

eXtreme Electronics

28 PIN AVR Development Board User Manual

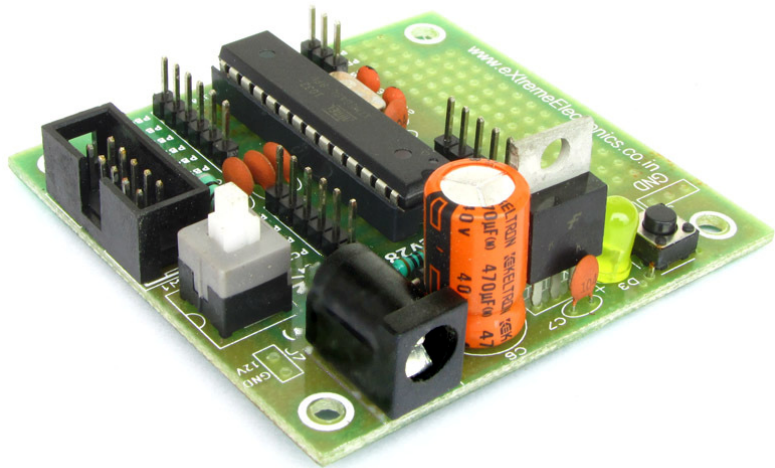


Table of Contents

Chapter 1. Introduction

- 1.1 Introduction
- 1.2 Know your development board

Chapter 2. Programming the Board

- 2.1 Programming the Board

Chapter 3. Development Process

- 3.1 Development Process
- 3.2 Software Tools
- 3.3 Entering and compiling code.
- 3.4 Electrical Connections.

28 PIN AVR Development Board User Manual

Introduction

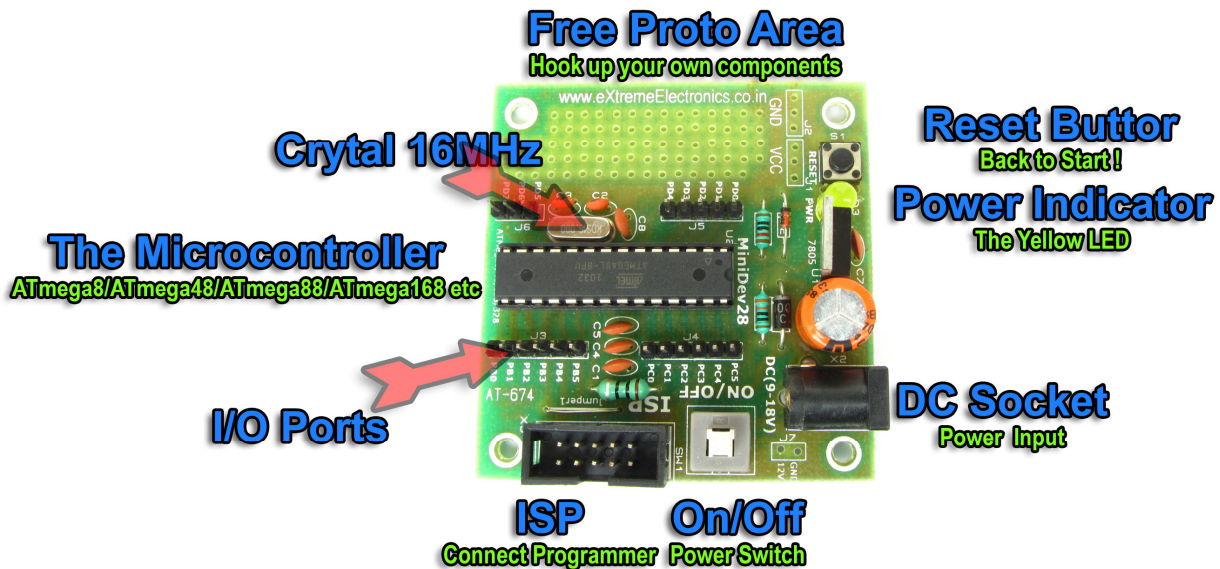
1.1 Introduction

28 PIN AVR Development Board helps user create and test applications using 28 PIN AVR MCUs. The board has the basic circuit required for the AVR to function. The board can be easily programmed using any in-circuit programmer (ISP). This features help you quickly update the program and test again and again during the program development phase. As you do not have to remove the microcontroller and put into the programmer each time. This greatly saves time and damages to MCU's pins.

You can power the board using standard 12V adaptor. The board has inbuilt regulator to provide clean 5v to the MCU. There is also a on/off switch and power indicator LED to further ease your life.

1.2 Know your development board.

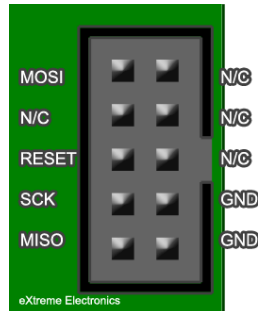
The image below show the parts on the board.



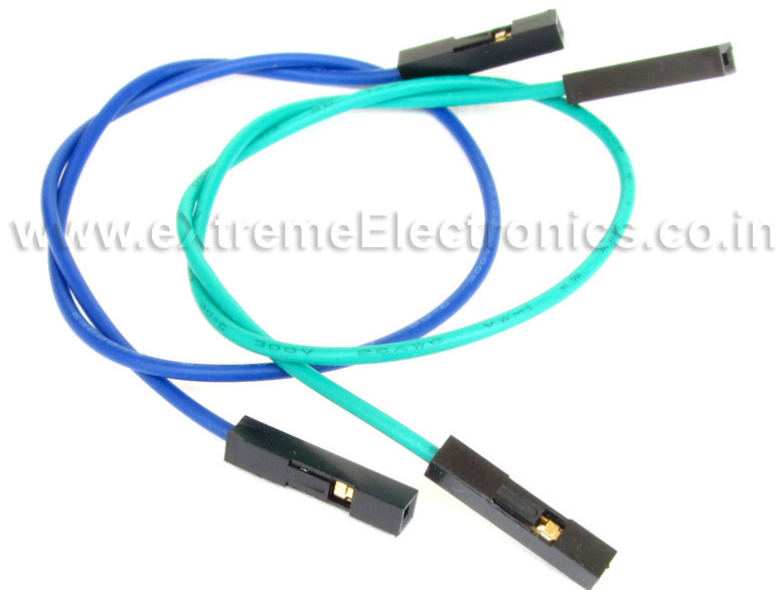
28 PIN AVR Development Board Overview

28 PIN AVR Development Board User Manual

- **Power In :** Standard DC socket for powering the board. Use a 12v 500ma adaptor with Center Positive pin.
- **On/Off Switch:** Main Power Control, used to turn on/off the entire board.
- **ISP:** This a 10 pin standard Atmel ISP. Used to connect with any ISP programmer for program downloading.



- **Power Indicator:** This LED indicates the power status of board.
- **Reset Button:** Press this to reset the MCU. The program will start to execute from the beginning.
- **Free Proto Area:** Here you can add more components to the board as required by your application. You can then connect them to the i/o ports. The Vcc and GND are available near by to power your extra devices. For example you may connect LEDs, Motor Driver etc in this free area.
- **I/O Ports:** The i/o ports of MCU is available in standard 0.1inch male headers. You may use single PIN female to female burg wires to connect these i/o ports with other modules.

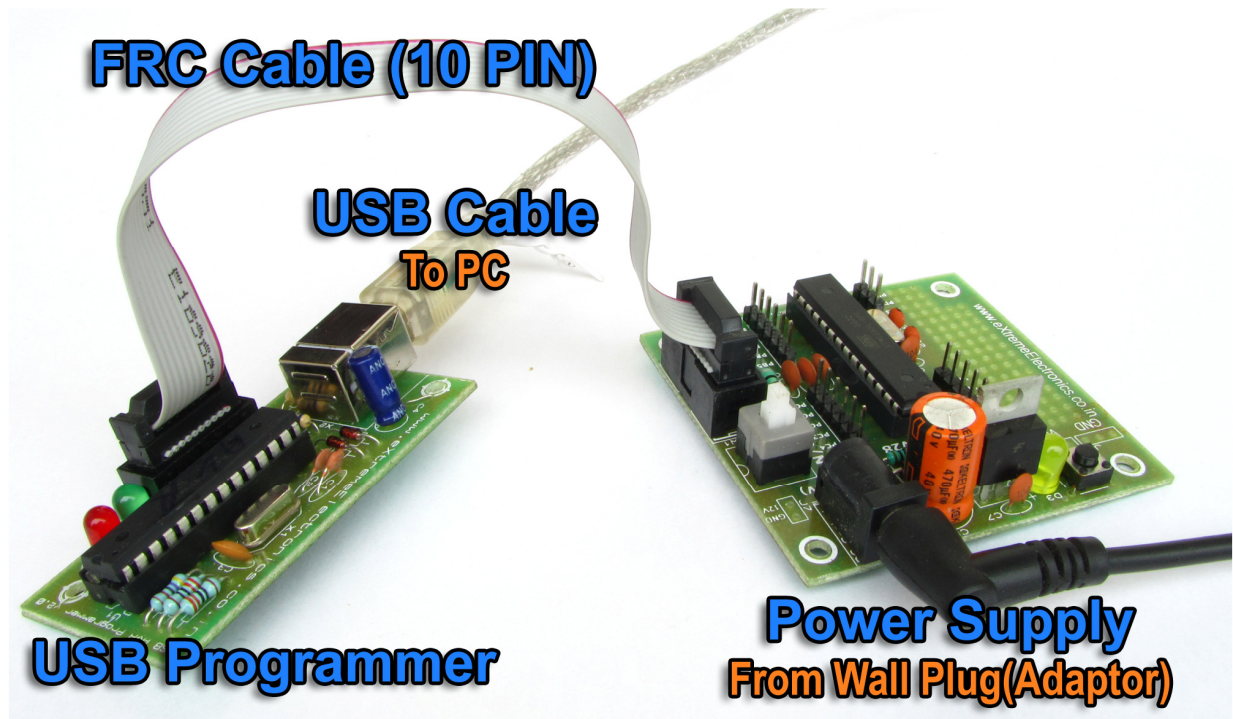


2.1 Programming the Board.

The devboard can be easily programmed by an ISP enabled programmer like **eXtreme Burner - AVR(USB)**.

Our programmer comes with the software which can be easily installed with a single setup file. The drives are built into the software so you do NOT need to install it separately.

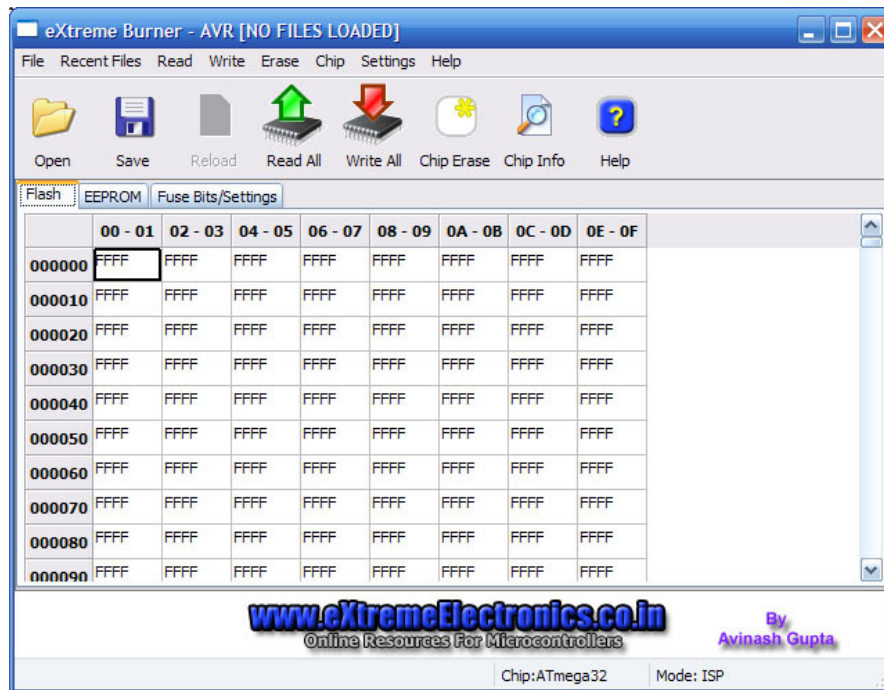
Connect the programmer with your PC using a USB Cable. The GREEN(Power LED) on the programmer should glow. Then connect the devboard with the Programmer by using an ISP Cable (10 PIN FRC Cable). **Power up the devboard.**



Programming Arrangement.

Launch the [eXtreme Burner AVR](#) from the Desktop Icon. You will get a screen like this.

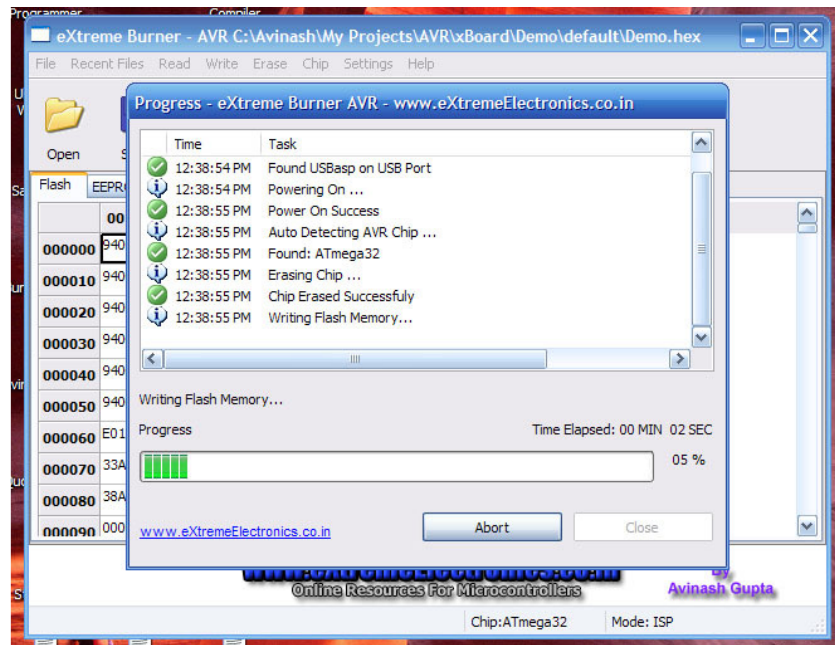
28 PIN AVR Development Board User Manual



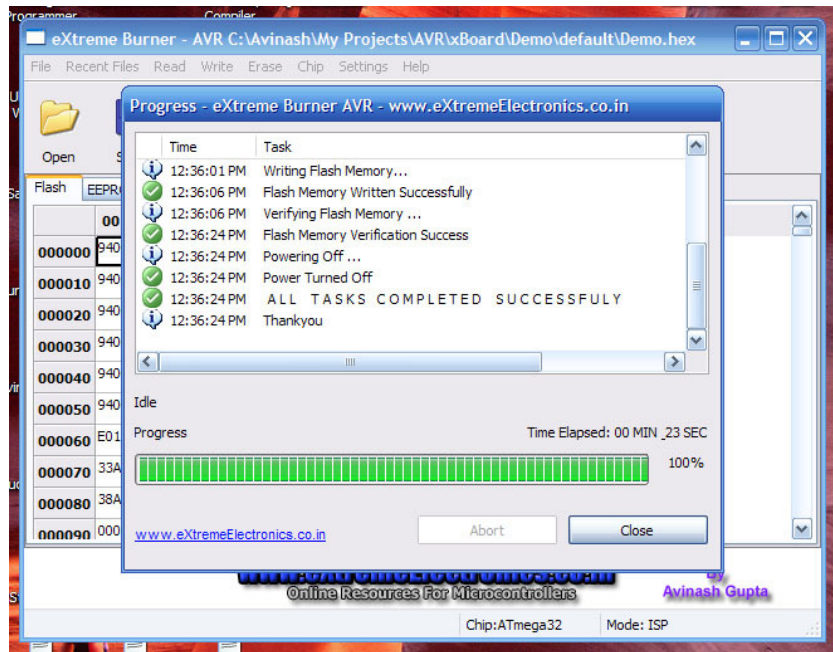
eXtreme Burner - AVR, Main Screen

Click **File->Open Flash** from the menu or **Open** from the toolbar. Select the hex files you wish to program.

After this select proper microcontroller using the **Chip** Menu. Click **Write->Flash**.



eXtreme Burner - AVR, Burn Progress.



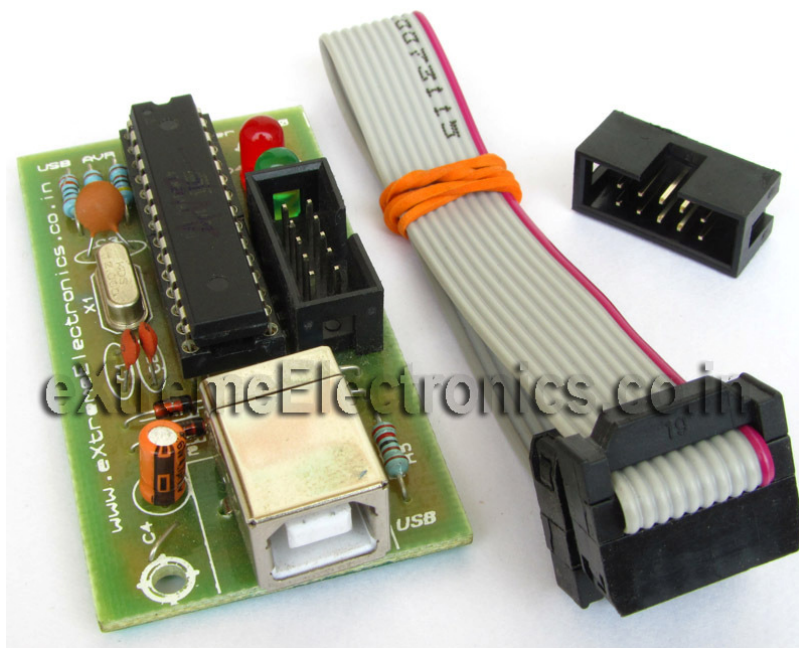
eXtreme Burner - AVR, Write Success.

If everything is ok you will get a success message. If you get some error please consult the manual of the programmer.

3.1 Development Process.

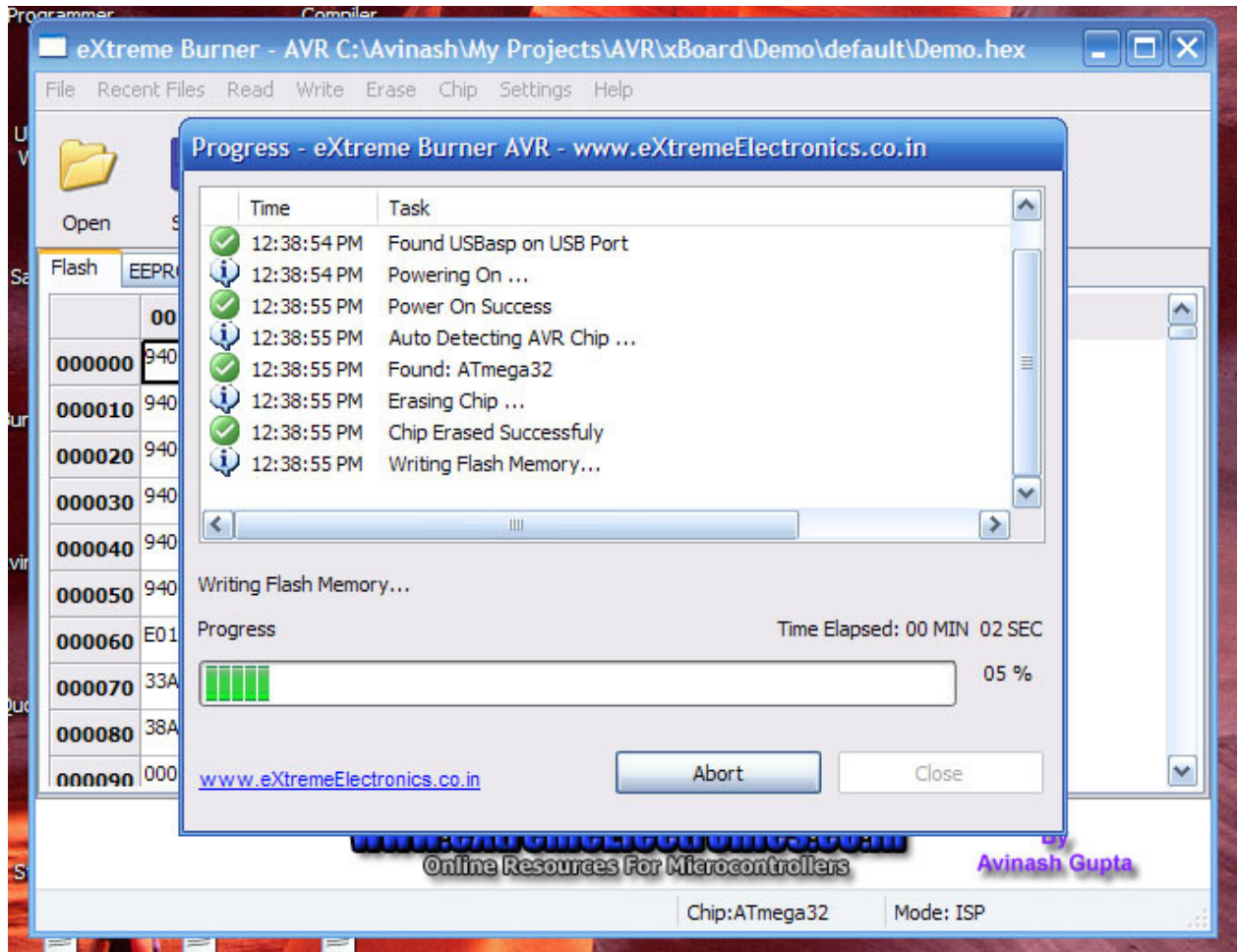
The development process with AVR is outlined below.

1. First you should know what you are trying to make. You should have some algorithm in your mind and clue on how the program should look. Or else you may have a ready made program, for example from a Website or Magazine. The human readable code is written in some high level Language like C,C++, BASIC etc. This code is called **Source Code**. We will use C language through out these tutorials.
2. You need a Text Edited to enter the **Source Code** and save in a computer file so these can be processed by computer software.
3. Then a **Compiler**, which is a software that input a Source code in a syntax of some programming language (in our case C) and generates a machine instructions for a specific machine(AVR MCU in our case). In Microcontroller world these file are called **HEX files**(the extension is .HEX eg. myprogram.hex)
4. After this we load this HEX file in a **programmer software**. We connect our programmer with PC and then Connect the programmer with devboard. The programmer software then uploads this file to the board, then the board will execute our program.



ISP Programmer used To Upload a HEX file.

28 PIN AVR Development Board User Manual



ISP Programmer used To Upload a HEX file.

3.2 Software Tools

The main software you will need are:

- **AVR Studio** – A GUI IDE for AVR. IDE or Intergated Development Environment help us to Enter, Edit, Complie and Debug a program from a single software interface. This also help in project based development, where more than one source file makes a complete project. Different source files contains code logically divided by the function they perform.
- **WinAVR** – a free C compiler for AVR's. This will convert a source program written in C language to a machine code that can be understood and executed by a machine (AVR MCU in our case).

We don't have to deal with a compiler manually. AVR Studio will take care of it.

- **eXtreme Burner - AVR** – This will upload the HEX file generated by the IDE/Compiler to the microcontroller.

You can get all the software from the folder named "Software" in the Support Disc. 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.

3.3 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.

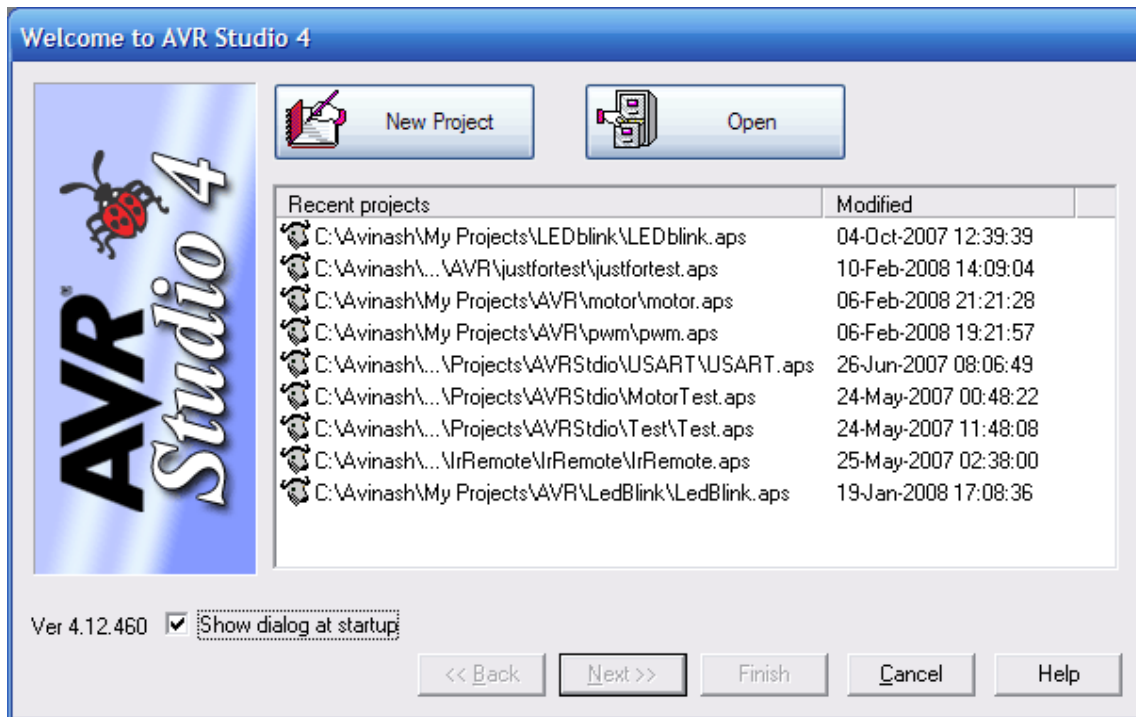


Fig - AVR Studio Project Wizard

28 PIN AVR Development Board User Manual

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.

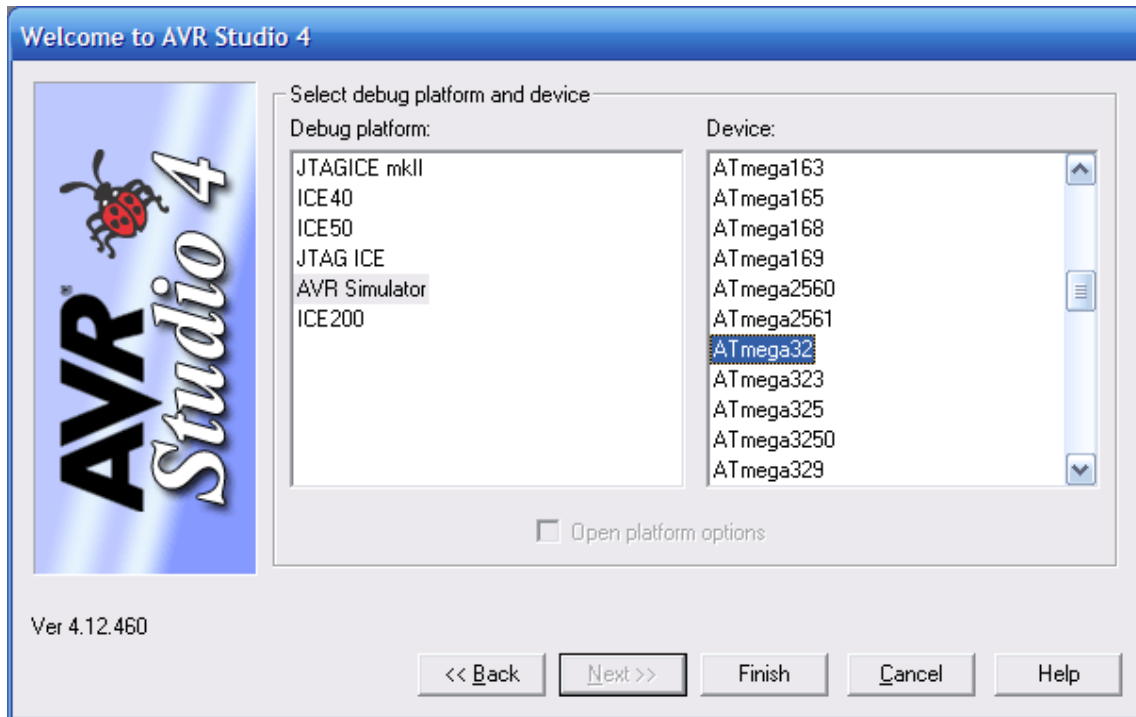


Fig - Device Selection

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. Click finish. After that you will be presented with an Integrated Development Environment-IDE. As shown below.

28 PIN AVR Development Board User Manual

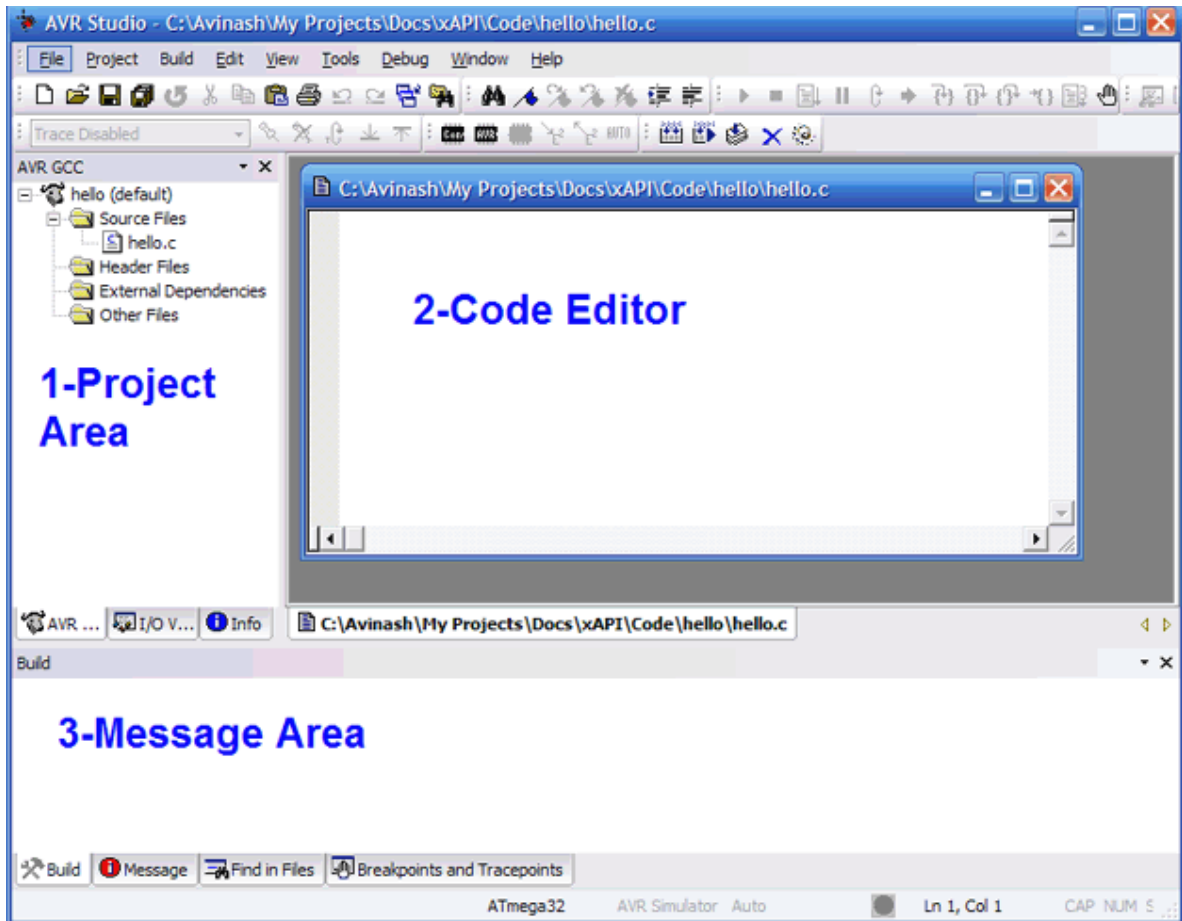


Fig - AVR Studio main window

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.

28 PIN AVR Development Board User Manual

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

void Wait()
{
    uint8_t i=0;
    for(;i<23;i++)
        _delay_loop_2(0);
}

void main()
{
    //Set PORTB1 as output

    DDRB=0b00000010;

    while(1)
    {
        //Set PB1=High(+5v)
        PORTB|=0b00000010;

        Wait();

        //Set PB1=Low(GND)
        PORTB&=0b11111101;

        Wait();
    }
}
```

If you don't understand the program don't worry, the next tutorial will teach you how to [setup and use general purpose IO ports in AVR](#).

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

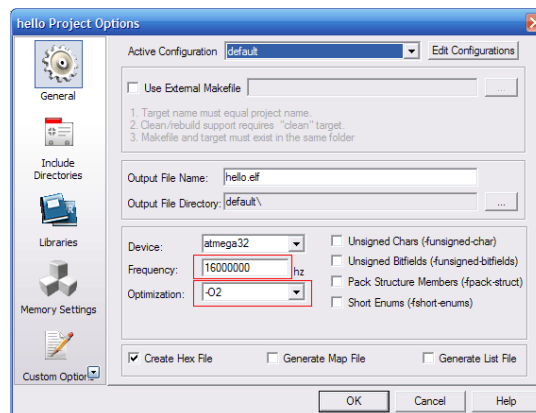


Fig - Setting CPU frequency and compiler optimization

28 PIN AVR Development Board User Manual

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.

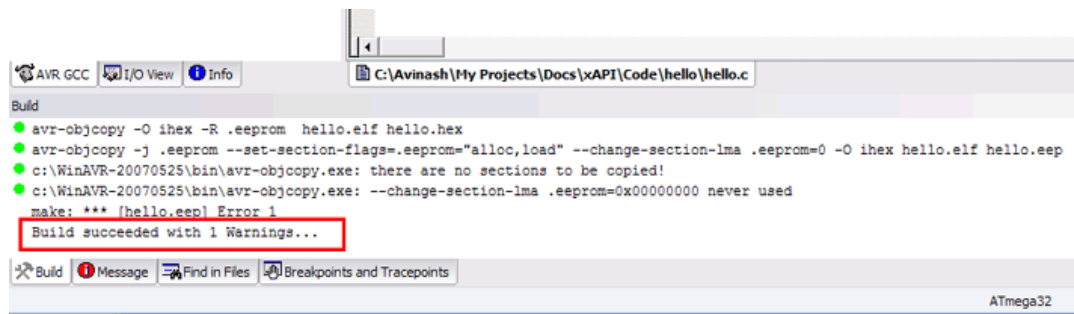
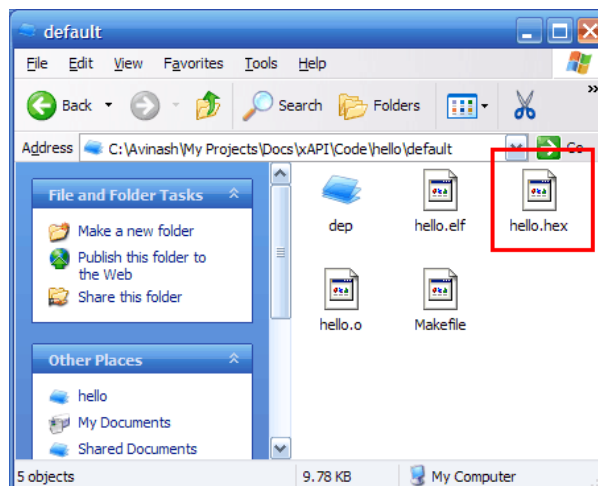


Fig - Message

“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 is a **“.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”**



3.4 Electrical Connections.

Connect a LED on PB1 (PortB bit 1) via a 330ohm resistor to GND. That means connect the anode (+) of LED to PB1 and the cathode (-) of led to R and the other lead of R to GND.

Now you can relax and enjoy that 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.

Note

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