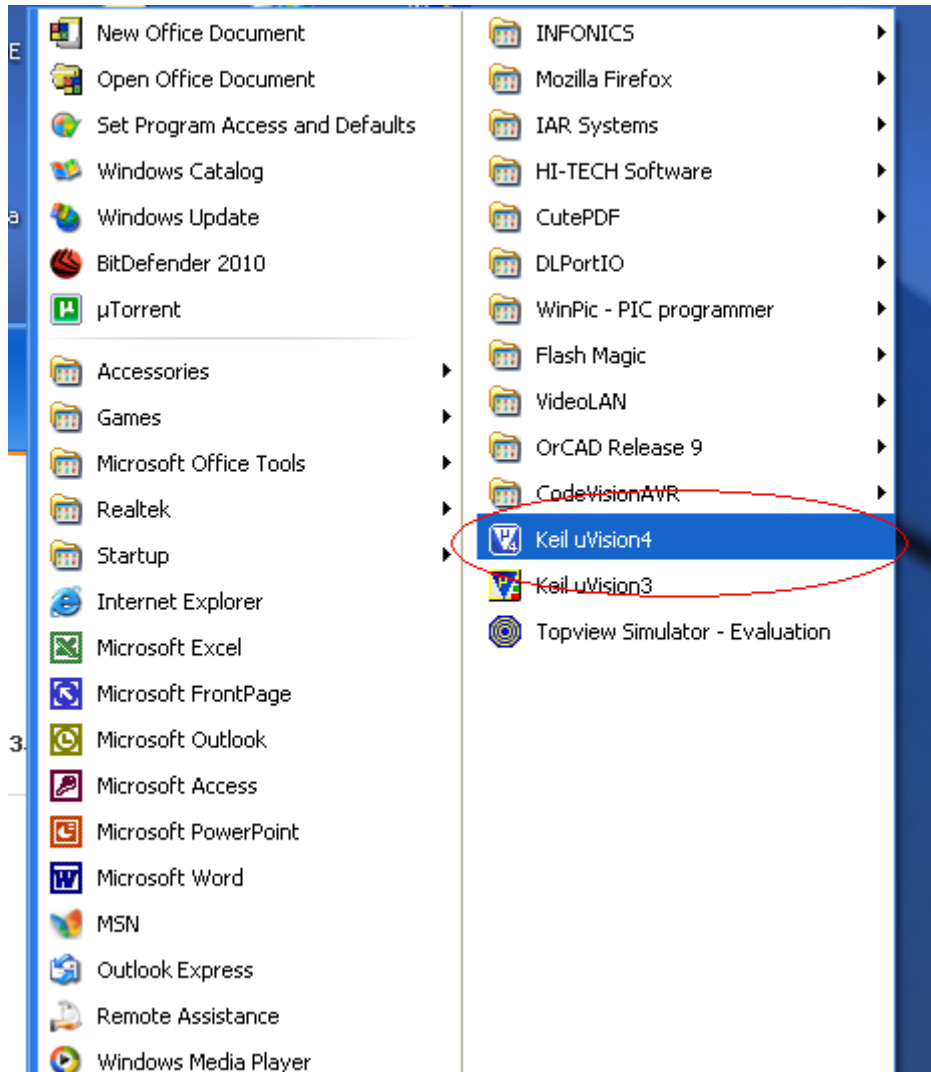# HOW TO USE KEIL µVISION4

For ARM7 (LPC2148)
Step By Step
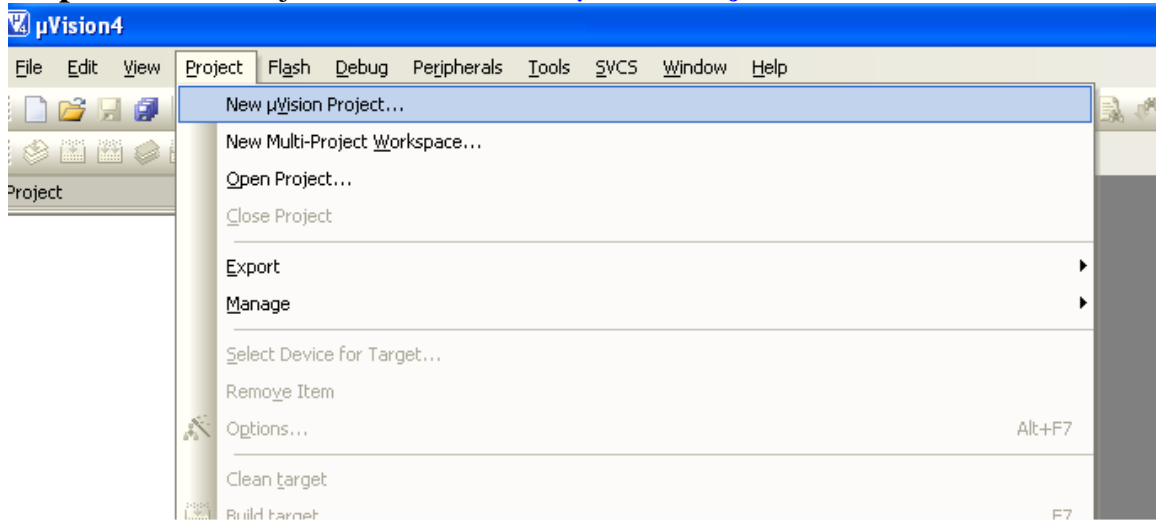
For Running Code in Flash Memory

ARM Training at:
IMBUENT TECHNOLOGIES PVT.LTD.
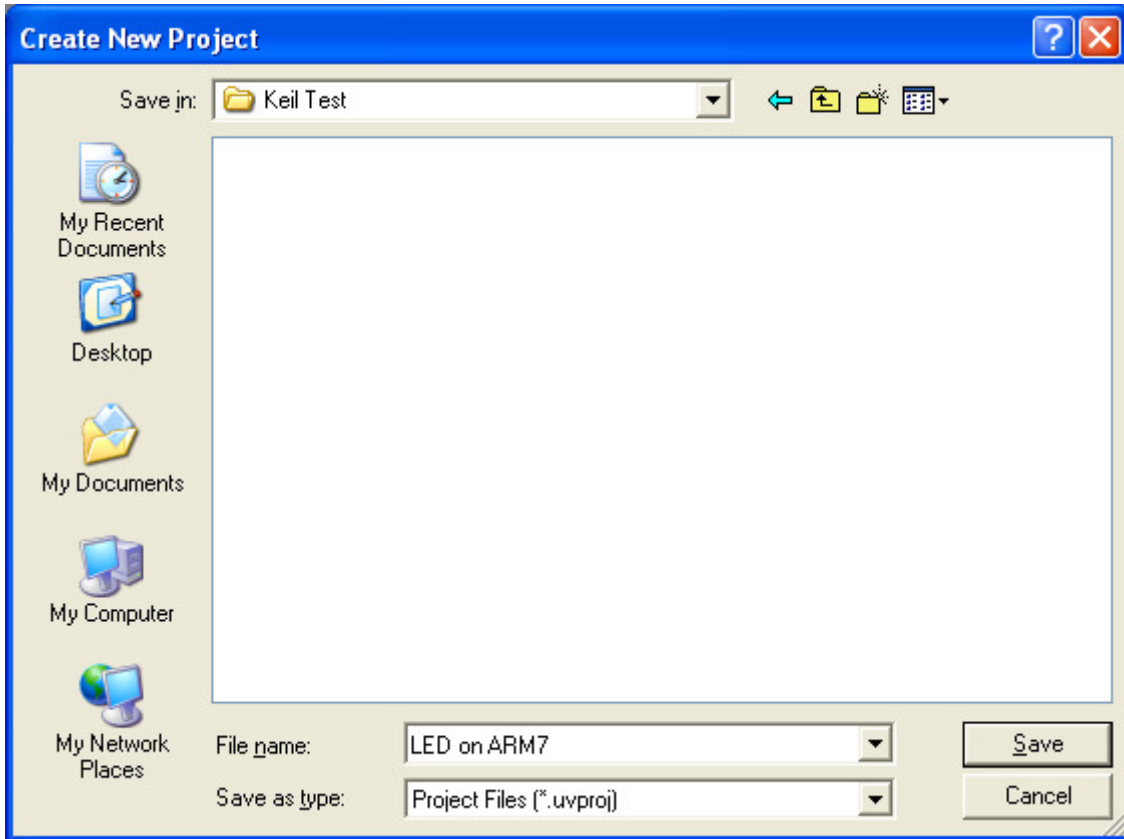Ludhiana (Punjab) India
www.imbuent.com

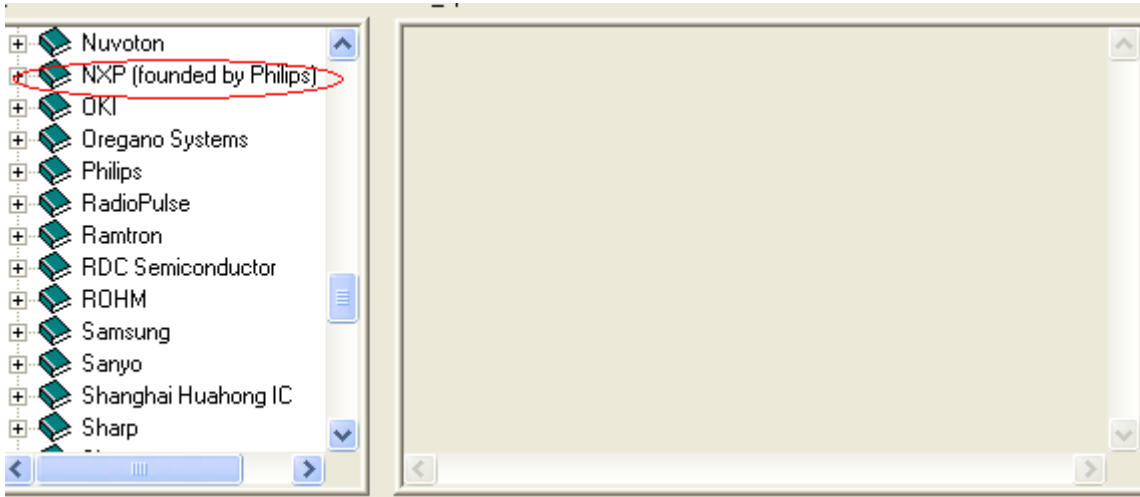**Step1:** Click for **KEIL µVISION4** Icon . Which appearing after Installing Keil
KEIL µVISION4.

Step2: Click on Project Menu, Then **New µVison Project**.

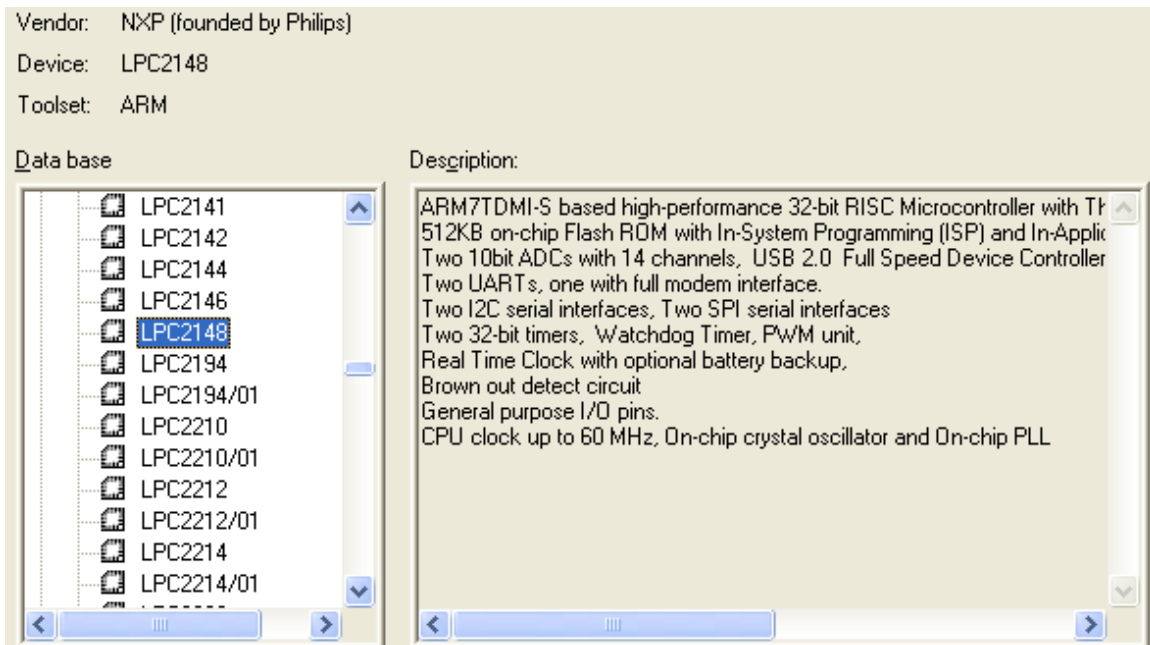**Step3:** Create New Project Folder named as "**Keil Test".**

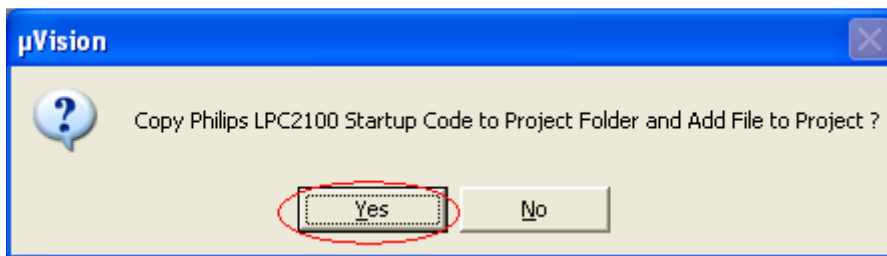**Step 4:** Select Target Device Vendor (i.e.).
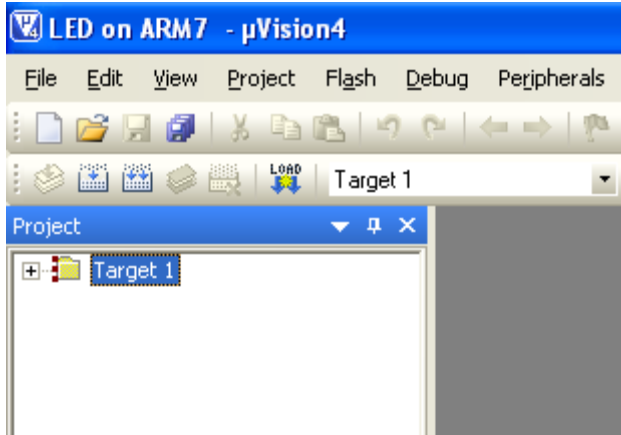

Step 5: Then select specific chip LPC2148.

Vendor: NXP (founded by Philips)
Device: LPC2148
Toolset: ARM

Data base
- LPC2141
- LPC2142
- LPC2144
- LPC2146
- **LPC2148**
- LPC2194
- LPC2194/01
- LPC2210
- LPC2210/01
- LPC2212
- LPC2212/01
- LPC2214
- LPC2214/01

Description:
ARM7TDMI-S based high-performance 32-bit RISC Microcontroller with TH
512KB on-chip Flash ROM with In-System Programming (ISP) and In-Applic
Two 10bit ADCs with 14 channels, USB 2.0 Full Speed Device Controller
Two UARTs, one with full modem interface.
Two I2C serial interfaces, Two SPI serial interfaces
Two 32-bit timers, Watchdog Timer, PWM unit,
Real Time Clock with optional battery backup,
Brown out detect circuit
General purpose I/O pins.
CPU clock up to 60 MHz, On-chip crystal oscillator and On-chip PLL

Step 6: Then select specific chip i.e. LPC2148.

µVision

? Copy Philips LPC2100 Startup Code to Project Folder and Add File to Project ?

[Yes]   [No]

**Step 7:** Then you will see following window.

Step 8: Now you see Startup.s is already added which is necessary for running code for Keil.

Note: Code wills Not Run without Startup.s
Startup.s is available in C:\Keil\ARM\Startup\Philips.

The startup-code executes immediately upon reset of the target system and performs the following operations:

1. Defines interrupt and exception vectors.
2. Configures the CPU clock source (on some devices).
3. Initializes the external bus controller.
4. Copies the exception vectors from ROM to RAM for systems with memory remapping.
5. Initializes other low level peripherals, if necessary.
6. Reserves and initializes the stack for all modes.
7. Reserves the heap.
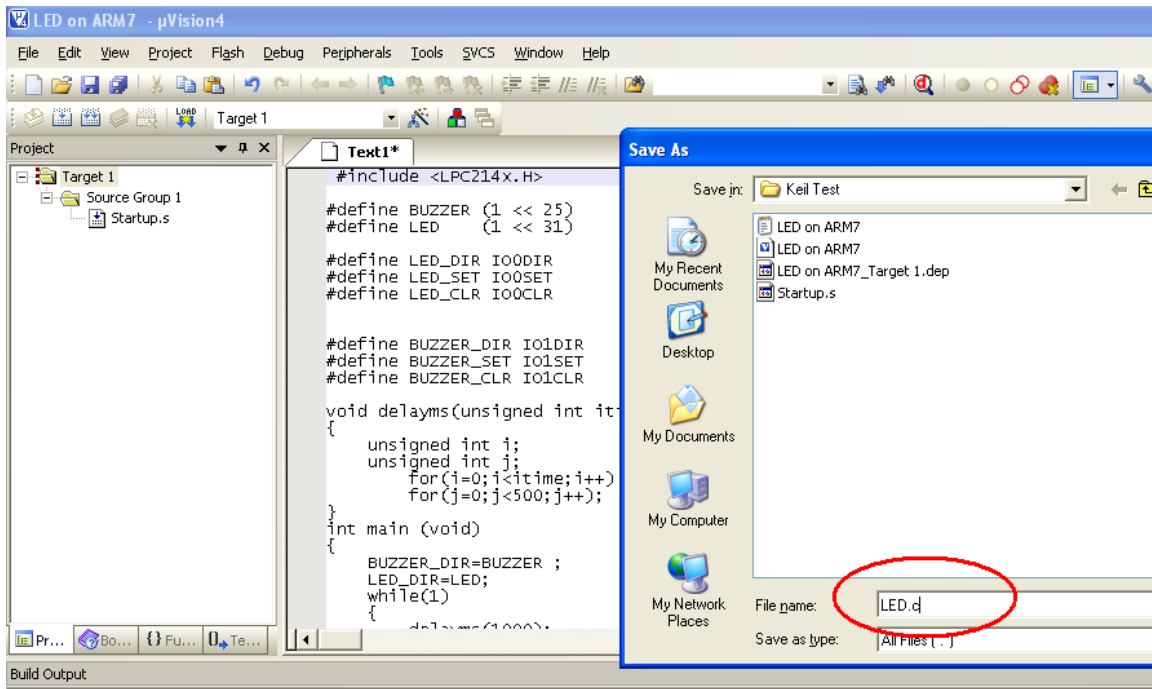8. Transfers control to the main C function.
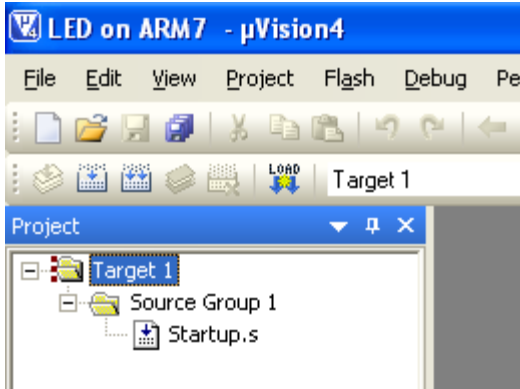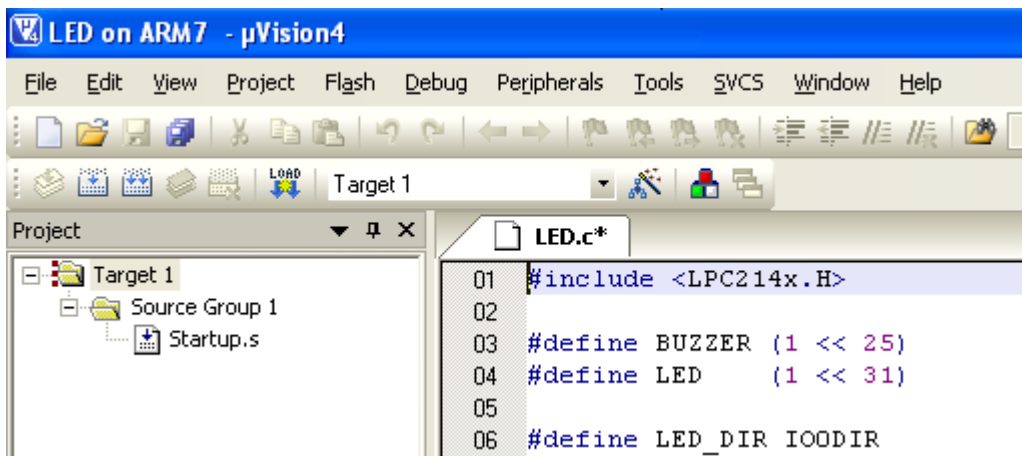
Step 9: Now Click on File Menu and Click on New.
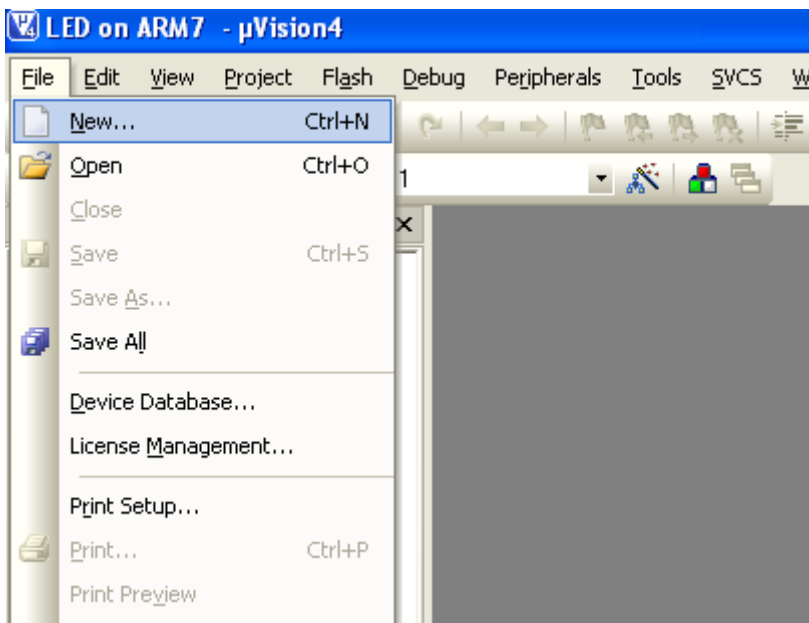
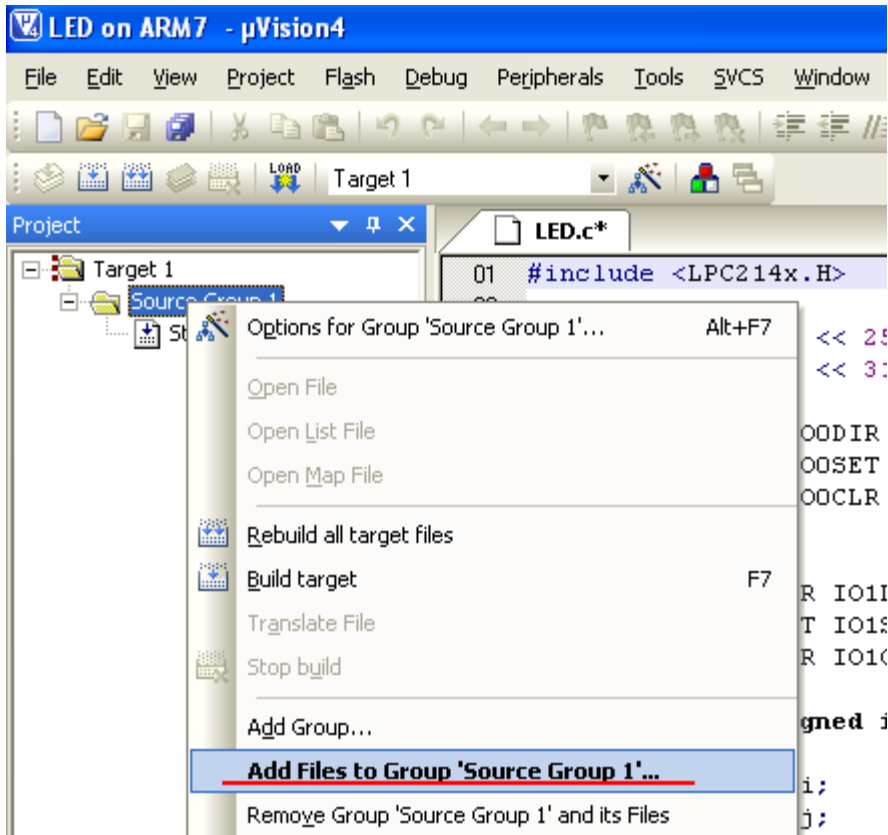**Step 10**: Write Code for Blink LED in C and **FileName.c** Save.
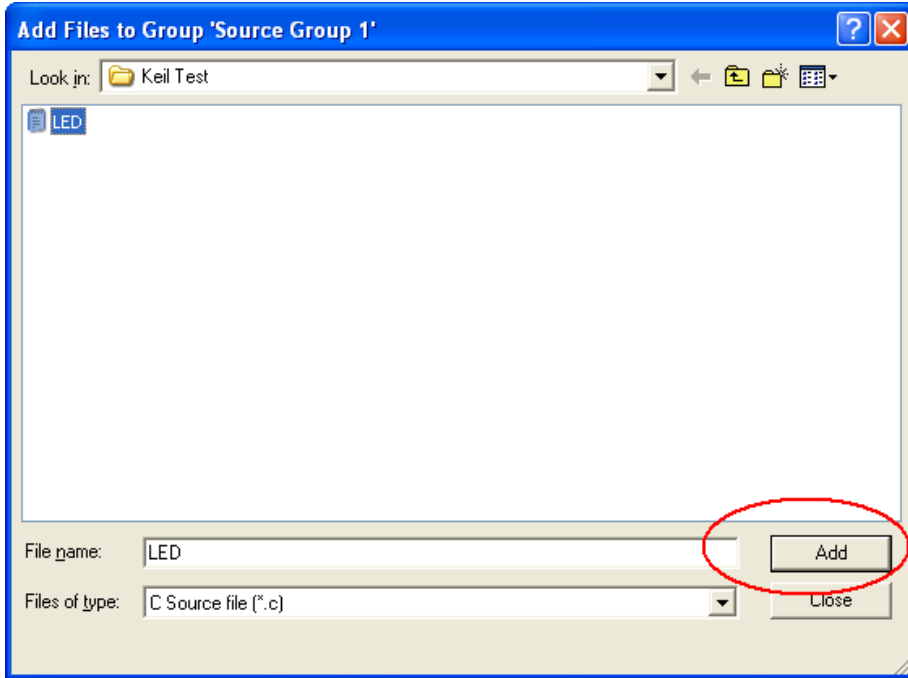
Note: Don't forget to save .c Extension.
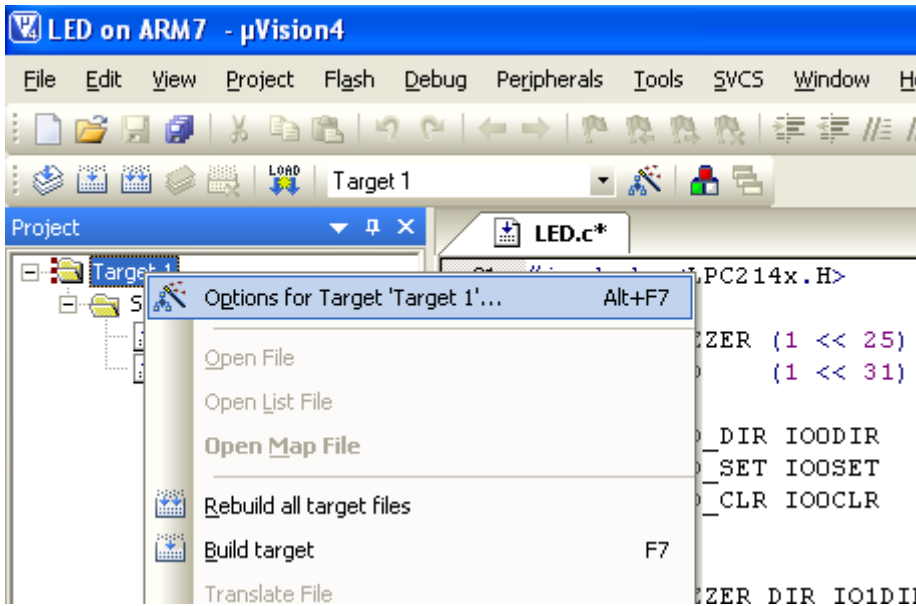
Step 11: Now you Window in C Syntax.

Step 12: Now you add LED.c file by adding Sourse Group 1 Add files to Group 'Source Group 1'.
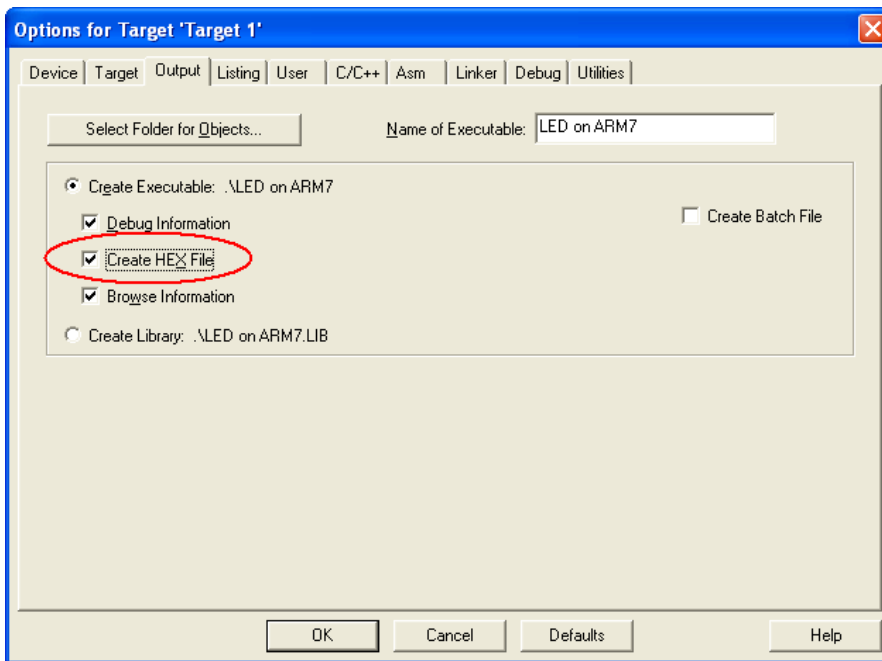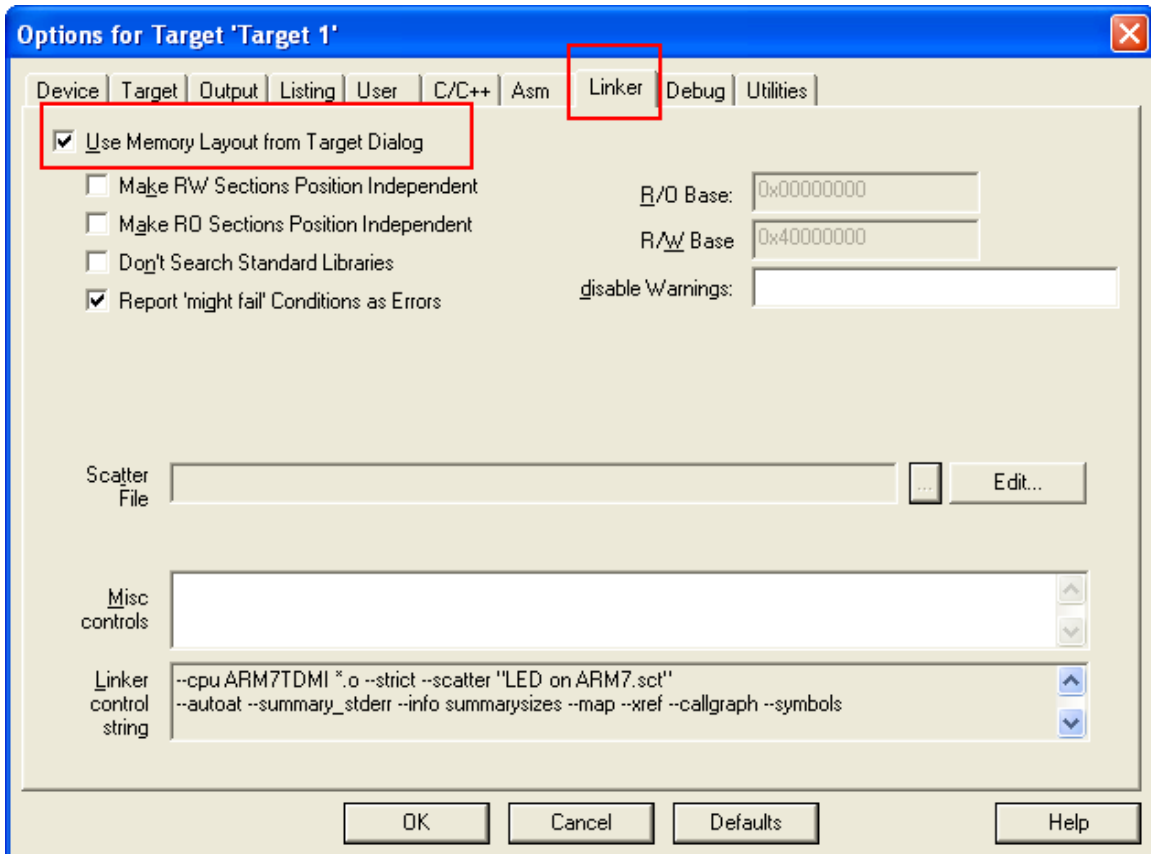
Step 13: Add LED.C file.

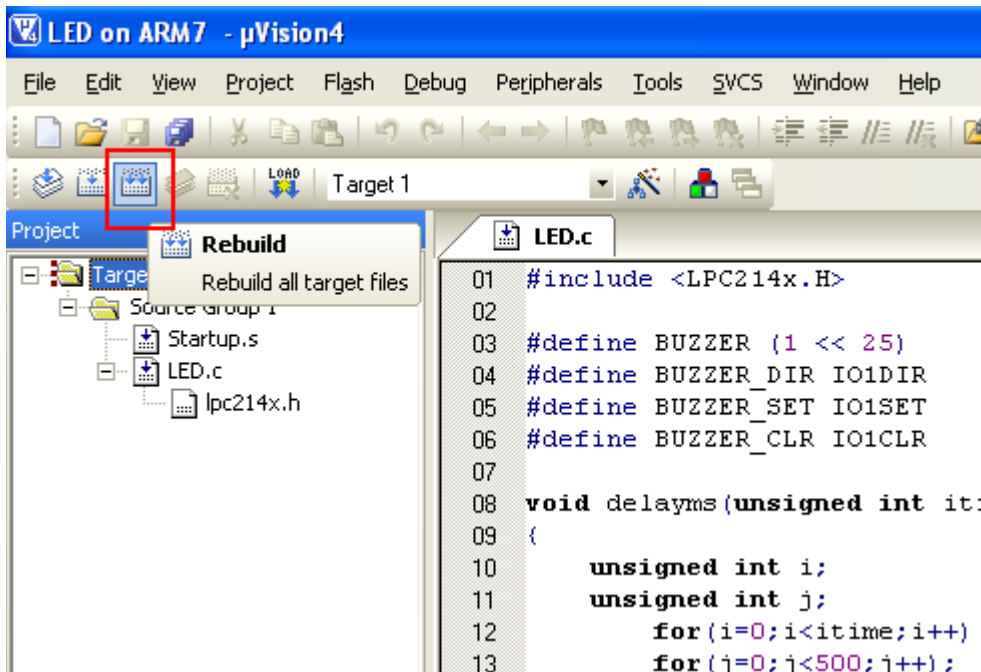**Step 14:** Now Click on Options for Target 'Target 1'.

Step 15: Go to Options for Target 'Target 1'. Click on Check Box Create HEX File.

Step 16: Then go to Linker. Click on Use Memory Layout for Target Dialog.

Step 16: Then Click on Rebuild All Target Files

Step 17: Now you see 0 Error(s), 0 Warning (s). Then Hex File will create in Specific Folder. Now to download it for you target hardware.

```
Build Output
Build target 'Target 1'
assembling Startup.s...
compiling LED.c...
linking...
Program Size: Code=880 RO-data=16 RW-data=0 ZI-data=1256
FromELF: creating hex file...
"LED on ARM7.axf" - 0 Error(s), 0 Warning(s).
```