"Hello world"

AVR Tutorial Series

Now you have the basic hardware tools, its time to setup the software environment. The main softwares you will need are:

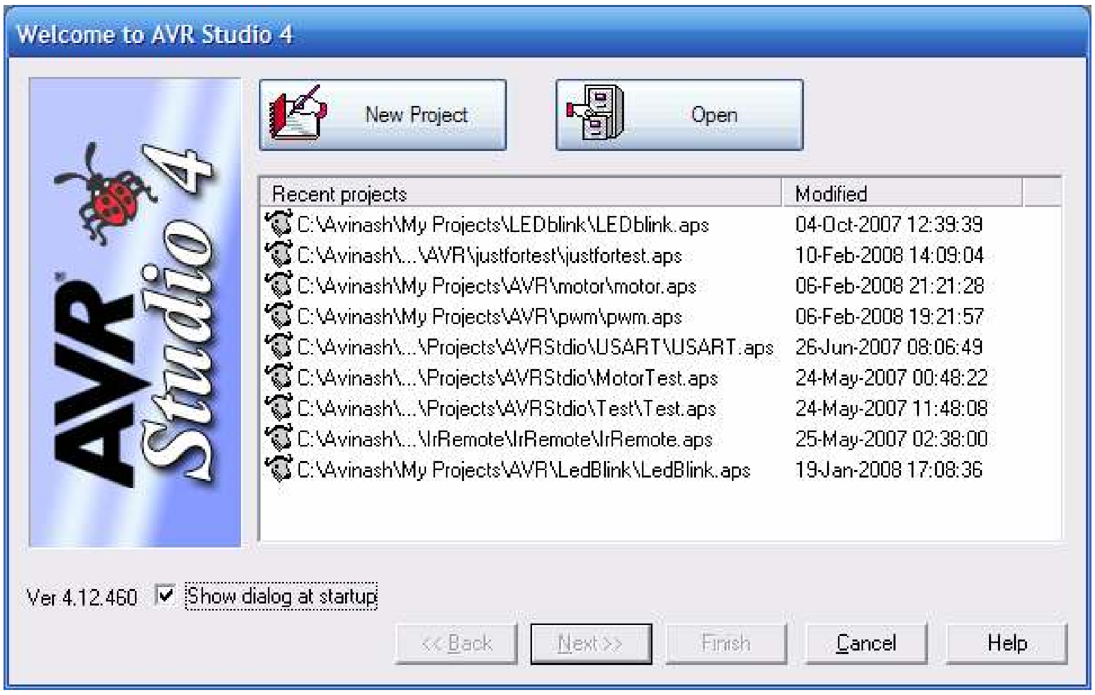
- AVR Studio – A GUI IDE for AVR(www.atmel.com, http://www.atmel.com/dyn/resources/prod_documents/AVRStudio4.13SP2.exe)
- WinAVR – a free C compiler for AVRs (http://winavr.sourceforge.net/download.html)
- AVR Dude GUI – a free software for burning ".hex" file generated by Compiler to AVR MCU (in the xBoard™ MINI). Through the programmer hardware.

All these software are provided in the CD-ROM.(Under Tools Folder). It is better to install WinAVR in root of a drive like c:\winavr or d:\winavr. Also please install WinAVR first then AVR Studio, this will let AVR Studio detect the compiler. Now you are ready to write you first microcontroller program !!!

In this tutorial, you will learn the basic steps required for any microcontrollers based project. We will write a basic "hello world" project, which is a simple LED blinker in MCU empire to demonstrate these basic steps.

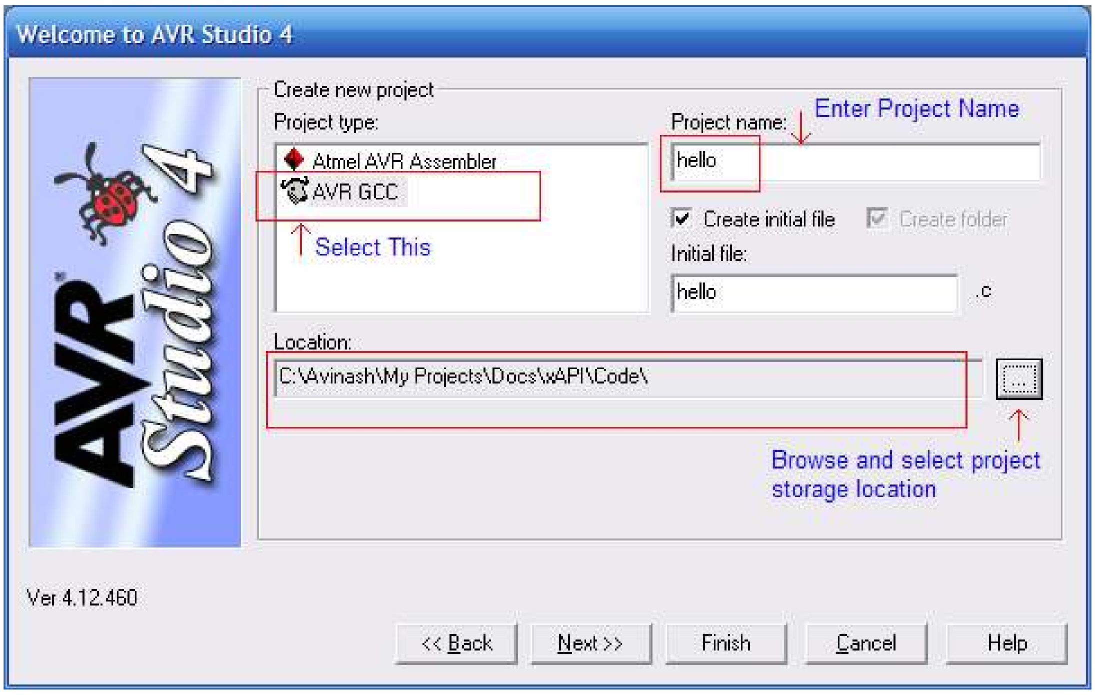
Step I Entering and compiling code.

Start "AVR Studio" from Start Menu->All programs->Atmel AVR Tools-> AVR Studio 4
You will be presented with a Project wizard as shown below.



Recent projects	Modified
C:\Avinash\My Projects\LEDblink\LEDblink.aps	04-Oct-2007 12:39:39
C:\Avinash\...\AVR\justfortest\justfortest.aps	10-Feb-2008 14:09:04
C:\Avinash\My Projects\AVR\motor\motor.aps	06-Feb-2008 21:21:28
C:\Avinash\My Projects\AVR\pwm\pwm.aps	06-Feb-2008 19:21:57
C:\Avinash\...\Projects\AVRStdio\USART\USART.aps	26-Jun-2007 08:06:49
C:\Avinash\...\Projects\AVRStdio\MotorTest.aps	24-May-2007 00:48:22
C:\Avinash\...\Projects\AVRStdio\Test\Test.aps	24-May-2007 11:48:08
C:\Avinash\...\IrRemote\IrRemote\IrRemote.aps	25-May-2007 02:38:00
C:\Avinash\My Projects\AVR\LedBlink\LedBlink.aps	19-Jan-2008 17:08:36

AVR Studio Project Wizard


Welcome to AVR Studio 4

Create new project

Project type:
- Atmel AVR Assembler
- AVR GCC

Project name: hello

Create initial file [✓] Create folder [✓]

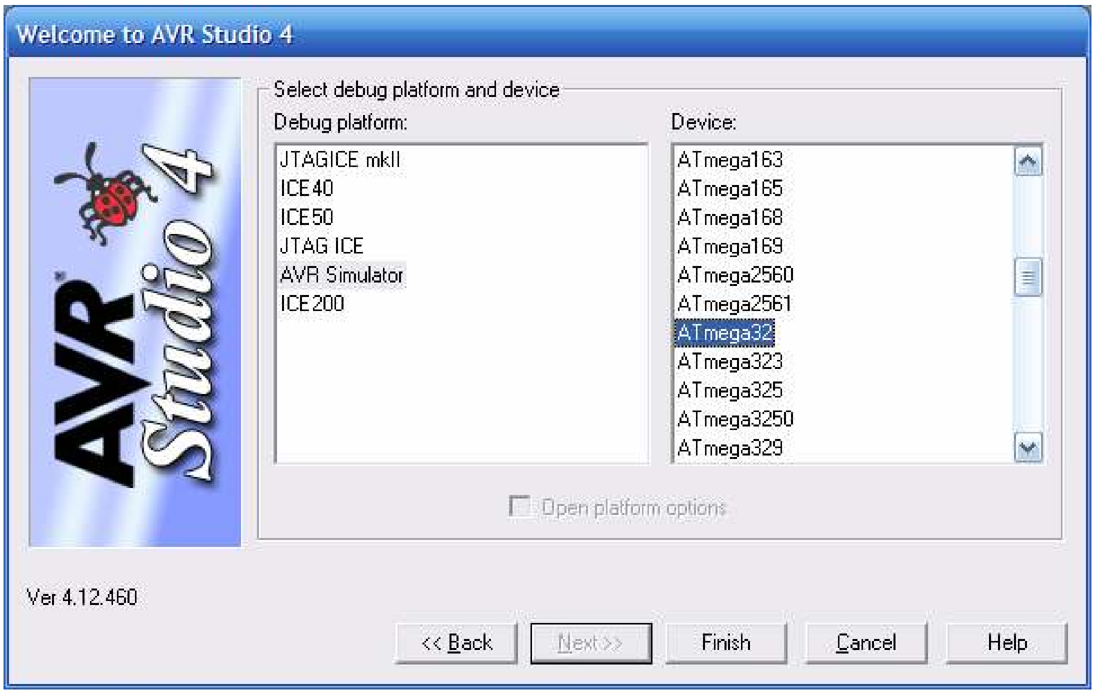
Initial file: hello .c

Location: C:\Avinash\My Projects\Docs\xAPI\Code\

Ver 4.12.460

Project Details

Select AVR GCC in Project type then enter a suitable project name say "hello" and select a location in you hard disk. Then click next. Make sure that "Create initial file" and "Create folder" option is checked.


Welcome to AVR Studio 4

Select debug platform and device

Debug platform:
- JTAGICE mkII
- ICE40
- ICE50
- JTAG ICE
- AVR Simulator
- ICE200

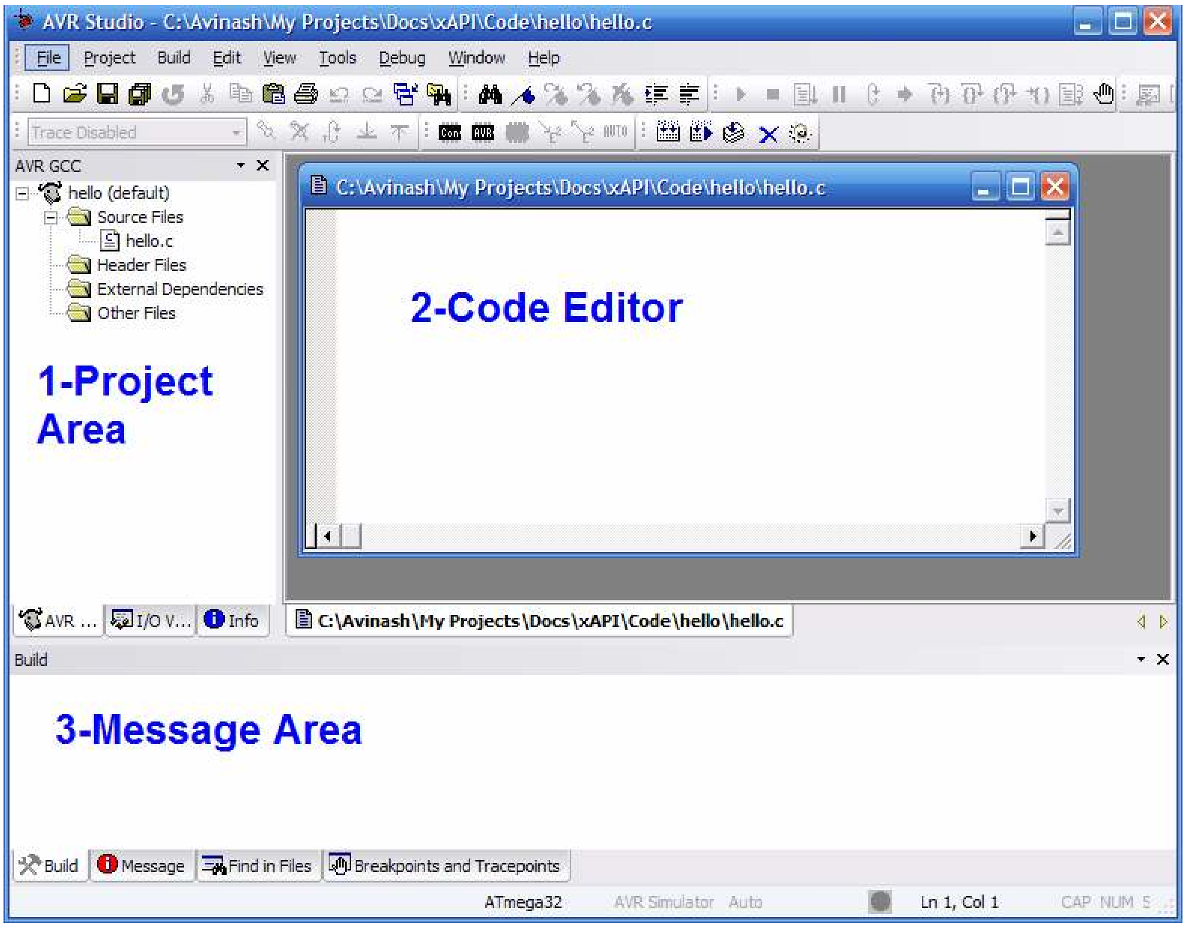
Device:
- ATmega163
- ATmega165
- ATmega168
- ATmega169
- ATmega2560
- ATmega2561
- ATmega32
- ATmega323
- ATmega325
- ATmega3250
- ATmega329

Open platform options

Ver 4.12.460

Device Selection

In this dialog box select AVR Simulator in "Debug Platform" list and select the AVR MCU depending on the type of MCU installed on your development board, in Device list. In xBoard™ MINI ATmega8 is used. Click finish. After that, you will be presented with an Integrated Development Environment-IDE. As shown below.


AVR Studio - C:\Avinash\My Projects\Docs\xAPI\Code\hello\hello.c
File Project Build Edit View Tools Debug Window Help
Trace Disabled
AVR GCC
hello (default)
Source Files
hello.c
Header Files
External Dependencies
Other Files
1-Project Area
C:\Avinash\My Projects\Docs\xAPI\Code\hello\hello.c
2-Code Editor
AVR ... I/O V... Info
Build
3-Message Area
Build Message Find in Files Breakpoints and Tracepoints
ATmega32 AVR Simulator Auto Ln 1, Col 1 CAP NUM

AVR Studio Main Window.

This IDE will help you in editing, modifying, and compile source program. After a project is compiled it gives you a ".hex" file ready to burn to your MCU. The main parts of the window are

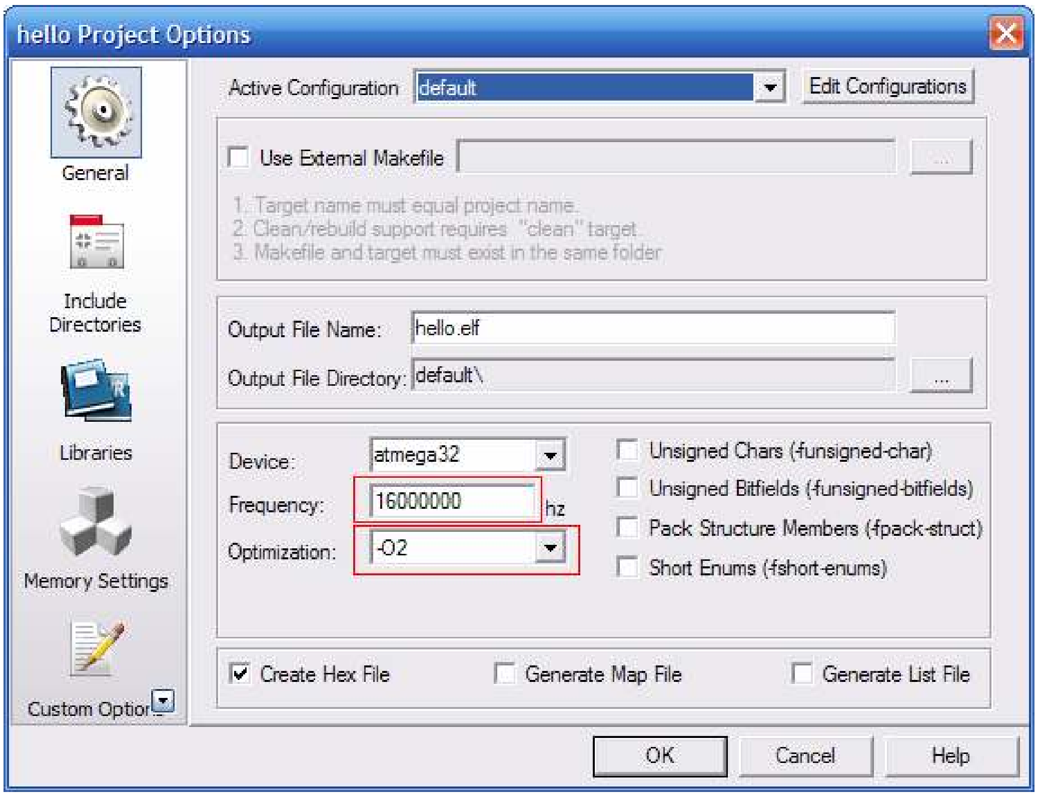
- **Project Area** – displays all the files source and header in the current project. You can add and remove files by the context menu of different groups like "source file" , "header files" etc. Double click a file to open it in the editor.
- **Code editor** – Here you enter and edit the files.
- **Message Area** – Here AVR Studio will show errors and warning generated by compile when it tries to compile a source file.

Now copy past or type the following code in the code editor.

Sample code to blink led on PORT C-0

```
/*

      A Program to blink LED on port C-0
_______________________________________________________________________

      Date  :     26 March 2008

      Copyright 2008 Avinash Gupta
      avinash@extremeelectronics.co.in

      extremeelectronics.co.in
*/


#include <avr/io.h>
#include <util/delay.h>

void Wait()
{
	int i=0;
	for( ;i<23;i++)
		_delay_loop_2(0);
}

void main()
{
	DDRC|=0b00000001;
	PORTC=0;

	while(1)
	{
	PORTC&=0b11111110;

	Wait();

	PORTC|=0b00000001;

	Wait();

	}
}
```

Go to **Project->Configuration Options** to bring the Project option dialog.



Active Configuration: default

☐ Use External Makefile

1. Target name must equal project name.
2. Clean/rebuild support requires "clean" target.
3. Makefile and target must exist in the same folder

Output File Name: hello.elf

Output File Directory: default\

Device: atmega32

Frequency: 16000000 hz

Optimization: -O2

☐ Unsigned Chars (-funsigned-char)

☐ Unsigned Bitfields (-funsigned-bitfields)

☐ Pack Structure Members (-fpack-struct)

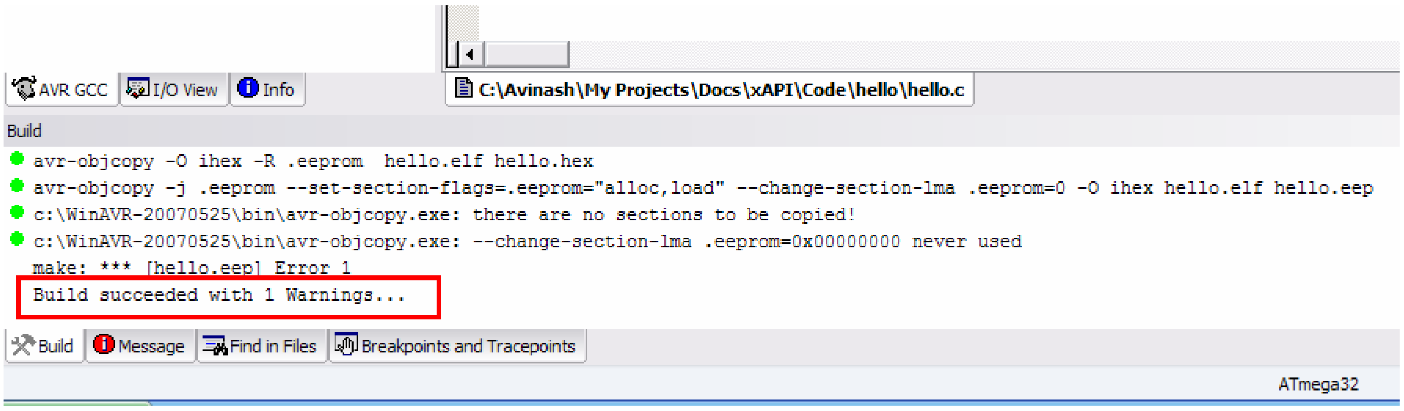
☐ Short Enums (-fshort-enums)

☑ Create Hex File ☐ Generate Map File ☐ Generate List File

Project option

Enter the CPU frequency. If you are using xBoard™ or xBoard™ MINI enter 16Mhz i.e. 16000000. In addition, select optimization as -O2. Click ok.

Now you have entered the code now time to compile and build the project. Press **F7** or **select Build->Build** or click the toolbar button for **Build active configuration.** If the code is error free AVR studio will show you the following message.

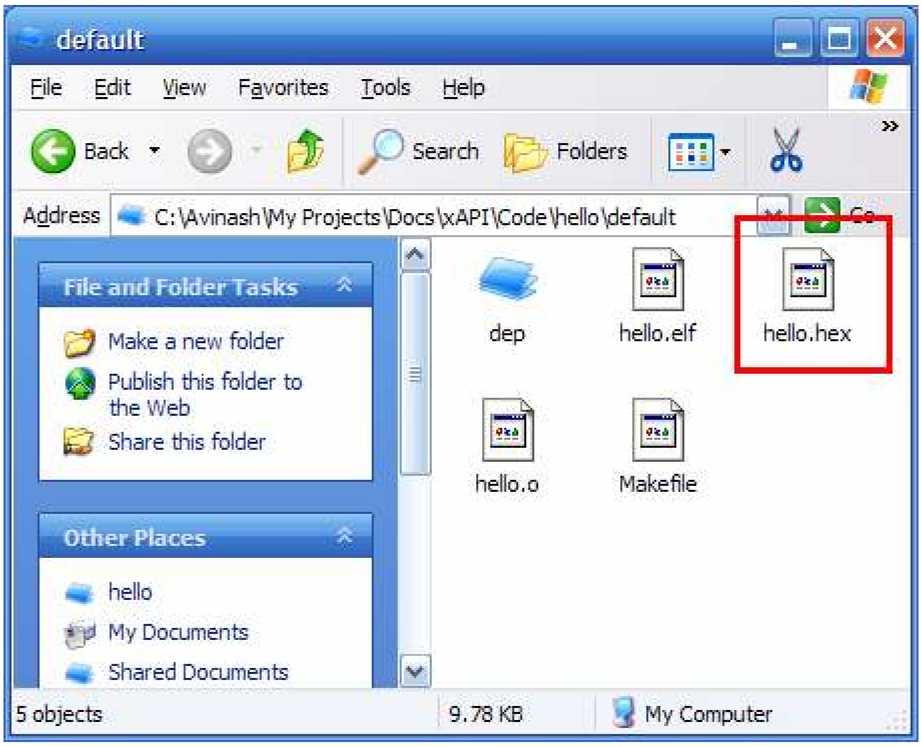


```
avr-objcopy -O ihex -R .eeprom  hello.elf hello.hex
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" --change-section-lma .eeprom=0 -O ihex hello.elf hello.eep
c:\WinAVR-20070525\bin\avr-objcopy.exe: there are no sections to be copied!
c:\WinAVR-20070525\bin\avr-objcopy.exe: --change-section-lma .eeprom=0x00000000 never used
make: *** [hello.eep] Error 1
Build succeeded with 1 Warnings...
```

AVR GCC
I/O View
Info
C:\Avinash\My Projects\Docs\xAPI\Code\hello\hello.c
Build
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega32

Message

"Build succeed with 1 warning. Don't worry about the one warning it is due to the fact that ANSI standard suggest that return type of **main()** must be one, but for MCU platform there is no environment or operating system that will receive this returned value. So return type of **main()** is **void.**
Now you have successfully compiled you first project, what you have got after compilation and build is ".hex file". You can find it in a folder named "default" in you project folder. It has same name as you project, in this case **"hello.hex"**



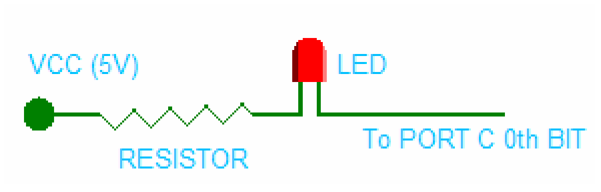
Finding the "hello.hex" file

Step II Programming the MCU with "hello.hex".

Now its time to burn this hex file to your MCU. To know how to burn the hex file to you MCU refer to you programmers manual. You can use **eXtreme electronics USB AVR Programmer** to burn hex files to you mcu(see http://shop.extremeelectronics.co.in). Detailed procedure is given in **"programmer"** folder support CD. Click here for tutorial on using our USB AVR Programmer.

Step III Electrical Connections.

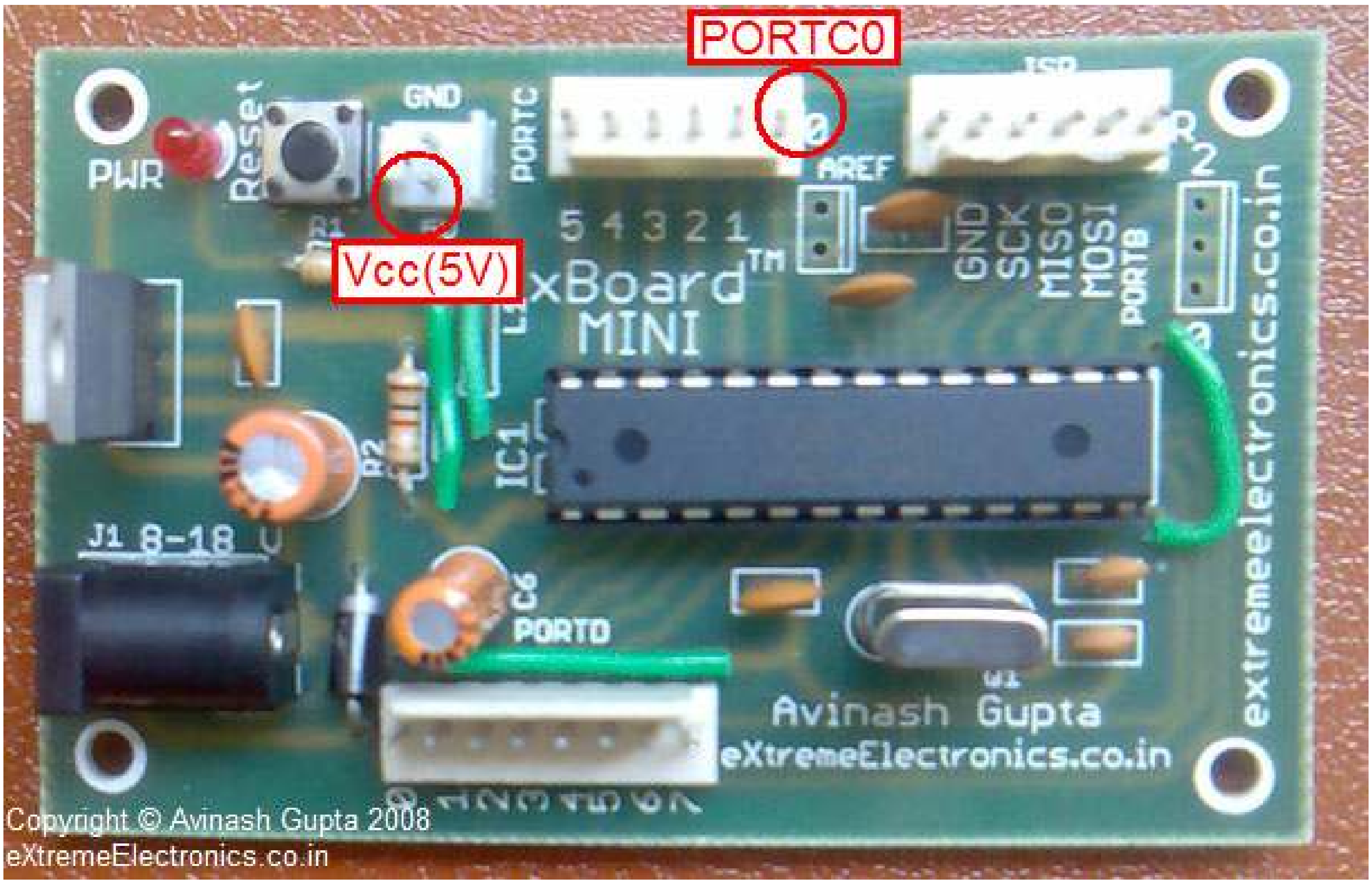
This is a very simple project you don't need any complicated connection. You may be using any of our development boards like **xBoard™, xBoard™ MINI** or simpler **xCards™.** You just need to figure out where is the MCU ports (which is programmed to blink LED, in this case PORT C'S 0TH BIT) connector on the board. They are clearly marked in the boards. After finding it, connect the LED as follows to the MCU port.



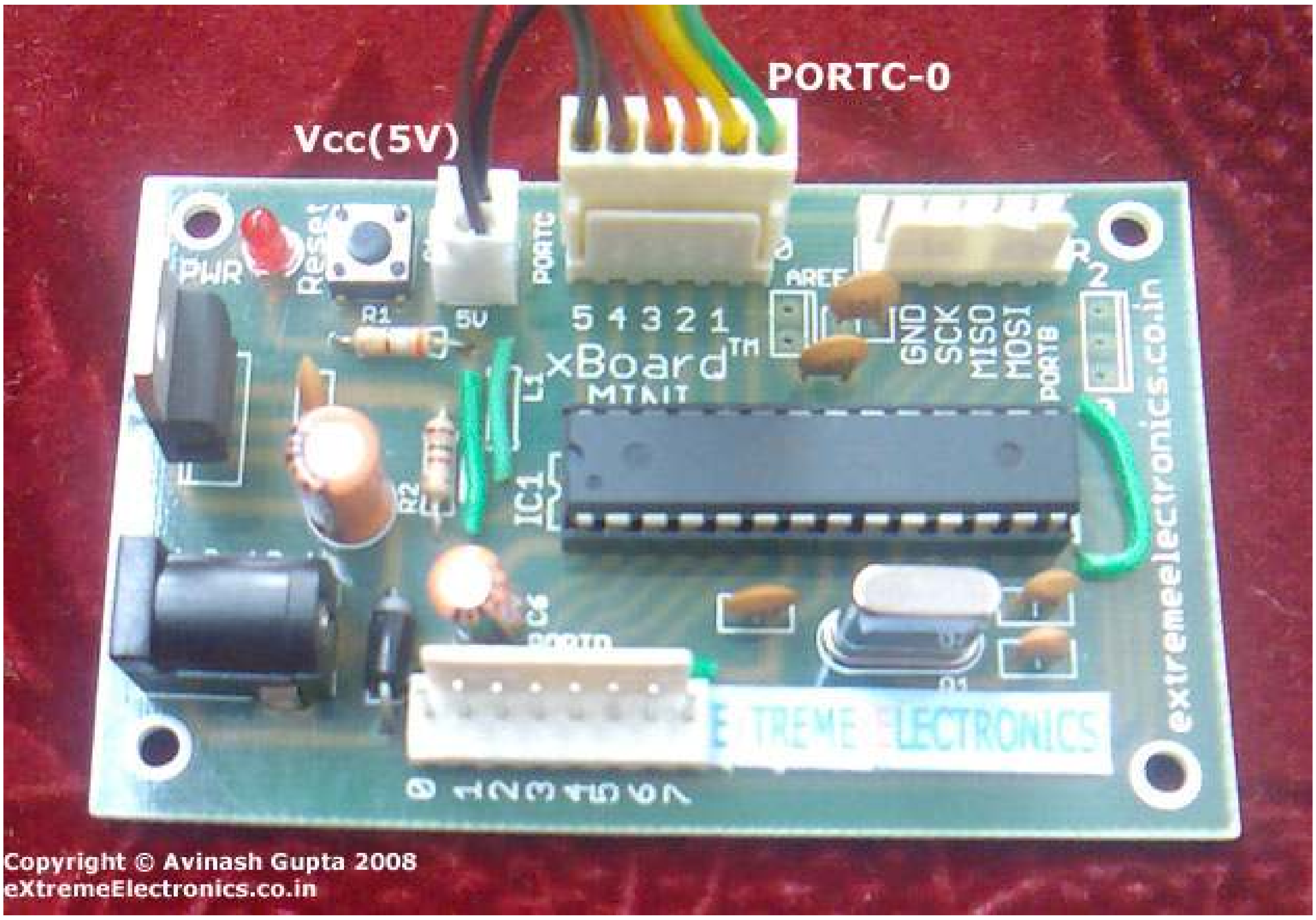
Connecting a LED

The VCC (5v) required is also available in the board and marked as "**spare 5V**" or something like that. After all connections connect the power adaptor to the board and switch on. The power indicator LED of board should glow and the LED connected to the MCU's port should blink on and off. Now you can relax and enjoy than light blinking with all that modern sophisticated technology behind it and you first dedicated embedded program doing all the magic. Now go on with your imagination tweak the program, modify delay and switching logic to get interesting patterns blinking. **The limit is your imagination.**

Pictures

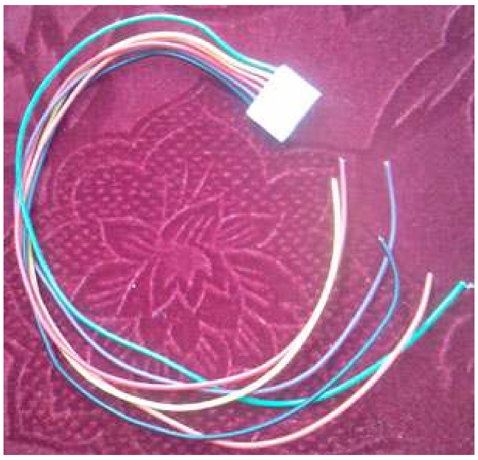


Finding Vcc and PORTC-0

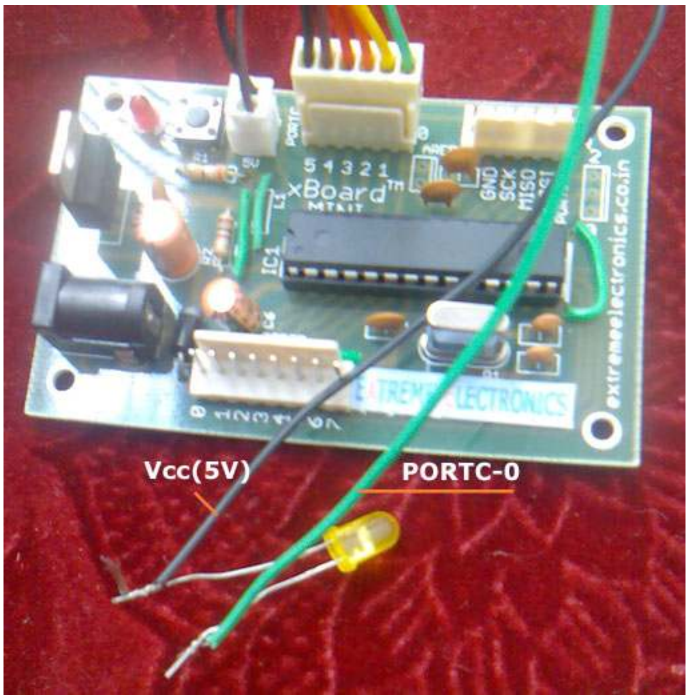


Connecting Connectors in PORTC and Spare Supply

Connector 2 PIN



Connector 6 PIN



Connecting the LED

Watch the LED Blinking Project Video !!!

Note: For More information on using AVR studio see AVR GCC Plugin Help in the Help Menu