Getting started with C Programming for the ATMEL AVR Microcontroller

By Lam Phung

Version 1.0

Created on May 14, 2008. Last updated January 15, 2010.

Latest version of this document is available at: http://www.elec.uow.edu.au/avr

Table of Contents

1. Introduction

This tutorial provides information on the tools and the basic steps that are involved in using the C programming language for the Atmel AVR microcontrollers. It is aimed at people who are new to this family of microcontrollers. The Atmel STK500 development board and the ATMEGA16 chip are used in this tutorial; however, it is easy to adopt the information given here for other AVR chips.

This tutorial requires the following:

- the AVR Studio produced by Atmel,
- the WinAVR package by Sourgeforge WinAVR project, and
- an STK500 development board produced by Atmel.

2. Installing tools for C programming

To work with the Atmel AVR microcontroller using the C programming language, you will need two tools: *AVR Studio* and *WinAVR*. Both tools are free at the links given below.

- AVR Studio is an integrated development environment that includes an editor, the assembler, HEX file downloader and a microcontroller emulator. AVR Studio setup file and service packs are available at

  http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

- WinAVR is for a GCC-based compiler for AVR. It appears in AVR Studio as a plug-in. WinAVR also includes a program called Programmer's Notepad that can be used to edit and compile C programs, independently of AVR Studio. WinAVR setup file is available at

  http://winavr.sourceforge.net/

Installing these tools is easy: just download and run the setup files, and accept the default installation options. Remember to install AVR Studio first before WinAVR.

3. Using AVR Studio for C programming

As an example, we will create a simple C program for the Atmel AVR that allows the user to turn on one of the eight Light Emitting Diodes (LEDs) on the STK500 development board, by pressing a switch. Next, you will be guided through four major stages:

- creating an AVR Studio project,
- compiling C code to HEX file,
- debugging C program using the simulator,
- downloading HEX file to the STK500 development board and running it.

3.1 Creating an AVR Studio project

Perform the following steps to create a simple AVR Studio project.

- Start the AVR Studio program by selecting **Start | Programs | Atmel AVR Tools | AVR Studio**.

- Select menu **Project | New Project**. In the dialog box that appears (see Figure 1), select AVR GCC as project type, and specify the project name and project location. If options 'Create initial file' and 'Create folder' are selected, an empty C file and containing folder will be created for you. In this case, we create a project called 'led'. Click button **Next** when you are ready.

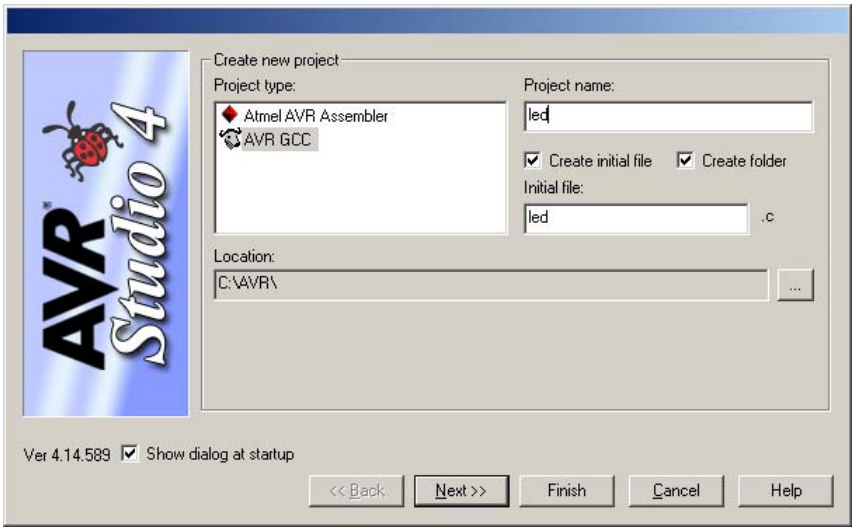


Figure 1: Entering project type, name and location.

- In the 'Select debug platform and device' dialog that appears (see Figure 2), choose 'AVR Simulator' as the debug platform and 'ATMEGA16' as the device. Click button **Finish**.

  Note: If you want to use other AVR chips such as ATMAGE8515, select it at this step. In this tutorial, we will use ATMEGA16 for both software simulation and hardware testing.

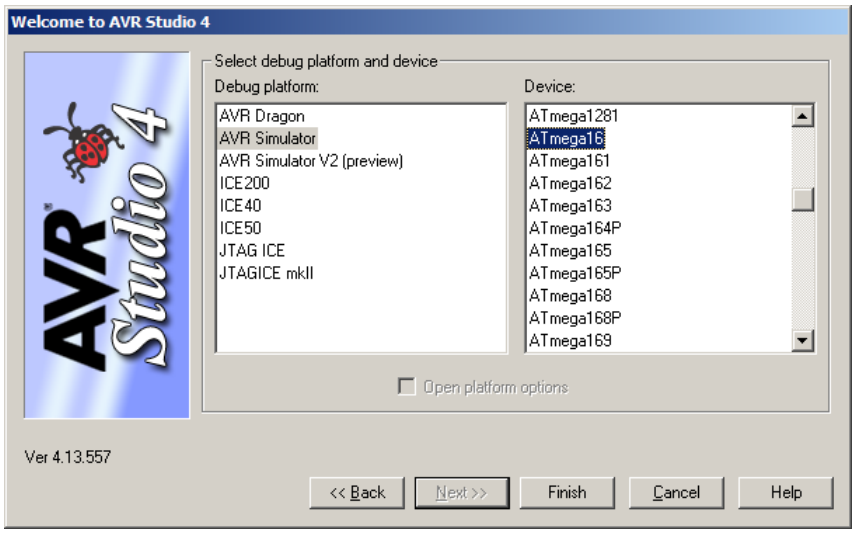


Figure 2: Selecting debug platform and device.

- A project file will be created and AVR Studio displays an empty file led.c (see Figure 3). Enter the C code shown in Figure 4. It is not important to understand the code at this stage, but you can do that by reading the C comments.

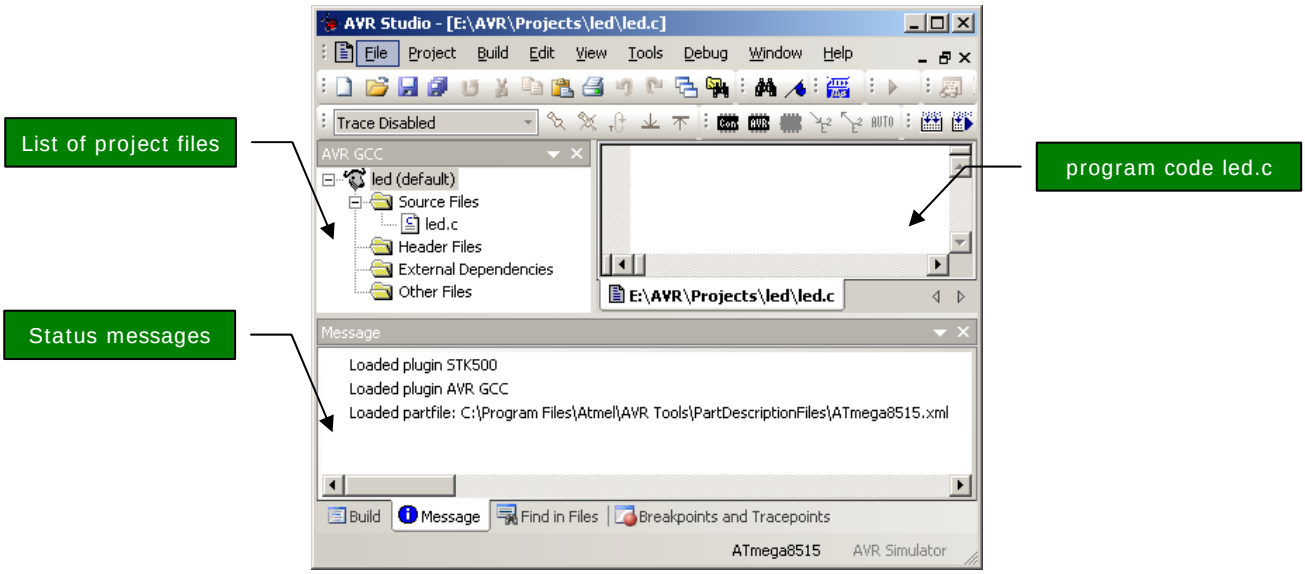


Figure 3: The AVR Studio with a project file open.

- Click menu **Project | Save Project** to save the project file and the C program. AVR Studio project files have extension ‘aps’.

```
// File:          led.c
// Description:   Simple C program for the ATMEL AVR uC (ATMEGA16 or ATMEGA8515 chip)
// This program lets the user turn on LEDs by pressing the switches on STK500 board
// Date modified: 13 May 2008

#include <avr/io.h>     // avr header file for IO ports
int main(void){
       unsigned char i; // temporary variable

       DDRA = 0x00;    // set PORTA for input
       DDRB = 0xFF;    // set PORTB for output

       PORTB = 0x00;  // turn ON all LEDs initially

       while(1){
              // Read input from PORTA.
              // This port will be connected to the 8 switches
              i = PINA;

              // Send output to PORTB.
              // This port will be connected to the 8 LEDs
              PORTB = i;
       }
       return 1;
}
```

Figure 4: Program code led.c

3.2 Compiling C code to HEX file

- Click menu **Build | Rebuild All** to compile the C code.
- If there is no error message, a file called led.hex will be produced (see Figure 5). This file contains the machine code that is ready to be downloaded to the ATMEGA16 microcontroller. The file is stored in sub-folder '\default' of your project.
- If there are error messages, check your C code. Most often, they are caused by some typos or syntax errors.

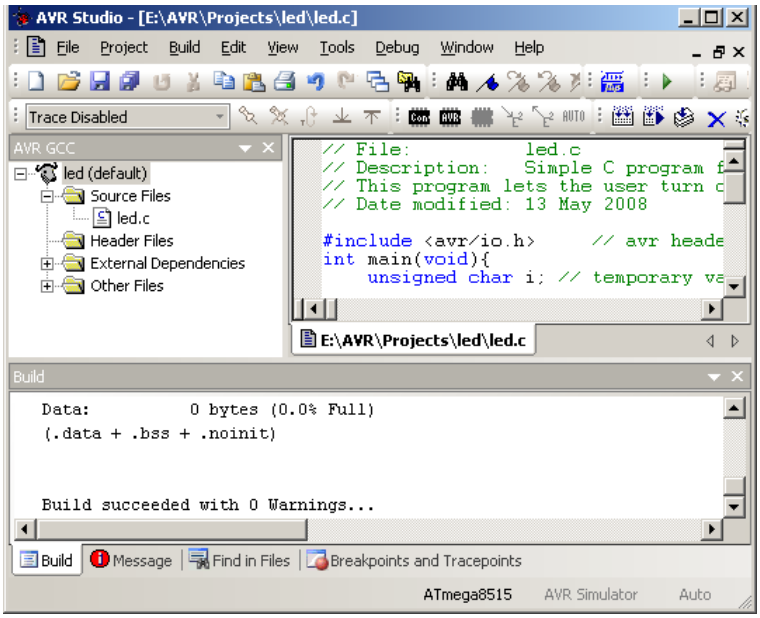


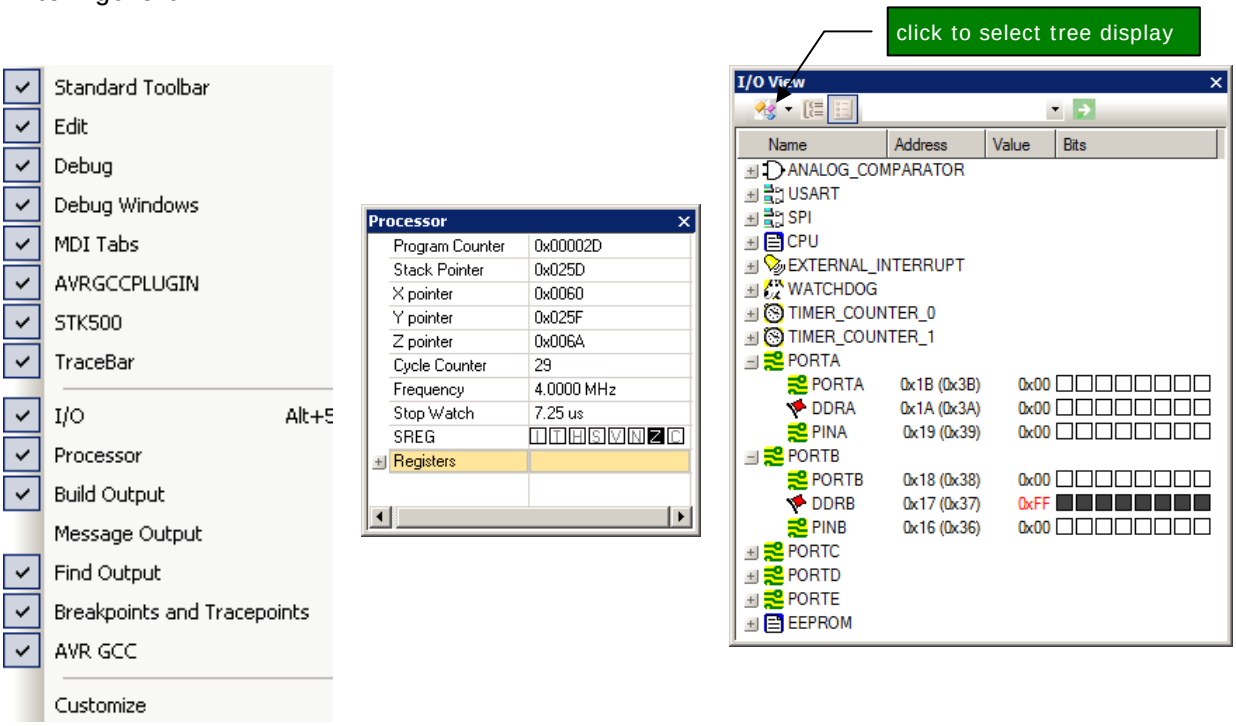
Figure 5: Selecting menu Build | Rebuild All to create HEX file.

3.3 Debugging C program using the simulator

Debugging is an essential aspect in any type of programming. This section will show you how to debug a C program at source-code level, using AVR Studio. You can execute a C program one line at a time, and observe the effects on the CPU registers, IO ports and memory. This is possible because AVR Studio provides a simulator for many AVR microcontrollers, including the ATMEGA16 and ATMEGA8515. Therefore, this debugging does not require the STK500 kit.

We will continue with the example project led.aps created in Section 3.2 of this tutorial.

- AVR Studio lets you examine the contents of CPU registers and IO ports. To enable these views, right click on the menu bar at the top and select 'I/O' and 'Processor' options. Refer to Figure 6.



(a) Selecting views (b) Processor view (c) IO view

Figure 6: Debugging views.

- Select menu **Debug | Start Debugging**. A yellow arrow will appear in the code window (Figure 7); it indicates the C instruction to be executed next.

- Select menu **Debug | Step Into** (or press hot-key F11) to execute the C instruction at the yellow arrow. Figure 6c shows the IO view after the following C instruction is executed:

```
DDRB = 0xFF;     // set PORTB for output
```

We can see that Port B Data Direction Register (DDRB) has been changed to 0xFF.

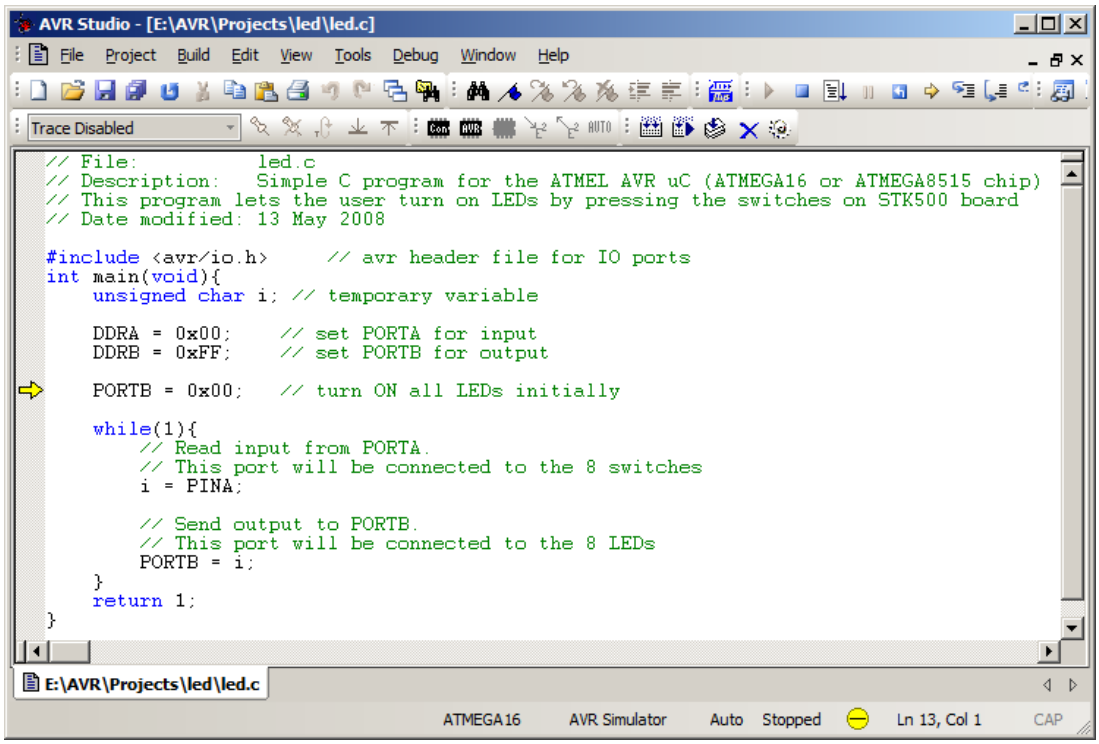
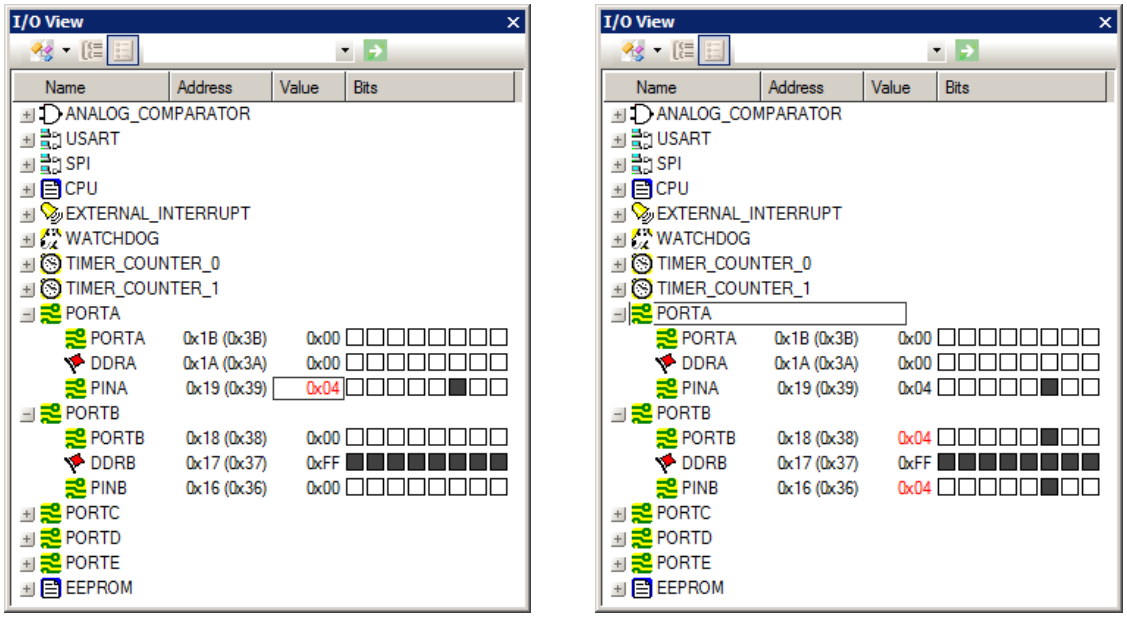


Figure 7: Stepping through a C program in debugging mode.

- While debugging the C program, you can change the contents of a register. For example, to change Port A Input Pins register (PINA), click on the value column of PINA and enter a new value (Figure 8a). This change takes effect immediately. Subsequently, the contents of PORTB will be 0x04 (see Figure 8b) after running the two C instructions:

```
i = PINA;
PORTB = i;
```



(a) changing PINA register to 0x04

(b) effects on PORTB after running `i = PINA; PORTB = i;`

Figure 8: Modifying registers manually.

- To monitor a C variable, select the variable name in the code window and click menu **Debug | Quick Watch**. The variable will be added to a watch window, as in Figure 9.

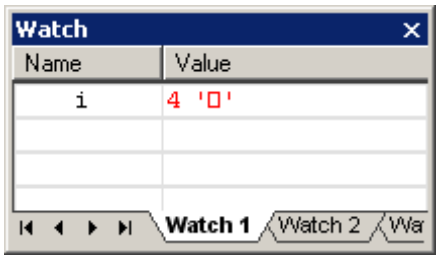


Figure 9: Watch window for C variables.

- Many other debugging options are available in the Debug menu, such as running up to a break point or stepping over a function or a loop. To view the assembly code along with the C code, select menu **View | Disassembler**.

3.4 Downloading and running HEX file on AVR board

To perform the steps in this section, you will need a STK500 development board from Atmel. The STK500 kit includes two AVR microcontroller chips: ATMEGA8515 and ATMEGA16.

- The ATMEGA8515 is installed on the development board by the manufacturer.
- The ATMEGA16 is installed on all development boards in SECTE laboratories.

Note:

- To install the ATmega16 chip, simply use a chip extractor tool to remove the existing ATMEGA8515 from its socket. Then place the ATMEGA16 in **socket SCKT3100A3**.
- If you use other AVR chips such as ATMEGA128, refer to Table 3.2 AVR Sockets, 'AVR STK500 User Guide' for the exact socket.

Hardware setup

Refer to Figure 10 when carrying out the following steps for hardware setup.

- Connect the SPRO3G jumper to the ISP6PIN jumper, using the supplied cable in the STK500 kit. This is needed to program the ATMEGA16 chip.

- Connect the board with the PC using a serial cable. Note that the STK500C has two RS232 connectors; we use only the connector marked with RS232 CTRL.

- Connect the SWITCHES jumper to PORTA jumper. This step is needed in our example because we want to connect 8 switches on the development board to port A of the microcontroller.

- Connect the LEDS jumper to PORTB jumper. This step is needed in our example because we want to connect 8 LEDs on the development board to port B of the microcontroller.

- Connect the board with 12V DC power supply and turn the power switch ON.

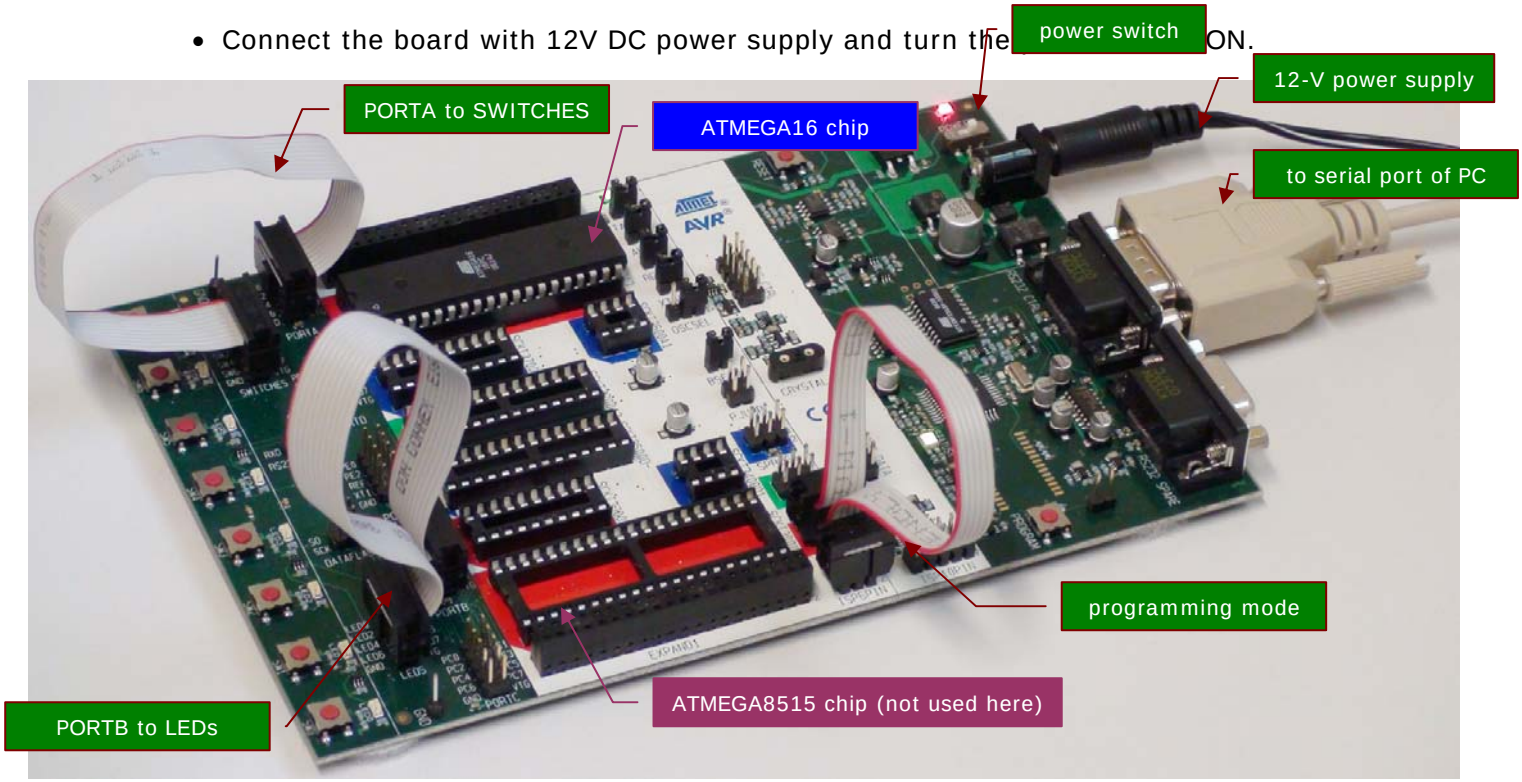


Figure 10: Setting up the STK500 for downloading and testing.

Downloading and running HEX file

- In AVR Studio, select menu **Tools | Program AVR | Connect**.
- In the 'Select AVR Programmer' dialog box, choose 'STK500 or AVRISP' as the platform and 'Auto' as Port (see Figure 11). Then click button **Connect**.

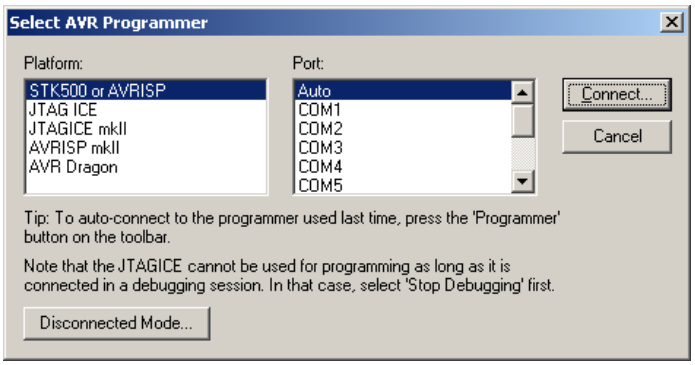


Figure 11: Selecting AVR programmer.

- Depending on the version of your AVR Studio, a message about firmware *may* appear. For now, this message can be discarded by clicking button **Cancel**. In the future, you may want to read this message carefully and perform the steps described there to perform firmware update.

- In the 'STK500' dialog box that appears, select led.hex as 'Input Hext File'. Then, click button **Program** to download the HEX file to the AVR chip (Figure 12).

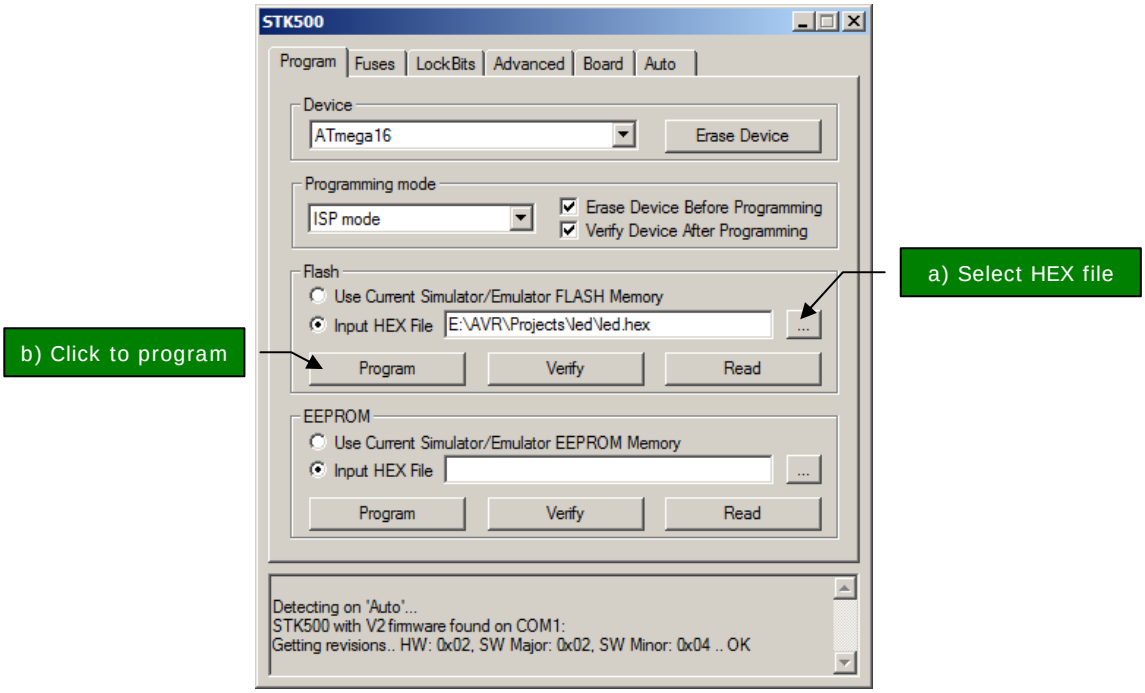


Figure 12: Selecting AVR programmer.

- The program will now run on the microcontroller. If you press and hold down one of the 8 switches on the development board, the corresponding LED will be turned on.

  A MPEG-4 video demo of the program is available at
  http://www.elec.uow.edu.au/avr/getdoc.php?doc=ecte333/lab07_task123.mp4

This is the end of this introductory tutorial. More in-depth information about programming Atmel AVR microcontrollers for embedded applications is provided in ECTE333 Digital Hardware course at the School of Electrical, Computer and Telecommunication Engineering, University of Wollongong, and also at our website http://www.elec.uow.edu.au/avr.

*** END ***