Getting Started with C Programming for the ATMEL AVR Microcontrollers

By Son Lam Phung

Version 2.0

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 tool and the basic steps for programming the Atmel AVR microcontrollers using C. 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.

2. Installing tool for C programming

To program Atmel AVR microcontrollers using C, you will need Atmel Studio software, which is freely available from the company website. Atmel Studio is an integrated development environment that includes the editor, C compiler, assembler, HEX file downloader, and a microcontroller emulator.

To install Atmel Studio, perform the following steps:

- Download setup files for Atmel Studio 6 from ATMEL (about 820MB):
  http://www.atmel.com/microsite/atmel_studio6/
- Download Microsoft Visual Studio 10 Service Pack 1 from Microsoft (about 1.6GB):
  http://www.microsoft.com/en-us/download/details.aspx?id=23691
- Run the setup file for Atmel Studio 6. Accept the default options.
- Run the setup file for Microsoft Visual Studio 10 Service Pack 1. Accept the default options.

3. Using Atmel 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 Atmel Studio project,
- compiling C code to produce a HEX file,
- debugging C program using the simulator,
- downloading HEX file to the STK500 development board and running it.

3.1 Creating an Atmel Studio project

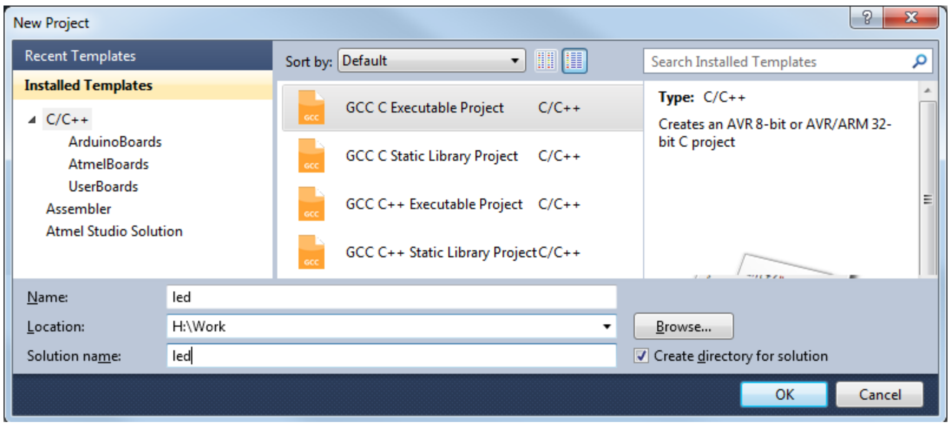
Perform the following steps to create a simple Atmel Studio project.

- Start the Atmel Studio 6 program by clicking its icon on the Windows Desktop.

- Select menu **File** | **New Project**. In the dialog box that appears (see Figure 1), select 'GCC C Executable Project', and specify the project name and project location.

  Select the option 'Create directory for solution' so that a folder will be created to store. In this case, we use 'led' as both the project name and the solution name[1].

  Click button **OK**.



New Project

Recent Templates
Installed Templates
C/C++
ArduinoBoards
AtmelBoards
UserBoards
Assembler
Atmel Studio Solution

Sort by: Default

GCC C Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++

Search Installed Templates

Type: C/C++
Creates an AVR 8-bit or AVR/ARM 32-bit C project

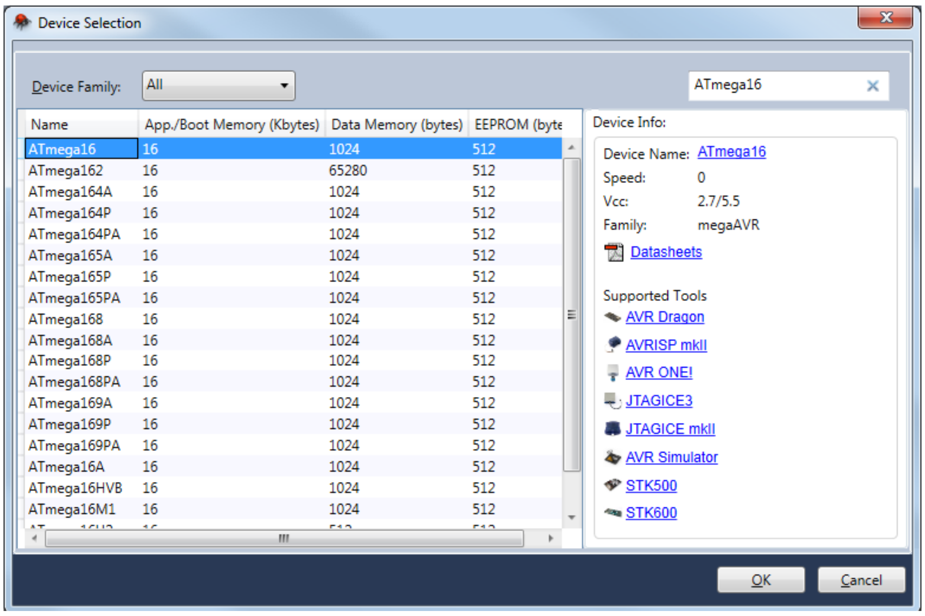
Name: led
Location: H:\Work
Solution name: led
Browse...
[✓] Create directory for solution
OK
Cancel

Figure 1: Entering project type, name and location.

- In the 'Device Selection' dialog that appears (see Figure 2), search for ATmega16 and then click button **OK**.

[1] In Atmel Studio 6, a *solution* may contain several *projects*.

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.



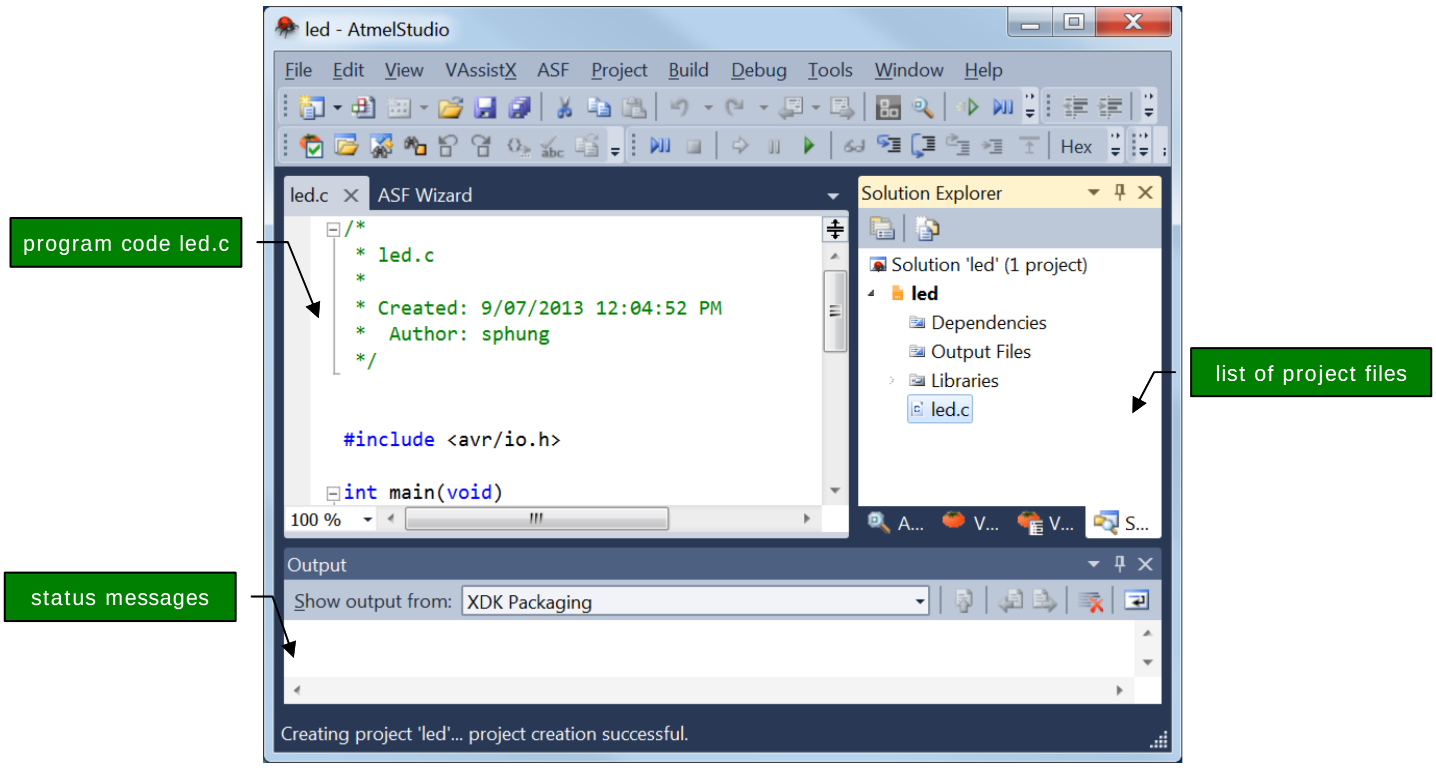
Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (byte
ATmega16	16	1024	512
ATmega162	16	65280	512
ATmega164A	16	1024	512
ATmega164P	16	1024	512
ATmega164PA	16	1024	512
ATmega165A	16	1024	512
ATmega165P	16	1024	512
ATmega165PA	16	1024	512
ATmega168	16	1024	512
ATmega168A	16	1024	512
ATmega168P	16	1024	512
ATmega168PA	16	1024	512
ATmega169A	16	1024	512
ATmega169P	16	1024	512
ATmega169PA	16	1024	512
ATmega16A	16	1024	512
ATmega16HVB	16	1024	512
ATmega16M1	16	1024	512

Device Info:
Device Name: ATmega16
Speed: 0
Vcc: 2.7/5.5
Family: megaAVR
Datasheets

Supported Tools
AVR Dragon
AVRISP mkII
AVR ONE!
JTAGICE3
JTAGICE mkII
AVR Simulator
STK500
STK600

Figure 2: Selecting device.

- A project file will be created and Atmel Studio displays an initial file led.c (see Figure 3).



```
/*
 * led.c
 *
 * Created: 9/07/2013 12:04:52 PM
 *  Author: sphung
 */


#include <avr/io.h>

int main(void)
```

Solution 'led' (1 project)
led
Dependencies
Output Files
Libraries
led.c

Show output from: XDK Packaging

Creating project 'led'... project creation successful.

Figure 3: The Atmel Studio with a project opened.

- 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.
- Click menu **File** | **Save All** to save all project files. Note that an Atmel Studio *solution* has extension '.atsln'; an Atmel Studio C *project* has extension '.cproj'.

```
// File: led.c
// Description: Simple C program for the ATMEL AVR uC (ATmega16 chip)
// This program lets the user turn on LEDs by pressing the switches on STK500 board

#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;
}
```

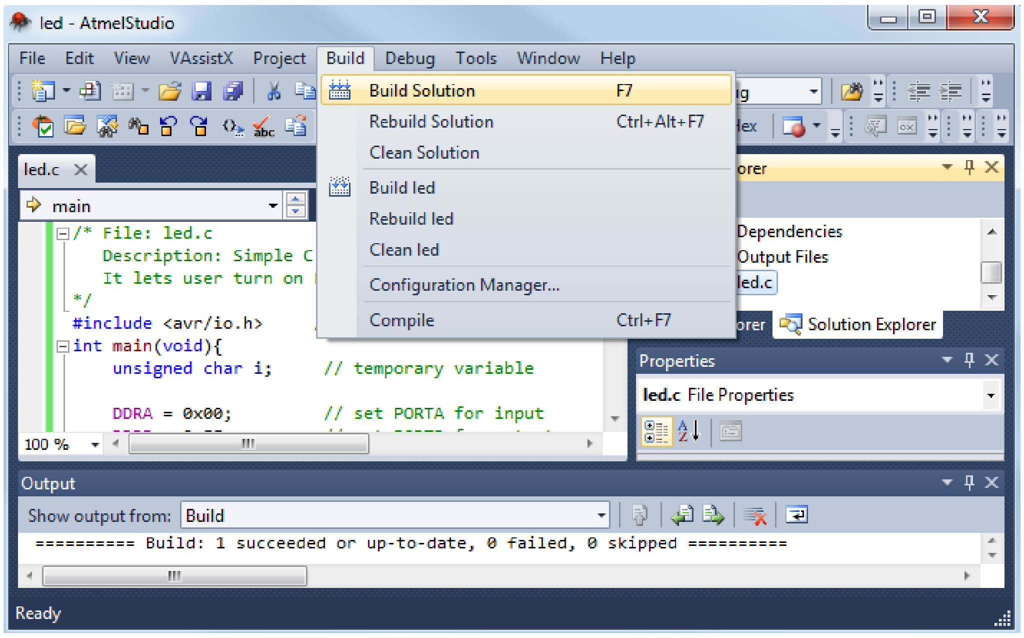
Figure 4: Program code led.c.

3.2 Compiling C code to HEX file

- Click menu **Build** | **Build Solution** to compile the C code (the hot-key for this is **F7**).

- 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 'debug' or 'release' of your project.

- If there are error messages, check your C code. Most often, error messages are caused by typographical or syntax errors. Atmel Studio will show the line numbers where errors appear in the C code.



```
/* File: led.c
   Description: Simple C
   It lets user turn on
*/
#include <avr/io.h>
int main(void){
    unsigned char i;      // temporary variable

    DDRA = 0x00;          // set PORTA for input
```

Show output from: Build

```
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

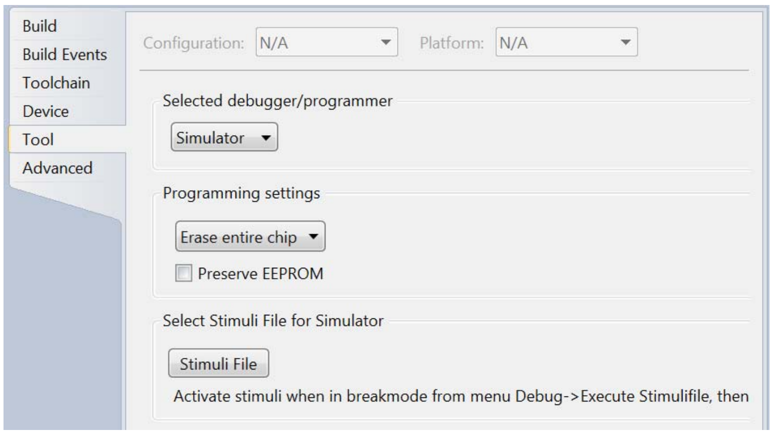
Figure 5: Selecting menu Build | Build Solution to create HEX file.

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 Atmel Studio. Basically, 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 Atmel Studio provides a software simulator for many AVR microcontrollers, including the ATmega16 chip. The following steps in this section do not require a STK500 board.

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

- Start the debugger by selecting menu **Debug** | **Start Debugging and Break**. Atmel Studio will require you to specify a debugger. Select 'Simulator', as shown in Figure 6.



Configuration: N/A Platform: N/A

Selected debugger/programmer: Simulator

Programming settings: Erase entire chip

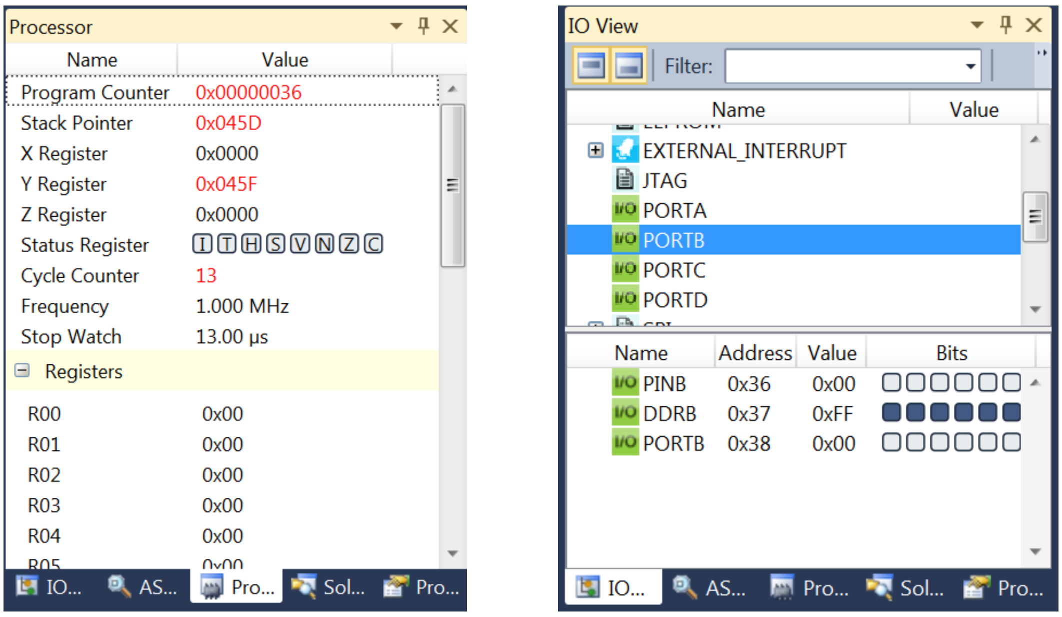
Preserve EEPROM

Select Stimuli File for Simulator: Stimuli File

Activate stimuli when in breakmode from menu Debug->Execute Stimulifile, then

Figure 6: Specifying the debugger to be 'Simulator'.

- Atmel Studio lets you examine the contents of CPU registers and IO ports. To enable these views, select menu **Debug** | **Windows** and then **Processor View** or **I/O View**. Refer to Figure 7.



Processor

Name	Value
Program Counter	0x00000036
Stack Pointer	0x045D
X Register	0x0000
Y Register	0x045F
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	13
Frequency	1.000 MHz
Stop Watch	13.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00

IO View

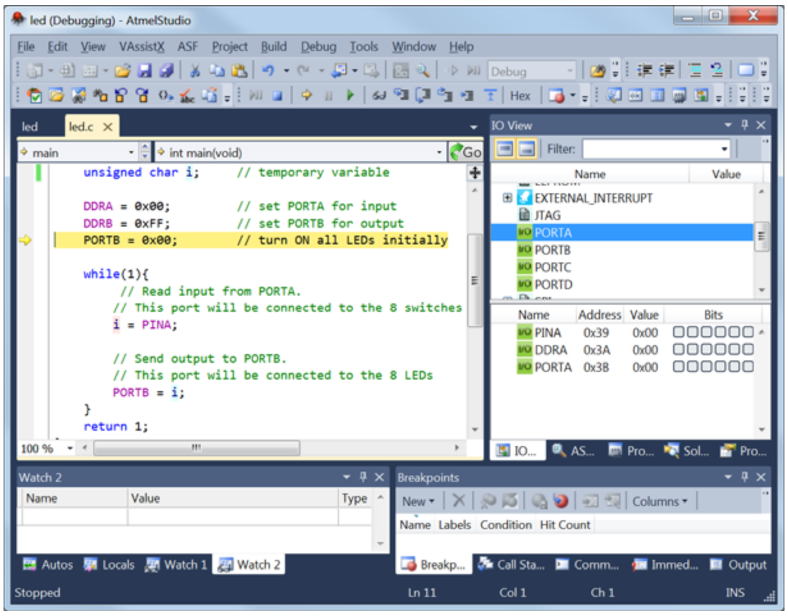
Name	Value
EXTERNAL_INTERRUPT	
JTAG	
PORTA	
PORTB	
PORTC	
PORTD	

Name	Address	Value	Bits
PINB	0x36	0x00	☐☐☐☐☐☐
DDRB	0x37	0xFF	■■■■■■
PORTB	0x38	0x00	☐☐☐☐☐☐

(a) Processor view (b) IO view

Figure 7: Debugging views.

- A yellow arrow will appear in the code window (Figure 8); it indicates the C instruction to be executed next.



```
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;
```

Name	Address	Value	Bits
PINA	0x39	0x00	☐☐☐☐☐☐
DDRA	0x3A	0x00	☐☐☐☐☐☐
PORTA	0x3B	0x00	☐☐☐☐☐☐

led (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
led.c
main
int main(void)
IO View
Watch 2
Breakpoints
Stopped Ln 11 Col 1 Ch 1 INS

Figure 8: Stepping through a C program in the debugging mode.

- Select menu **Debug** | **Step Into** (or press hot-key F11) to execute the C instruction at the yellow arrow. Figure 7b shows the IO view after the following C instruction is executed:

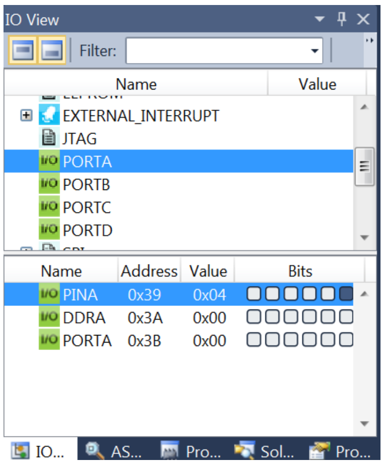
```
DDRB = 0xFF;     // set PORTB for output
```

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

- 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 9a). This change will take effect immediately. Subsequently, the contents of PORTB will be 0x04 (see Figure 9b) after running the two C instructions:

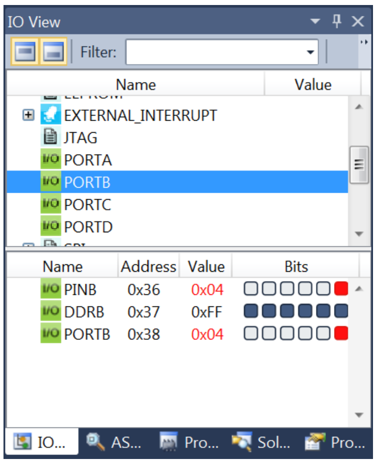
```
i = PINA;
PORTB = i;
```



Name	Value
EXTERNAL_INTERRUPT	
JTAG	
PORTA	
PORTB	
PORTC	
PORTD	

Name	Address	Value	Bits
PINA	0x39	0x04	▢▢▢▢▢■
DDRA	0x3A	0x00	▢▢▢▢▢▢
PORTA	0x3B	0x00	▢▢▢▢▢▢

(a) changing PINA register to 0x04



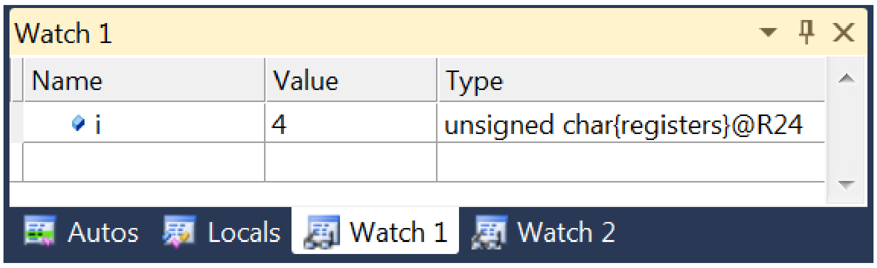
Name	Value
EXTERNAL_INTERRUPT	
JTAG	
PORTA	
PORTB	
PORTC	
PORTD	

Name	Address	Value	Bits
PINB	0x36	0x04	▢▢▢▢▢🟥
DDRB	0x37	0xFF	■■■■■■
PORTB	0x38	0x04	▢▢▢▢▢🟥

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

Figure 9: Modifying registers manually.

- To monitor a C variable, select the variable name in the code window, click menu **Debug | Quick Watch**, and then click button **Add Watch**. The variable will be added to a watch window, as in Figure 10.



Name	Value	Type
i	4	unsigned char{registers}@R24

Figure 10: Watch window for C variables.

- The Debug menu provides many other debugging options, 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 **Debug | Windows | Disassembly**.
- To stop debugging, select menu **Debug | Stop Debugging**.

3.4 Downloading and running HEX file on AVR board

To perform the steps in this section, you will need an STK500 development board from Atmel and the ATmega16 chip. The ATMEGA16 should be placed in **socket SCKT3100A3**.

Note: 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 11 when carrying out the following steps.

- Step 1: Connect the SPROG3 jumper to the ISP6PIN jumper, using the supplied cable in the STK500 kit. This step is needed to program the ATmega16 chip.
- Step 2: Connect the board with the PC using a serial cable. Note that the STK500 has two RS232 connectors; we use only the connector marked with RS232 CTRL.
- Step 3: 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.
- Step 4: 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.
- Step 5: Connect the board with 12V DC power supply and turn the power switch ON.

All testing involving the ATmega16 chip require Steps 1, 2, and 5.

Steps 3 and 4 are needed only for this particular example.

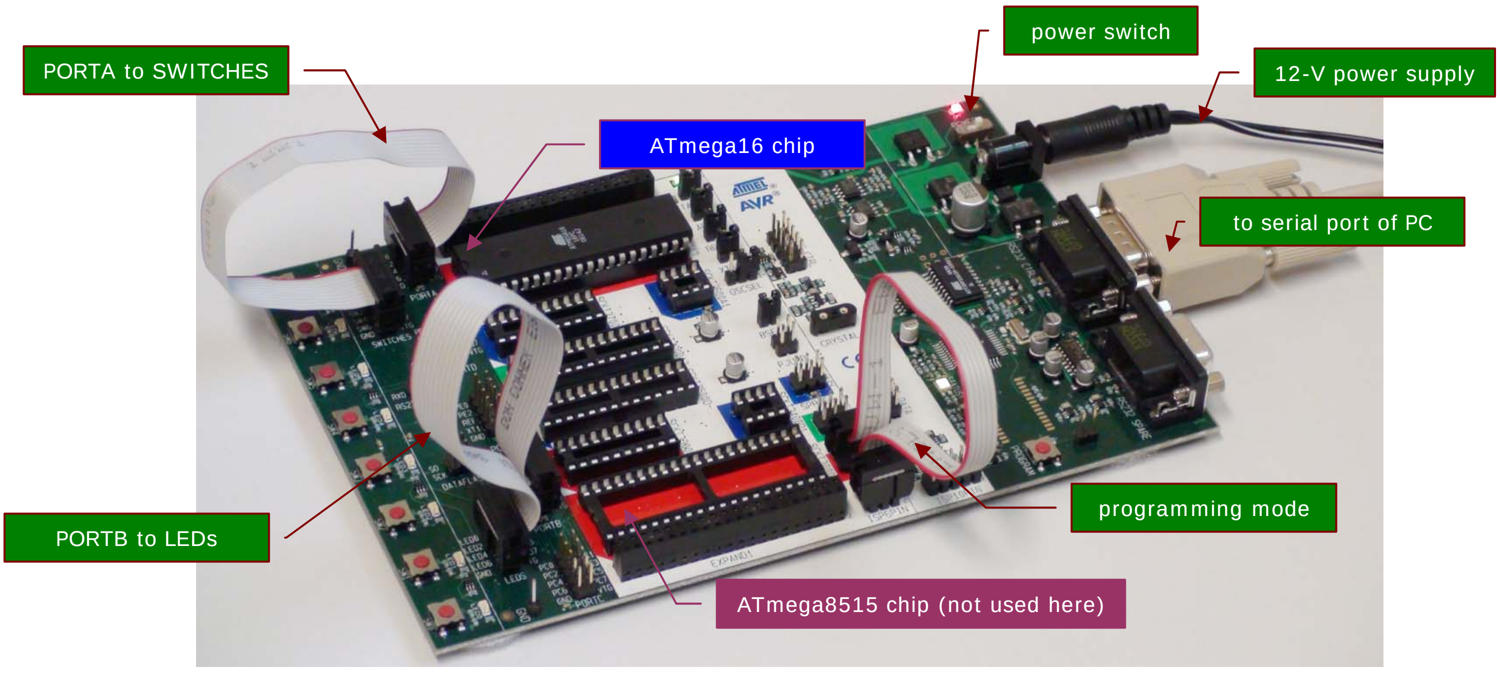
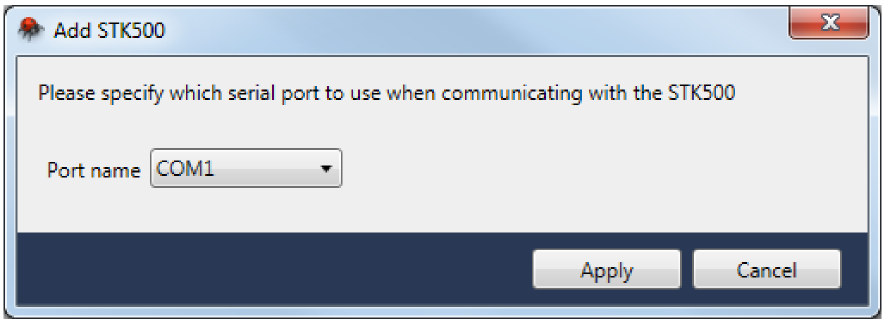


Figure 11: Setting up the STK500 for downloading and testing.

Downloading and running HEX file

- In Atmel Studio, select menu **Tools | Add STK500**[2].

- In the 'Add STK500' dialog box that appears (see Figure 12), select the correct serial port and click button '**Apply**'.
  - For your home PC with inbuilt serial port, the serial port is usually COM1.
  - For a computer using a USB-to-Serial cable, the serial port can be a different number.
  - For computers in SECTE Digital Lab 35.129, the serial port is COM5 or COM4.

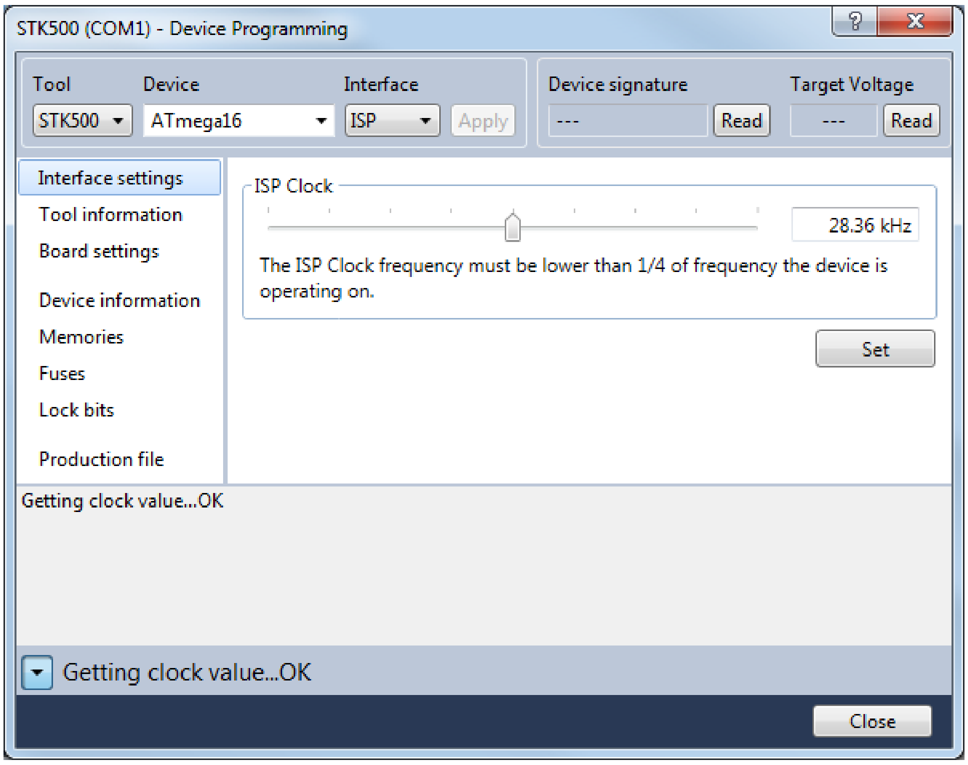

Add STK500

Please specify which serial port to use when communicating with the STK500

Port name COM1

Figure 12: Adding STK500 board.

- In the 'Device Programming' dialog box that appears (see Figure 13), select 'Tool' = STK500, 'Device' = ATmega16, and 'Interface' = ISP.


STK500 (COM1) - Device Programming

Tool STK500 Device ATmega16 Interface ISP Device signature --- Target Voltage ---

ISP Clock 28.36 kHz

The ISP Clock frequency must be lower than 1/4 of frequency the device is operating on.

Getting clock value...OK

Getting clock value...OK

Figure 13: Device programming.

[2] In Atmel Studio 6.1, select menu **Tools | Add Target**.

- In ‘Memories’ tab, select the HEX file and click ‘Program’ (see Figure 14).

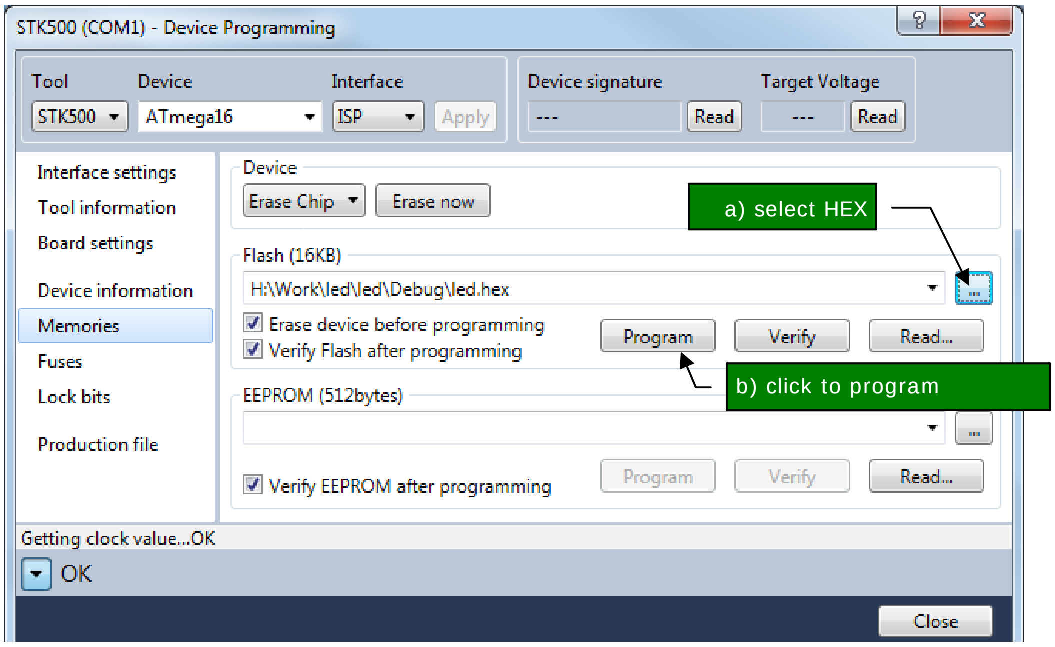


Figure 14: Programming the ATmega16 chip.

- The program will now run on the microcontroller. If you press and hold down one of the 8 switches on the STK500 board, the corresponding LED will be turned on.

A video demo of the program is available at: http://youtu.be/XlqmbExF1mU

This is the end of this introductory tutorial. More information about programming Atmel AVR microcontrollers for embedded applications is provided in ECTE333 Microcontroller Architecture and Applications subject, School of Electrical, Computer and Telecommunication Engineering, University of Wollongong, and also at http://www.uow.edu.au/~phung.

Version history

Version	Date	Description
2.0	16/09/2013	Updated guide, using Atmel Studio 6.x
1.0	14/05/2008 14/01/2010	Initial guide, using Atmel Studio 4.x

*** END ***