FIRE BIRD V

LPC2148 ARM7
ROBOTIC RESEARCH PLATFORM
Software Manual

Designed By:

ERTS Lab, CSE, IIT Bombay
www.it.iitb.ac.in/~erts

NEX ROBOTICS
www.nex-robotics.com

Manufactured By: NEX Robotics Pvt. Ltd.

FIRE BIRD V

SOFTWARE MANUAL

Version 1.0
November 22, 2010

Documentation author
Rohit Chauhan, NEX Robotics
Sachitanand Malewar, NEX Robotics Pvt. Ltd.
Aditya Sharma, NEX Robotics Pvt. Ltd.

Credits (Alphabetically)
Aditya Sharma, NEX Robotics
Amey Apte, NEX Robotics
Anant Malewar, EE, M.Tech, IIT Bombay
Ashish Gudhe, CSE, M.Tech, IIT Bombay
Behlul Sutarwala, NEX Robotics
Gurulingesh R. CSE, M.Tech, IIT Bombay
Inderpreet Arora, EE, M.Tech, IIT Bombay
Prof. Kavi Arya, CSE, IIT Bombay
Prof. Krithi Ramamritham, CSE, IIT Bombay
Nandan Salunke, RA, CSE, IIT Bombay
Pratim Patil, NEX Robotics
Preeti Malik, RA, CSE, IIT Bombay
Prakhar Goyal, CSE, M.Tech, IIT Bombay
Raviraj Bhatane, RA, CSE, IIT Bombay
Rajanikant Sawant, NEX Robotics
Saurabh Bengali, RA, CSE, IIT Bombay
Vaibhav Daghe, RA, CSE, IIT Bombay
Vibhooti Verma, CSE, M.Tech, IIT Bombay

Notice

The contents of this manual are subject to change without notice. All efforts have been made to ensure the accuracy of contents in this manual. However, should any errors be detected, NEX Robotics welcomes your corrections. You can send us your queries / suggestions at info@nex-robotics.com

⚠ **Robot's electronics is static sensitive. Use robot in static free environment.**
⚠ **Read the hardware and software manual completely before start using this robot**

Recycling:

Almost all of the robot parts are recyclable. Please send the robot parts to the recycling plant after its operational life. By recycling we can contribute to cleaner and healthier environment for the future generations.

Index

1.	**Introduction**	**6**
2.	**Programming the Fire Bird V ARM7 LPC2148 Robot**	**10**
3.	**Input / Output Operations On the Robot**	**46**
4.	**Sample Applications**	**55**

1 Introduction

Thanks for choosing the Fire Bird V mobile robot platform. Fire Bird V will give you good exposure to the world of robotics and embedded systems. Thanks to its innovative architecture and adoption of the 'Open Source Philosophy' in its software and hardware design, you will be able to create and contribute to complex applications that run on this platform, helping you acquire expertise as you spend more time with them.

Origins of the Fire Bird V

The Fire Bird V robot is the 5th in the Fire Bird series of robots. First two versions of the robots were designed for the Embedded Real-Time Systems Lab of Department of Computer Science and Engineering, IIT Bombay. Theses platforms made commercially available from the version 3 onwards.

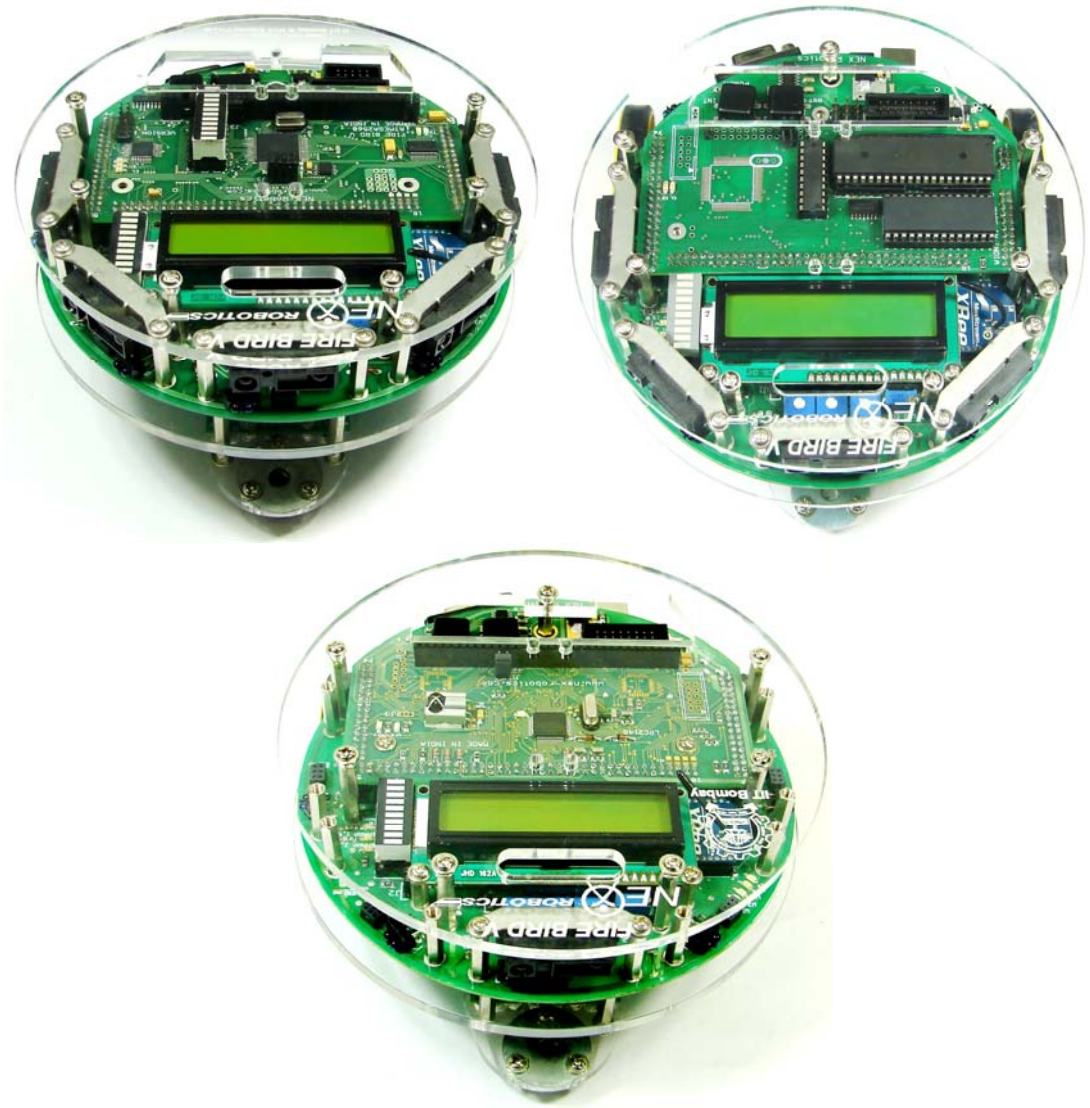

Figure 1.1: Fire Bird V Robots

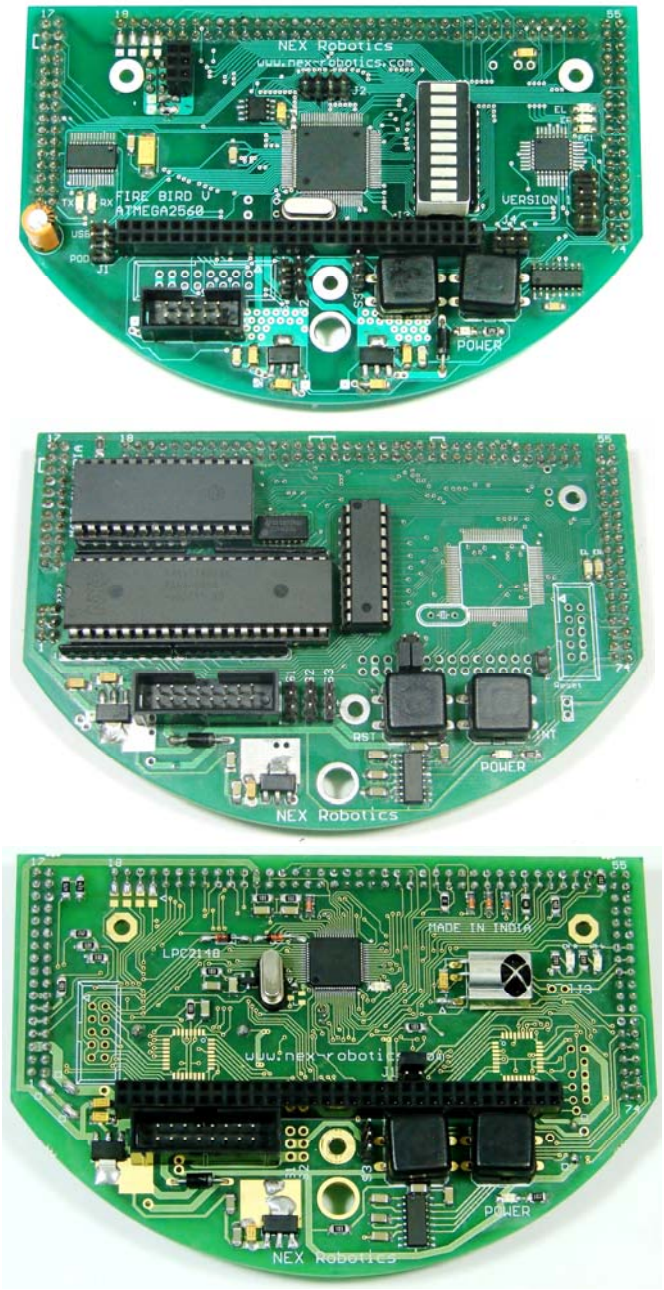

Figure 1.2: ATMEGA2560 (AVR), LPC2148 (ARM7) and P89V51RD2 (8051) microcontroller adaptor board

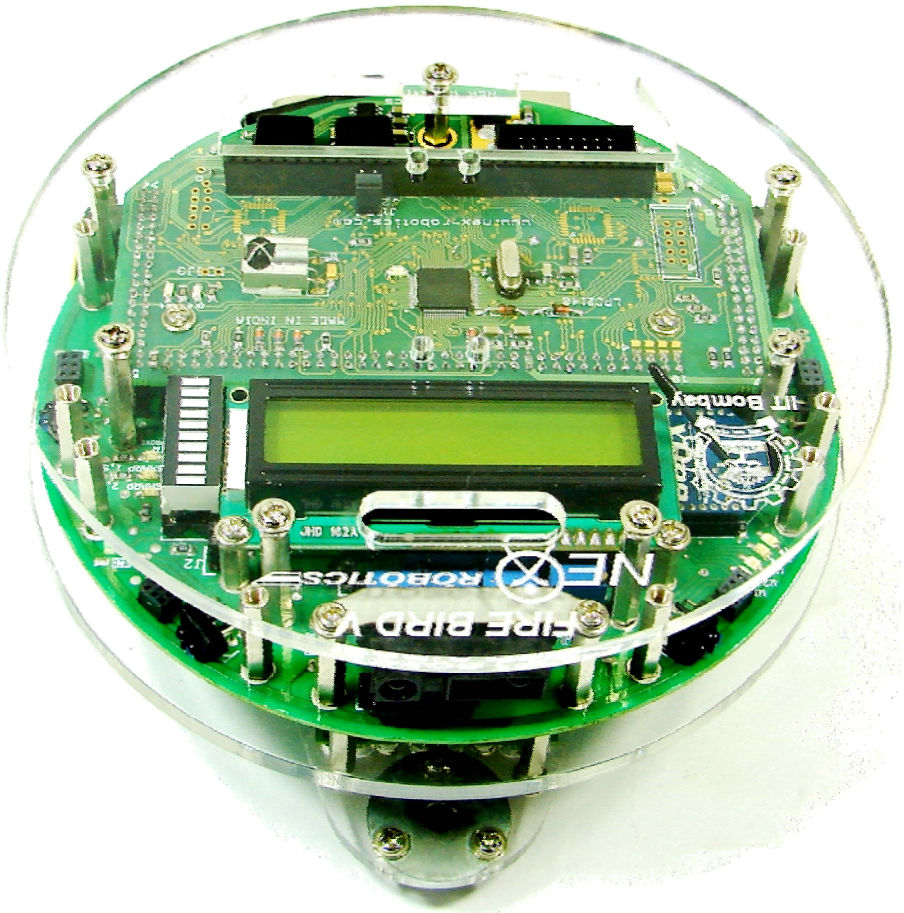

Figure 1.3: Fire Bird V ARM7 LPC218 robot

1.1 Fire Bird V Block Diagram:

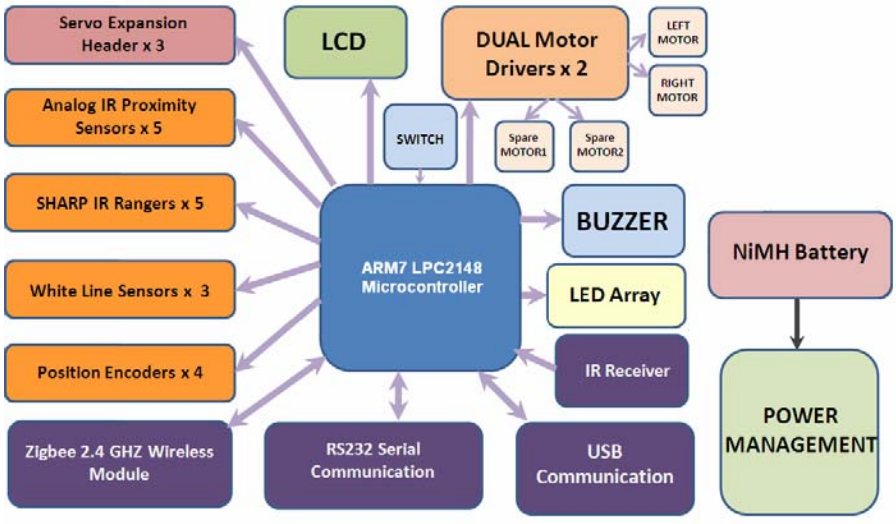

Figure 1.4: Fire Bird V ARM7 LPC2148 robot block diagram

1.2 Fire Bird V ARM7 LPC2148 technical specification

Microcontroller:
NXP's ARM7 LPC2148 as Master microcontroller (ARM architecture based Microcontroller)

Sensors:
Three white line sensors
Three Sharp GP2D12 IR range sensor (One in default configuration)
Two analog IR proximity sensors
Two analog directional light intensity sensors
Two position encoders
Battery voltage sensing
Current Sensing (Optional)

Indicators:
2 x 16 Characters LCD
Indicator LEDs
Buzzer

Control:
Autonomous Control
PC as Master and Robot as Slave in wired or wireless mode

Communication:
Wireless ZigBee Communication (2.4GHZ) (if ZigBee wireless module is installed)
Wired RS232 (serial) communication
Simplex infrared communication (From infrared remote to robot)

Dimensions:
Diameter: 16cm
Height: 10cm
Weight: 1300gms

Power:
9.6V, 2100mAh Nickel Metal Hydride (NiMh) battery pack and external Auxiliary power using battery charger.

Battery Life:
2 Hours while motors are operational at 75% of time

Locomotion:
Two DC geared motors in differential drive configuration and caster wheel at front as support

- Top Speed: 24 cm / second
- Wheel Diameter: 51mm
- Position encoder: 30 pulses per revolution
- Position encoder resolution: 5.44 mm

2. Programming the Fire Bird V ARM7 LPC2148 Robot

The LPC2148 microcontroller is supported by various commercially available IDEs for compiling and debugging of the code. Keil being one of them is the widely used IDE for LPC family of microcontrollers. The µVision4 IDE is Windows-based software development platforms that combines a robust editor, project manager, and make facility. µVision4 integrates all tools including the C compiler, macro assembler, linker/locator, and HEX file generator. The evaluation version of Keil µVision4 IDE is used for demonstrating the sample codes that are included in the CD. The evaluation version has the code size limit of 32KB above which it is restricted to compile the code. The open source community has been doing a lot in the development of open source tools for ARM architecture based microcontrollers. The open source tools are available at zero cost and are being improved with time. Eclipse being one of them and is most commonly used IDE due to its unique features like auto complete, project tree, etc. It requires GCC tool chain for code compilation.

The LPC series of microcontrollers are preloaded with the boot loader firmware which allows self programming of microcontrollers using serial port. Flash magic is a utility which provides an interface for reading, writing and verifying the flash memory of the microcontroller.

2.1 Installing Keil µVision4 IDE

To install **Keil µVision4 IDE**, Go to "Software" folder in the documentation CD and locate "**mdk412.exe"** file. Click on "**mdk412.exe**" to start the installation process. Once the installation process is started Keil µVision welcome screen will appear.

Please read the instructions on the welcome window and click **Next>>** to start the installation.

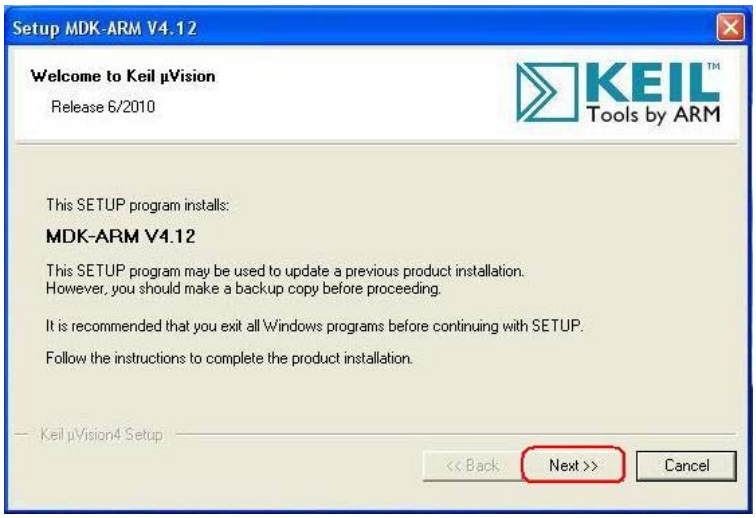

Figure 2.1

Please read the license agreement carefully. If it is acceptable click the check box and click **Next>>** to continue.

Figure 2.2

Select the destination folder where setup will install files. It is always recommended to select the default location. To create backup of old installation select the backup option and click **Next>>** to continue.

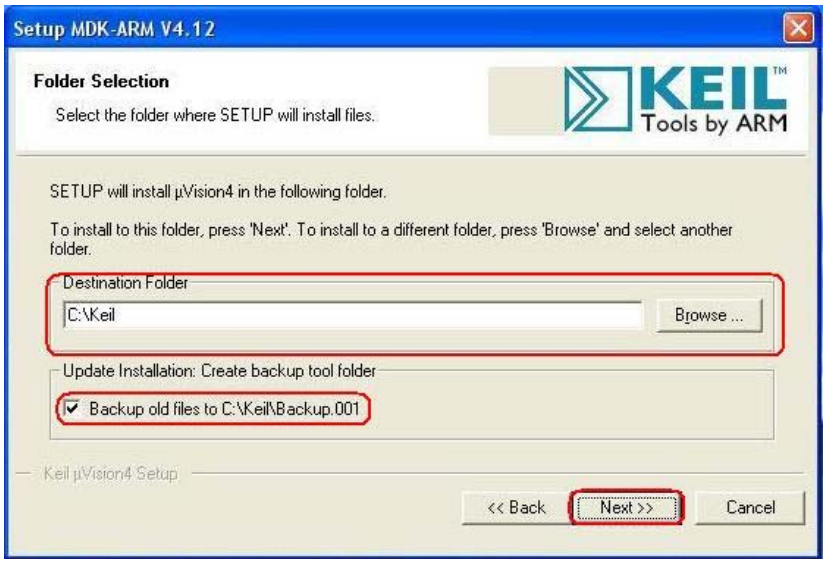

Figure 2.3

In the next window enter your information and click **Next>>** to continue.

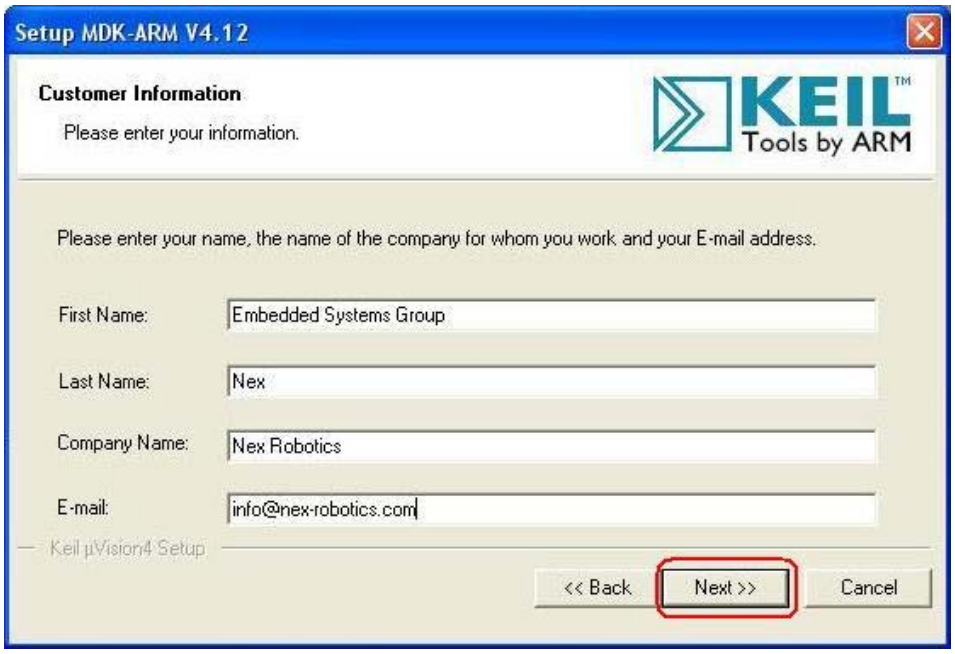

Figure 2.4

On clicking next the file copying process will begin. Wait till setup is complete.

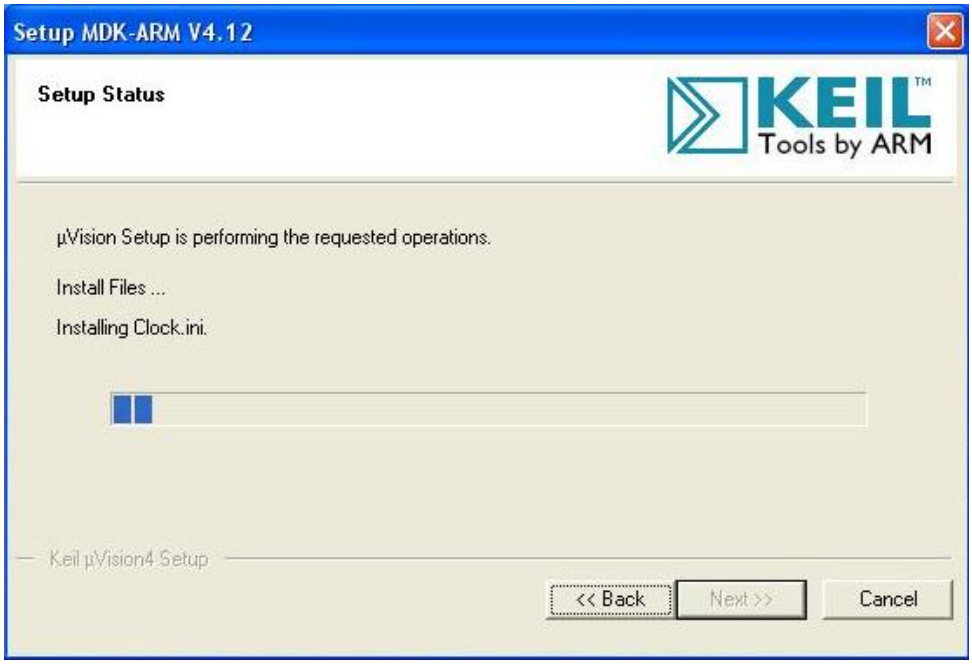

Figure 2.5

Click **Finish** to complete installation process.

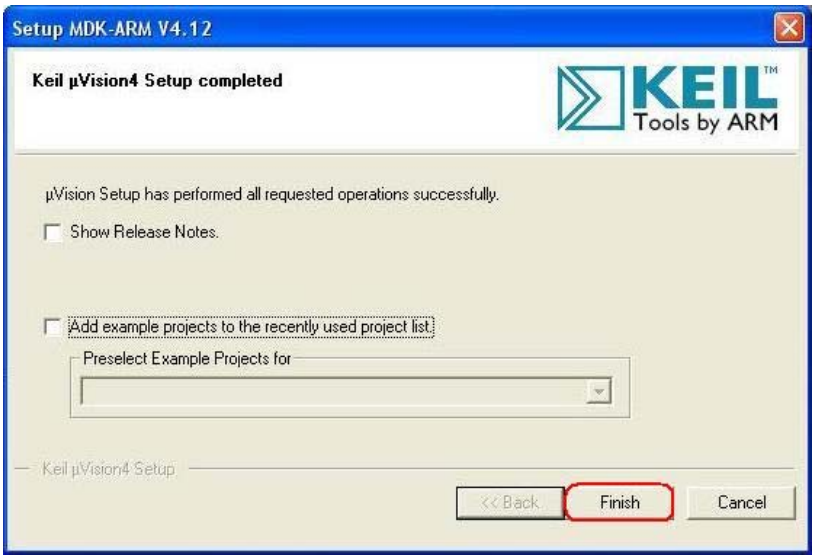

Figure 2.6

2.2 Installing Flash Magic Tool

To install **Flash Magic**, Go to "Software" folder in the documentation CD and locate "**FlashMagic.exe"** file. Click on "**FlashMagic.exe**" to start the installation process. Once the installation process is started Flash Magic welcome screen will appear.

Figure 2.7

Please read the license agreement carefully. If it is acceptable click the radio button and click **Next>>** to continue.

Figure 2.8

Select the destination folder and click **Next>** to continue.

Figure 2.9

In the next window choose the appropriate folder and click **Next>** to continue.

Figure 2.10

Select the desired option and click **Next>** to continue.

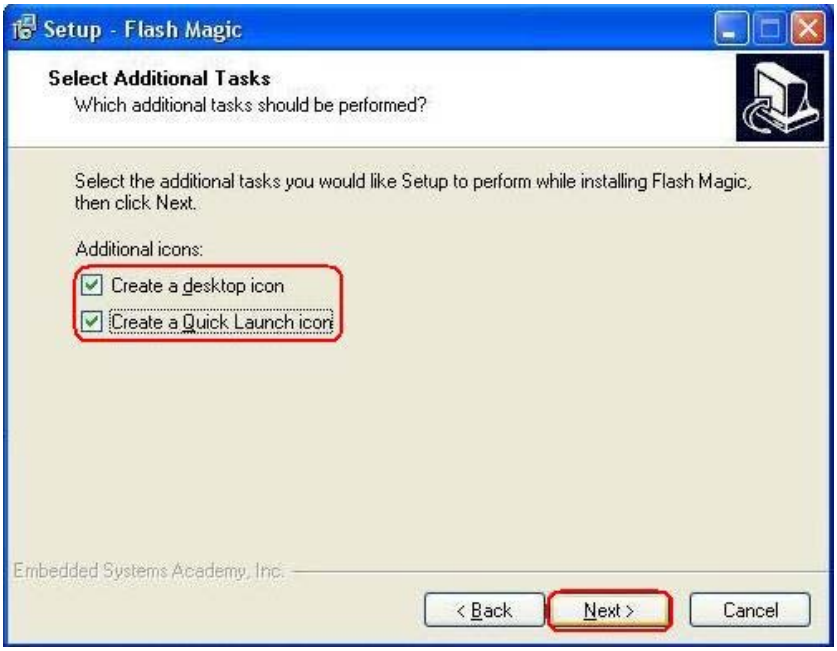

Figure 2.11

Setup is now ready to begin installing flash magic. Click **Install** to continue.

Figure 2.12

Wait till setup installs Flash Magic on your computer.

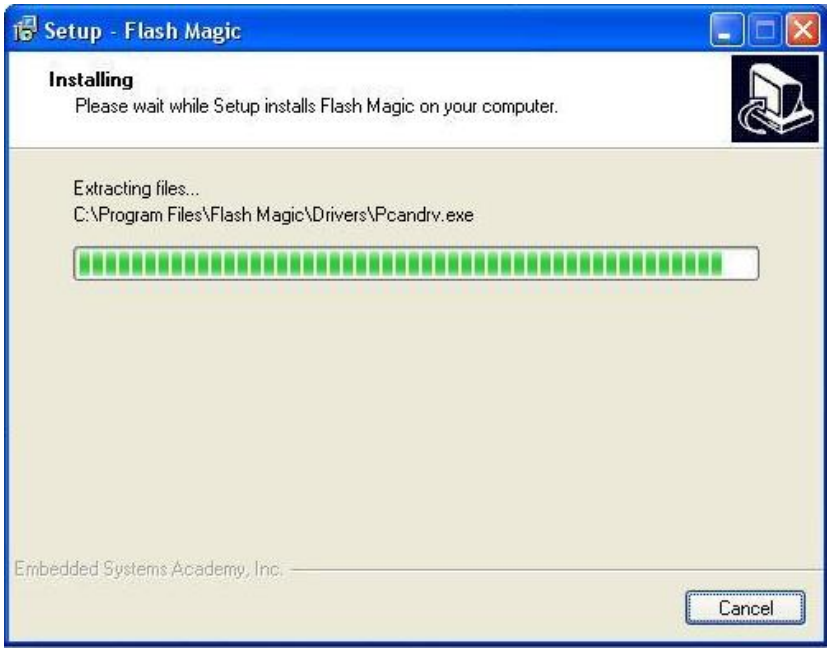

Figure 2.13

Click Finish to complete the installation process

Figure 2.14

2.3 Overview of Keil µVision4 IDE

To start Keil IDE click Start>Programs>Keil µVision4. The initial screen will appear followed by the main window.

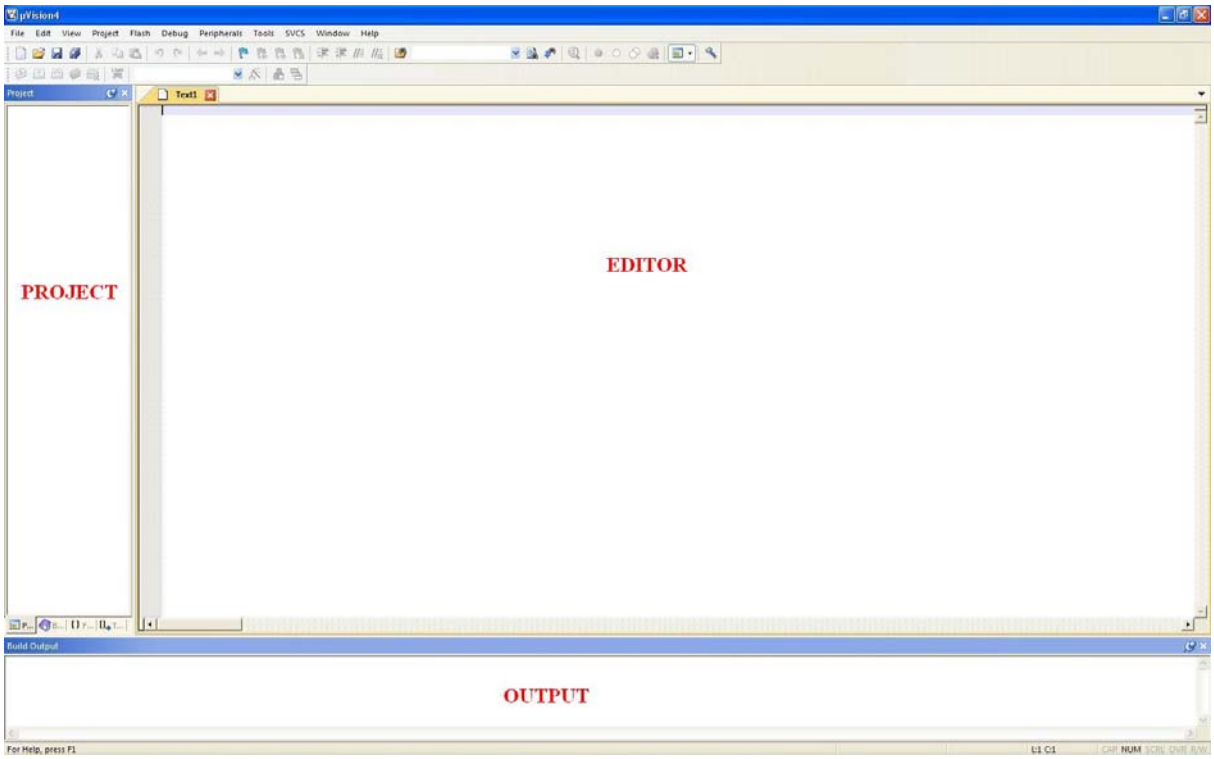

Figure 2.15

The Keil IDE main window in basic configuration is mainly divided into three areas.

Editor - It is the area where .c and .h files of the project are edited.
Project Explorer- It shows the project tree.
Output Window- This window shows the messages related to compiling, project building and debugging.

2.3.1 Setting up Project in Keil uVision

1. To create a new project, Select **Project>New uVision Project** from the main menu.

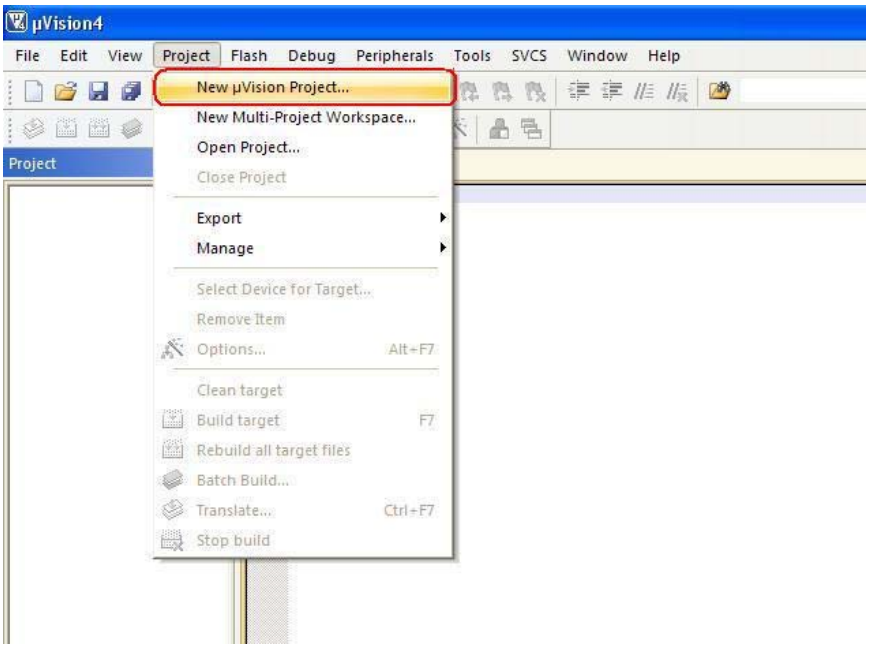

Figure 2.16

2. Create a new directory and name it as **First_Project**. Click open to enter in to this directory.

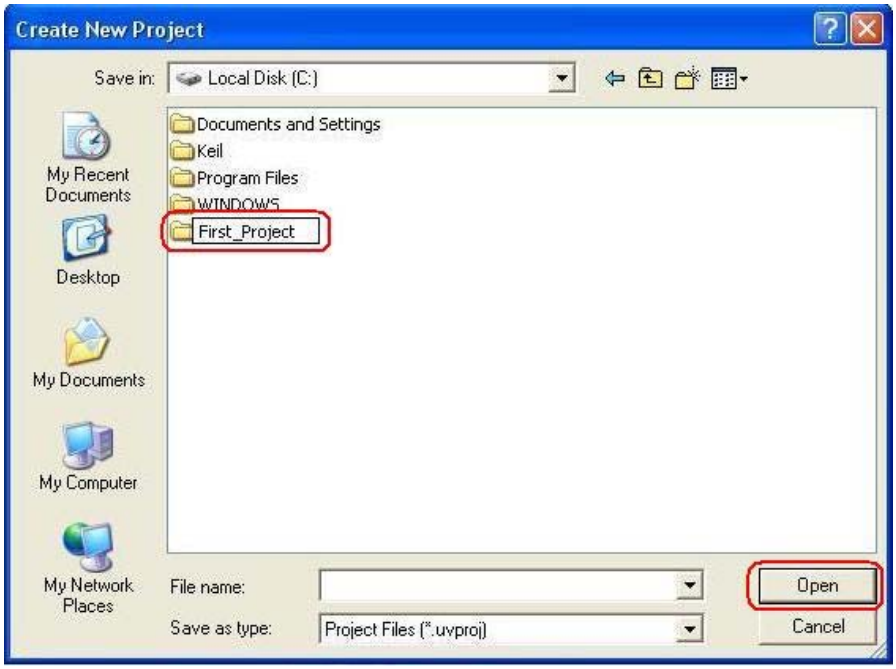

Figure 2.17

3. Inside this directory create a new project and name it as **First_Project** and click **Save** to continue.

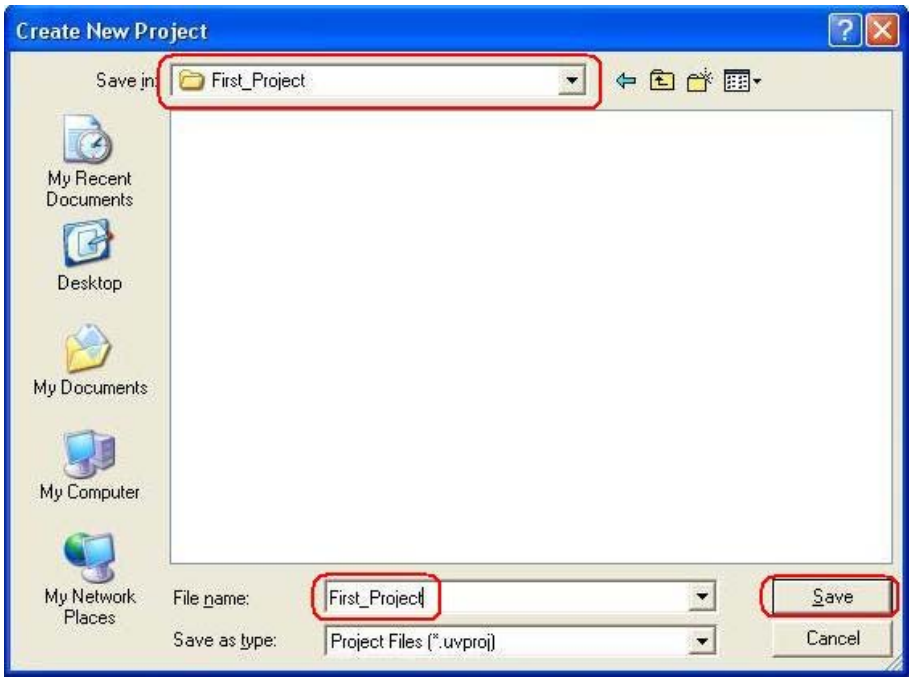

Figure 2.18

4. In the next window locate **NXP (founded by Philips)** tree and expand it.

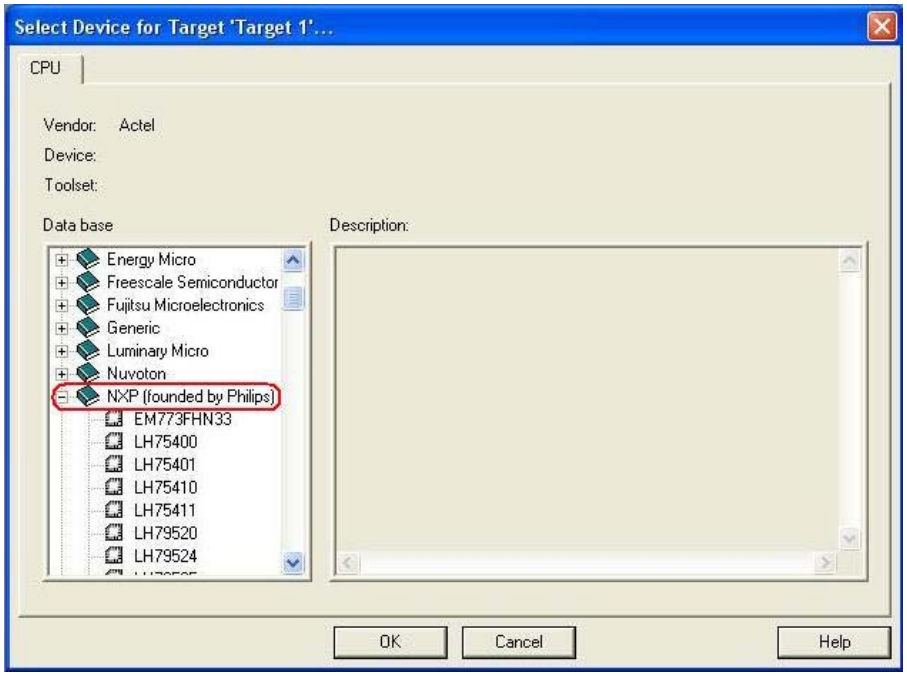

Figure 2.19

5. Now select target device as **LPC2148** and click **OK** to continue.

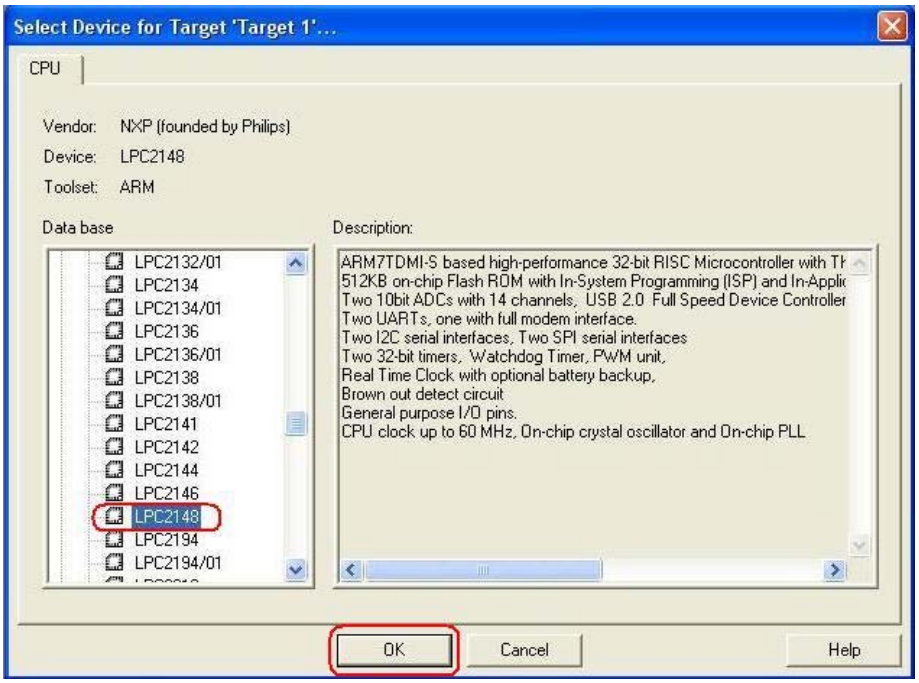

Figure 2.20

6. Click **Yes** to copy **Startup.s** file to project folder. This file configures stack, PLL and maps memory as per the configurations in the wizard. It is discussed in the later sections.

Figure2.21

7. Observe the project explorer area in the main window.

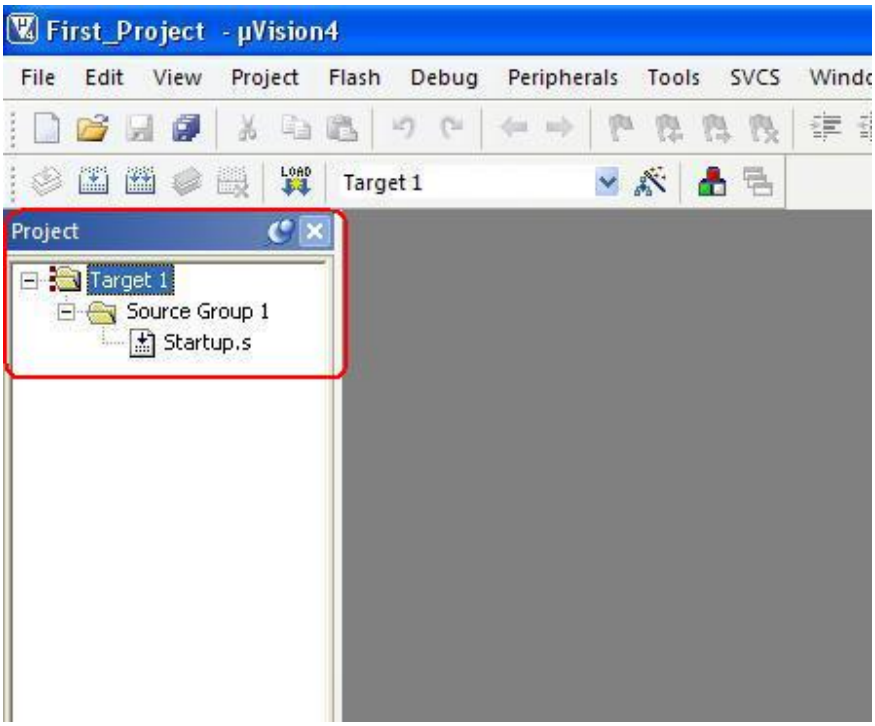

Figure 2.22

8. Now click **Project>Manage>Components, Environments, Books** from the main menu to ensure compiler settings.

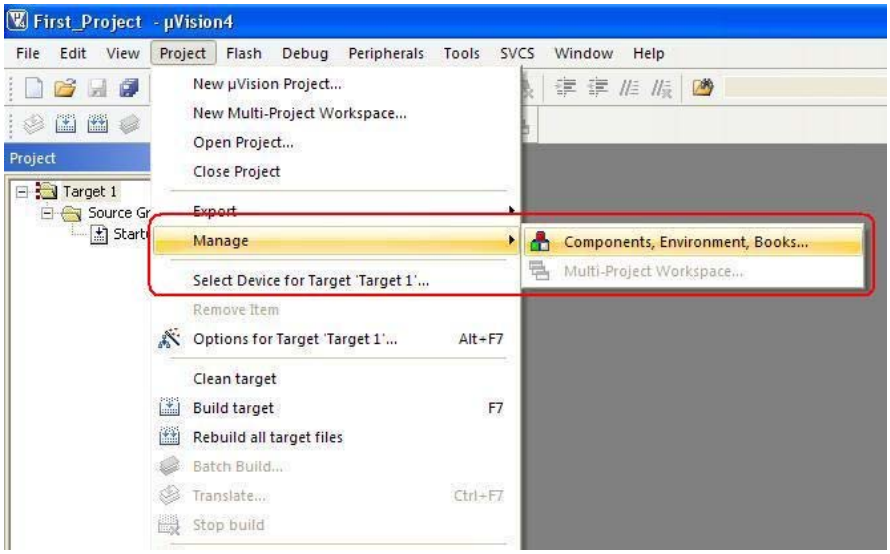

Figure 2.23

9. In the **Folders/Extensions** tab ensure the compiler settings are as shown in the Figure below. If you have installed keil software at a different location then change **Tool Base Folder** location. Click **OK** to continue.

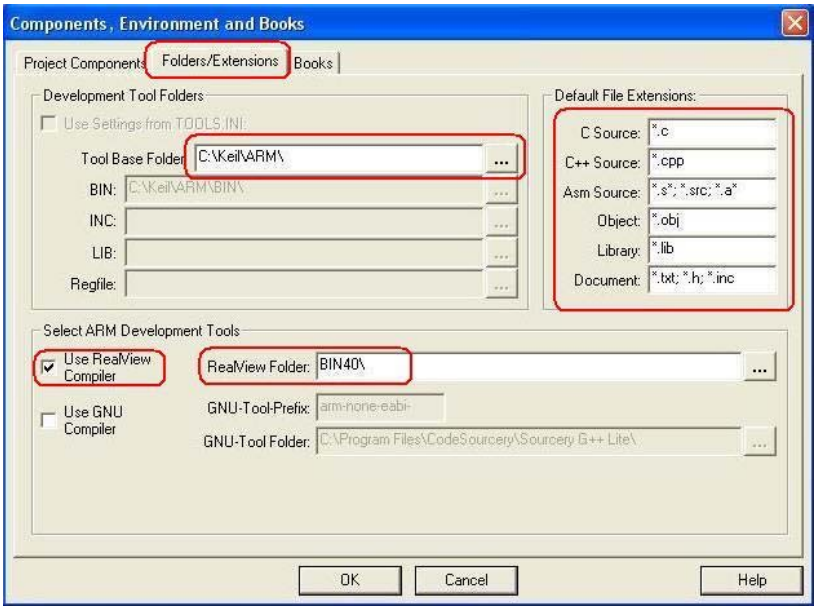

Figure 2.24

10. Now click **File>New** to create a new file.

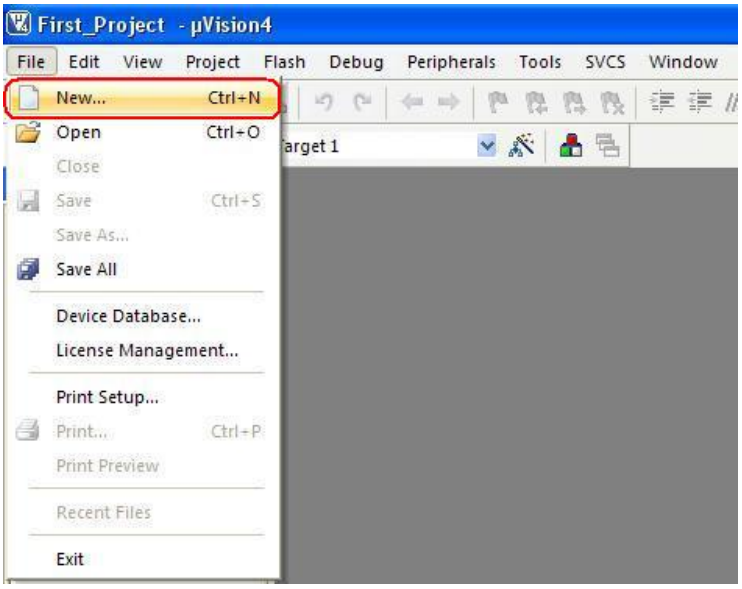

Figure 2.25

11. Save the new file in to the same folder that was created earlier and name it as **main.c** and click **Save** to continue.

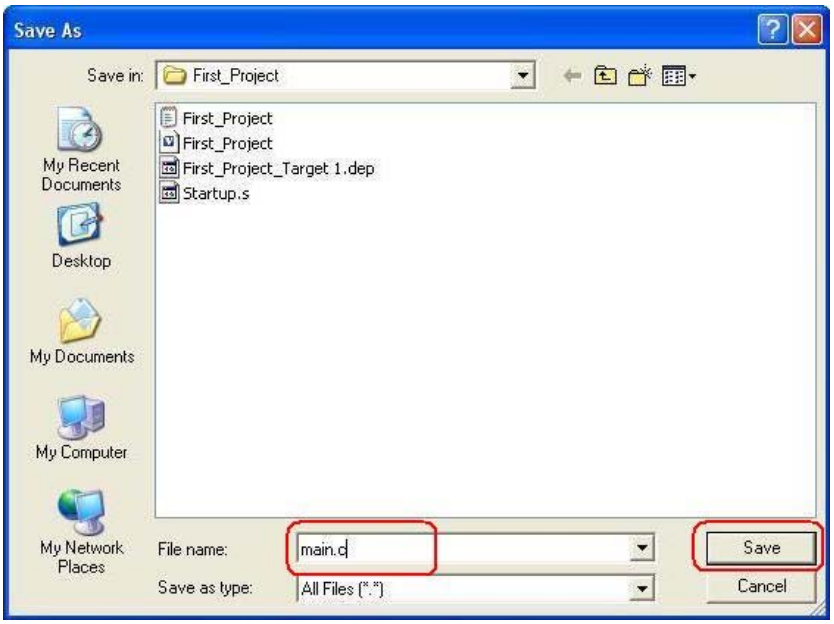

Figure 2.26

12. Now add "**main.c**" to the source group by right clicking on the **Source Group 1** from the project explorer and select the highlighted option as shown in the Figure below.

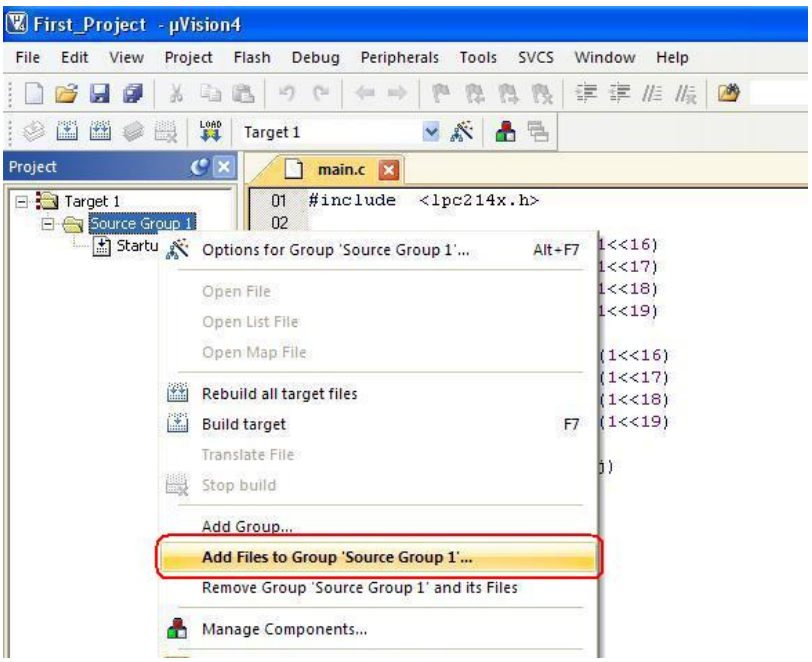

Figure 2.27

13. Select "**main.c**" file to be added and click **ADD** to continue.

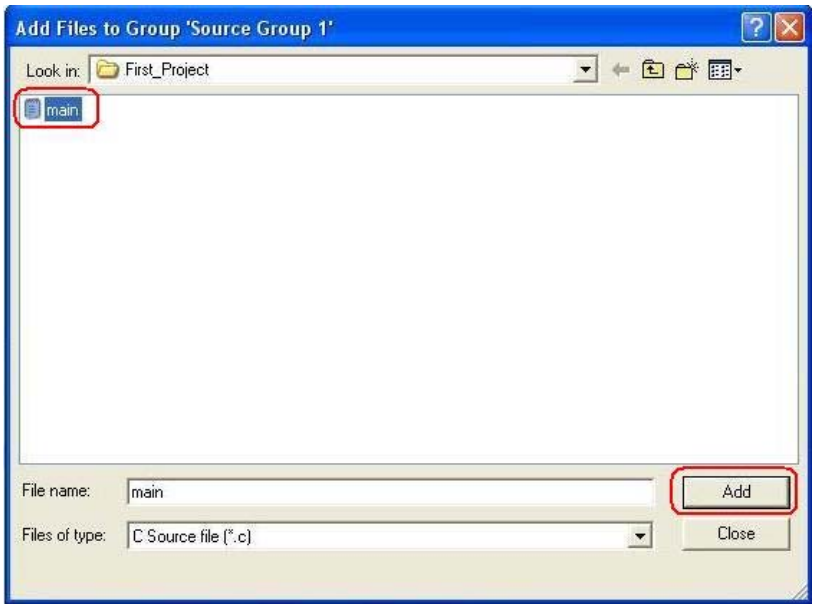

Figure 2.28

14. Observe that **"main.c"** file is added to the source group in the project explorer window.

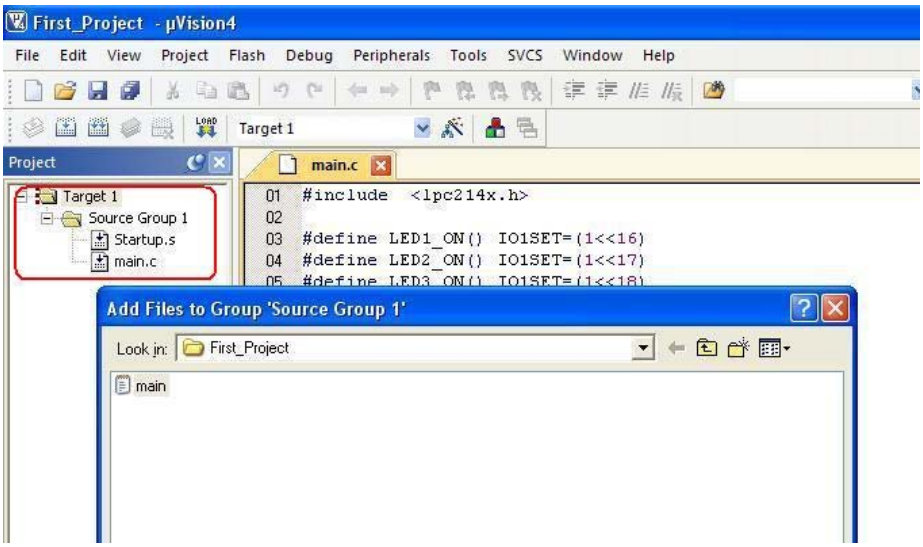

Figure 2.29

15. Right click **Target1** in the project explorer window and select the highlighted option as shown in the Figure below.

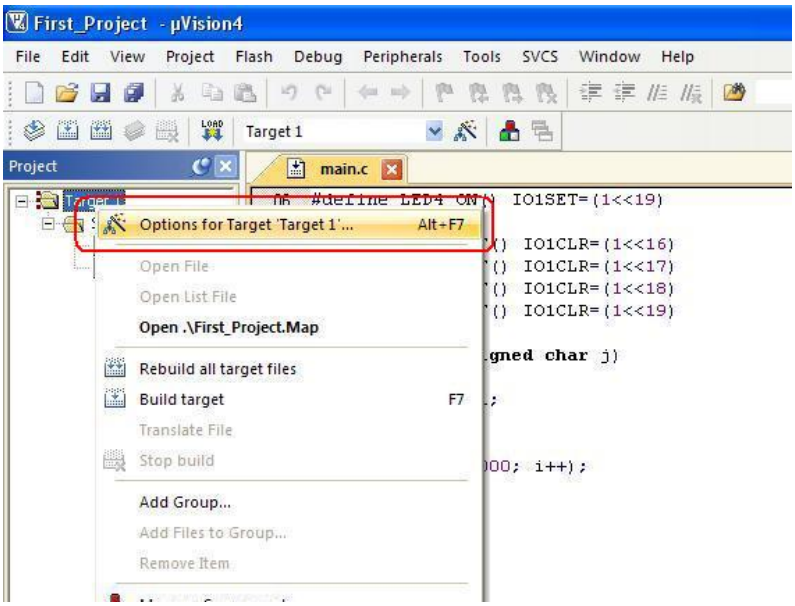

Figure 2.30

16. In the appearing window select **Target** tab and set Xtal. frequency as 12MHz.

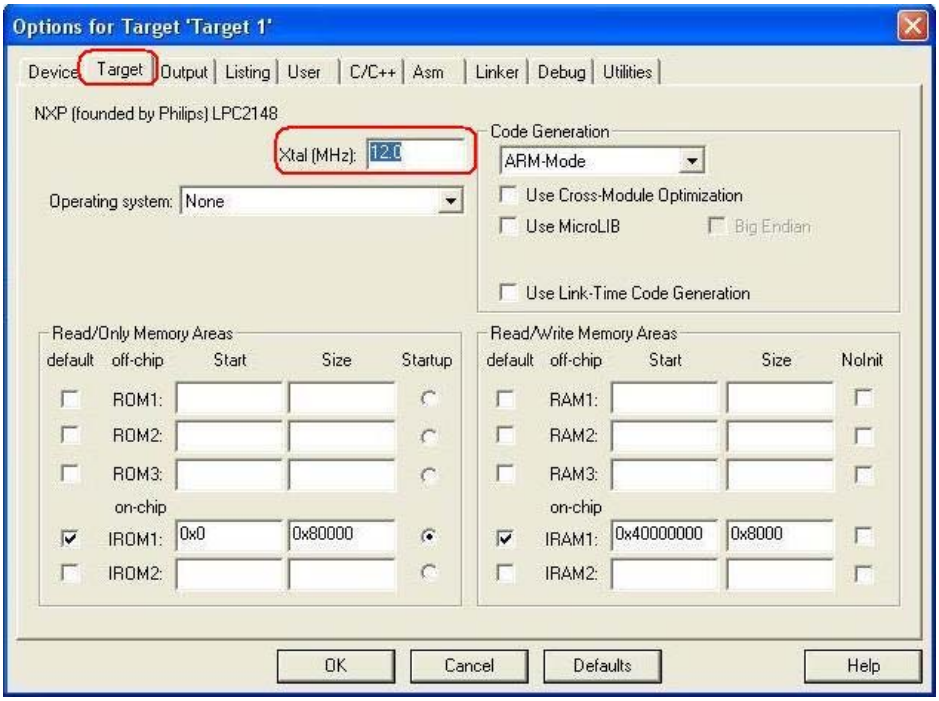

Figure 2.31

17. In the **Output** tab ensure that **Create HEX File** option is selected.

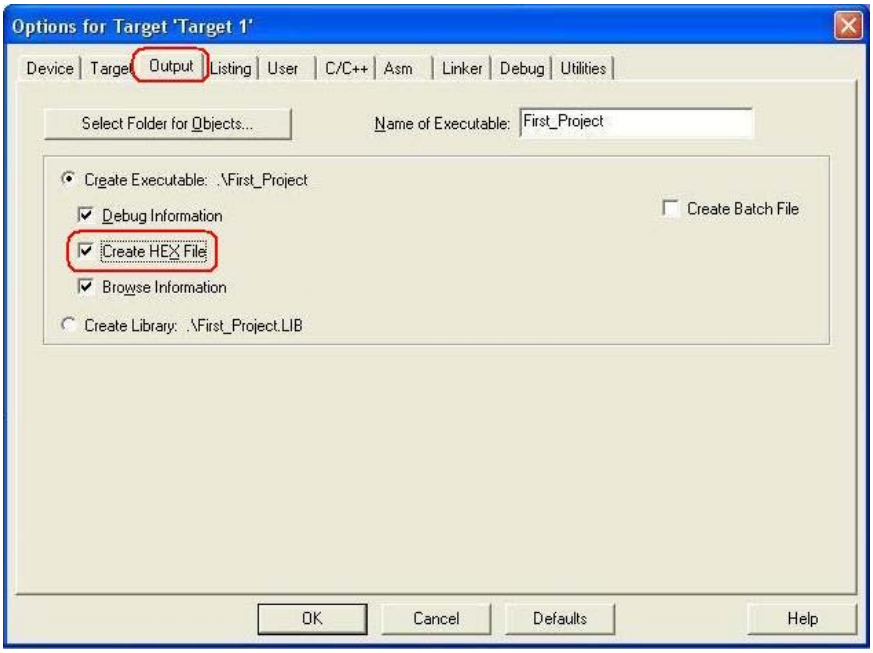

Figure 2.32

18. In the **Linker** tab ensure that the highlighted option is selected and click **OK** to continue.

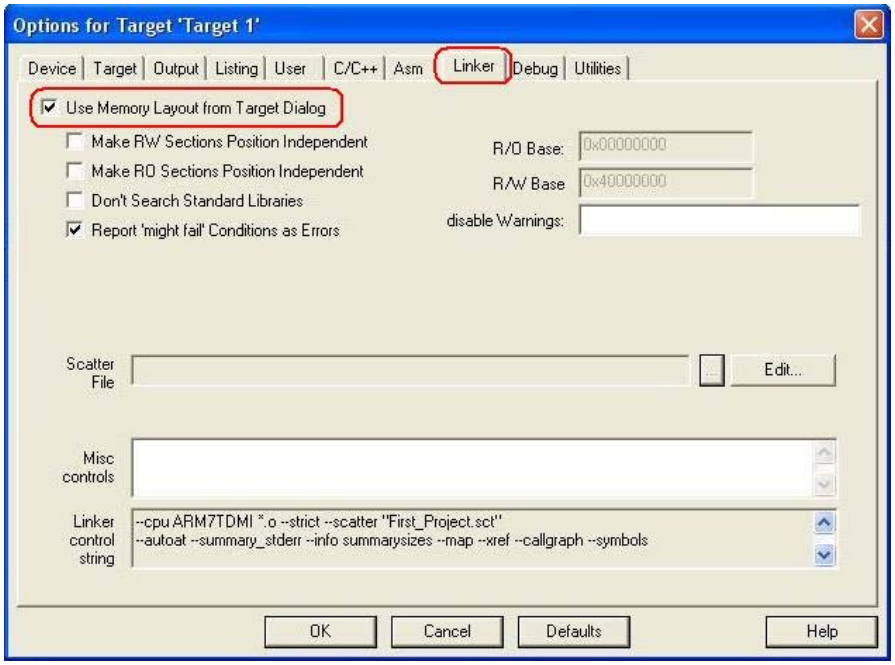

Figure 2.33

2.4 Writing your first code in Keil

Now since the project is almost setup we can start writing code in the "**main.c**" file that was created earlier. For demonstration purpose you can copy the following code and paste it in the "**main.c**" file. This code is written to blink bargraph LEDs on the robot in the sequential order starting from top to bottom.

```
#include <lpc214x.h>


/**********Macros*****************/

#define BUZZER_OFF() IO0CLR=(1<<25)          //Macro to turn OFF buzzer
#define BUZZER_ON() IO0SET=(1<<25)           //Macro to turn ON buzzer

/*********************************/


/****Function Prototypes**********/

void Buzzer_Delay(void);
void Init_Buzzer_Pin(void);
void Init_Peripherals(void);
void Init_Ports(void);

/*********************************/


/*********************************************************

      Function          : Buzzer_Delay
      Return type       : None
      Parameters        : None
      Description       : Provides small amount of delay between
                          buzzer toggles.
*********************************************************/
void Buzzer_Delay(void)
{
 unsigned int  i,j;
 for(j=0;j<20;j++)
 {
  for(i=0; i<60000; i++);
 }
}


/*********************************************************

      Function          : Init_Buzzer_Pin
      Return type       : None
      Parameters        : None
      Description       : Initialises Buzzer pin
*********************************************************/
```

```
void Init_Buzzer_Pin(void)
{
 PINSEL1&=0xFFF3FFFF;
 PINSEL1|=0x00000000;         //Set P0.25 as GPIO
 IO0DIR&=0xFDFFFFFF;
 IO0DIR|= (1<<25);            //Set P0.25 as Output
 BUZZER_OFF();                      //Initially turn OFF buzzer
}


void Init_Ports(void)
{
 Init_Buzzer_Pin();
}

void Init_Peripherals(void)
{
 Init_Ports();
}

int main(void)
{
 PINSEL0 = 0x00000000;        // Reset all pins as GPIO
 PINSEL1 = 0x00000000;
 PINSEL2 = 0x00000000;
 Init_Peripherals();        // Init Port pins and Peripherals
 while(1)
 {
  BUZZER_ON();                              //Turn ON buzzer
  Buzzer_Delay();                   //Wait
  BUZZER_OFF();                             //Turn OFF Buzzer
  Buzzer_Delay();                   //Wait
 }
}
```

We are now going to compile this code to generate the hex file which we will load on the Robot's microcontroller. To build the project click on **Rebuild** button on the main toolbar.

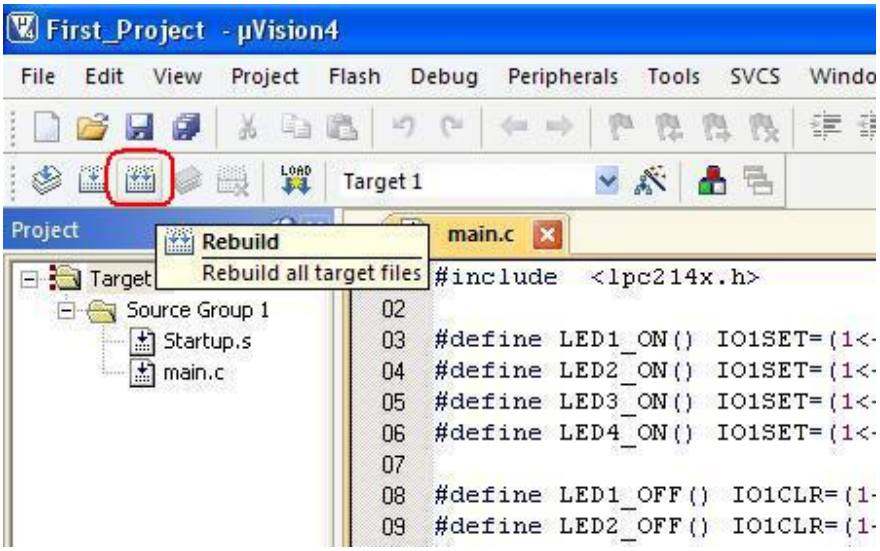

Figure 2.34

You can alternatively build project by clicking on **Project>Build Target** from the main menu.

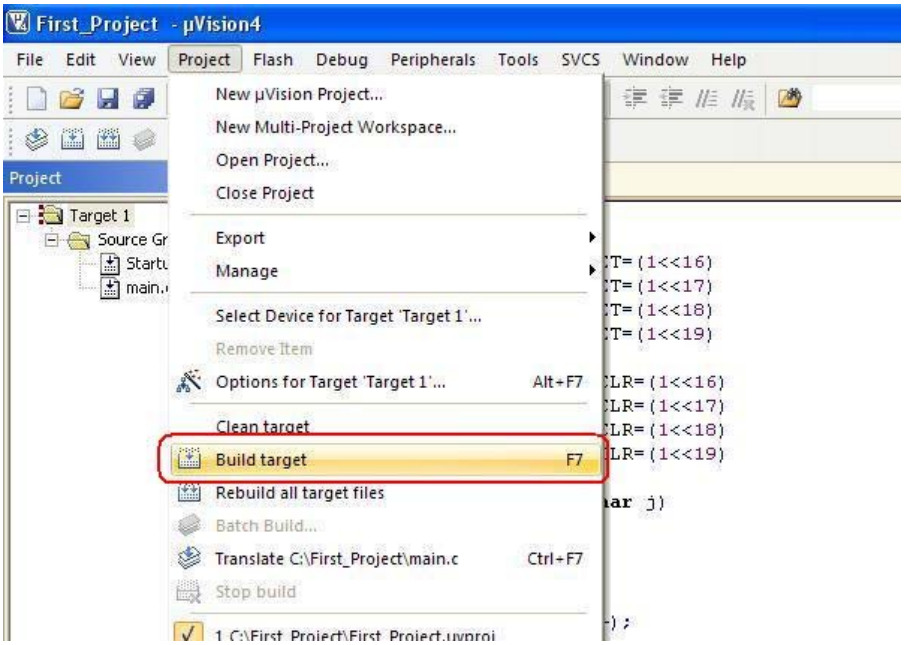

Figure 2.35

You can observe the build process in the output window. If any errors, rectify it by double clicking on it and you will be pointed to the erroneous line.

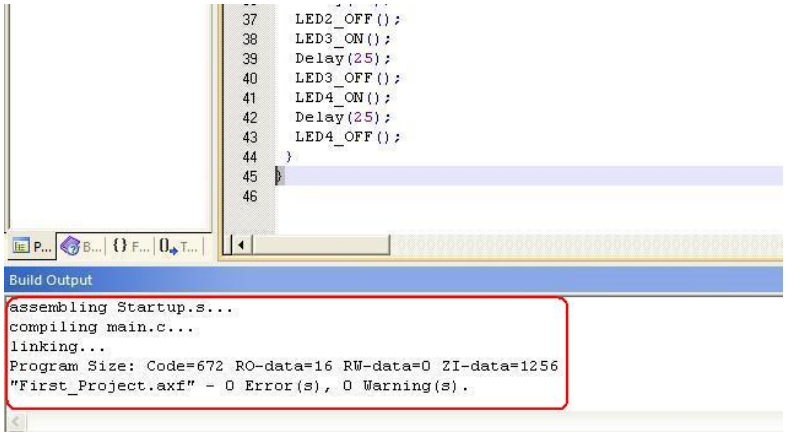

Figure 2.36

You can also verify that “first_project.hex” file is generated in the folder you have selected.

2.5 Debugging the code in uVision in Simulator mode

To use Debugger in simulator mode first we need to setup the debugger so as to use simulator.

1. Right click **Target 1** and select the highlighted option as shown in the figure below.

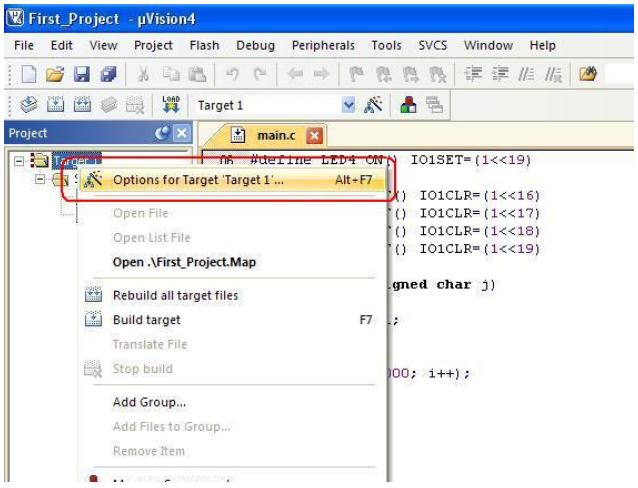

Figure 2.37

2. In the debug tab make sure that the highlighted settings are done. Click **OK** to continue.

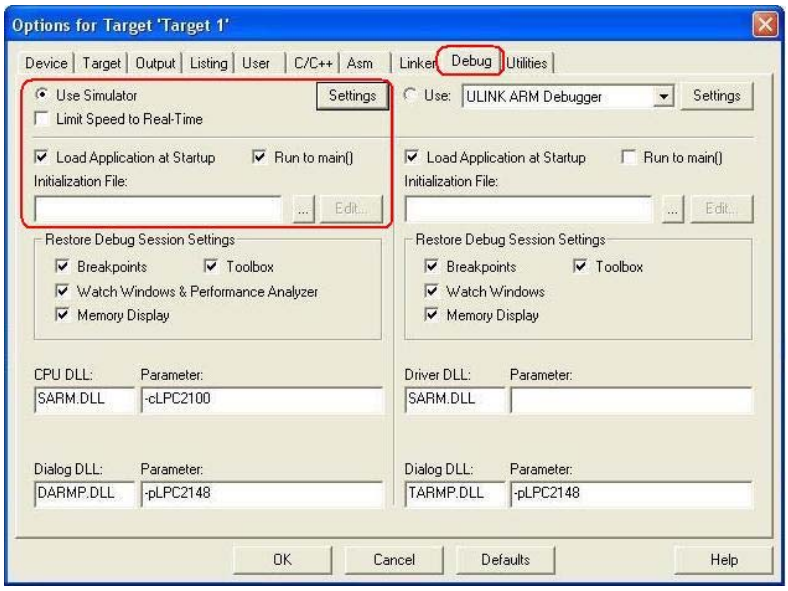

Figure 2.38

3. Start Debug session by clicking on **Debug>Start/Stop Debug Session** from the main menu. Alternatively you can press Ctrl+F5 to toggle between starting and stopping of debug session.

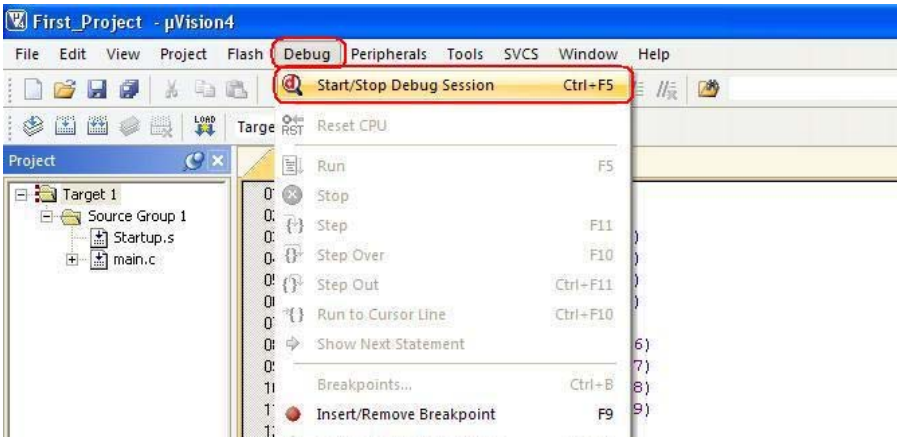

Figure 2.39

4. Click **OK** to continue.

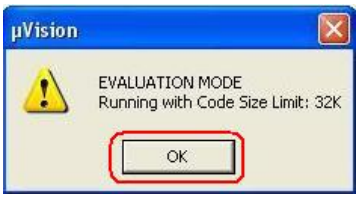

Figure 2.40

5. Go to **Peripherals>System Control Block>Phase Locked Loop 0** to observe the PLL settings on the microcontroller.

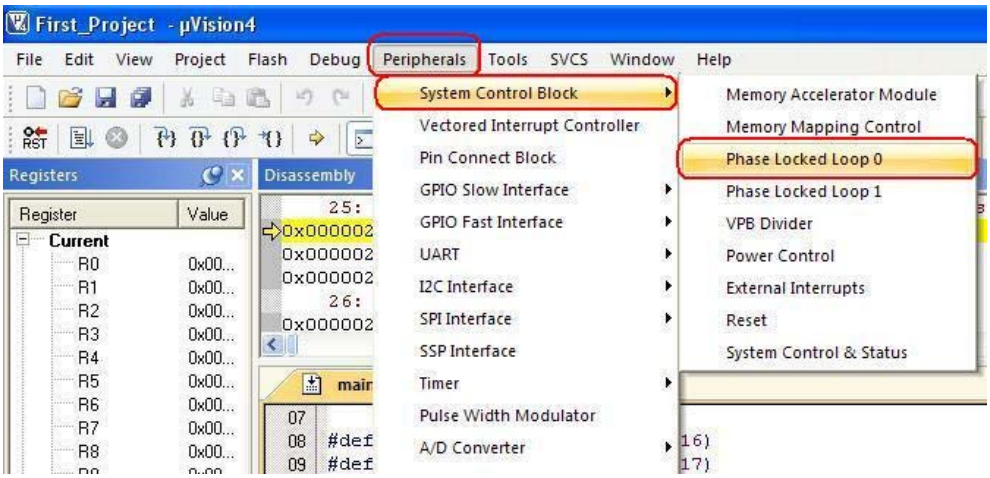

Figure 2.41

6. In the **Crystal Oscillator & Processor Clock** section observe that XTAL is 12MHz and CCLK is 60MHz. This is because the debugger has already executed the start up code when we started the debug session. It also ensures that PLL settings that we had done in the configuration wizard of the startup code are properly working.

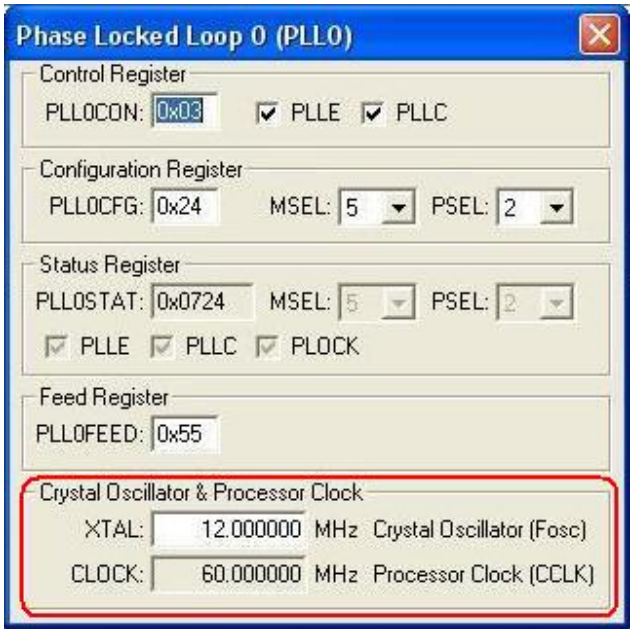

Figure 2.42

7. Now click **Peripherals>Pin Connect Block** to include pin connect block,

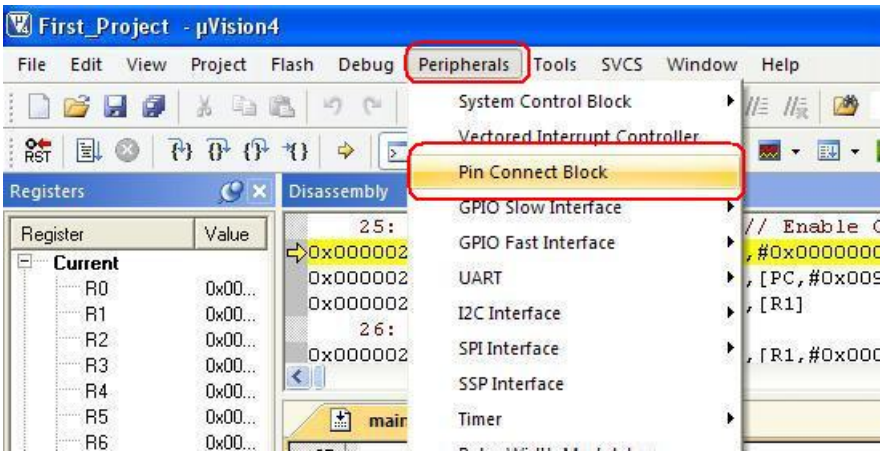

Figure 2.43

8. Similarly select **Port 1 window.**

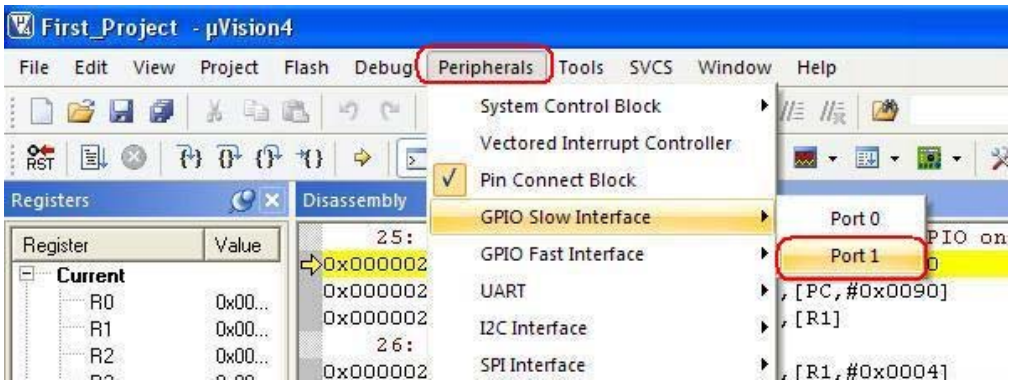

Figure 2.44

9. Arrange the windows as shown in figure below.

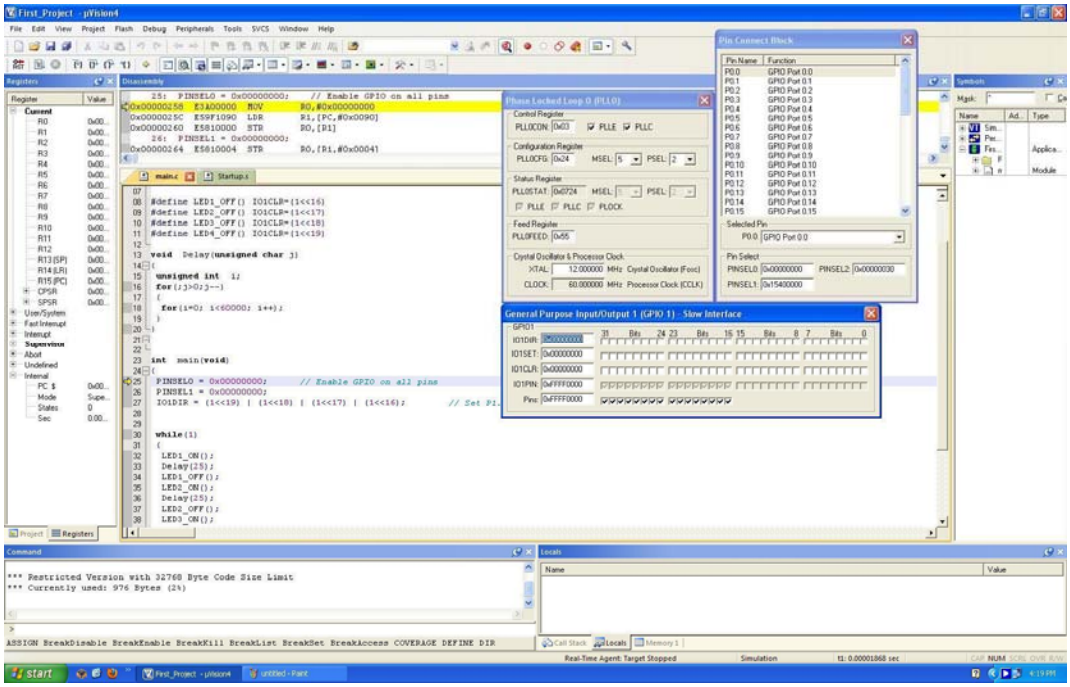

Figure 2.45

10. Now use **Step** function to step through the code. Alternatively you can press Ctrl+F11. Observe the changes in the Pin connect block and Port1 block as you step through the code.

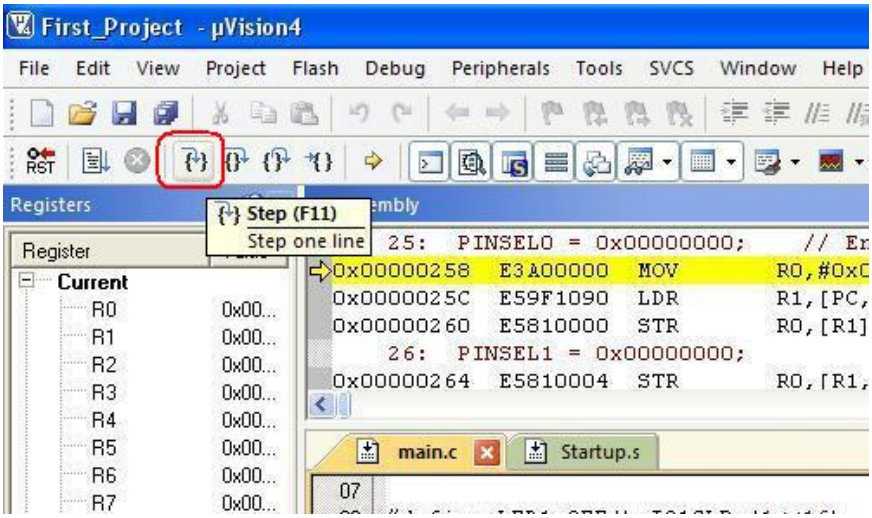

Figure 2.46

11. When you encounter the delay function you can simply use Step Out function. It will execute the delay function and take execution to the next line immediately after the delay function.

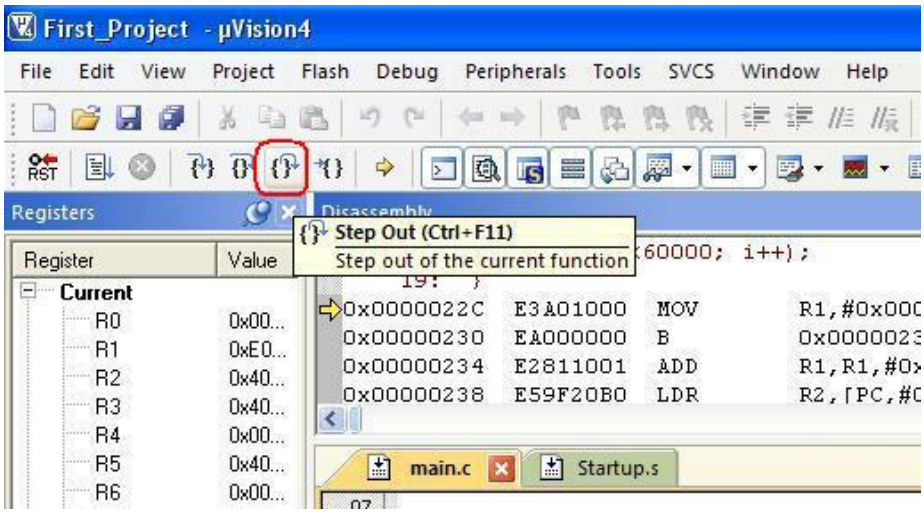

Figure 2.47

12. Now press the Reset button to Reset the CPU and observe the PLL window.

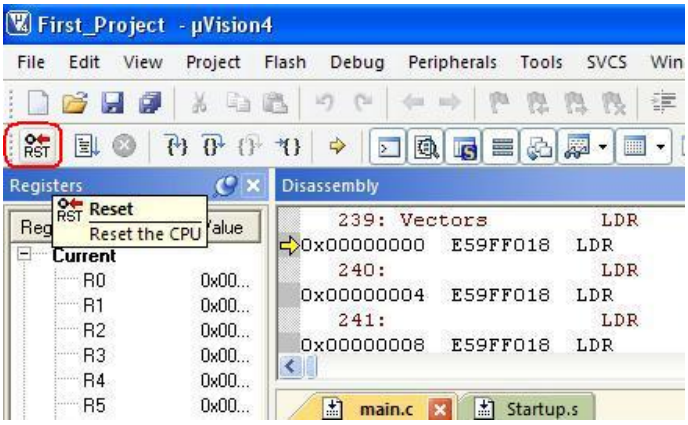

Figure 2.48

13. Observe that CCLK is also reset and now it is equal to XTAL frequency. Similarly the GPIO and pin connect block vales are also reset.

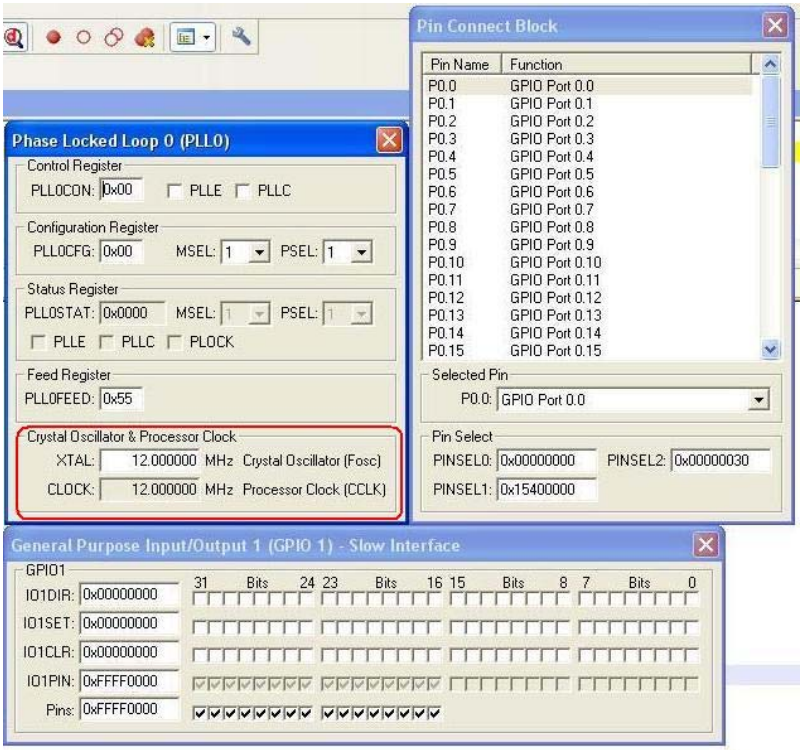

Figure 2.49

14. Now let us setup a break point at line no. 146 i.e. at the beginning of the main function. It will halt the execution when the debugger encounters this line.

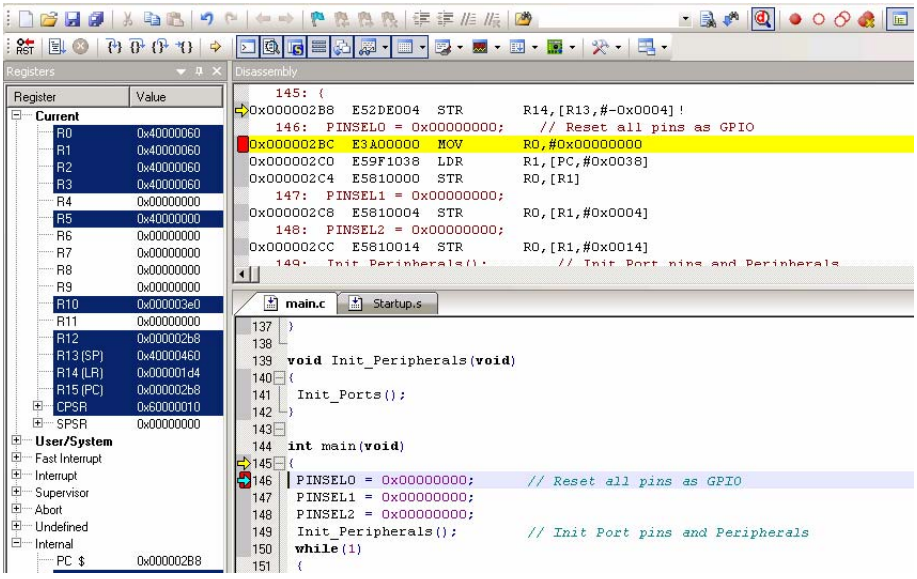

Figure 2.50

15. Now click Run to start the execution.

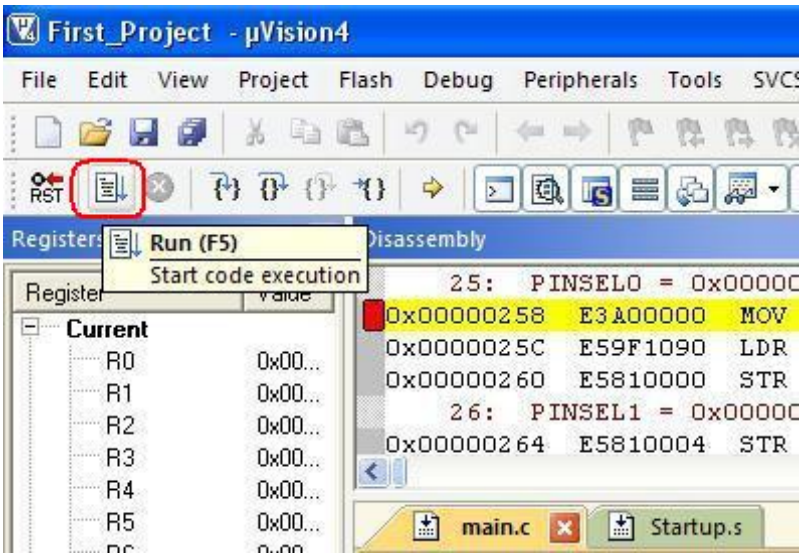

Figure 2.51

16. After clicking on RUN observe the change in the CCLK field. The execution has also halted at the breakpoint that was setup earlier. Now again if you click Run the code will continue to execute.

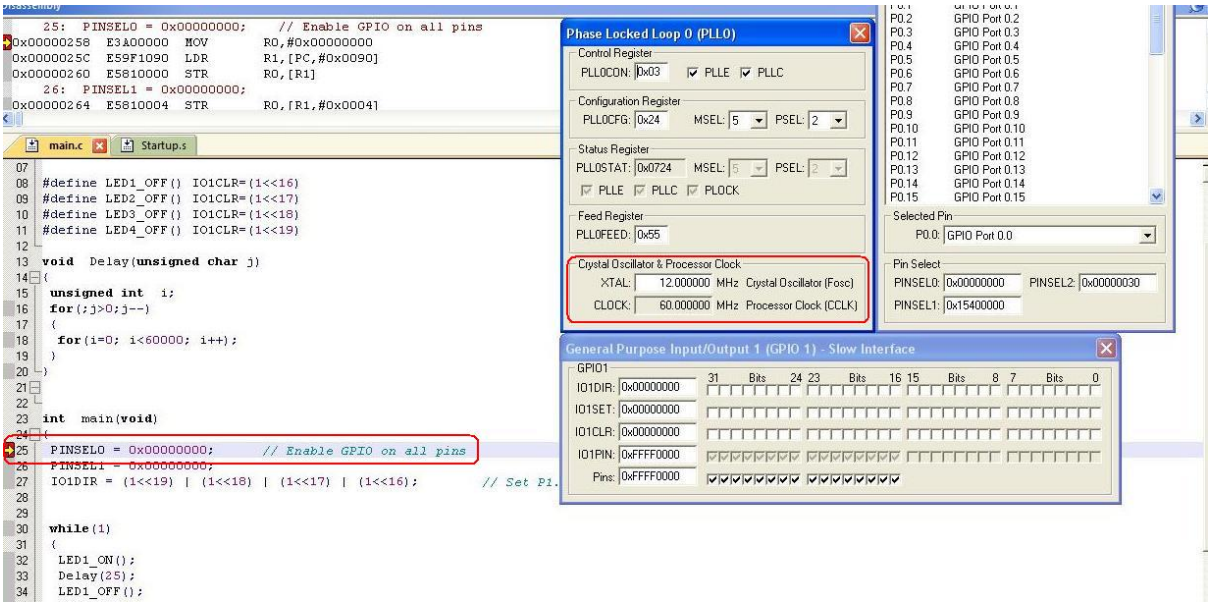

Figure 2.52

After debugging is done select “Stop Debugging” from Debug menu. Alternatively you can press Ctrl+F5 to Start/Stop debugging.

2.6 Loading Hex file on the microcontroller using on-chip boot loader

LPC2148 microcontroller incorporates a capability to self program itself using the on-chip UART boot loader. This eliminates the need of an external programming hardware. The boot loader code is dedicated to use UART0 on the microcontroller. Once the code is loaded on the microcontroller UART0 is free to be used in the application. The Flash Magic software is used on the computer side for loading the hex file. The installation procedure is already explained in section 4.2. Before loading the hex file it is essential to setup Flash Magic for LPC2148 microcontroller. The microcontroller can be entered into the boot loader mode by pressing the Boot switch and then pressing Reset switch. This process is explained in detail in later sections.

1. Click **Start>Programs>Flash Magic>Flash Magic** to start Flash Magic utility.
2. Click **Select Device** to select LPC2148 microcontroller.

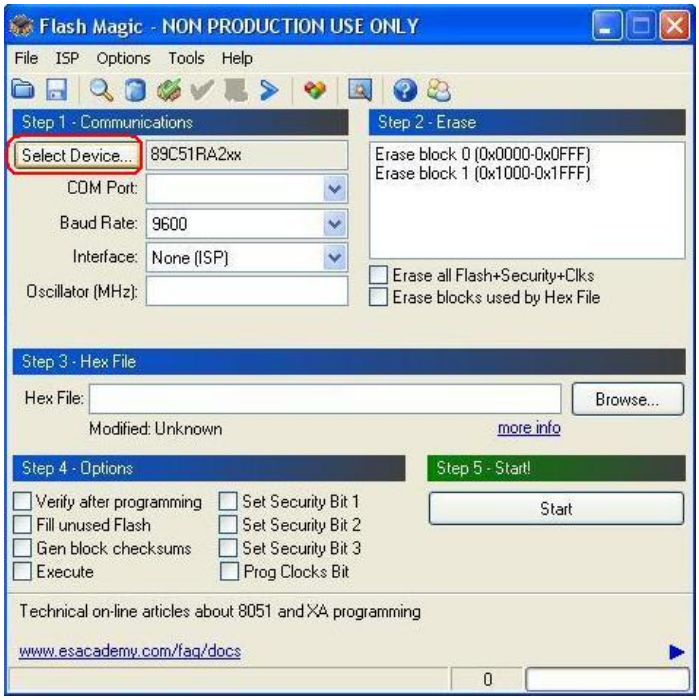

Figure 2.53

3. Expand the ARM7 tree and select **LPC2148** from the displayed list of microcontrollers.

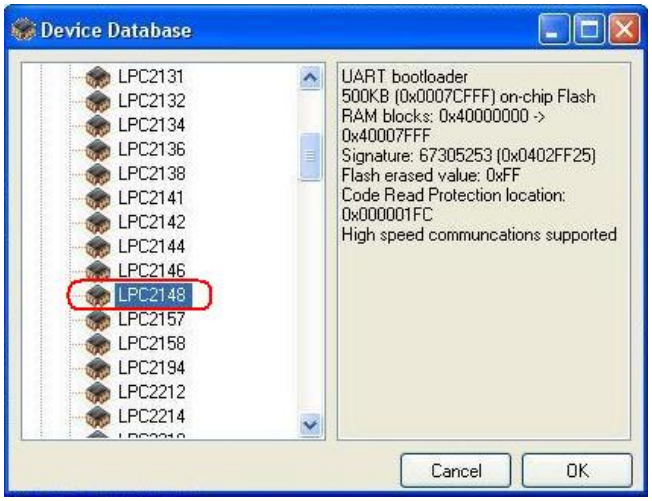

Figure 2.54

4. After selecting the microcontroller, select the COM port that you are going to use and ensure that other settings are as shown in the figure below.

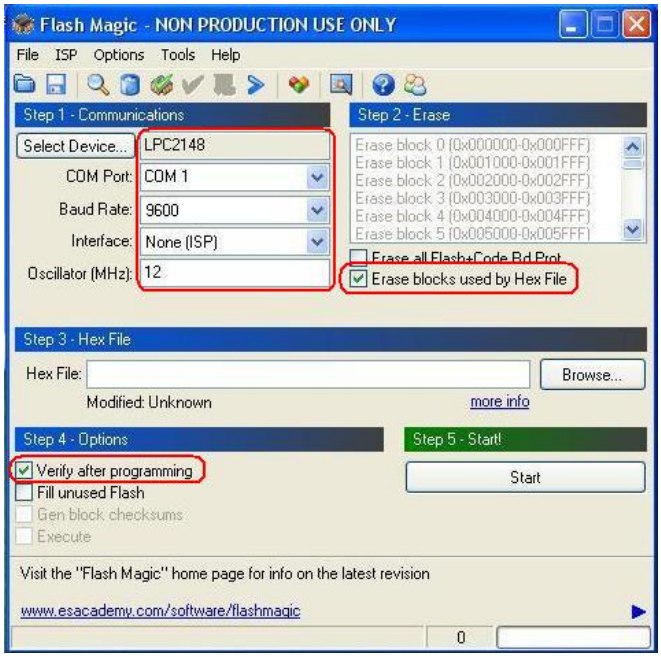

Figure 2.55

5. To select the hex file click **Browse** and point to the folder that was created earlier and select the hex file.

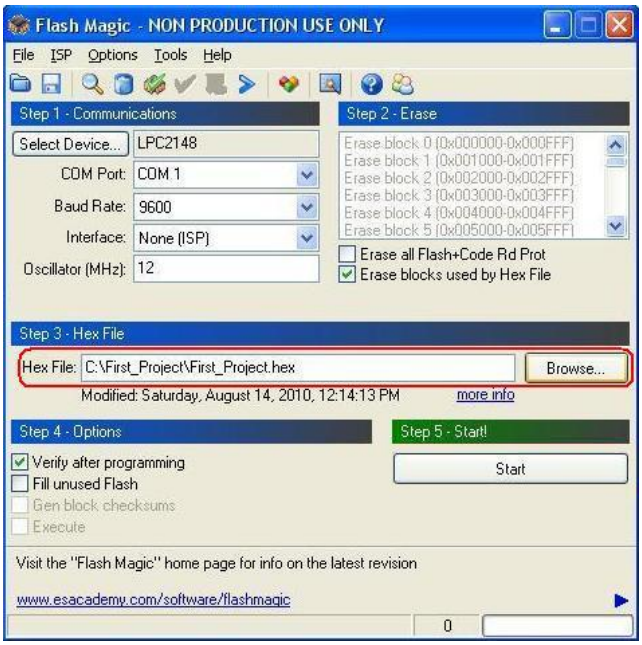

Figure 2.56

6. Connect a DB9 cable between the selected COM port on your PC and robot's serial port and turn ON the robot.

7. Press the Boot switch and while keeping it pressed, press Reset switch once. This will make LPC2148 enter into the boot loader mode

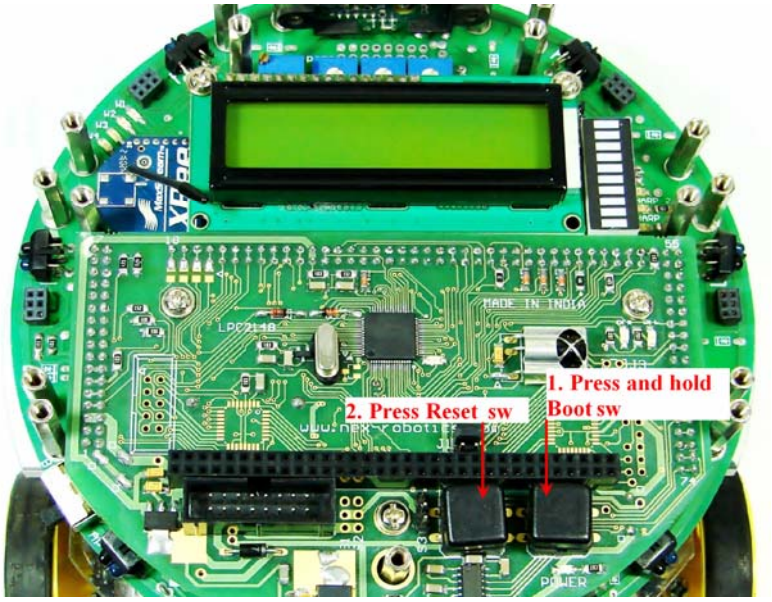

Figure 2.57

8. After ensuring the hardware setup in step 7 click start to begin programming the hex file.

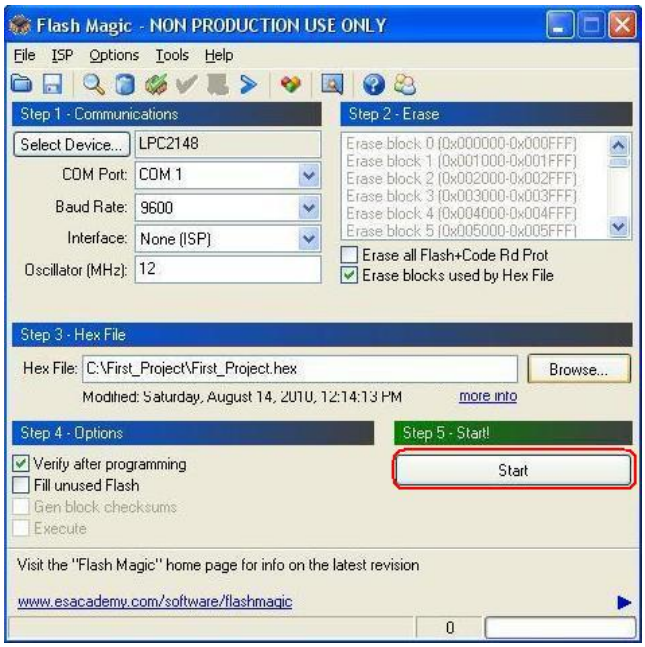

Figure 2.58

9. Observe the status bar at the bottom of the Flash Magic tool. Wait for the "**Finished"** status and press "**Reset"** on the LPC2148 development board and observe the LEDs.

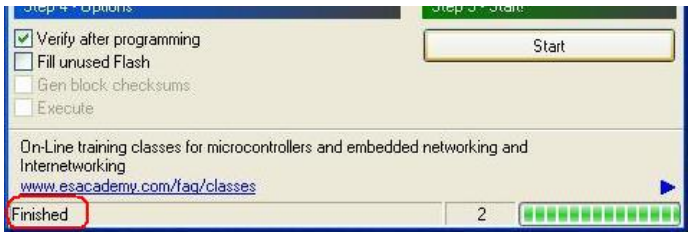

Figure 2.59

If not successful then repeat again from step 7.

2.7 Configuring Startup.s File using Configuration wizard

The "**startup.s**" is the collection of assembly instructions that is supposed to be executed at the start-up of the application code. The startup file contains information about the PLL settings, peripheral clock divider, stack configuration, etc. This file can be found in the Keil installation directory. On creating a new project keil allows you to copy this file to the project source folder so that you can modify the settings as per your application requirements without modifying the original file. To execute the start up code on reset please ensure that following setting is done in the target

options window. To open this window goto Projects-> Options for Target ‘Target1’ from the menu bar.

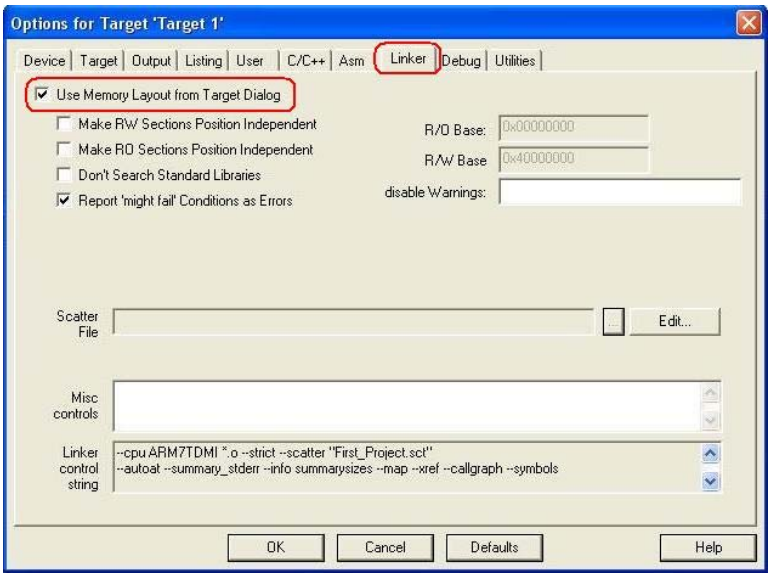

Figure 2.60

1. In the Project explorer double click on **“startup.s”** file.

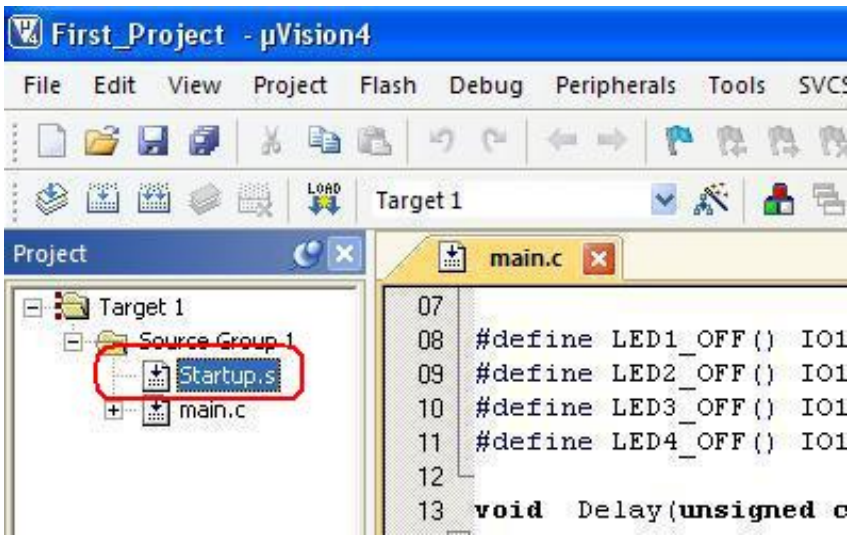

Figure 2.61

2. The startup.s file will open up as assembly file. To open the wizard click **Configuration Wizard** as shown in figure below.

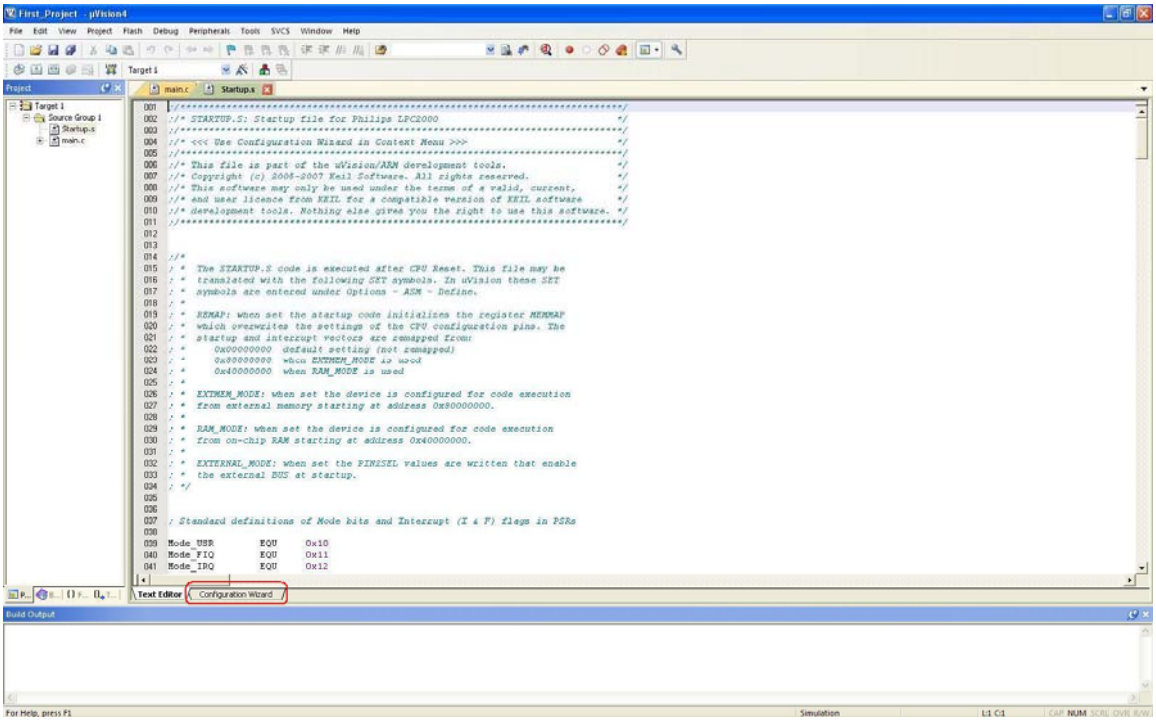

Figure 2.62

3. Expand the PLL Setup tree to observe the default the PLL settings.

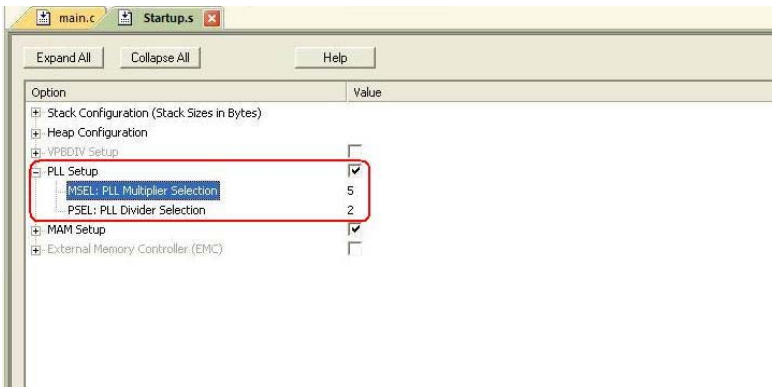

Figure 2.63

To include the PLL code in the startup file select the checkbox next to PLL setup heading. Here MSEL setting indicates that the onboard crystal frequency (12MHz) will be multiplied by 5. This indicates that the CPU core frequency will be 60MHz and this is the maximum operating frequency for LPC2148 microcontroller. Care should be taken while selecting the multiplier value to ensure that the CPU core frequency is not beyond the maximum value. When using the PLL setup the minimum onboard crystal frequency should be at least 10 MHz.

The PLL divider selection maintains the frequency of internal current controlled oscillator in the range of 150-320MHz. For more information please refer LPC2148

user manual. The default setting for **MSEL** is 5 and for **PSEL** it is 2. So the core frequency (FCLK) is 60MHz.
To change any setting just click on it and make use of the up/down arrows.

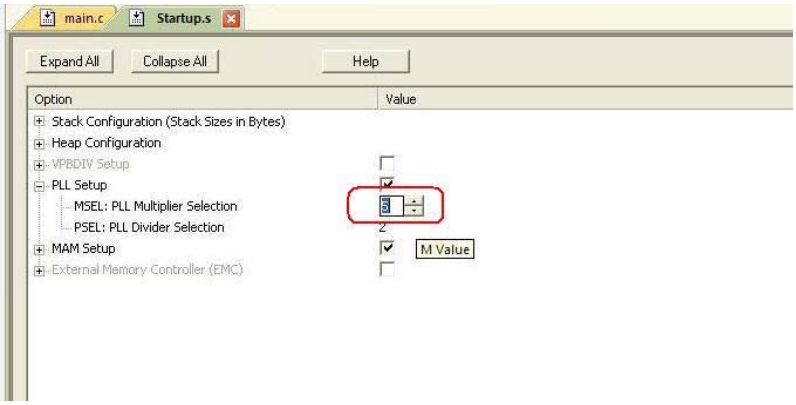

Figure 2.64

4. Now expand VPBDIV Setup tree and observe its settings. It is used to prescale the peripheral clock (PCLK).

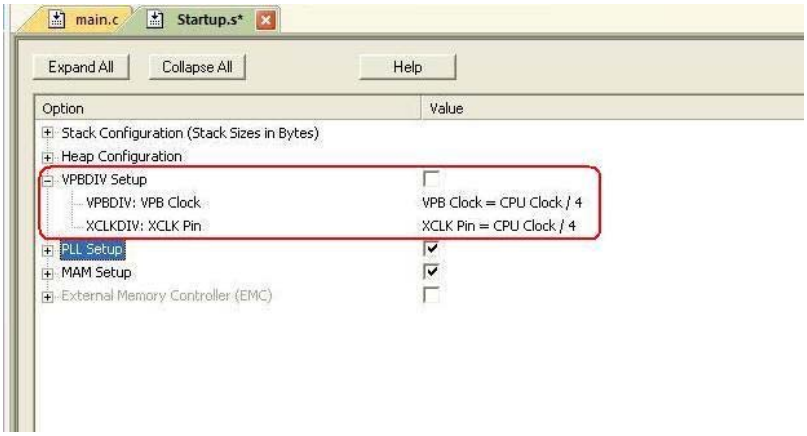

Figure 2.65

Observe that in the default mode it is not included but at reset the VPBDIV loads the default value that is ¼ of the CPU clock. Using the configuration wizard you can set the peripheral clock as:
¼ of CPU clock or
½ of CPU clock or
equal to CPU clock.

For other settings please refer LPC2148 microcontroller user manual.
After you have done using configuration wizard rebuild the project.

3. Input / Output Operations on the Robot

LPC2148 microcontroller has two 32 bit ports as P0 and P1. However pins P0.24, P0.26, P0.27 on P0 and pins P1.0 through P1.15 on P1 are not available. Input/output operations are the most basic, easy and essential operations.

We will need frequent I/O operations mainly to do following tasks:

Function	Pins	Input / Output	Recommended Initial State
Robot Direction control	P0.22. P1.21, P0.10, P0.11	Output	Logic 0
LCD display control	P1.17 to P1.19 P1.22 to P1.25	Input / Output	Logic 0
On Board Interrupt Switch	P0.14(EINT1)	Input	Default
Buzzer control	P0.25	Output	Logic 1
Sharp & White line sensor control	P1.16	Output	Logic 1 *

Table 3.1

Note:

* Power to the Sharp IR range sensor 2, 3, 4 and red LEDs of the white line sensor can be turned on permanently by jumper 2 marked as J2 on the main board. In order to control illumination by microcontroller jumper J2 must be removed. For more details refer to the section 8.3 of chapter 8 of the Fire Bird V hardware manual.

By disabling these sensors we can reduce current consumption by 110 mA. Using this feature and wireless communication many robots can use the sensors at the same time using time division multiplexing. For more details refer to the section 8.3 of chapter 8 of the Fire Bird V hardware manual.

3.1 Registers for using PORTs of the LPC2148 microcontroller

Each pin of the port can be addressed individually. Each pin individually can be configured as input or as output. While pin is input it can be kept floating or even pulled up by using internal pull-up. While pin is in the output mode it can be logic 0 or logic 1. To configure these ports as input or output each of the port has five associated registers. These are GPIO Port Pin value register (IOxPIN), GPIO Port Output Set register (IOxSET), GPIO Port Direction control register (IOxDIR), GPIO Port Output Clear register (IOxCLR) where ‘x’ is 0 or 1 indicating particular port number. The fifth one being Pin function select register (PINSELx) where ’x’ indicates register number.

A. GPIO Port Pin value register (IOxPIN)

This register provides the value of port pins that are configured to perform only digital functions. The register will give the logic value of the pin regardless of whether the pin is configured for input or output, or as GPIO or an alternate digital function. As an example,

a particular port pin may have GPIO input, GPIO output, UART receive, and PWM output as selectable functions. Any configuration of that pin will allow its current logic state to be read from the IOPIN register.
If a pin has an analog function as one of its options, the pin state cannot be read if the analog configuration is selected. Selecting the pin as an A/D input disconnects the digital features of the pin. In that case, the pin value read in the IOPIN register is not valid. Writing to the IOPIN register stores the value in the port output register, bypassing the need to use both the IOSET and IOCLR registers to obtain the entire written value. This feature should be used carefully in an application since it affects the entire port.
For e.g.

```
IO0PIN = (IO0PIN && 0xFFFF00FF) || 0x0000A500
```

The above code will preserve existing output on PORT0 pins P0.[31:16] and P0.[7:0] and at the same time set P0.[15:8] to 0xA5, regardless of the previous value of pins P0.[15:8].

B. GPIO Port Output Set register (IOxSET)

This register is used to produce a HIGH level output at the port pins configured as GPIO in an OUTPUT mode. Writing 1 produces a HIGH level at the corresponding port pins. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing 1 to the corresponding bit in the IOSET has no effect. Reading the IOSET register returns the value of this register, as determined by previous writes to IOSET and IOCLR (or IOPIN as noted above). For example,

```
IO0DIR = 0x00000080; pin P0.7 configured as output
IO0SET = 0x00000080; P0.7 goes HIGH
```

C. GPIO Port Direction control register (IOxDIR)

This word accessible register is used to control the direction of the pins when they are configured as GPIO port pins. Direction bit for any pin must be set according to the pin functionality. Bit ‘0’ of IOxDIR refers to bit ‘0’ of PORTx where ‘x’ indicates port number. If any bit in IOxDIR is set to ‘1’ the corresponding port pin direction is configured as output. If ‘0’ then direction of pin is set as input. For e.g.

```
IO0DIR = 0x00000080; pin P0.7 configured as output
```

D. GPIO Port Output Clear register (IOxCLR)

This register is used to produce a LOW level output at port pins configured as GPIO in an OUTPUT mode. Writing 1 produces a LOW level at the corresponding port pin and clears the corresponding bit in the IOSET register. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to IOCLR has no effect.

E. Pin function select register (PINSELx)

The pin function registers allows selected pins of the microcontroller to have more than one function. This registers control the multiplexers to allow connection between the pin and the on chip peripherals. Peripherals should be connected to the appropriate pins prior to being activated, and prior to any related interrupt(s) being enabled. Activity of any enabled peripheral function that is not mapped to a related pin should be considered undefined. Selection of a single function on a port pin completely excludes all other functions otherwise available on the same pin. The Pin Control Module contains 2 registers namely PINSEL0 and PINSEL1.
The PINSEL0 register controls the functions of the pins as per the settings listed in Figure 3.2. The direction control bit in the IO0DIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Bit	Symbol	Value	Function	Reset value
1:0	P0.0	00	GPIO Port 0.0	0
		01	TXD (UART0)	
		10	PWM1	
		11	Reserved	
3:2	P0.1	00	GPIO Port 0.1	0
		01	RxD (UART0)	
		10	PWM3	
		11	EINT0	
5:4	P0.2	00	GPIO Port 0.2	0
		01	SCL0 (I^2C0)	
		10	Capture 0.0 (Timer 0)	
		11	Reserved	
7:6	P0.3	00	GPIO Port 0.3	0
		01	SDA0 (I^2C0)	
		10	Match 0.0 (Timer 0)	
		11	EINT1	
9:8	P0.4	00	GPIO Port 0.4	0
		01	SCK0 (SPI0)	
		10	Capture 0.1 (Timer 0)	
		11	AD0.6	
11:10	P0.5	00	GPIO Port 0.5	0
		01	MISO0 (SPI0)	
		10	Match 0.1 (Timer 0)	
		11	AD0.7	
13:12	P0.6	00	GPIO Port 0.6	0
		01	MOSI0 (SPI0)	
		10	Capture 0.2 (Timer 0)	
		11	Reserved[1][2] or AD1.0[3]	
15:14	P0.7	00	GPIO Port 0.7	0
		01	SSEL0 (SPI0)	
		10	PWM2	
		11	EINT2	
17:16	P0.8	00	GPIO Port 0.8	0
		01	TXD UART1	
		10	PWM4	
		11	Reserved[1][2] or AD1.1[3]	

19:18	P0.9	00	GPIO Port 0.9	0
		01	RxD (UART1)	
		10	PWM6	
		11	EINT3	
21:20	P0.10	00	GPIO Port 0.10	0
		01	Reserved[1][2] or RTS (UART1)[3]	
		10	Capture 1.0 (Timer 1)	
		11	Reserved[1][2] or AD1.2[3]	
23:22	P0.11	00	GPIO Port 0.11	0
		01	Reserved[1][2] or CTS (UART1)[3]	
		10	Capture 1.1 (Timer 1)	
		11	SCL1 (I^2C1)	
25:24	P0.12	00	GPIO Port 0.12	0
		01	Reserved[1][2] or DSR (UART1)[3]	
		10	Match 1.0 (Timer 1)	
		11	Reserved[1][2] or AD1.3[3]	
27:26	P0.13	00	GPIO Port 0.13	0
		01	Reserved[1][2] or DTR (UART1)[3]	
		10	Match 1.1 (Timer 1)	
		11	Reserved[1][2] or AD1.4[3]	
29:28	P0.14	00	GPIO Port 0.14	0
		01	Reserved[1][2] or DCD (UART1)[3]	
		10	EINT1	
		11	SDA1 (I^2C1)	
31:30	P0.15	00	GPIO Port 0.15	0
		01	Reserved[1][2] or RI (UART1)[3]	
		10	EINT2	
		11	Reserved[1][2] or AD1.5[3]	

[1] Available on LPC2141.

[2] Available on LPC2142.

[3] Available on LPC2144/6/8.

Figure 3.1 PINSEL0 function select register bit description

The PINSEL1 register controls the functions of the pins as per the settings listed in Figure 3.2. The direction control bit in the IO0DIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically.

Bit	Symbol	Value	Function	Reset value
1:0	P0.16	00	GPIO Port 0.16	0
		01	EINT0	
		10	Match 0.2 (Timer 0)	
		11	Capture 0.2 (Timer 0)	
3:2	P0.17	00	GPIO Port 0.17	0
		01	Capture 1.2 (Timer 1)	
		10	SCK1 (SSP)	
		11	Match 1.2 (Timer 1)	
5:4	P0.18	00	GPIO Port 0.18	0
		01	Capture 1.3 (Timer 1)	
		10	MISO1 (SSP)	
		11	Match 1.3 (Timer 1)	
7:6	P0.19	00	GPIO Port 0.19	0
		01	Match 1.2 (Timer 1)	
		10	MOSI1 (SSP)	
		11	Capture 1.2 (Timer 1)	
9:8	P0.20	00	GPIO Port 0.20	0
		01	Match 1.3 (Timer 1)	
		10	SSEL1 (SSP)	
		11	EINT3	
11:10	P0.21	00	GPIO Port 0.21	0
		01	PWM5	
		10	Reserved[1][2] or AD1.6[3]	
		11	Capture 1.3 (Timer 1)	
13:12	P0.22	00	GPIO Port 0.22	0
		01	Reserved[1][2] or AD1.7[3]	
		10	Capture 0.0 (Timer 0)	
		11	Match 0.0 (Timer 0)	
15:14	P0.23	00	GPIO Port 0.23	0
		01	V_{BUS}	
		10	Reserved	
		11	Reserved	
17:16	P0.24	00	Reserved	0
		01	Reserved	
		10	Reserved	
		11	Reserved	
19:18	P0.25	00	GPIO Port 0.25	0
		01	AD0.4	
		10	Reserved[1] or Aout(DAC)[2][3]	
		11	Reserved	

21:20	P0.26	00	Reserved	0
		01	Reserved	
		10	Reserved	
		11	Reserved	
23:22	P0.27	00	Reserved	0
		01	Reserved	
		10	Reserved	
		11	Reserved	
25:24	P0.28	00	GPIO Port 0.28	0
		01	AD0.1	
		10	Capture 0.2 (Timer 0)	
		11	Match 0.2 (Timer 0)	
27:26	P0.29	00	GPIO Port 0.29	0
		01	AD0.2	
		10	Capture 0.3 (Timer 0)	
		11	Match 0.3 (Timer 0)	
29:28	P0.30	00	GPIO Port 0.30	0
		01	AD0.3	
		10	EINT3	
		11	Capture 0.0 (Timer 0)	
31:30	P0.31	00	GPO Port only	0
		01	UP_LED	
		10	CONNECT	
		11	Reserved	

[1] Available on LPC2141.

[2] Available on LPC2142.

[3] Available on LPC2144/6/8.

Figure 3.2 PINSEL1 function select register bit description

3.2 LPC2148 microcontroller pin configuration

Pin no.	Pin name	USED FOR
1	P0.21/PWM5/AD1.6/CAP1.3	PWM for right motor.
2	P0.22/AD1.7/CAP0.0/MAT0.0	Logic input 1 for Left motor (Left back)
3	RTXC1	RTC Crystal(32.768KHz) pin 1
4	P1.19/TRACEPKT3	LCD control line RS (Register Select)
5	RTXC2	RTC Crystal(32.768KHz) pin 2
6	VSS	**Ground:** 0 V reference
7	VDDA	Analog 3.3 V Power Supply
8	P1.18/TRACEPKT2	LCD control line RW(Read/Write Select)
9	P0.25/AD0.4/Aout	Buzzer
10	D+	USB bidirectional D+ line.
11	D-	USB bidirectional D- line.
12	P1.17/TRACEPKT1	LCD control line EN(Enable Signal)
13	P0.28/AD0.1/CAP0.2/MAT0.2	ADC input for white line sensor 2
14	P0.29/AD0.2/CAP0.3/MAT0.3	ADC input for white line sensor 3
15	P0.30/AD0.3/EINT3/CAP0.0	TSOP 1738 Input Capture pin via jumper J3. GPIO pin available on expansion slot.*
16	P1.16/TRACEPKT0	Sharp IR ranges sensor 2, 3, 4 and red LEDs of white line sensor 1, 2, 3 enable/disable pin** Turns off these sensors when output is logic 1
17	P0.31	USB Good Link LED indicator. It is ON during USB communication, otherwise it is OFF.
18	VSS	**Ground:** 0 V reference
19	P0.0/TXD0/PWM1	Transmitter output for UART0/PWM for servo motor***
20	P1.31/TRST	TSOP1738 Input Pin via jumper J1 and J3****
21	P0.1/RxD0/PWM3/EINT0	Receiver input for UART0
22	P0.2/SCL0/CAP0.0	I2C bus / GPIOs/Input Capture (Available on expansion slot of the microcontroller socket)
23	VDD	3.3 V Power Supply
24	P1.26/RTCK	GPIO/JTAG Pin(Available on expansion slot of the microcontroller socket)
25	VSS	**Ground:** 0 V reference
26	P0.3/SDA0/MAT0.0/EINT1	I2C bus / GPIOs/Input Capture (Available on expansion slot of the microcontroller socket)
27	P0.4/SCK0/CAP0.1/AD0.6	ADC input for Sharp IR range sensor 2 or IR proximity sensor 2 via 0E jumper*****
28	P1.25/EXTIN0	LCD data line(DB7)
29	P0.5/MISO0/MAT0.1/AD0.7	ADC input for Sharp IR range sensor 4 or IR proximity sensor 4 via 0E jumper*****
30	P0.6/MOSI0/CAP0.2/AD1.0	ADC input for Sharp IR range sensor 3
31	P0.7/SSEL0/PWM2/EINT2	PWM for left motor.
32	P1.24/TRACECLK	LCD data line(DB6)
33	P0.8/TXD1/PWM4/AD1.1	UART 1 transmit for XBee wireless module (if installed)
34	P0.9/RxD1/PWM6/EINT3	UART 1 receive for XBee wireless module

		(if installed)
35	P0.10/RTS1/CAP1.0/AD1.2	Logic input 2 for Right motor (Right forward)
36	P1.23/PIPESTAT2	LCD data line(DB5)
37	P0.11/CTS1/CAP1.1/SCL1	Logic input 1 for Right motor (Right back)
38	P0.12/DSR1/MAT1.0/AD1.3	ADC input for white line sensor 1
39	P0.13/DTR1/MAT1.1/AD1.4	ADC input for battery voltage monitoring
40	P1.22/PIPESTAT1	LCD data line(DB4)
41	P0.14/DCD1/EINT1/SDA1	Boot loader entry switch. External interrupt for on board boot switch in application mode.**********
42	VSS	**Ground:** 0 V reference
43	VDD	3.3 V Power Supply
44	P1.21/PIPESTAT0	Logic input 1 for Left motor (Left back)
45	P0.15/RI1/EINT2/AD1.5	External Interrupt for the left motor's position encoder
46	P0.16/EINT0/MAT0.2/CAP0.2	External Interrupt for the right motor's position encoder
47	P0.17/CAP1.2/SCK1/MAT1.2	(Available on expansion slot of the microcontroller socket)
48	P1.20/TRACESYNC	GPIO/JTAG Pin(Available on expansion slot of the microcontroller socket)
49	VBAT	RTC Power Supply from 3V cell or 3.3V power supply.
50	VSS	**Ground:** 0 V reference
51	VDD	3.3 V Power Supply
52	P1.30/TMS	GPIO/JTAG Pin(Available on expansion slot of the microcontroller socket)
53	P0.18/CAP1.3/MISO1/MAT1.3	(Available on expansion slot of the microcontroller socket)
54	P0.19/MAT1.2/MOSI1/CAP1.2	(Available on expansion slot of the microcontroller socket)
55	P0.20/MAT1.3/SSEL1/EINT3	Not Connected
56	P1.29/TCK	GPIO/JTAG Pin(Available on expansion slot of the microcontroller socket)
57	RESET	Microcontroller Reset pin connected to onboard Reset Switch
58	P0.23	Indicates the presence of USB bus power
59	VSSA	**Analog Ground:** 0 V reference.
60	P1.28/TDI	GPIO/JTAG Pin(Available on expansion slot of the microcontroller socket)
61	XTAL2	Crystal 12.0 MHz
62	XTAL1	
63	VREF	ADC Reference Voltage(3.3V)
64	P1.27/TDO	GPIO/JTAG Pin(Available on expansion slot of the microcontroller socket)

Table 3.5: LPC2148 microcontroller pin connections

Note:

***** Pin 15 is tied to TSOP1738 output pin by shorting pads of jumper J3 on the microcontroller board. This pin is used to capture pulses generated by TSOP1738.

****** Sensor's switching can be controlled only if jumper J2 on main board is kept open.

******* Disconnect servo motor during ISP (In System Programming) or UART0 serial communication.

******** This pin is tied to TSOP1738 output pin via Jumper J2 and J3 on the microcontroller board. It is used to sense positive or negative edge of pulses coming from TSOP1738.

********* IR Proximity sensor no. 2 & 4 are connected to AD0.6 and AD0.7 via 0E resistors. These pins are directly connected to Sharp IR range finder no. 2 & 4. So to use Sharp IR range finder 2 & 3 instead of IR proximity 2 & 4 desolder these resistors from the main board. Refer Section 8.7.11 to know more about IR Proximity Sensors.

********** In application mode switch connected to this pin can be used to interrupt microcontroller. Otherwise it is used as entry mechanism for boot loader.

4. Sample Applications

4.1 Buzzer Beep

This application example is located in the folder “Experiments \ Buzzer_Beep” folder in the documentation CD.

In the previous chapter, we have loaded buzzer beep code in Fire Bird V. Now we will see in detail the structure of this code.

This experiment demonstrates the simple operation of Buzzer ON/OFF with approximately 1 second delay. Buzzer is connected to P0.25 pin of the LPC2148 microcontroller.

Concepts covered: Output operation, generating delay

Note:
1. Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Steup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4
For more details refer section 4.8 in the hardware manual.

```
#include <lpc214x.h>


/**********Macros*****************/

#define BUZZER_OFF() IO0CLR=(1<<25)          //Macro to turn OFF buzzer
#define BUZZER_ON() IO0SET=(1<<25)           //Macro to turn ON buzzer

/*********************************/


/****Function Prototypes**********/

void Buzzer_Delay(void);
void Init_Buzzer_Pin(void);
```

```
void Init_Peripherals(void);
void Init_Ports(void);

/*********************************/


/*************************************************************

      Function          : Buzzer_Delay
      Return type       : None
      Parameters        : None
      Description       : Provides small amount of delay between
                               buzzer toggles.
*************************************************************/
void Buzzer_Delay(void)
{
 unsigned int  i,j;
 for(j=0;j<20;j++)
 {
  for(i=0; i<60000; i++);
 }
}


/*************************************************************

      Function          : Init_Buzzer_Pin
      Return type       : None
      Parameters        : None
      Description       : Initialises Buzzer pin
*************************************************************/

void Init_Buzzer_Pin(void)
{
 PINSEL1&=0xFFF3FFFF;
 PINSEL1|=0x00000000;         //Set P0.25 as GPIO
 IO0DIR&=0xFDFFFFFF;
 IO0DIR|= (1<<25);            //Set P0.25 as Output
 BUZZER_OFF();                      //Initially turn OFF buzzer
}


void Init_Ports(void)
{
 Init_Buzzer_Pin();
}

void Init_Peripherals(void)
{
 Init_Ports();
}

int main(void)
{
 PINSEL0 = 0x00000000;        // Reset all pins as GPIO
 PINSEL1 = 0x00000000;
 PINSEL2 = 0x00000000;
```

```
 Init_Peripherals();        // Init Port pins and Peripherals
 while(1)
 {
  BUZZER_ON();                          //Turn ON buzzer
  Buzzer_Delay();                  //Wait
  BUZZER_OFF();                         //Turn OFF Buzzer
  Buzzer_Delay();                  //Wait
 }
}
```

In this code, first line represents processor specific header file declaration. The # include directive is used for including header files in the existing code. These file contains the definitions of special function registers related to this device. This file can be found in \ Keil\ARM\INC\Philips subfolder of **Keil** installation directory.

The next two lines,

```
#define BUZZER_ON() IO0CLR=(1<<25)     //Macro to turn ON buzzer
#define BUZZER_OFF() IO0SET=(1<<25)    //Macro to turn OFF buzzer
```

declares macros to turn ON and OFF buzzer. It mainly controls the state of pin (P0.25) that is connected to buzzer.

The function `Buzzer_Delay(x)` generates app 1 mSec delay. The integer value passed to this function determines the delay period.
In all the codes we will configure pins related to any particular module in the `Init_xxxx_Pin()` functions. In this example code we have used the function `Init_Buzzer_Pin`. Buzzer is connected to P0.25 pin of the microcontroller. P0.25 is configured as output with the initial value set to logic 1 to keep buzzer off at the time of port initialization. All the `Init_xxxx_Pin()` functions will be initialized in the `Init_Peripherals()` function in all the codes as a convention.

In the above code buzzer is turned on by calling macro `BUZZER_ON()`
`Buzzer_Delay(1000)` introduces delay of approximately 1 second.
Buzzer is turned off by calling macro `BUZZER_OFF()`
`Buzzer_Delay(1000)` introduces delay of approximately 1 second.

All these statements are written in `while(1)` loop construct to make buzzer on and off periodically.

4.2 Simple Input – Output operation

This sample application is located in the folder “Experiments \ I-O Interfacing” folder in the documentation CD.

This experiment demonstrates simple Input and Output operation. When switch is pressed buzzer turned on. When switch is released buzzer is turned off. Refer to folder “Experiments \ I-O Interfacing” folder in the documentation CD to look at the program.

Concepts covered:
Input and Output operations

Connections:
Buzzer: P0.25
Interrupt switch: P0.14

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Steup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

4.3 Motion control

This application code is located in the folder “Experiments \ Motion_Control_Simple” folder in the documentation CD.

Hardware aspects of the motion control are covered in detail in the chapter 3 and 8 in the hardware manual. Robot’s motors are controlled by L293D motor controller from ST Microelectronics. Using L293D, microcontroller can control direction and velocity of both the motors. To change the direction appropriate logic levels (High/Low) are applied to IC L293D’s direction pins. Velocity control is done using pulse width modulation (PWM) applied to Enable pins of L293D IC.

ROBOT DIRECTION	**LEFT BWD (LB) P0.22**	**LEFT FWD(LF) P1.21**	**RIGHT FWD(RF) P0.10**	**RIGHT BWD(RB) P0.11**	**PWM P0.7 for left motor P0.21 for right motor**
FORWARD	0	1	1	0	As per velocity requirement
REVERSE	1	0	0	1	As per velocity requirement
RIGHT *(Left wheel forward, Right wheel backward)*	0	1	0	1	As per velocity requirement
LEFT*(Left wheel backward, Right wheel forward,)*	1	0	1	0	As per velocity requirement
SOFT RIGHT*(Left wheel forward,, Right wheel stop)*	0	1	0	0	As per velocity requirement
SOFT LEFT*(Left wheel stop, Right wheel forward,)*	0	0	1	0	As per velocity requirement
SOFT RIGHT 2 *(Left wheel stop, Right wheel backward)*	0	0	0	1	As per velocity requirement
SOFT LEFT 2 *(Left wheel backward, Right wheel stop)*	1	0	0	0	As per velocity requirement
HARD STOP	0	0	0	0	As per velocity requirement
SOFT STOP (Free running stop)	X	X	X	X	0

Table 4.1: Logic levels for robot motion control

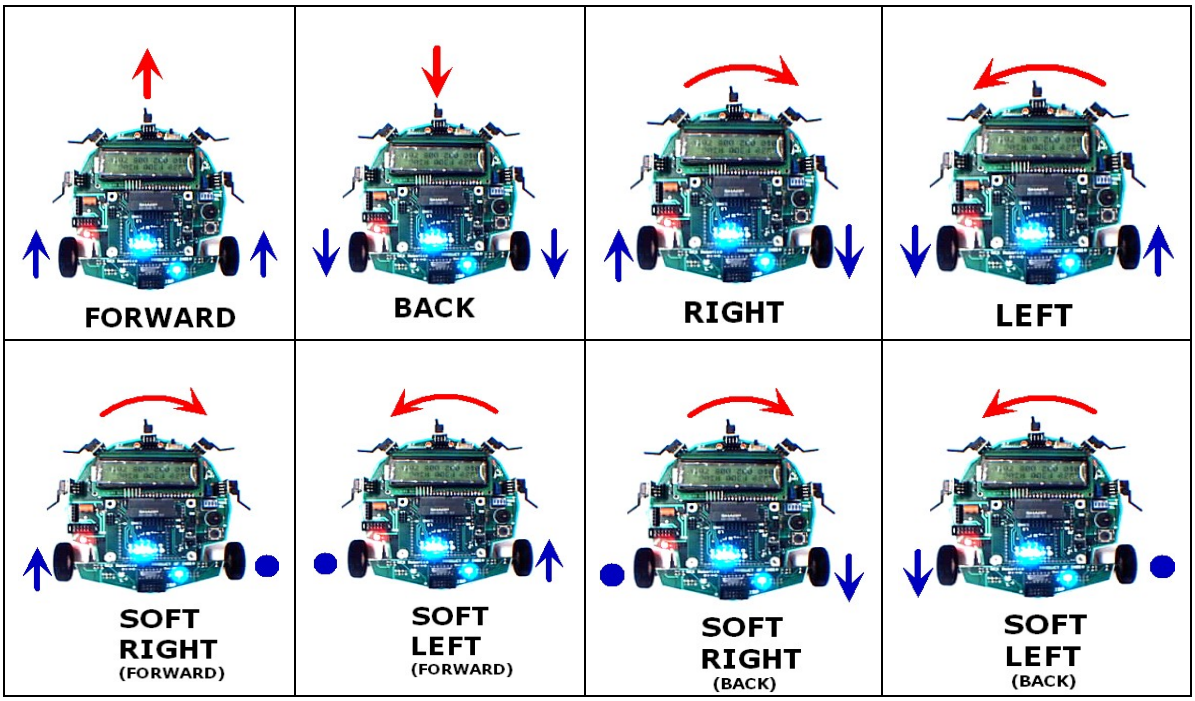

Figure 4.1

Note:

- All the soft turns should be used when you need more accuracy during turning
- Soft left 2 and Soft right 2 motions are very useful in grid navigation.

Concepts covered: Simple motion control using I-O interfacing

There are two components to the motion control:

1. Direction control using pins P0.22, P1.21, P0.10, P0.11
2. Velocity control by PWM on pins P0.7/PWM2 and P0.21/PWM5.

In this experiment for the simplicity P0.7 and P0.21 are kept at logic 1.

Connections:

Microcontroller Pin	Function
P0.7 (PWM2)	Pulse width modulation for the left motor (velocity control)
P0.21 (PWM5)	Pulse width modulation for the right motor (velocity control)
P0.22	Left motor direction control
P1.21	Left motor direction control
P0.10	Right motor direction control
P0.11	Right motor direction control

Table 4.2: Pin functions for the motion control

Note:
1. Note: Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Steup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. Auxiliary power can supply current up to 1 Ampere while Battery can supply current up to 2 Ampere. When robot is on auxiliary power and robot changes is direction suddenly from forward to backward or vice versa, it will cause large current surge and robot's microcontroller might get reset. This will happen repeatedly and you might think that problem is in the software. To avoid this problem use stop for at least 0.5 second before changing direction. This problem will not occur when robot is on battery power and batteries have sufficiently charge. But still it is a good practice to stop robot for 0.5 seconds before changing the direction to increase the life of the motor.

4.4 Sensor Switching

This application code is located in the folder “Experiments \ Sensor_Switching” folder in the documentation CD.

This experiment demonstrates switching on and off of MOSFET switches to turn on and off the active part of the sensors.

Connections:
P1.16 --> Power control of Sharp IR Range sensor 2, 3, 4 and red LEDs of the white line sensors

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. Make sure that Jumper J2 is not inserted on the main board. J2 bypasses software control

4. Logic 1 on pin P1.16 turns OFF the power to the sensor.
Logic 0 on pin P1.16 turns ON the power to the sensor.

4.5 Position Control using Interrupts

This application code is located in the folder “Experiments \ Position_Con_Interrupts” folder in the documentation CD.

This experiment demonstrates use of position encoders and external interrupts used for motion control.

Microcontroller pins used:

P0.22, P1.21, P0.10, P0.11: Robot direction control
P0.7/PWM2 and P0.21/PWM5: Robot velocity control. Currently set to 1 as PWM is not used
P0.15 (EINT2): External interrupt for left motor position encoder
P0.16 (EINT0): External interrupt for the right position encoder

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. Auxiliary power can supply current up to 1 Ampere while Battery can supply current up to 2 Ampere. When robot is on auxiliary power and robot changes is direction suddenly from forward to backward or vice versa, it will cause large current surge and robot's microcontroller might get reset. This will happen repeatedly and you might think that problem is in the software. To avoid this problem use stop for at least 0.5 second before changing direction. This problem will not occur when robot is on battery power and batteries have sufficiently charge. But still it is a good practice to stop robot for 0.5 seconds before changing the direction to increase the life of the motor.

4.6 Velocity Control using PWM

This application code is located in the folder “Experiments \ Velocity_Control_using PWM” folder in the documentation CD.

This experiment demonstrates robot velocity control using PWM.

There are two components to the motion control:
1. Direction control using pins P0.22, P1.21, P0.10, P0.11
2. Velocity control by PWM on pins P0.7/PWM2 and P0.21/PWM5.

In this experiment for the simplicity P0.7 and P0.21 are kept at logic 1.

Connection Details: L-1---->P1.21; L-2---->P0.22;
R-1---->P0.10; R-2---->P0.11;
P0.7 (PWM2) ----> Logic 1; P0.21 (PWM5) ----> Logic 1;

Note:

1. Note: Make sure that in the Target options following settings are done for proper operation of the code.

Microcontroller: LPC2148
Xtal Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Steup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. Auxiliary power can supply current up to 1 Ampere while Battery can supply current up to 2 Ampere. When robot is on auxiliary power and robot changes is direction suddenly from forward to backward or vice versa, it will cause large current surge and robot's microcontroller might get reset. This will happen repeatedly and you might think that problem is in the software. To avoid this problem use stop for at least 0.5 second before changing direction. This problem will not occur when robot is on battery power and batteries have sufficiently charge. But still it is a good practice to stop robot for 0.5 seconds before changing the direction to increase the life of the motor.

4.7 Servo Motor Control using PWM

This application code is located in the folder “Experiments \ Servo_Motor_Control_using_PWM” folder in the documentation CD.

This experiment demonstrates Servo motor control using PWM module.

Fire Bird V ARM7 LPC2148 microcontroller board has connection for a single RC servo motor. Servo motors move between 0 to 180 degrees proportional to the pulse train with the on time of 1 to 2 ms with the frequency between 40 to 60 Hz. 50Hz is most recommended. We are using PWM1 to generate servo control waveform. In this mode servo motors can be controlled with the angular resolution of 2.25 degrees. Although angular resolution is less this is very simple method. There are better ways to produce very high resolution PWM but it involves interrupts at the frequency of the PWM. High resolution PWM is used for servo control in the Hexapod robot.

Connection Details: P0.0/TXD0 PWM1 --> Servo 1: Camera pod pan servo

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Steup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. **5V** supply to these motors is provided by separate low drop voltage regulator "5V Servo" which can supply maximum of 800mA current. It is a good practice to move one servo at a time to reduce power surge in the robot's supply lines. Also preferably take ADC readings while servo motor is not moving or stopped moving after giving desired position.

4. The pin used by servo motor control is also used for loading hex file during ISP. After programming is finished remove DB9 cable from the RS-232 port and then run the servo code by pressing reset

4.8 Timer0 Match Interrupt

This application code is located in the folder “Experiments \ Timer0_Match_Interrupt” folder in the documentation CD.

This experiment demonstrates use of timer match interrupt. In this example timer 0's Match0 interrupt is used to turn on and off buzzer with the time period of 1 second.

Connections: Buzzer is connected to the P0.25.

Note:
1. Make sure that in the Target options following settings are done for proper operation of the code.

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Steup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4
For more details refer section 4.8 in the hardware manual.

4.9 LCD Interfacing

This application code is located in the folder “Experiments \ LCD_Interfacing” folder in the documentation CD.

This experiment demonstrates LCD interfacing in 4 bit mode

LCD Connections:

LCD	Microcontroller Pins
RS	P1.19
RW	P1.18
EN	P1.17
DB7	P1.25
DB6	P1.24
DB5	P1.23
DB4	P1.22

Table 4.3

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code.

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

4.10 Analog to Digital Conversion of Sensor data and Display on LCD

This application code is located in the folder "Experiments \ ADC_Sensor_Display_on_LCD" folder in the documentation CD.

In this experiment ADC captures the analog sensor values and displays it on the LCD

LCD Connections:

LCD	Microcontroller Pins
RS	P1.19
RW	P1.18
EN	P1.17
DB7	P1.25
DB6	P1.24
DB5	P1.23
DB4	P1.22

Table 4.4

ADC Connection:

Sensor	ADC Channel No.
Battery Voltage	AD1.4(P0.13)
White line sensor 1	AD1.3(P0.12)
White line sensor 2	AD0.1(P0.28)
White line sensor 3	AD0.2(P0.29)
IR Proximity analog sensor 2*****	AD0.6(P0.4)
IR Proximity analog sensor 4*****	AD0.7(P0.5)
Sharp IR range sensor 2	AD0.6(P0.4)
Sharp IR range sensor 3	AD1.0(P0.6)
Sharp IR range sensor 4	AD0.7(P0.5)

Table 4.5

***** For using Analog IR proximity (2 and 4) sensors ensure that OE resistors are soldered and remove the respective sharp sensors.

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code.

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. Make sure that you copy the lcd.c file in your folder
4. Distance calculation is for Sharp GP2D12 (10cm-80cm) IR Range sensor

4.11 White Line Following

This application code is located in the folder "Experiments \ White_Line_Following" folder in the documentation CD.

This experiment demonstrates the application of a simple line follower robot. The robot follows a white line over a black background.

Connection Details: L-1---->P1.21; L-2---->P0.22;
R-1---->P0.10; R-2---->P0.11;
P0.7 (PWM2) ----> Logic 1; P0.21 (PWM5) ----> Logic 1;

LCD Connections:

LCD	Microcontroller Pins
RS	P1.19
RW	P1.18
EN	P1.17
DB7	P1.25
DB6	P1.24
DB5	P1.23
DB4	P1.22

Table 4.6

ADC Connection:

Sensor	ADC Channel No.
Battery Voltage	AD1.4(P0.13)
White line sensor 1	AD1.3(P0.12)
White line sensor 2	AD0.1(P0.28)
White line sensor 3	AD0.2(P0.29)
IR Proximity analog sensor 2*****	AD0.6(P0.4)
IR Proximity analog sensor 4*****	AD0.7(P0.5)

Sharp IR range sensor 2	AD0.6(P0.4)
Sharp IR range sensor 3	AD1.0(P0.6)
Sharp IR range sensor 4	AD0.7(P0.5)

Table 4.7

***** For using Analog IR proximity (2 and 4) sensors ensure that OE resistors are soldered and remove the respective sharp sensors.

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code.

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. Make sure that you copy the lcd.c file in your folder
4. Distance calculation is for Sharp GP2D12 (10cm-80cm) IR Range sensor

4.12 Adaptive Cruise Control

This application code is located in the folder “Experiments \ Adaptive_Cruise_Control” folder in the documentation CD.

This experiment demonstrates the application of a simple line follower robot. The robot follows a white line over a black background at the same time it maintains safe distance with the robot ahead.

Connection Details: L-1---->P1.21; L-2---->P0.22;
R-1---->P0.10; R-2---->P0.11;
P0.7 (PWM2) ----> Logic 1; P0.21 (PWM5) ----> Logic 1;

LCD Connections:

LCD	Microcontroller Pins
RS	P1.19
RW	P1.18
EN	P1.17
DB7	P1.25
DB6	P1.24
DB5	P1.23
DB4	P1.22

Table 4.8

ADC Connection:

Sensor	ADC Channel No.
Battery Voltage	AD1.4(P0.13)
White line sensor 1	AD1.3(P0.12)
White line sensor 2	AD0.1(P0.28)
White line sensor 3	AD0.2(P0.29)
IR Proximity analog sensor 2*****	AD0.6(P0.4)
IR Proximity analog sensor 4*****	AD0.7(P0.5)
Sharp IR range sensor 2	AD0.6(P0.4)
Sharp IR range sensor 3	AD1.0(P0.6)
Sharp IR range sensor 4	AD0.7(P0.5)

Table 4.9

***** For using Analog IR proximity (2 and 4) sensors ensure that OE resistors are soldered and remove the respective sharp sensors.

Note:

1. Make sure that in the Target options following settings are done for proper operation of the code.

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

3. Make sure that you copy the lcd.c file in your folder
4. Distance calculation is for Sharp GP2D12 (10cm-80cm) IR Range sensor

4.13A Serial Communication over RS-232 and UART0

This application code is located in the folder “Experiments \ Serial_Communication_UART0” folder in the documentation CD.

This example demonstrates motion control of using by sending data from the PC/laptop over the serial port.

Serial Communication:
P0.1 --> RXD1 UART1 receive for RS232 serial communication
P0.0 --> TXD1 UART1 transmit for RS232 serial communication

Serial communication baud rate: 9600bps

To control robot use number pad of the keyboard which is located on the right hand side of the keyboard and make sure that NUM lock is on.

Commands:

Key	HEX value	Action
8	0x38	Forward
2	0x32	Backward
4	0x34	Left
6	0x36	Right
5	0x35	Stop
7	0x37	Buzzer on
9	0x39	Buzzer off

Table 4.10

Note:

1. Note: Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:

Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.

4.13B Serial Communication over Xbee wireless module

This application code is located in the folder “Experiments \ Serial_Communication_XBEE” folder in the documentation CD.

This example demonstrates motion control of using by sending data from the PC/laptop over the Xbee wireless module.

Serial Communication:
P0.1 --> RXD1 UART1 receive for RS232 serial communication
P0.0 --> TXD1 UART1 transmit for RS232 serial communication

Serial communication baud rate: 9600bps

To control robot use number pad of the keyboard which is located on the right hand side of the keyboard and make sure that NUM lock is on.

Commands:

Key	HEX value	Action
8	0x38	Forward
2	0x32	Backward
4	0x34	Left
6	0x36	Right
5	0x35	Stop
7	0x37	Buzzer on
9	0x39	Buzzer off

Table 4.11

Note:

1. Note: Make sure that in the Target options following settings are done for proper operation of the code

Microcontroller: LPC2148
XTAL Frequency: 12 Mhz
Create Hex File: Checked (For more information read section 4.3.1 "Setting up Project in Keil uVision" in the hardware manual)

2. Ensure that following settings are done in Startup.s configuration wizard:
Clock Settings:

PLL Setup >> MSEL=5, PSEL=2
VPBDIV Setup >> VPBCLK = CPU Clock/4

For more details refer section 4.8 in the hardware manual.