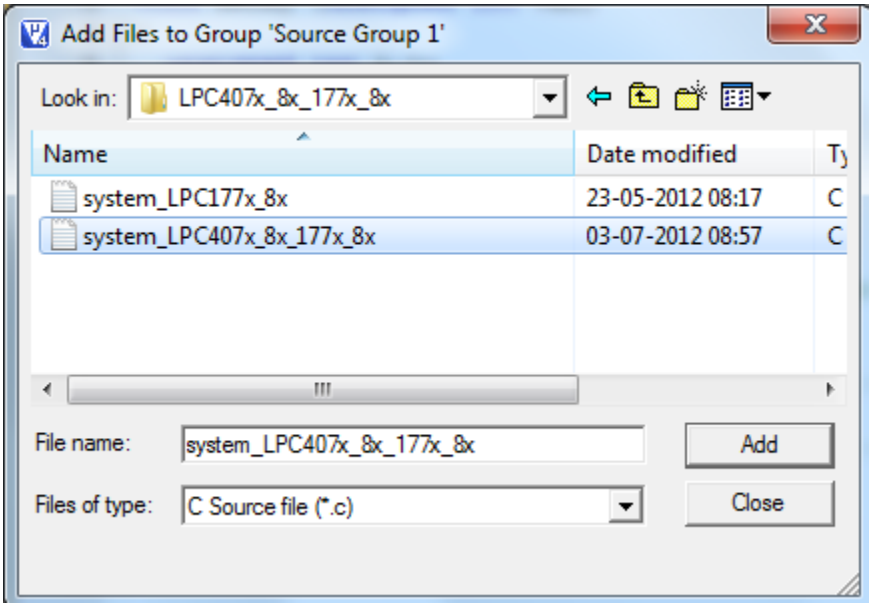
STEP 14: Paste it in your project directory



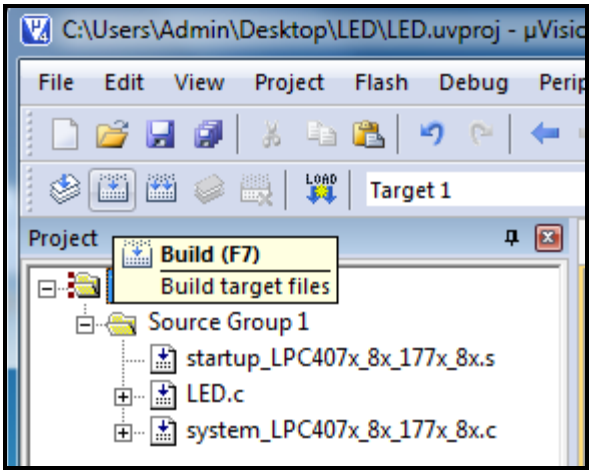
STEP 15: Right Click on Source group once again and select add files to group



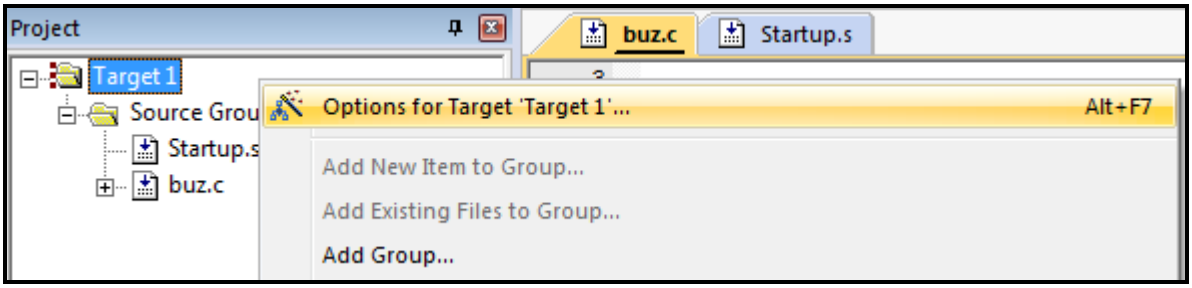
STEP 16: Now add this "system_LPC407x_8x_177x_8x.c" file to your project



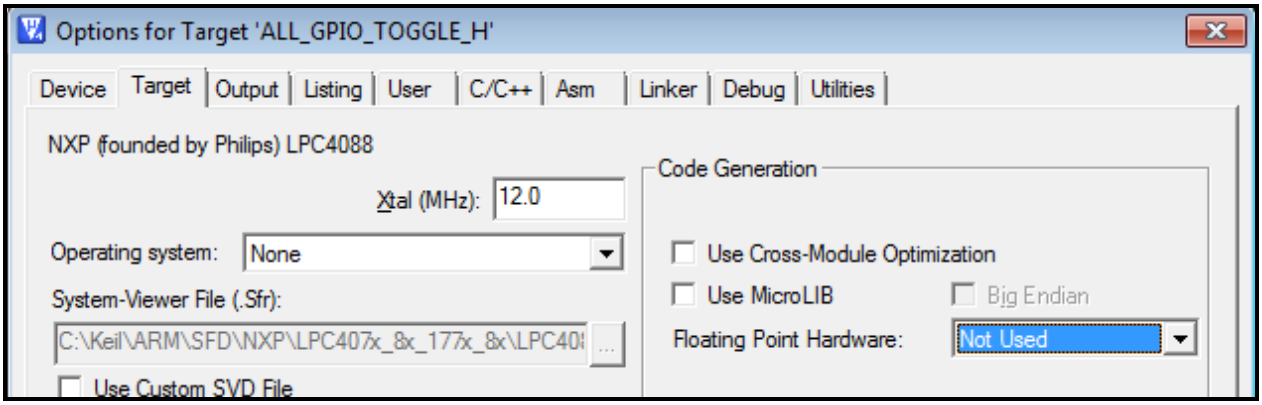
STEP 17: Build the Project



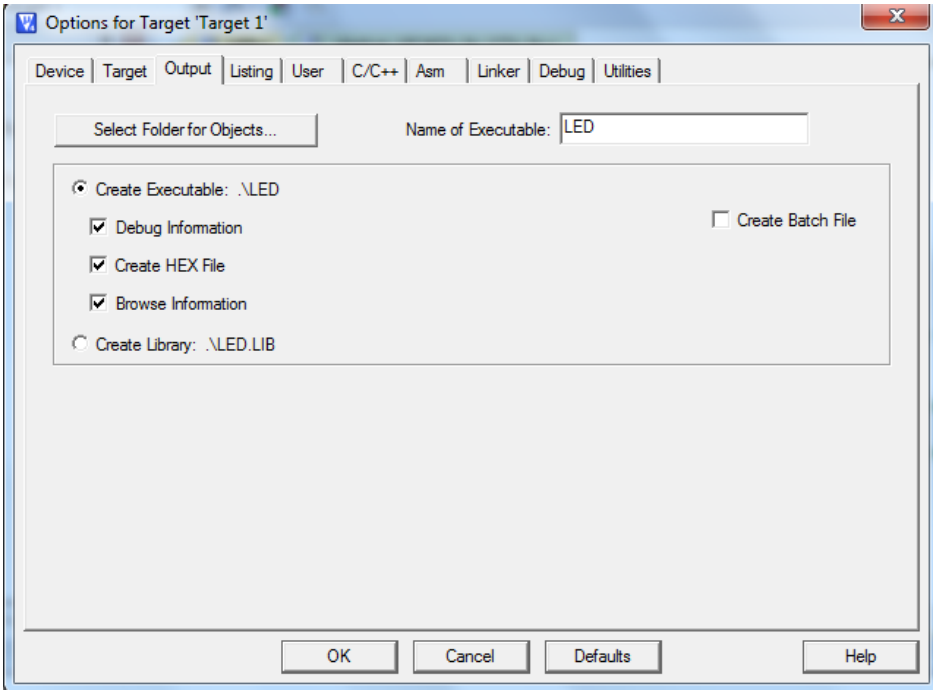
STEP 18: Right Click "Target1" on the Project Tab and select Options for Target



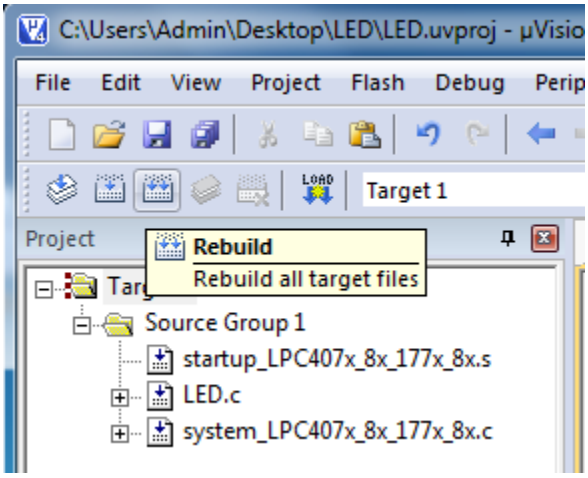
STEP 19: On the Target Tab, Select the Floating point hardware as "Not used"



STEP 20: Select Output Tab and put a check mark on "Create HEX file". The Name in the Text box "Name of Executable" will be used to name the Hex file.



STEP 20: Click Rebuild



That's all. Now write your hex file into ARM CORTEX-M4 using Flash magic

CHAPTER 5: Example Programs and Connections

Pre-requisites:

Place a Jumper at JP13 and Turn ON the DIP switch SW29 pins 7 and 8. And keep these settings for all the examples

EX1. ADC and Temperature Sensor

Aim:

This example describes how to use ADC conversion in polling mode.

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable

Procedure:

1. Power: In the PCONP register, set the PCADC bit
2. Configure pins P0.23 (CH0), P0.24 (CH1) and P0.25 (CH2) as analog pins using their respective IOCON registers i.e. IOCON_P0_23, IOCON_P0_24 and IOCON_P0_25.
3. Select ADC conversion Clock rate (400 KHz-12bit conversion) using ADC control register.
4. Start ADC conversion
5. Wait for the DONE bit. If the DONE bit is HIGH, the conversion is completed
6. Read the data register and store them in a variable
7. Do the Steps 4 to 6 for the remaining 2 channels.
8. Write these data's in HyperTerminal via UART0

Connections:

<table>
<tr><th colspan="2">HARDWARE PIN OUT</th><th>CONNECTIONS</th><th>OUTPUT</th></tr>
<tr><td>LM35</td><td>P0.23</td><td rowspan="3">Place all the Jumpers @J6 (ADC)

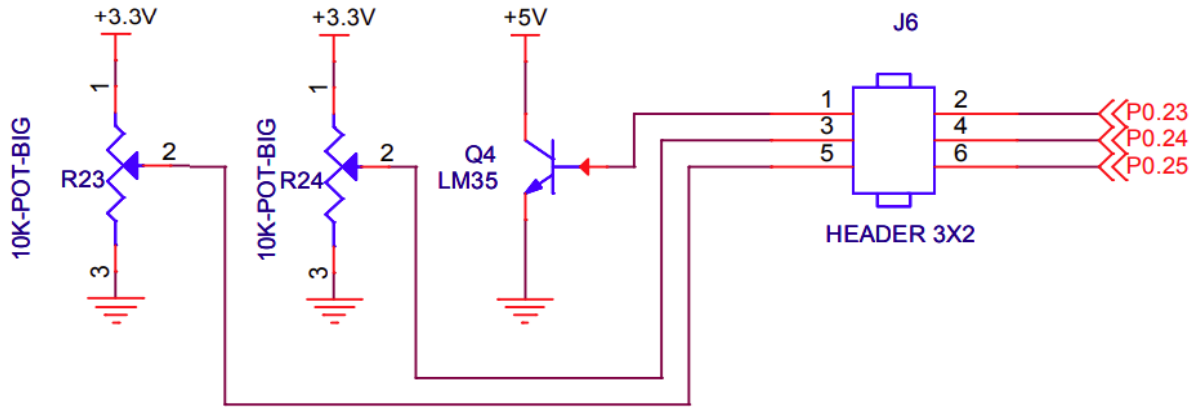
Turn ON DIP Switch (SW29) Pins 7-(0.2) and 8-(0.3).

Configure HyperTerminal @9600 baud rate.

Press RESET.</td><td rowspan="3">Digital Values of all the Three channels will be displayed in UART

Adjust the POT1 and POT2 or Apply temperature on LM35 to see the changes</td></tr>
<tr><td>POT1</td><td>P0.24</td></tr>
<tr><td>POT2</td><td>P0.25</td></tr>
</table>

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"
#include<stdio.h>

unsigned int CH0;//, CH1, CH2;
unsigned char Str[5];

unsigned char UART0_Receive(void);
void UART0_Txmt(unsigned char Chr);
void init_serial(void);
void UART0_puts(unsigned char *string);
void init_adc(void);
unsigned int ADC_Getdata(unsigned char channel);
void delay_ms(long ms) ;
void itoa(unsigned int val, unsigned char *str);

int main(void)
{
                        init_adc();
                        init_serial();
                        while(1)
                        {
                                CH0=ADC_Getdata(0);
        //Get data from channel0
                                itoa(CH0,Str);
        //Convert the val to string
                                UART0_Txmt('T');
                                UART0_Txmt(':');
                                UART0_puts(Str);
        //Step 9-Send the string to UART0
                                UART0_Txmt('\r');
        //Carriage return, Enter Key
                        UART0_Txmt('\n');
                                delay_ms(500);

                                CH1=ADC_Getdata(1);
        //Step 8-Convert the 2nd channel
                                itoa(CH1,Str);
                                UART0_Txmt('A');
                                UART0_Txmt('1');
                                UART0_Txmt(':');
                                UART0_puts(Str);
        //Step 9-Send ADC data to UART
                                UART0_Txmt(' ');

                                CH2=ADC_Getdata(2);
```

```
		//Step 8-Convert the 3rd channel
				itoa(CH2,Str);
				UART0_Txmt('A');
				UART0_Txmt('2');
				UART0_Txmt(':');
				UART0_puts(Str);
		//Step 9-Send ADC data to UART
				UART0_Txmt('\r');
		//Carriage return, Enter Key
				UART0_Txmt('\n');
				delay_ms(500);
			}
}

void init_adc(void)
{
		//1. set the PCADC bit
			LPC_SC->PCONP |= (1 << 12);
			/* enable power to ADC*/

		//2. Configure pins P0.23, P0.24 and P0.25 as ADC input pins
			LPC_IOCON->P0_23 =  1;                                          /* Pin P0.23
used as ADC0, IN0 */
			LPC_IOCON->P0_24 =  1;                                          /* Pin P0.24
used as ADC0, IN1 */
			LPC_IOCON->P0_25 =  1;
		/* Pin P0.25 used as ADC0, IN2 */
//
		//3. ADC conversion rate = 400Khz
		//30MHz/12.4MHz = 2.4-1 = 1 (8 to 15)
		//4. Enable PDN bit (21 bit)
			LPC_ADC->CR  = 0;
			LPC_ADC->CR |= ((1<<21) | (1<<8));
}

unsigned int ADC_Getdata(unsigned char channel)
{
			//5.Select the Channel and Start conversion
			LPC_ADC->CR |= (1<<24) | (1<<channel);
			//6. Wait for the DONE bit (31). If the DONE bit is HIGH, the conversion is completed
			while((LPC_ADC->DR[channel] & 0x80000000)==0);
			//Deselect the channel and Stop Conversion
			LPC_ADC->CR &= ~((1<<24) | (1<<channel));
			//7. Read the data
			return ((LPC_ADC->DR[channel]>>4)&0xFFF);
}

void init_serial(void)
{
			//set bit PCUART0
			LPC_SC->PCONP |= (1 << 3);
		/* enable power to UART0*/

			//Configure pins P0.2 and P0.3 as UART0 TX and RX
			LPC_IOCON->P0_2 |=  1;                                          /* Pin P0.2
used as TXD0 */
			LPC_IOCON->P0_3 |=  1;                                          /* Pin P0.3
used as RXD0 */

			//Select Clock source and frequency=PCLK ie 30MHz
			/* 8 bits, no Parity, 1 Stop bit   */
			LPC_UART0->LCR    = 0x83;

			//Derive baud rate from the UART clock source, Set DLAB=1 to access baud rate
			//Register
			//DLM:DLL=PCLK/(16*baud)= 30Mhz/(16*115200)= 16
			LPC_UART0->DLL    = 16;                 /* 115200 Baud Rate @ 30.0 MHZ PCLK*/
			LPC_UART0->DLM    = 0;                   /* MSB = 0 */
			LPC_UART0->LCR    = 0x03;                 /* DLAB = 0*/
}
```

```
//Transmit a character
void UART0_Txmt(unsigned char Chr)
{
                        while((LPC_UART0->LSR & 0x20)==0);                      //Bit5-THRE, Check THR empty or not
                        LPC_UART0->THR = Chr;
        //Send the next character
}

//Receive a character
unsigned char UART0_Receive(void)
{
                        while((LPC_UART0->LSR & 0x01)==0);                      //Bit0-RDR, Check receive data ready?
                        return(LPC_UART0->RBR);
        //Read the data
}

//Transmit a string
void UART0_puts(unsigned char *string)
{
                        while(*string)
                        UART0_Txmt(*string++);
}

void delay_ms(long ms)                                  // delay 1 ms per count @ CCLK 120 MHz
{
                        long i,j;
                        for (i = 0; i < ms; i++ )
                        for (j = 0; j < 26659; j++ );
}

//Int to string (ascii)
void itoa(unsigned int val, unsigned char *str)
{
        //Separate the single digit and add 0x30 (where the ascii '0' starts
        str[3] = val%10+'0';
        //Get the remainder
        val = val/10;
        //Separate the single digit of remainder ie 10th digit of val
        str[2] = val%10+'0';
        //Get the remainder
        val = val/10;
        //Separate the single digit of remainder ie 100th digit of val
        str[1] = val%10+'0';
        //Get the remainder, its the 1000th digit
        str[0] = val/10+'0';
        //End the string with NULL character
        str[4] = '\0';
}
```

EX3. LED and SWITCH

Aim:

This example describes how to interface LED and SWITCHES

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable

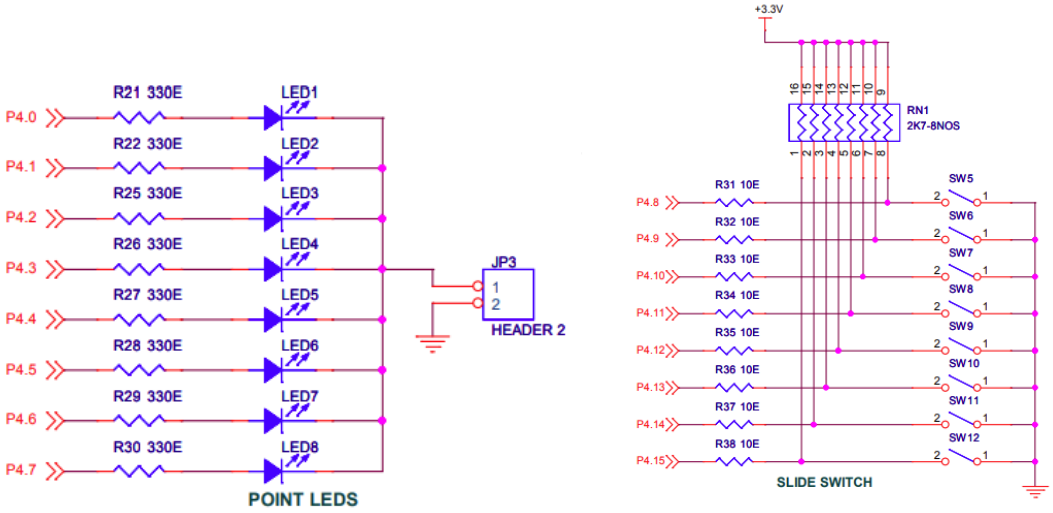
Procedure:

1. Configure pins P4.0 to P4.15 as GPIO pins i.e. IOCON_P4_0 to IOCON_P0_24.
2. Select LED pins P4.0 to P4.7 as output and SWITCH pins P4.8 to P4.15 as input
3. Turn ON/OFF LEDs depends on Switch status (MSB-LSB respectively)

Connections:

HARDWARE PIN OUT				CONNECTIONS	OUTPUT
LED1	P4.0	SW5	P4.8	Place a Jumper @JP3	Turn ON and OFF the switches SW5-12, The LED1-8 will be turned ON/OFF. I.e. LED displays the Switch Status.
LED2	P4.1	SW6	P4.9		
LED3	P4.2	SW7	P4.10		
LED4	P4.3	SW8	P4.11		
LED5	P4.4	SW9	P4.12		
LED6	P4.5	SW10	P4.13		
LED7	P4.6	SW11	P4.14		
LED8	P4.7	SW12	P4.15		

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"

void delay_ms(long ms);

int main(void)
{
		//1. Set the PCGPIO bit
		LPC_SC->PCONP |= (1<<15);

		//2. Configure pins P4.0 to P4.15 as GPIO pins
		LPC_IOCON->P4_0 = 0;
		LPC_IOCON->P4_1 = 0;
		LPC_IOCON->P4_2 = 0;
		LPC_IOCON->P4_3 = 0;
		LPC_IOCON->P4_4 = 0;
		LPC_IOCON->P4_5 = 0;
		LPC_IOCON->P4_6 = 0;
		LPC_IOCON->P4_7 = 0;
		LPC_IOCON->P4_8 = 0;
		LPC_IOCON->P4_9 = 0;
		LPC_IOCON->P4_10 = 0;
		LPC_IOCON->P4_11 = 0;
		LPC_IOCON->P4_12 = 0;
		LPC_IOCON->P4_13 = 0;
		LPC_IOCON->P4_14 = 0;
		LPC_IOCON->P4_15 = 0;

		//3. Configure the pins P4.0 to P4.7 as output
		//   SWITCH pins P4.8 to P4.15 as input
		LPC_GPIO4->DIR= 0x00FF;

		while(1)
		{
			//4. Read the Switch Status and display it in LEDs
			LPC_GPIO4->PIN >>= 8;
			delay_ms(50);
		}
}

void delay_ms(long ms)                                  // delay 1 ms per count @ CCLK 120 MHz
{
		long i,j;
		for (i = 0; i < ms; i++ )
		for (j = 0; j < 26659; j++ );
}
```

EX4. PWM INTERFACE

Aim:

This example describes how to generate a PWM with variable duty cycle

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable, CRO

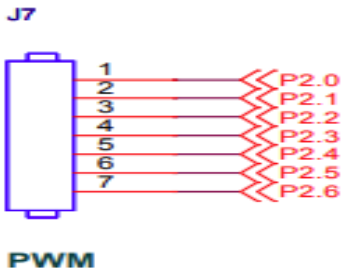
Procedure:

1. Power: In the PCONP register, set bit PCPWM1
2. Configure pins P2.0 to P2.6 for PWM using IOCON_P2_0 to IOCON_P2_6.
3. Select the clock source and clock value for PWM (peripheral clock).
4. Set match value for PWM match channel 0 (Frequency of PWM ie period)
5. Select and update the duty cycle for each channel (Match registers)
6. Select the single edge mode of PWM
7. Enable PWM Channel Output and Interrupts (If needed)

Connections:

HARDWARE PIN OUT (PWM1)		CONNECTIONS	OUTPUT
CH1	P2.0	J7-PWM	Measure the pulse width and duty cycle using CRO @J7 pins 1-7 Each pulse has different duty cycle
CH2	P2.1		
CH3	P2.2		
CH4	P2.3		
CH5	P2.4		
CH6	P2.5		
CH7	P2.6		

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"
#define PWMPRESCALE 30              //30 PCLK cycles to increment TC by 1 i.e 1 Micro-second
void init_pwm(void);
int main(void)
{
```

```
		LPC_SC->PCLKSEL = 4;
			init_pwm();
			while(1);
	}
	void init_pwm(void)
	{
		//1. set bit PCPWM1
			LPC_SC->PCONP |= (1 << 6);
				/* enable power to PWM1*/

		//2. Configure Pins P2.0 to P2.5 for PWM using IOCON_P2_0 to P2_5
			LPC_IOCON->P2_0 =  0x1;
			/* Pin P2.0 to P2.6 used as PWM */
			LPC_IOCON->P2_1 =  0x1;
			LPC_IOCON->P2_2 =  0x1;
			LPC_IOCON->P2_3 =  0x1;

			LPC_IOCON->P2_4 =  0x1;
			LPC_IOCON->P2_5 =  0x1;

			LPC_PWM1->PCR  = 0x0;
		// Step-3. Select Single Edge PWM - by default its single Edged so this line can be removed
			LPC_PWM1->PR   = PWMPRESCALE-1;                                  // Step-4. 1us resolution
	(30MHz/30)=1MHz
			LPC_PWM1->MR0  = 10000;                                                  // Step-5.
	10ms=10000us period duration ie 100Hz

			//6.Select the duty cycle for each channel
			LPC_PWM1->MR1  = 9500;
		// 9.5ms - pulse duration i.e Duty=95%
			LPC_PWM1->MR2  = 8000;
		// 8.0ms - pulse duration
			LPC_PWM1->MR3  = 6000;
		// 6.0ms - pulse duration
			LPC_PWM1->MR4  = 3000;
		// 3.0ms - pulse duration
			LPC_PWM1->MR5  = 1500;
		// 1.5ms - pulse duration
			LPC_PWM1->MR6  = 500;
		// 0.5ms - pulse duration i.e Duty=5%
			LPC_PWM1->LER  = 0x7F;
		// Update Period and Duty cycle

			//7.Enable PWM Channel Output
			LPC_PWM1->PCR  = (0x3F<<9);                                              // enable
	PWM1 to PWM6 output
			LPC_PWM1->TCR  = (1<<0) | (1<<3);                  // enable counters and PWM Mode

			//8.Reset the Counter
		//	LPC_PWM1->TCR  = (1<<1) ;                                                //
	Reset PWM TC & PR
}
```

EX6a. UART0

Aim:

This example describes how to make UART communication via UART0

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable

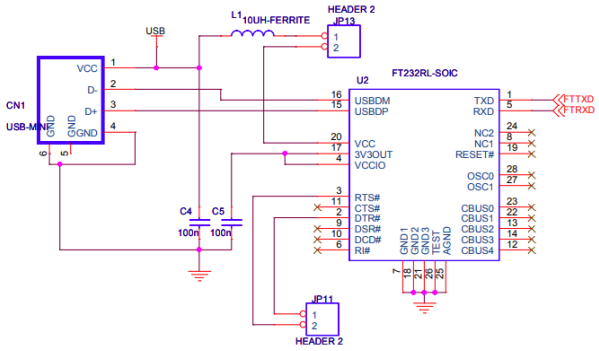
Procedure:

1. Power: In the PCONP register, set bit PCUART0
2. Configure pins P0.2 and P0.3 as UART0 TX and RX using IOCON_P0_2 and IOCON_P0_3.
3. Select Clock source and frequency
4. Derive baud rate from the UART clock source
5. Set no. of data bits, stop bit, parity and flow control (optional)
6. Enable transmit and receive interrupt
7. Transmit a character or string to communicate
8. Receive a character and process it by adding some task

Connections:

HARDWARE PIN OUT		CONNECTIONS	OUTPUT
U0TXD	P0.2	Turn ON 7-P0.2 and 8-P0.3 (UART0) pins of CONFIG switch SW29. Open HyperTerminal and Configure USB-UART COM Port Number, 9600, 8, N, 1, N. Press RESET Once (If Necessary)	A Welcome string will be displayed and then Type a character, the same character will be returned from Board.
U0RXD	P0.3		

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"
unsigned char recval=0, welstring[]="Hi: Press a Key\r\n";
unsigned char UART0_Receive(void);
void UART0_Txmt(unsigned char Chr);
void init_serial(void);
void UART0_puts(unsigned char *string);
int main(void)
```

```
{
		init_serial();
		UART0_puts(welstring);
		while(1)
		{
			recval=UART0_Receive();
			UART0_Txmt(recval);
		}
}
void init_serial(void)
{
		LPC_SC->PCONP |= (1 << 3);		//1. set bit PCUART0/* enable power to UART0*/
		//2. Configure pins P0.2 and P0.3 as UART0 TX and RX
		LPC_IOCON->P0_2 |=  1;					/* Pin P0.2 used as TXD0 */
		LPC_IOCON->P0_3 |=  1;					/* Pin P0.3 used as RXD0 */
		//3.Select Clock source and frequency=PCLK ie 30MHz	/* 8 bits, no Parity, 1 Stop bit   */
		// Set DLAB=1 to access baud rate Register
		LPC_UART0->LCR    = 0x83;		//4. Derive baud rate from the UART clock source,
		//DLM:DLL=PCLK/(16*baud)= 30Mhz/(16*9600)= 195
		LPC_UART0->DLL    = 195;		/* 9600 Baud Rate @ 30.0 MHZ PCLK*/
		LPC_UART0->DLM    = 0;		/* MSB = 0 */
		LPC_UART0->LCR    = 0x03;		/* DLAB = 0*/
}
void UART0_Txmt(unsigned char Chr)    //Transmit a character
{
	//6. Check the Transmitter flag
	//   If it is High Transmit a character
		while((LPC_UART0->LSR & 0x20)==0);
		LPC_UART0->THR = Chr;
}
unsigned char UART0_Receive(void)    //Receive a character
{
		//7.Check the Receiver flag for data ready
		//  and Read the character
		while((LPC_UART0->LSR & 0x01)==0);
		return(LPC_UART0->RBR);
}
void UART0_puts(unsigned char *string)    //Transmit a string
{
		while(*string)
		UART0_Txmt(*string++);
}
```

EX7. 4x4 MATRIX KEYPAD

Aim:

This example describes how to use 4x4 matrix keypad and display its result in LCD

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable, LCD

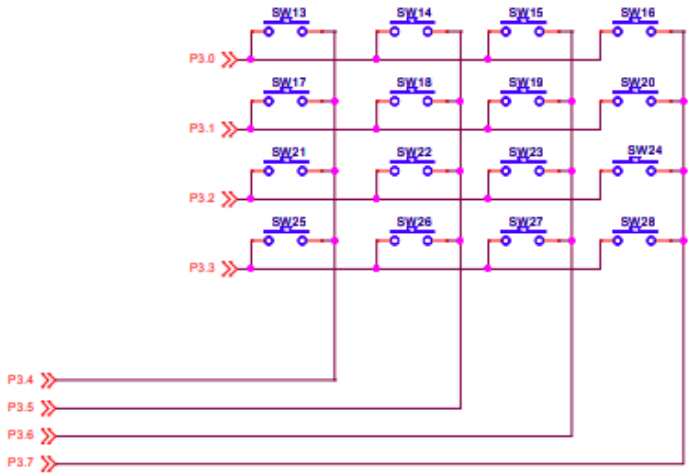
Procedure:

1. Power: In the PCONP register, set the PCGPIO bit
2. Configure all the row pins as output and make it HIGH
3. Configure all the column pins as input and make it HIGH
4. Configure all the LCD pins as output and write necessary software routines for LCD initialization, command and data
5. Make a row pin Zero and check all the four column lines for zero
6. If any key is pressed, the corresponding column will get zero
7. Read the key value and display it in LCD

Connections:

HARDWARE PIN OUT		CONNECTION	OUTPUT
ROW1	P3.0	Turn ON DIP Switch (SW29) Pins 2-LCD+	Press a Key (4x4 matrix keypad) and the number will be displayed LCD
ROW2	P3.1		
ROW3	P3.2		
ROW4	P3.3		
COL1	P3.4		
COL2	P3.5		
COL3	P3.6		
COL4	P3.7		

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"
#define RS_PORT                 (0)
#define RS_PIN          (4)
#define RS_VAL          (1 << RS_PIN)
#define EN_PORT                 (0)
#define EN_PIN          (5)
#define EN_VAL                  (1 << EN_PIN)
#define DATA     LPC_GPIO4->PIN
#define keyportc  LPC_GPIO3->CLR
```

```
#define keyports  LPC_GPIO3->SET
#define COL1                    (LPC_GPIO3->PIN & 0x10)
#define COL2                    (LPC_GPIO3->PIN & 0x20)
#define COL3                    (LPC_GPIO3->PIN & 0x40)
#define COL4                    (LPC_GPIO3->PIN & 0x80)
unsigned char k=0,key=0,i=0;
unsigned char Lcd_LINE1[]={"KEYPAD DEMO:    "};
void delay_ms(long ms);
void lcd_command(unsigned char cmd);
void lcd_data(unsigned char data);
void lcd_init(void);
unsigned char get_key(void);
int main(void)
{
        LPC_SC->PCONP |= (1<<15); //1. Set the PCGPIO bit
        LPC_IOCON->P3_0 = 0x00; //2. Configure all Keypad pins as GPIO
        LPC_IOCON->P3_1 = 0x10;
        LPC_IOCON->P3_2 = 0x10;
        LPC_IOCON->P3_3 = 0x10;
        LPC_IOCON->P3_4 = 0x10;
        LPC_IOCON->P3_5 = 0x10;
        LPC_IOCON->P3_6 = 0x10;
        LPC_IOCON->P3_7 = 0x10;
//3.Configure all the row pins as output and make it HIGH Configure all the column pins as input and make it HIGH
                LPC_GPIO3->DIR = 0x0F;
                LPC_GPIO3->PIN = 0x0F;

                //4. Configure all the LCD pins as outputs   //   RS-P0.4, EN-P0.5, Datalines-P4.28 to P4.31
                LPC_IOCON->P0_4 = 0; LPC_IOCON->P0_5 = 0;
                LPC_IOCON->P4_28 = 0; LPC_IOCON->P4_29 = 0;
                LPC_IOCON->P4_30 = 0; LPC_IOCON->P4_31 = 0;
                //5. Configure the pins P4.28 to P4.31 as output        //Configure the pins P0.4 and P0.5 as output
                LPC_GPIO0->DIR |= 0x30;         LPC_GPIO4->DIR  = (0xF << 28);
                //5. Initialize LCD lcd_init();         //Send First line command
                lcd_command(0x80);      //Send 16 characters from the line1 array
                for(i=0;Lcd_LINE1[i]!='\0';i++)
                {
                        lcd_data(Lcd_LINE1[i]);
                        delay_ms(50);
                }
                delay_ms(1000);
                while (1)
                {
                                if(get_key()!=0){
                                        lcd_command(0xc0);
                                        if(key<=10)lcd_data(key-1+'0');
                                        else lcd_data(key-11+'A');
                                }
                         delay_ms(100);
                }
}
void lcd_command(unsigned char cmd)
{
                //Clear RS pin--Command mode
                LPC_GPIO0->CLR |= RS_VAL;
                //Place the MSB 4bits on data lines
```

```
                DATA=cmd<<24;  //Pulse the EN pin
                LPC_GPIO0->PIN |= EN_VAL;
                LPC_GPIO0->CLR |= EN_VAL;
                //Place the LSB 4bits on data lines
                DATA=cmd<<28;//Pulse the EN pin
                LPC_GPIO0->PIN |= EN_VAL;
                LPC_GPIO0->CLR |= EN_VAL;
                delay_ms(5);
}
void lcd_data(unsigned char data)
{
                //Set RS pin--Command mode
                LPC_GPIO0->PIN |= RS_VAL;
                //Place the MSB 4bits on data lines
                DATA=data<<24;
                LPC_GPIO0->PIN |= EN_VAL;
                LPC_GPIO0->CLR |= EN_VAL;
                //Place the LSB 4bits on data lines
                DATA=data<<28; //Pulse the EN pin
                LPC_GPIO0->PIN |= EN_VAL;
                LPC_GPIO0->CLR |= EN_VAL;
                delay_ms(5);
}
void lcd_init(void)
{
                delay_ms(100);                          //LCD power up time
                lcd_command(0x33);                              //Wake up
                lcd_command(0x32);                              //Wake up
                lcd_command(0x28);                              //4bit mode
                lcd_command(0x0C);                              //display on and cursor off
                lcd_command(0x06);                              //Entry mode and shift
                lcd_command(0x01);                              //Clear LCD
                delay_ms(200);                                          //Give more time to settle
}




void delay_ms(long ms)                                  // delay 1 ms per count @ CCLK 120 MHz
{
                long i,j;
                for (i = 0; i < ms; i++ )
                for (j = 0; j < 26659; j++ );
}


unsigned char get_key(void)
{
                                        LPC_GPIO3->PIN=0x0F;
      k=1;
      for(i=0;i<4;i++){
            keyportc |=(0x01<<i);
          //Scan for a Key by sending '0' on ROWS
```

```
                                                        lcd_command(0xc0);
                if(COL1==0){                                                    //when a
key pressed numbers 1--16 will be returned
                        key = k+0;
                        while(COL1==0);
                        return key;
                }
                if(COL2==0){
                        key = k+1;
                        while(COL2==0);
                        return key;
                }
                if(COL3==0){
                        key = k+2;
                        while(COL3==0);
                        return key;
                }
                if(COL4==0){
                        key = k+3;
                        while(COL4==0);
                        return key;
                }
            k+=4;
            keyports |=(0x01<<i);
        }
        return 0;
}
```

EX8. LCD

Aim:

This example describes how to display a string in LCD using 4 bit mode

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable, LCD

Procedure:

1. Power: In the PCONP register, set the PCGPIO bit
2. Configure all the LCD pins as outputs
3. Store all the initialization command and data to be displayed in an array
4. Make RS pin low and send a command, pulse the EN pin ON/OFF for a moment
5. Repeat step 4 for all the initialization commands
6. Send First line command using step 4

7. Make RS pin high and send a character to be displayed, pulse the EN pin ON/OFF for a moment
8. Repeat step 5 until the end of first line (16 characters)
9. Send the second line command using step 4
10. Make RS pin high and send a character to be displayed, pulse the EN pin ON/OFF for a moment
11. Repeat step 5 until the end of second line (16 characters)

Connections:

HARDWARE PIN OUT		CONNECTIONS	OUTPUT
CONTROL LINES		Turn ON DIP Switch (SW29) Pins 2-LCD+	The Strings "CORTEX DEV BOARD" and "LCD DEMO PROGRAM" will be displayed on LCD.
RS	P0.4		
RW	GND		
EN	P0.5		
DATA LINES			
D0	-		
D1	-		
D2	-		
D3	-		
D4	P4.28		
D5	P1.29		
D6	P1.30		
D7	P1.31		

EX9. I2C EEPROM

Aim:

This example describes how to make a communication with I2C EEPROM

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable

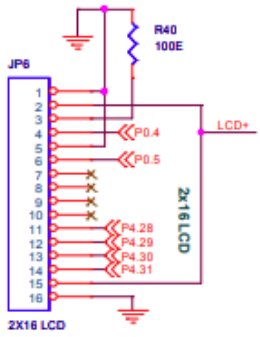
Procedure:

1. Power: In the PCONP register, set the PCI2C0 bit
2. Configure pins P0.27 (CH0) and P0.28 for I2C function using IOCON registers
3. Select Clock source and frequency for I2C
4. Derive the required baud rate for I2C and UART0 from clock source
5. Initialize I2C and enable .I2C interrupt
6. Write a string of data to I2C EEPROM
7. Read back the data from I2C
8. Configure the UART0 using EX6
9. Send the data read from I2C to the HyperTerminal via UART0

Connections:

HARDWARE PIN OUT		CONNECTIONS	OUTPUT			
CLK (6)	P0.28	Turn ON 3-EE, 7-P0.2 and 8-P0.3 (UART0) pins of CONFIG switch SW29. Open HyperTerminal and Configure USB-UART COM Port Number, as 9600, 8, N, 1, N. Place a Jumper @JP3 (LED) Make all the switches SW5-7 HIGH LED4 is for Error indication	SW7	SW6	SW5	
DATA (5)	P0.27					
SW5	P4.8					
SW6	P4.9		1	1	0	Writes EEPROM LED1 Blinks, Successful message will be displayed in HyperTerminal
SW7	P4.10					
LED1	P4.0		1	0	1	Reads EEPROM LED2 Blinks, Successful message will be displayed in HyperTerminal
LED2	P4.1					
LED3	P4.2		0	1	1	Erases EEPROM LED3 Blinks, Successful message will be displayed in HyperTerminal
LED4	P4.3					

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"
#define RS_PORT                 (0)
#define RS_PIN          (4)
#define RS_VAL                  (1 << RS_PIN)

#define EN_PORT                 (0)
#define EN_PIN          (5)
```

```
#define EN_VAL                  (1 << EN_PIN)

#define DATA    LPC_GPIO4->PIN
//data array
unsigned char Lcd_LINE1[]={"CORTEX DEV BOARD"},i=0;
unsigned char Lcd_LINE2[]={"LCD DEMO PROGRAM"};

void delay_ms(long ms);
void lcd_command(unsigned char cmd);
void lcd_data(unsigned char data);
void lcd_init(void);

int main(void)
{

        //1. Set the PCGPIO bit
        LPC_SC->PCONP |= (1<<15);

        //2. Configure all the LCD pins as GPIO
        //      RS-P0.4, EN-P0.5, Datalines-P4.28 to P4.31
        LPC_IOCON->P0_4 = 0;
        LPC_IOCON->P0_5 = 0;
        LPC_IOCON->P4_28 = 0;
        LPC_IOCON->P4_29 = 0;
        LPC_IOCON->P4_30 = 0;
        LPC_IOCON->P4_31 = 0;


        //3. Configure the pins P4.28 to P4.31 as output
        //      Configure the pins P0.4 and P0.5 as output
        LPC_GPIO0->DIR |= 0x30;
        LPC_GPIO4->DIR  = (0xF << 28);

        //4. Initialize LCD by sending Initialisation commands
        lcd_init();

        while (1)
        {
                //11. Send First line command
                lcd_command(0x80);
                //12. Send 16 characters from the line1 array
                for(i=0;Lcd_LINE1[i]!='\0';i++)
                {
                        lcd_data(Lcd_LINE1[i]);
                        delay_ms(50);
                }

                //13. Send Second line command
                lcd_command(0xc0);
```

```
                                        //14. Send 16 characters from the line2 array
                                        for(i=0;Lcd_LINE2[i]!='\0';i++)
                                        {
                                                lcd_data(Lcd_LINE2[i]);
                                                delay_ms(50);
                                        }
                                        delay_ms(1000);
                                        //Clear the display and repeat
                                        lcd_command(0x01);
                        }
}

void lcd_command(unsigned char cmd)
{
                        //5. Clear RS pin--Command mode
                        LPC_GPIO0->CLR |= RS_VAL;

                        //6. Place the MSB 4bits on data lines
                        DATA=cmd<<24;
                //6. Pulse the EN pin
                        LPC_GPIO0->PIN |= EN_VAL;
                        LPC_GPIO0->CLR |= EN_VAL;

                        //7. Place the LSB 4bits on data lines
                        DATA=cmd<<28;
                        //7. Pulse the EN pin
                        LPC_GPIO0->PIN |= EN_VAL;
                        LPC_GPIO0->CLR |= EN_VAL;
                        delay_ms(5);
}

void lcd_data(unsigned char data)
{
                        //8. Set RS pin--data mode
                        LPC_GPIO0->PIN |= RS_VAL;

                        //9. Place the MSB 4bits on data lines
                        DATA=data<<24;
                        //9. Pulse the EN pin
                        LPC_GPIO0->PIN |= EN_VAL;
                        LPC_GPIO0->CLR |= EN_VAL;

                        //10. Place the LSB 4bits on data lines
                        DATA=data<<28;
                        //10. Pulse the EN pin
                        LPC_GPIO0->PIN |= EN_VAL;
                        LPC_GPIO0->CLR |= EN_VAL;
                        delay_ms(5);
}
```

```
void lcd_init(void)
{
        delay_ms(100);                  //LCD powerup time
        lcd_command(0x33);              //Wake up
        lcd_command(0x32);              //Wake up
        lcd_command(0x28);              //4bit mode
        lcd_command(0x0C);              //display on and cursor off
        lcd_command(0x06);              //Entry mode and shift
        lcd_command(0x01);              //Clear LCD
        delay_ms(200);                          //Give more time to settle
}

void delay_ms(long ms)                  // delay 1 ms per count @ CCLK 120 MHz
{
        long i,j;
        for (i = 0; i < ms; i++ )
        for (j = 0; j < 26659; j++ );
}
```

EX10. EXTERNAL INTERRUPT

Aim:

This example describes how to use the External interrupts by using SW4

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable

Procedure:

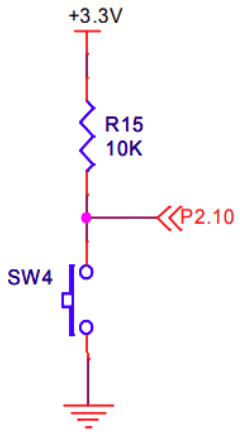
1. Power: In the PCONP register, set the PCGPIO bit
2. Configure pins P4.0 to P4.7 as GPIO pins using their respective IOCON registers IO-CON_P4_0 to IOCON_P4_7
3. Configure the pins P4.0 to P4.7 as output pins using their direction registers
4. Select the priority of external interrupt
5. Select the triggering of interrupt whether edge or level triggering
6. Enable interrupt
7. Go to the interrupt service routine and do some task
8. Clear the Interrupt

Connections:

HARDWARE PIN OUT	CONNECTIONS	OUTPUT

EINT0	P2.10	Place a Jumper @JP3 (LED)	Press SW4 (EINT0) and the interrupt count is displayed in LED

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"
unsigned int count=0;
void delay_ms(long ms);
void Init_LED(void);
void Init_EINT0(void);
void EINT0_IRQHandler(void)
{
        //6. the interrupt service routine
        LPC_SC->EXTINT  |= 1;                   //7. Clear Interrupt flag
        count++;
        while((LPC_GPIO2->PIN & 0x400)==0);
}

int main(void)
{
        Init_LED();
        Init_EINT0();
        while(1)
        {
                LPC_GPIO4->PIN = count;
        }
}

void delay_ms(long ms)                          // delay 1 ms per count @ CCLK 120 MHz
{
        long i,j;
        for (i = 0; i < ms; i++ )
        for (j = 0; j < 26659; j++ );
}

void Init_LED(void)
```

```
{
                    //Set the PCGPIO bit
                    LPC_SC->PCONP |= (1<<15);

                    //Configure pins P4.0 to P4.7 as GPIO pins
                    LPC_IOCON->P4_0 = 0;        LPC_IOCON->P4_1 = 0;
                    LPC_IOCON->P4_2 = 0;        LPC_IOCON->P4_3 = 0;
                    LPC_IOCON->P4_4 = 0;        LPC_IOCON->P4_5 = 0;
                    LPC_IOCON->P4_6 = 0;        LPC_IOCON->P4_7 = 0;
                    //Configure the pins P4.0 to P4.7 as output
                    LPC_GPIO4->DIR= 0xFF;
}
void Init_EINT0(void)
{
                    //1. Configure pin P2.10 as EINT0 pin
                    LPC_IOCON->P2_10 = 1;
                    //2. Configure the pin P2.10 as input
                    LPC_GPIO2->DIR  &= ~(1<<10);
                    //3. Select Edge Triggering Mode and Select Falling Edge
                    LPC_SC->EXTMODE = 1;
                    LPC_SC->EXTPOLAR= 0;
                    //4. Enable Interrupt and Set Priority
                    NVIC->ISER[0] |= (1<<18);                                   //Enable EINT0
                    NVIC->IP[4]   |= (0x3<<19);                 //Interrupt Priority 0-High, 31-low
                    //5. Clear Interrupt Flag
                    LPC_SC->EXTINT  = 1;
}
```

EX12. FLASHING LEDS

Aim:

This example describes how to use Flash an LED.

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable

Procedure:

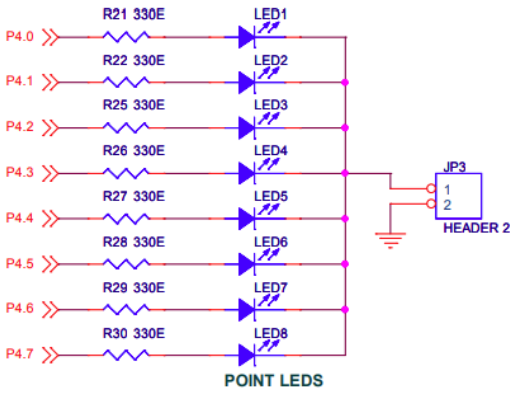
1. Power: In the PCONP register, set the PCGPIO bit
2. Configure pins P4.0 to P4.7 as GPIO pins using IOCON registers IOCON_P4_0 to IO-CON_P4_7
3. Configure the pins P4.0 to P4.7 as output pins using their direction registers
4. Make all the P4.0 to P4.7 pins as high
5. Pause the system for few milliseconds
6. Make all the P4.0 to P4.7 pins as low
7. Pause the system for few milliseconds
8. Repeat steps 4 to 7

Connections:

HARDWARE PIN OUT		CONNECTIONS	OUTPUT
LED1	P4.0	Place a Jumper @JP3	LED's will be Turned ON and OFF at 500ms interval.
LED2	P4.1		

LED3	P4.2		
LED4	P4.3		
LED5	P4.4		
LED6	P4.5		
LED7	P4.6		
LED8	P4.7		

Schematic:



Program:

```
#include "LPC407x_8x_177x_8x.h"


void delay_ms(long ms);

int main(void)
{
		//1. Set the PCGPIO bit
		LPC_SC->PCONP |= (1<<15);

		//2. Configure pins P4.0 to P4.7 as GPIO pins
		LPC_IOCON->P4_0 = 0;
		LPC_IOCON->P4_1 = 0;
		LPC_IOCON->P4_2 = 0;
		LPC_IOCON->P4_3 = 0;
		LPC_IOCON->P4_4 = 0;
		LPC_IOCON->P4_5 = 0;
		LPC_IOCON->P4_6 = 0;
		LPC_IOCON->P4_7 = 0;

		//3. Configure the pins P4.0 to P4.7 as output
		LPC_GPIO4->DIR= 0xFF;

		while(1)
		{
			//4. Send a High
			LPC_GPIO4->PIN= 0xFF;
			//5.Pause the system for few milliseconds
			delay_ms(500);
			//6.Send a low
```

```
                    LPC_GPIO4->PIN= 0x00;
                    //7.Pause the system for few milliseconds
                    delay_ms(500);
                }
}

void delay_ms(long ms)                          // delay 1 ms per count @ CCLK 120 MHz
{
                long i,j;
                for (i = 0; i < ms; i++ )
                for (j = 0; j < 26659; j++ );
}
```

EX13. STEPPER

Aim:

This example describes how to run a stepper motor.

Components Required:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable, Stepper motor

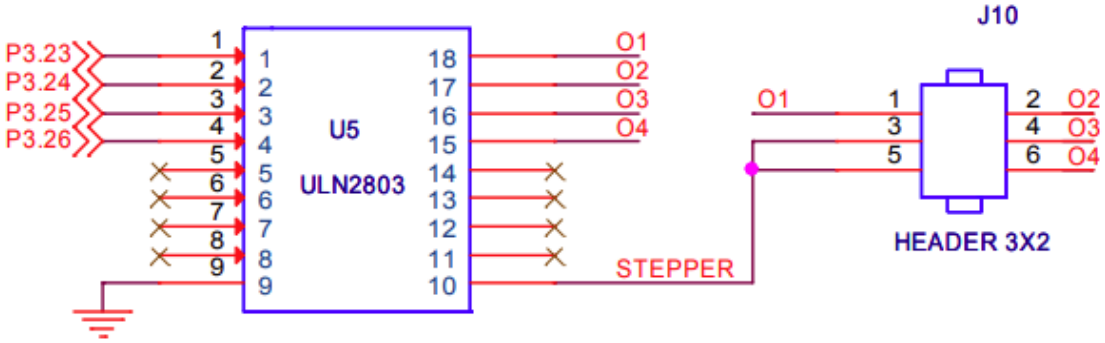
Procedure:

1. Power: In the PCONP register, set the PCGPIO bit
2. Configure pins P3.23 to P3.26 as GPIO pins using IOCON registers IOCON_P3_23 to IO-CON_P3_26
3. Configure the pins P3.23 to P3.24 as output pins using their direction registers.
4. Send the stepper sequence via IO lines

Connections:

HARDWARE PIN OUT		CONNECTIONS	OUTPUT
PHASE.1	P3.23	Connect a Stepper Motor @J10 (6 wire) Turn ON DIP Switch (SW29) Pin1 (5V-STP).	The Stepper motor will rotate in both clockwise and counter clockwise. The direction will be changed after few rotations in either direction
PHASE.2	P3.24		
PHASE.3	P3.25		
PHASE.4	P3.26		

Schematic:


P3.23
P3.24
P3.25
P3.26
U5
ULN2803
O1
O2
O3
O4
STEPPER
J10
HEADER 3X2

Program:

```
#include "LPC407x_8x_177x_8x.h"
void delay_ms(long ms);
int main(void)
{
        //1. Set the PCGPIO bit
        LPC_SC->PCONP |= (1<<15);
        //2. Configure pins P3.23 to P3.26 as GPIO pins
        LPC_IOCON->P3_23 = 0x0;
        LPC_IOCON->P3_24 = 0x0;
        LPC_IOCON->P3_25 = 0x0;
        LPC_IOCON->P3_26 = 0x0;
        //3.Configure the pins P3.23 to P3.24 as output pins
        LPC_GPIO3->DIR = (0x0F<<23);

        while(1)
        {
                //4. Send the stepper sequence//Stepper Squence 1001,1100,0110,0011
                LPC_GPIO3->PIN=(0x09<<23);
                delay_ms(5);
          LPC_GPIO3->PIN=(0x0c<<23);
          delay_ms(5);
          LPC_GPIO3->PIN=(0x06<<23);
          delay_ms(5);
          LPC_GPIO3->PIN=(0x03<<23);
                delay_ms(5);
        }
}

void delay_ms(long ms)                  // delay 1 ms per count @ CCLK 120 MHz
{
        long i,j;
        for (i = 0; i < ms; i++ )
        for (j = 0; j < 26659; j++ );
}
```

EX14. ZIGBEE

Aim:

This example describes how to make a wireless communication between two CORTEX Boards using ZIGBEE

Components Required:

2 no of PS- CORTEX-M4-TYRO-V4r2, 2- Mini USB cable, 2-ZIGBEE modules

Procedure:

1. Power: In the PCONP register, set bit PCUART3
2. Configure pins P0.0 and P0.1 as UART3 TX and RX using IOCON_P0_0 and IOCON_P0_1.
3. Select Clock source and frequency
4. Derive baud rate from the UART clock source
5. Set no. of data bits, stop bit, parity and flow control (optional)
6. Enable transmit and receive interrupt
7. Read the Slide switch positions
8. Transmit over ZIGBEE

Connections:

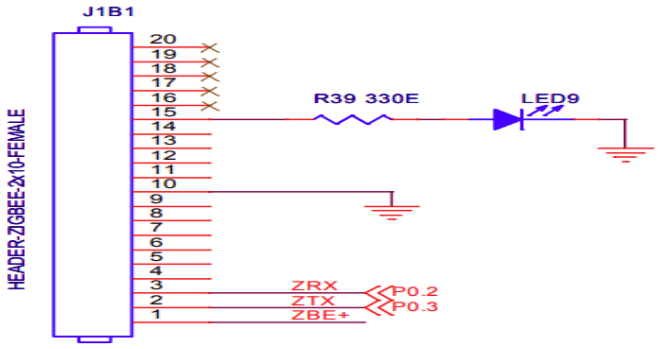
Transmitter

HARDWARE PIN OUT		CONNECTIONS	OUTPUT
U3TXD	P0.0	Zigbee module connects directly with P0.0 and P0.1 so no connections needed	Transmitter transmits the Slide Switch position over Zigbee
U3RXD	P0.1		

Receiver

HARDWARE PIN OUT		CONNECTIONS	OUTPUT
U3TXD	P0.0	Zigbee module connects directly with P0.0 and P0.1 so no connections needed Place a Jumper at JP3 (LED)	Receiver receives the switch positions from transmitter and displays its value in LED
U3RXD	P0.1		

Schematic:

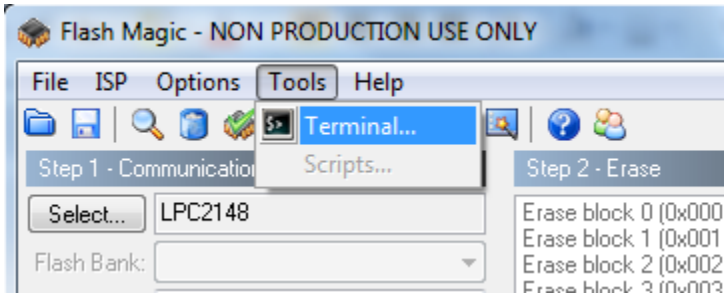


CHAPTER 5: Setting up HyperTerminal

There is no HyperTerminal in Windows 7 and the later versions of windows,

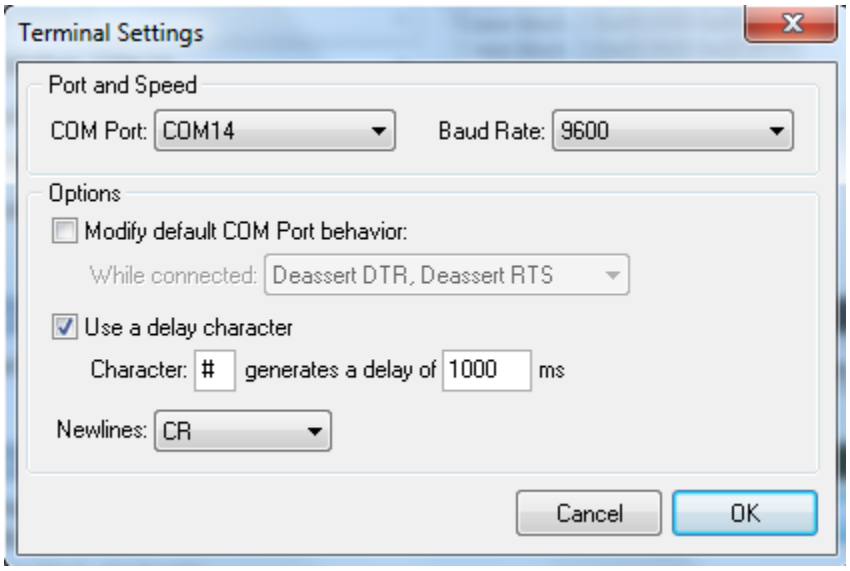
So you can use Flash Magic Terminal Program

1. Click Tools→ Terminal

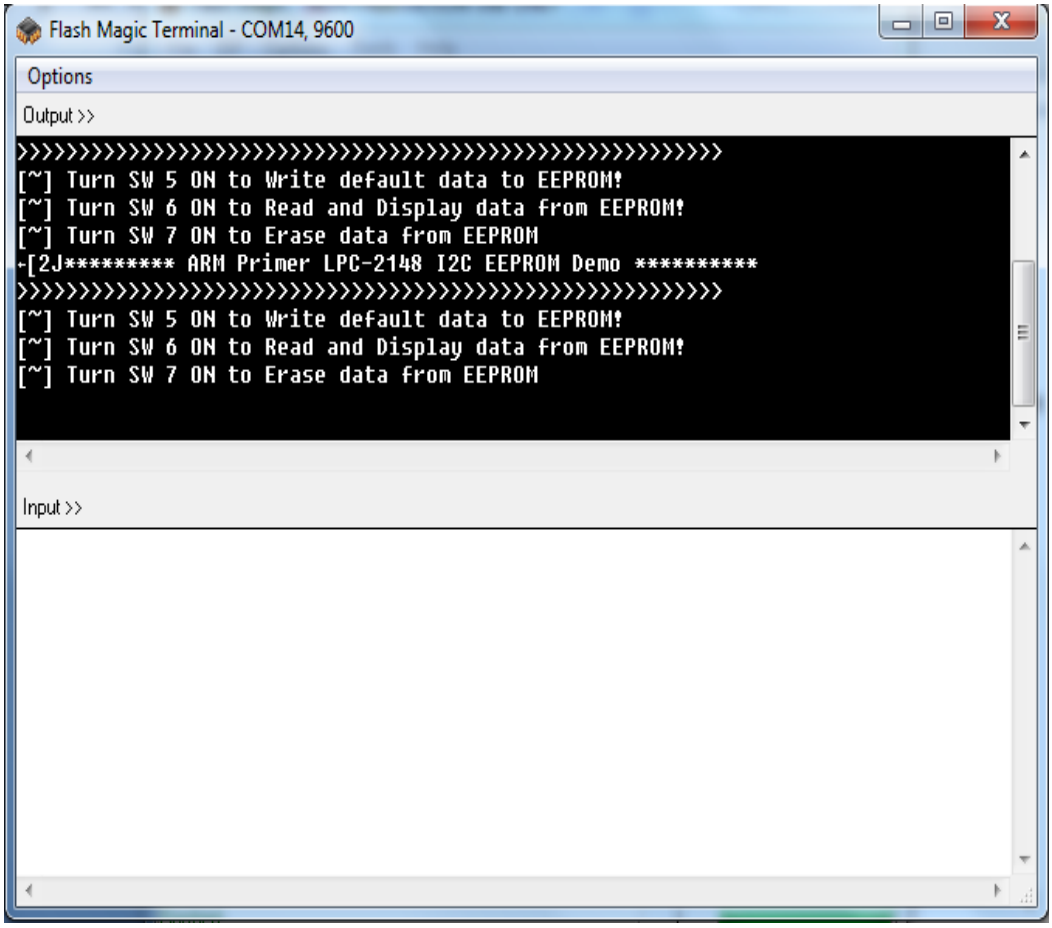


2. Select your COM port number and Baud rate

   Check the box "Use a delay character" and Select CR in Newlines (just for new line character)



3. The output will display the received messages
   The input textbox used to transmit characters. Just type something if you need to transmit data's



CHAPTER 6: Possible Errors and Solutions

6.1 PROBLEM CAUSED BY FT232

1. This error may be due to incorrect driver installed.
2. Windows 7, 8 and 10 will automatically install the latest driver; this may cause error if it is not properly updated. Turn OFF automatic driver installation in Windows. Uninstall all the FT232 drivers and install FT232 v2.8 driver.

6.2 SHORTCUTS

1. Use the jumper JP13 to restart the FT232 instead of removing and reinserting the USB cable (in case of restarting the FT232).
2. Buzzer can be used with any Port pin by removing the jumper @JP10 and connecting a hook wire between middle pin of JP10 and any port pin.