USB ASP Programmer & Atmega Programming – Tutorial #3

T.K. HAREENDRAN

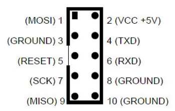
AVR tutorial

Share this:

Share More

It’s assumed that you have procured one USBasp Programmer from your trusted component vendor! Connecting the programmer to your computer comprises of two steps. First step is the physical connection of the programmer to the USB port using a suitable USB cable, and the second step is the installation of device drivers in order for it to work. After making the physical connection, go through the manufacture’s instruction manual to install device drivers for the USBasp Programmer. This tutorial presumes that you have sufficient privileges to install device drivers in Windows Operating Systems (whilst the USBasp programmer will work on a wide variety of operating systems, my preference is Windows 7-64bit).

In my new USBasp programmer (**http://www.robomart.com/avr-usb-programmer.html**), “6-pin ISP” connection provides the interface to the micro controller. That’s why I am using a 6-pin male header with my breadboard. However if you are using an odd model programmer in which “10-pin ISP” connection provides the interface, carefully re-wire the breadboard section to lodge a 0-pin connection. In this case it is required to make two more connections (from Pin 2 (RxD) and Pin 3 (TxD) of the Atmega8 IC) in the breadboard. 10-pin ISP connection pinout is shown below.



Atmega Programming

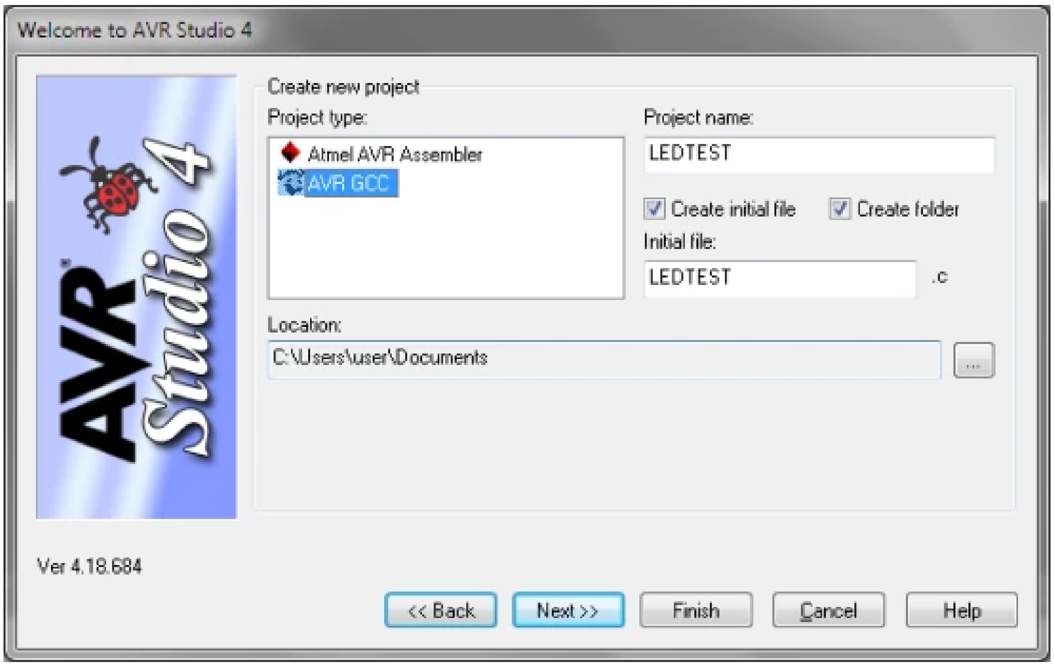
Kudos! Now the USBasp programmer and the programming adaptor are ready for use. Next step is the preparation of the first AVR C program. AVR Studio provides an IDE for writing, debugging, and simulating programs. We will use the WinAVR GCC C compiler toolset with AVR Studio via plug-in module.

You can find these at:

- AVRStudio: http://www.atmel.in/System/BaseForm.aspx?target=tcm:26-41049
- WinAVR: http://sourceforge.net/projects/winavr/files/latest/download?source=files

Install WinAVR first and AVR Studio second (use the default locations so AVRStudio can find WinAVR).

- Click on the AVR Studio desktop icon. It opens with “Welcome to AVR Studio 4”. Click on the “New Project” button
- In the “Create new project” window,click on “AVR GCC”,add the “Project name”: “LEDTEST” and set the “Location” to a convenient spot, then click “Next”


Welcome to AVR Studio 4

Create new project

Project type:
Atmel AVR Assembler
AVR GCC

Project name: LEDTEST

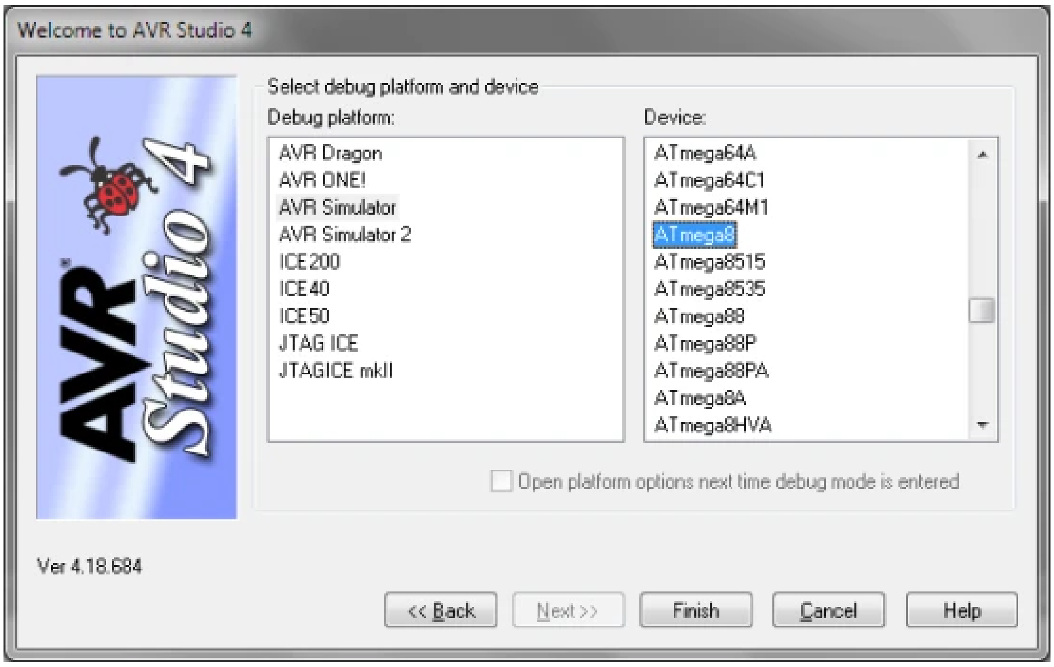
[✓] Create initial file [✓] Create folder

Initial file: LEDTEST .c

Location: C:\Users\user\Documents

Ver 4.18.684

- Choose “AVR simulator” as “Debug” Platform and “Atmega8” as device in device list, then click “Finish”


Welcome to AVR Studio 4

Select debug platform and device

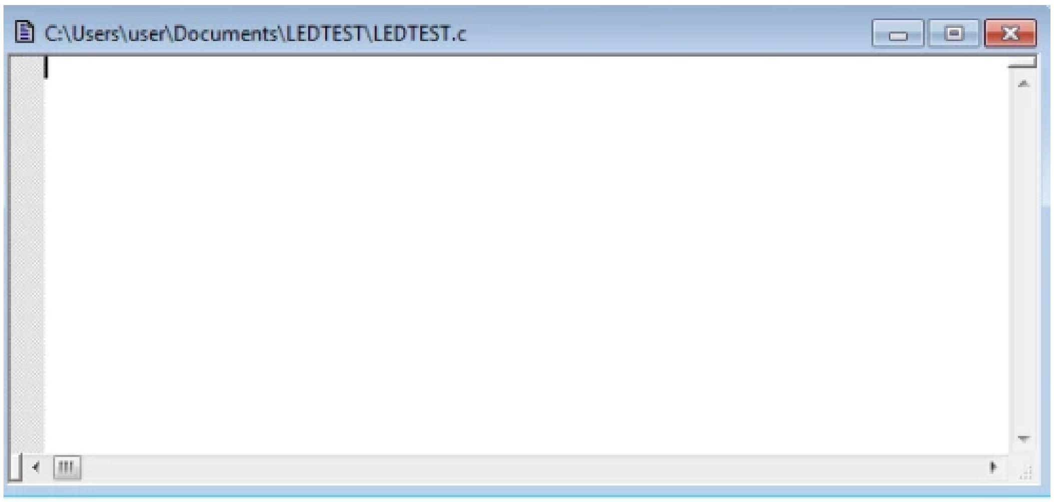
Debug platform:
AVR Dragon
AVR ONE!
AVR Simulator
AVR Simulator 2
ICE200
ICE40
ICE50
JTAG ICE
JTAGICE mkII

Device:
ATmega64A
ATmega64C1
ATmega64M1
ATmega8
ATmega8515
ATmega8535
ATmega88
ATmega88P
ATmega88PA
ATmega8A
ATmega8HVA

[ ] Open platform options next time debug mode is entered

Ver 4.18.684

- Now the project is ready for writing the program. If everything correct, you must be at this screen (infact this screen is a complex IDE with lots of tools – only code window is shown here)



Now write this code in centre code window, compile the program by clicking on "Build" button (or by pressing shortcut key F7)

```
#define F_CPU 1000000UL
#include
#include
int main(void) {
   //Set the Data Direction Register to output
   DDRC |= (1<<5);
   while (1) {
      //Set the signal to high
      PORTC |= (1<<5);
      //wait 0.5 sec
      _delay_ms(500);
      //Set the signal to low
      PORTC &= ~(1<<5);
      //wait 0.5 sec
      _delay_ms(500);
   }
}
```

If everything OK, you should get a message in the "Build" window.

```
(.text + .data + .bootloader)
Data: 0 bytes (0.0% Full)
(.data + .bss + .noinit)
Build succeeded with 0 Warnings...
```

Now go to the project folder to find a folder named " Default" inside it. Open the folder to view a file "LEDTEST.hex". Open this file to view the hex code (hexadecimal machine code of the program). Now you only need to burn this code into the flash memory of Atmega8 fitted in the breadboard! But How?