AVR030: Getting Started with IAR Embedded Workbench for Atmel AVR

8-bit AVR® Microcontrollers

Application Note

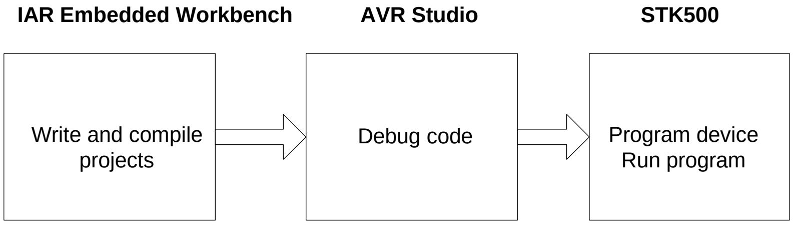
Features

- How to open a new workspace and project in IAR Embedded Workbench
- Description and option settings for compiling the c-code
- Setting up the STK500
- Using AVR Studio for debugging and programming

Introduction

The purpose of this application note is to guide new users through the initial settings of IAR Embedded Workbench, and compile a simple C-program. The application note shows setup of a new project to generate a debug target that can be loaded into AVR Studio. AVR Studio is used for both debugging and to download the code into the device installed on the STK500. A simple ledchaser is used as an example, written for the ATmega16.

Figure 1. Tools overview.



Rev. 1483D-AVR-10/04

Preparations

This application note is written for the IAR Embedded Workbench 3.20. From this point it's assumed that a working copy of this IDE is installed. Both IAR's homepage www.iar.com and AVRFreaks.net are suitable sources for download. You also need to install AVR Studio. A copy could be downloaded free from www.atmel.com or installed from the cd following the STK500 development board. The development board should be working with it's default settings, so just unpack and follow the instructions given later in this text.

Getting Started

You have now installed all tools needed for this getting started, as described in preparations. When working with programming in general it's important to have some structure in your coming projects and code. IAR Embedded Workbench is made to support such demands. The upper abstraction of a task is called a workspace, within each workspace you can add projects. The projects added in a workspace could be supporting the same device or have something in common. Each project contains code and settings for each target. So what we need to do is first make a workspace, then add a new project to this workspace. When this is done you should be able to include an application code to your project and make all the settings for the target AVR on your STK500 board. A sample application code is found together with this document, and should be unzipped and saved at a proper location, ex. "C:\users\code".

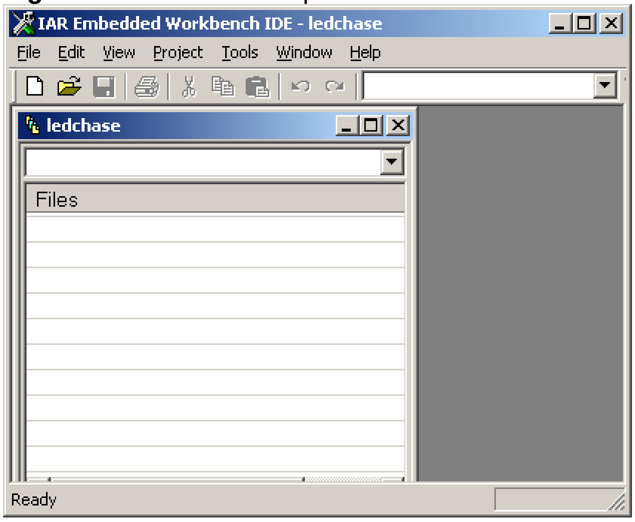
Creating a new workspace in IAR Embedded Workbench

Open IAR Embedded Workbench. Follow the list below to make your first project, the "ledchaser":

1. Click new under the File menu.
2. Select "Workspace" and click OK.
3. Next dialog is for selecting name of the new workspace, type in "ledchase". Set the "Save in" path to: "C:\users\code" and hit OK.

Your IAR IDE window should now look like that of Figure 2.

Figure 2. The IAR Workspace.


IAR Embedded Workbench IDE - ledchase
File Edit View Project Tools Window Help
ledchase
Files
Ready

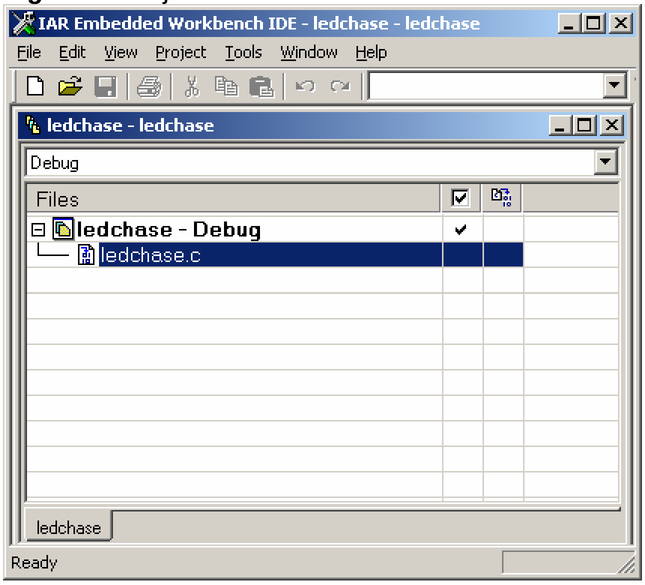
Creating a new project in IAR Embedded Workbench

When the new workspace is created, you'll see a pane to the left in the IAR window named "ledchase". To make a new project, "ledchaser", do the following:

1. Click "Create new Project" under the Project menu.
2. In the next dialog, select the "ledchase" directory as your working directory, in this case: "C:\users\code", and type "ledchase" in the project name field.
3. Click "Add Files" under the Project menu, and select the "ledchase.c" source file.
4. Click the open button and this file will be added to the project.
5. Double click the "ledchase.c" under the "ledchase" project, and feel free to resize, move etc.
6. Click "Save all" in the File menu. Your project is now completed.

Figure 3 shows what this will look like in IAR Embedded Workbench.

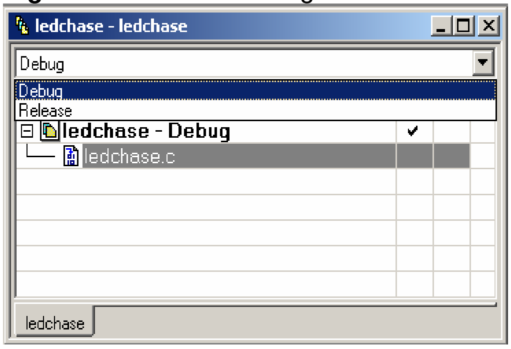
Figure 3. Project with ledchase.c included.



Setting up targets in IAR Embedded Workbench

You should now have a workspace which contains the "ledchase" project, where your source file is located. The next step is to tell what you need from the IAR compiler and linker. In this context it's valuable to work with two targets and their configuration. The IAR Embedded Workbench has two default targets; Debug and Release. The Debug target will make an object file that AVR Studio can use for debugging. The Release target will be a binary file for loading into the flash of your selected AVR. The settings for both targets are done under the Project menu. The steps must be repeated twice; once for the Debug target and once for the Release target.

Figure 4. Selection of target.



Be sure the Debug target is selected, see Figure 4, and go through the list below:

1. Make sure the right target is selected in the drop-down menu of the project box in your IDE.
2. Click “Options” under the Project menu.
3. In the “General” category, under the “Target” pane, select processor configuration for ATmega16.
4. Chose memory model “small”.
5. Under the “Library Configuration” pane, check to enable bit-definitions in IO-included files.
6. Select the “XLINK” category.
7. Set the output format to “Other” –“ubrof8”. This produces an object file that AVR Studio will recognize when the project is built.
8. Close the target options dialog by clicking “OK”.

The same steps should be repeated for Release target, except:

- In step 7, select “Other” – “Intel-standard” as output format. This setting tells the compiler to make a hex file that can be loaded into the flash.
- In the “ICCAVR” category, under the “Optimization” pane, select Medium optimization. This way the compiler may be optimized for code size or execution speed. Low degree of optimization gives best support for debugging.

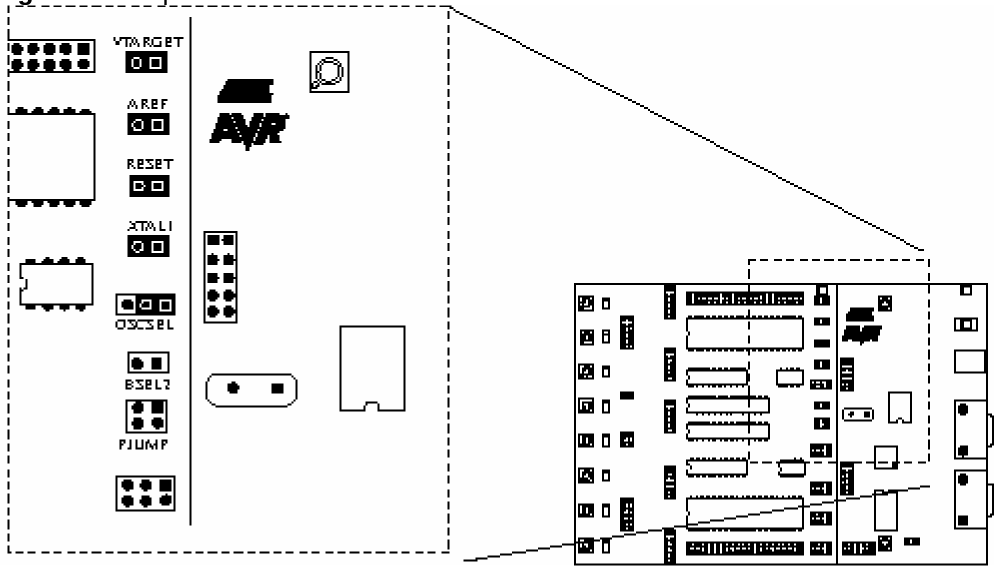
When this is done, save all once more. Then select the Debug target and click “Make” in the Project menu or press F7 to build the project.

Setting up the STK500

Before using the STK500 board be sure the power is turned off. The board should be supplied with an ATmega16 16PC microcontroller. If not already installed, carefully insert into the socket marked SCKT3100A3. The microcontroller has a notch that should be pointing the same way as the similar notch found on the socket. When all set, you should inspect that the following jumpers are set:

- VTARGET: On-board VTARGET supply connected.
- AREF: On-board Analog Voltage Reference connected.
- RESET: On-board reset system connected.
- XTAL1: On-board clock system connected.
- OSCEL: Between pin 2 and 3. Enables software clock signal.
- Remove any other jumpers.

Figure 5. Jumpers on stk500



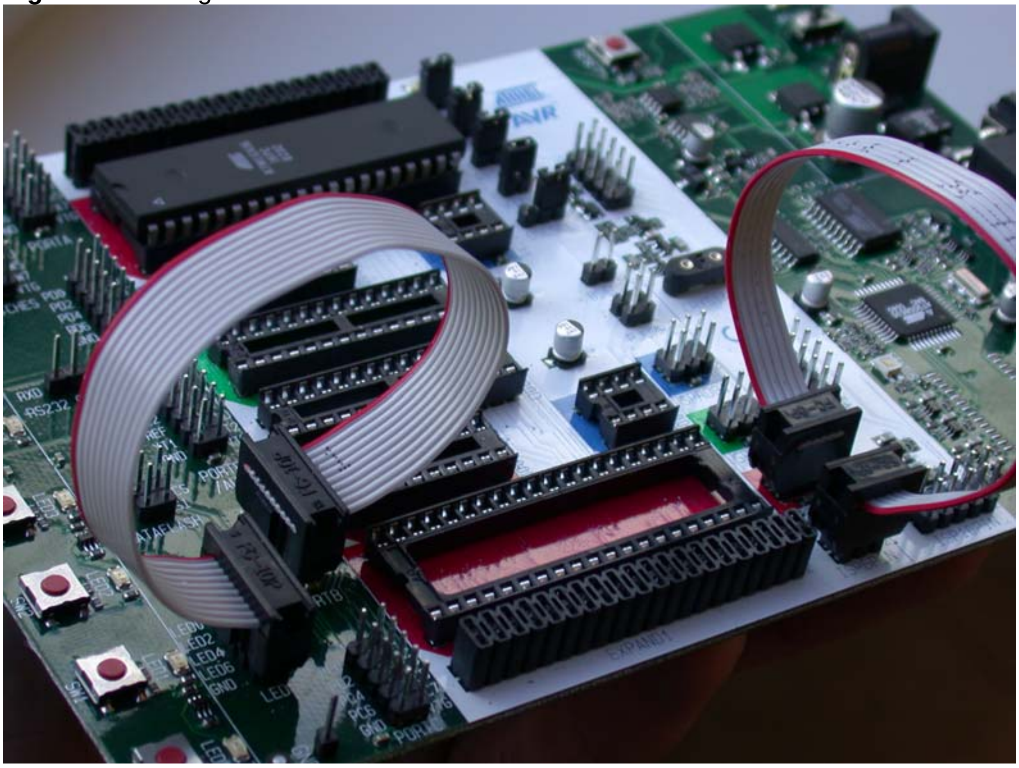
For further reading on this subject, please consult AVR Studio help under “STK500 User’s Guide” and “Jumper Settings and Special Cabling”.

Connect the 10-pin ribbon cable between PORTB and LEDs headers. This will allow displaying the state of the ATmega16’s PORTB output.

Connect the 6-pin ribbon between ISP6PIN and SPROG3 headers. This will allow the AVR Studio programming the microcontroller.

Connect a RS232 cable from your computer to the connector market “RS232 CTRL” on the STK500 board. Turn power on! Now your jumper section should look like that in Figure 6.

Figure 6. Cabling on STK500.



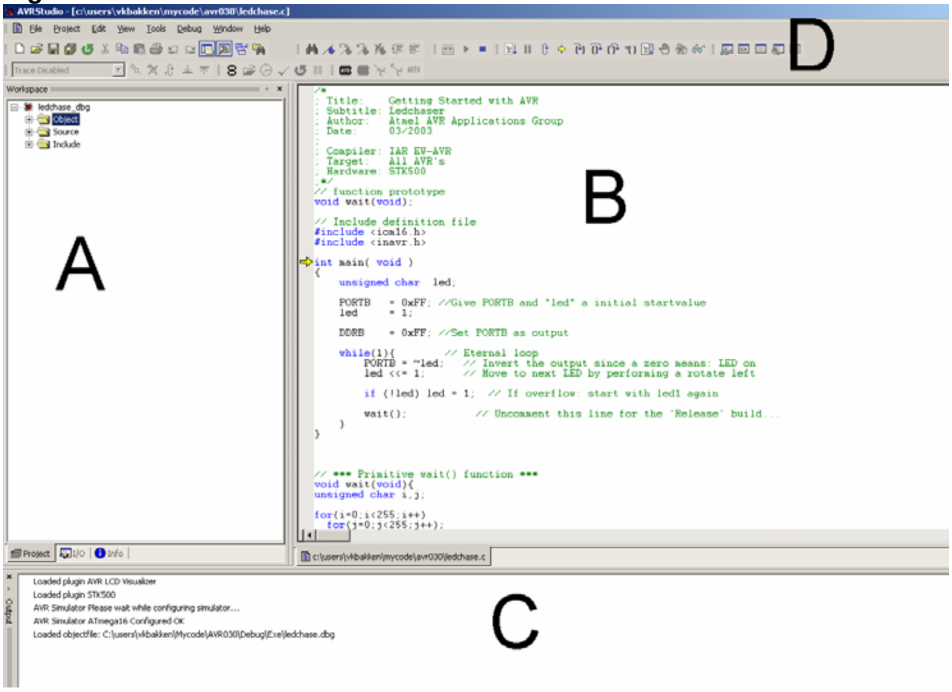
Using AVR Studio

IAR Embedded Workbench AVR Studio is project based, so debugging and development is done within a project handling your files. Open AVR Studio, and the project manager will be launched, and new projects can be made or old projects loaded. To open the “ledchase” project:

- Select open and browse to the file \Debug\Exe\ledchase.dbg in your projects directory.
- Select AVR Simulator as debug platform and ATmega16 as device.
- Press Finish
- The project is now ready to be debugged in the simulator or programmed into an AVR device.

Take a moment and read through the ledchaser code. Try understanding what it will do, before simulating or programming. The ledchaser application is just a simple ‘0’ walking on PORTB. ‘0’ is used because the LEDs on the STK500 are in a sink configuration, and will be lit when given a ‘0’ value.

Figure 7. AVR Studio.



AVRStudio - [c:\users\vkbakken\mycode\avr030\ledchase.c]
File Project Edit View Tools Debug Window Help
Trace Disabled
Workspace
ledchase_dbg
Object
Source
Include
Project
I/O
Info
c:\users\vkbakken\mycode\avr030\ledchase.c
Output
A
B
C
D

```
/*
; Title:    Getting Started with AVR
; Subtitle: Ledchaser
; Author:   Atmel AVR Applications Group
; Date:     03/2003
;
; Compiler: IAR EW-AVR
; Target:   All AVR's
; Hardware: STK500
;*/
// function prototype
void wait(void);

// Include definition file
#include <iom16.h>
#include <inavr.h>

int main( void )
{
    unsigned char  led;

    PORTB   = 0xFF; //Give PORTB and "led" a initial startvalue
    led     = 1;

    DDRB    = 0xFF; //Set PORTB as output

    while(1){         // Eternal loop
        PORTB = ~led;   // Invert the output since a zero means: LED on
        led <<= 1;      // Move to next LED by performing a rotate left

        if (!led) led = 1;  // If overflow: start with led1 again

        wait();             // Uncomment this line for the 'Release' build...
    }
}


// *** Primitive wait() function ***
void wait(void){
unsigned char i,j;

for(i=0;i<255;i++)
  for(j=0;j<255;j++);
```

```
Loaded plugin AVR LCD Visualizer
Loaded plugin STK500
AVR Simulator Please wait while configuring simulator...
AVR Simulator ATmega16 Configured OK
Loaded objectfile: C:\users\vkbakken\Mycode\AVR030\Debug\Exe\ledchase.dbg
```

AVR Studio is Windows based, and its editor GUI can be divided into four main sections, as seen in Figure 7.

A. The Workspace contains information about the files in the project, IO view, and info about the selected AVR.

B. The editor contains the code being edited or debugged.

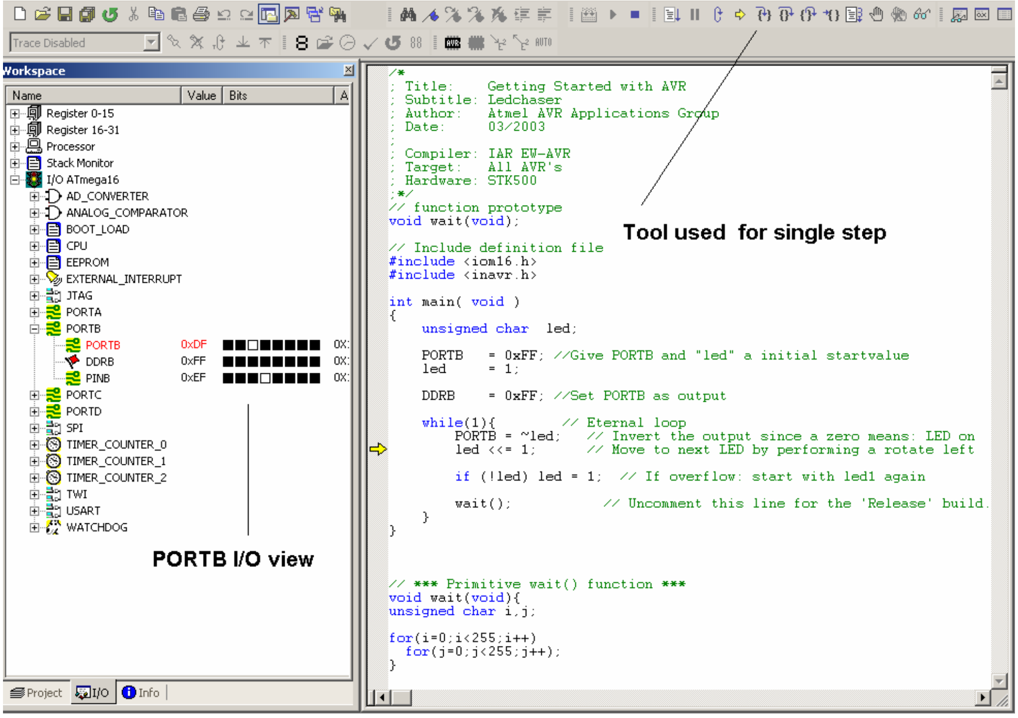
C. The output window displays status information while the System Tray displays information about which mode AVR Studio is running in.

D. Menus included standard Windows menus like save and load file, cut & paste, and other AVR Studio specific menus like Assembler options and debug functions. The toolbars act as shortcuts to menu functions.

Before programming the microcontroller it is valuable using the debugger to verify that the code works. Single step through the code and verify that:

1. PORTB and DDRB is set to 0xFF.
2. See that a PORTB has a running ‘0’. Use Step Into (F11) to see this. See Figure 8 for reference.

Figure 8. Tools and I/O view.

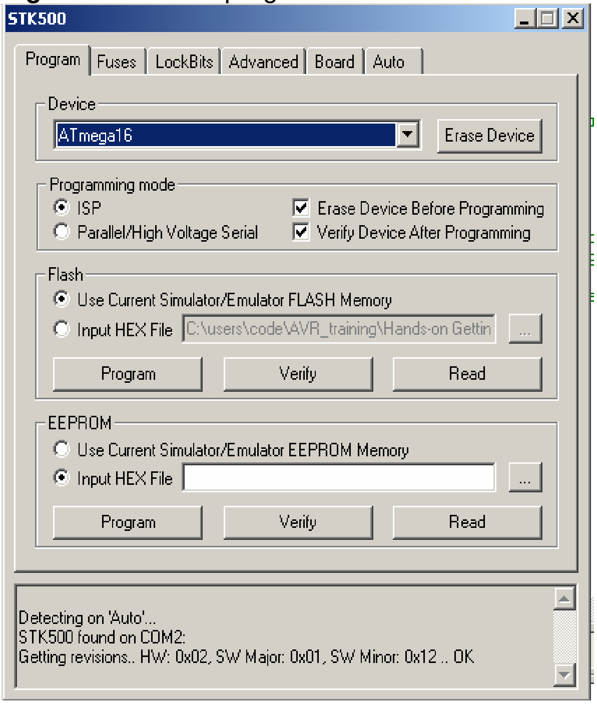


Name	Value	Bits	A
⊞ Register 0-15			
⊞ Register 16-31			
⊞ Processor			
⊞ Stack Monitor			
⊟ I/O ATmega16			
⊞ AD_CONVERTER			
⊞ ANALOG_COMPARATOR			
⊞ BOOT_LOAD			
⊞ CPU			
⊞ EEPROM			
⊞ EXTERNAL_INTERRUPT			
⊞ JTAG			
⊞ PORTA			
⊟ PORTB			
PORTB	0xDF	■■□■■■■■	0X:
DDRB	0xFF	■■■■■■■■	0X:
PINB	0xEF	■■■□■■■■	0X:
⊞ PORTC			
⊞ PORTD			
⊞ SPI			
⊞ TIMER_COUNTER_0			
⊞ TIMER_COUNTER_1			
⊞ TIMER_COUNTER_2			
⊞ TWI			
⊞ USART			
⊞ WATCHDOG			

```
/*
; Title:    Getting Started with AVR
; Subtitle: Ledchaser
; Author:   Atmel AVR Applications Group
; Date:     03/2003
;
; Compiler: IAR EW-AVR
; Target:   All AVR's
; Hardware: STK500
;*/
// function prototype
void wait(void);

// Include definition file
#include <iom16.h>
#include <inavr.h>

int main( void )
{
    unsigned char  led;

    PORTB   = 0xFF; //Give PORTB and "led" a initial startvalue
    led     = 1;

    DDRB    = 0xFF; //Set PORTB as output

    while(1){         // Eternal loop
        PORTB = ~led;   // Invert the output since a zero means: LED on
        led <<= 1;      // Move to next LED by performing a rotate left

        if (!led) led = 1;  // If overflow: start with led1 again

        wait();            // Uncomment this line for the 'Release' build.
    }
}


// *** Primitive wait() function ***
void wait(void){
unsigned char i,j;

for(i=0;i<255;i++)
  for(j=0;j<255;j++);
}
```

The code is verified and working, and its time to power up the STK500 board and load the program into the ATmega16. When power is on, a small led in the upper right corner should be lit red. To program the device select “STK500/AVRISP/JTAG ICE” from the Tools menu. A program manager will pop up, looking like that of Figure 9.

Figure 9. STK500 program window.



Device: ATmega16

Programming mode

● ISP ☑ Erase Device Before Programming

○ Parallel/High Voltage Serial ☑ Verify Device After Programming

Flash

● Use Current Simulator/Emulator FLASH Memory

○ Input HEX File C:\users\code\AVR_training\Hands-on Gettin

EEPROM

○ Use Current Simulator/Emulator EEPROM Memory

● Input HEX File

```
Detecting on 'Auto'...
STK500 found on COM2:
Getting revisions.. HW: 0x02, SW Major: 0x01, SW Minor: 0x12 .. OK
```

Make sure that the ATmega16 is selected as device, programming mode is set to “ISP” and under Flash “Use Current Simulator/Emulator FLASH Memory”. This will load the current code into device flash. Press the “Program” button, and wait while the Atmega16 is being programmed. Soon you should see the led start walking.

This note is completed and you’ve made your very first AVR project, using the IAR Embedded Workbench. For more information regarding IAR Embedded Workbench visit IAR’s homepage www.iar.com or have a look under the help menu. AVR Studio also includes a great source for help. The online help found under AVR Studio’s help menu, will solve many problems. Further reading and application notes can be found at www.atmel.com and www.avrfreaks.net.

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe

Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia

Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan

9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive

Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom

Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests

www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.