AVR033: Getting Started with the CodeVisionAVR C Compiler

8-bit AVR® Microcontrollers

Application Note

Features

- **Installing and Configuring CodeVisionAVR to Work with the Atmel STK®500 Starter Kit and AVR Studio® Debugger**
- **Creating a New Project Using the CodeWizardAVR Automatic Program Generator**
- **Editing and Compiling the C Code**
- **Loading the Executable Code into the Target Microcontroller on the STK500 Starter Kit**

1 Introduction

The purpose of this application note is to guide the user through the preparation of an example C program using the CodeVisionAVR C Compiler. The example, which is the subject of this application note, is a simple program for the Atmel ATmega8515 microcontroller on the STK500 starter kit.

Questions regarding CodeVisionAVR C Compiler should be addressed to HP InfoTech S.R.L. (http://www.hpinfotech.ro).

Rev. 2500C–AVR–04/08

2 Preparation

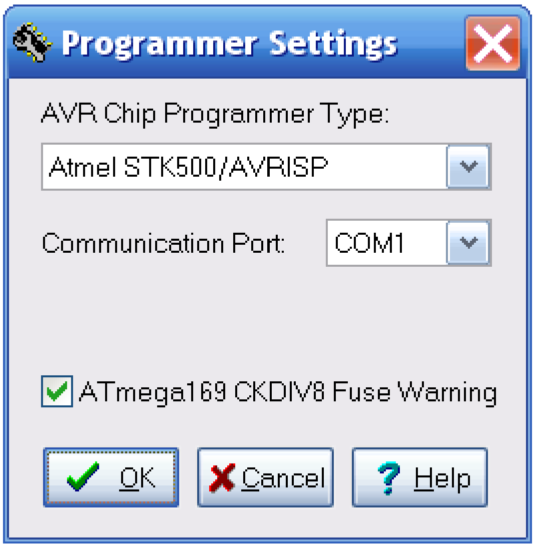
Install the CodeVisionAVR C Compiler in the default directory (“C:\cvavr”), and the Atmel AVR Studio debugger in the default directory (“C:\Program Files\Atmel\AVRTools\AVR Studio4”).

Set up the starter kit according to the instructions in the STK500 User Guide. Make sure that the power is off before inserting the ATmega8515 chip into the appropriate socket (marked SCKT3000D3). Set the VTARGET, RESET, and XTAL1 jumpers. Also, set the OSCSEL jumper between pins 1 and 2.

Connect one 10-pin ribbon cable between the PORTB and LEDs headers to allow for the state of ATmega8515’s PORTB outputs to be displayed. Then connect one 6-pin ribbon cable between the ISP6PIN and SPROG3 headers. This will allow the CodeVisionAVR IDE to automatically program the AVR chip after a successful compilation, if the programmer is correctly configured.

To configure the programmer, start the CodeVisionAVR IDE and select the “Settings→Programmer” menu option. The dialog window shown in Figure 2-1 will open.

Figure 2-1. Programmer Settings.



Programmer Settings

AVR Chip Programmer Type: Atmel STK500/AVRISP

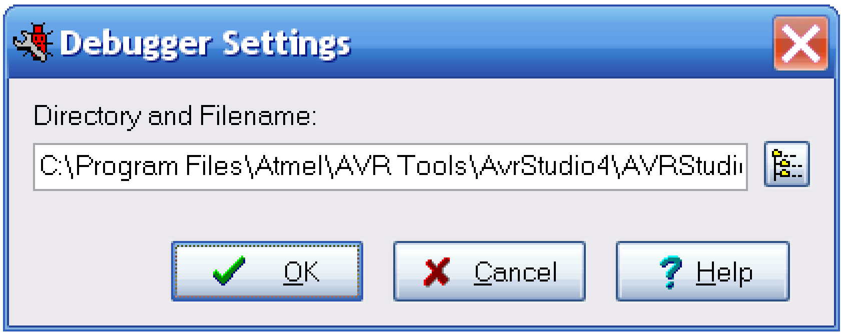
Communication Port: COM1

☑ ATmega169 CKDIV8 Fuse Warning

Set the AVR Chip Programmer Type to “Atmel STK500/AVRISP”, and the Communication Port to the one used with the STK500 starter kit on your system.

In order to be able to invoke the AVR Studio debugger from within the CodeVisionAVR IDE, the location of AVR Studio must be set. To do this, select the “Settings→Debugger” menu option. The dialog window as shown in Figure 2-2 will open.

Figure 2-2. Debugger Settings.



Directory and Filename:

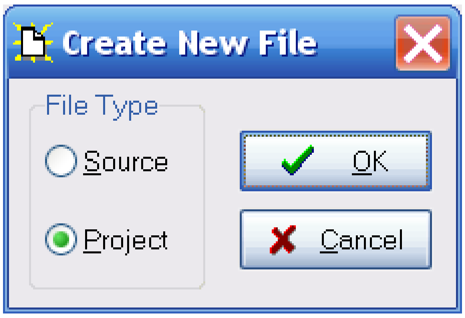
C:\Program Files\Atmel\AVR Tools\AvrStudio4\AVRStudi

Select "C:\Program Files\Atmel\AVRTools\AVR Studio4\AvrStudio.exe" using the button, then press the "OK" button to confirm.

3 Creating a New Project

In order to create a new project, select the "File→New" menu option or press the toolbar button. The dialog window shown in Figure 3-1 will be displayed.

Figure 3-1. New Project Dialog.



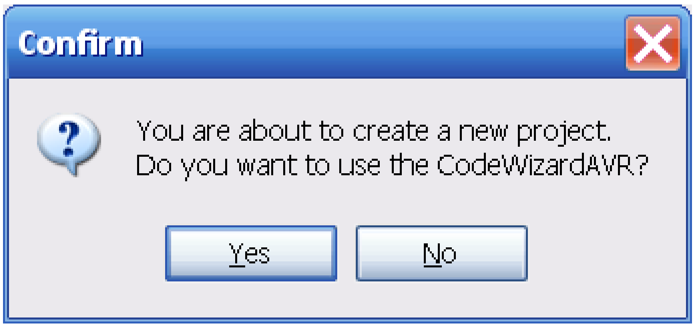
File Type

○ Source

● Project

Select "Project", press "OK", and the dialog window shown in Figure 3-2 will be displayed.

Figure 3-2. Confirmation Dialog.



You are about to create a new project.
Do you want to use the CodeWizardAVR?

Press "Yes" to use the CodeWizardAVR Automatic Program Generator, and the dialog window shown in Figure 4-1 will open.

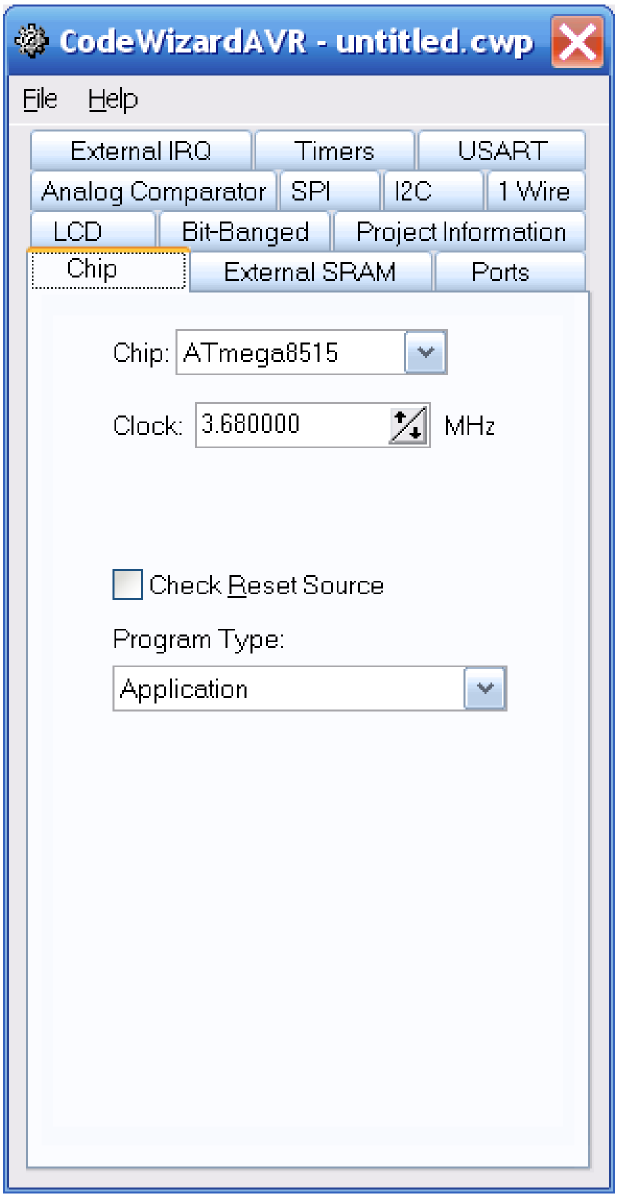
4 Using the CodeWizardAVR Automatic Program Generator

The CodeWizardAVR simplifies the task of writing start-up code for different AVR microcontrollers.

4.1 Configuring the Chip and Clock Settings

For this example project, we shall use the ATmega8515 microcontroller and the clock rate 3.68 MHz, since that is the clock rate on the STK500 starter kit. The resulting settings window is shown in Figure 4-1.

Figure 4-1. Chip Settings for CodeWizardAVR.



Chip: ATmega8515

Clock: 3.680000 MHz

□ Check Reset Source

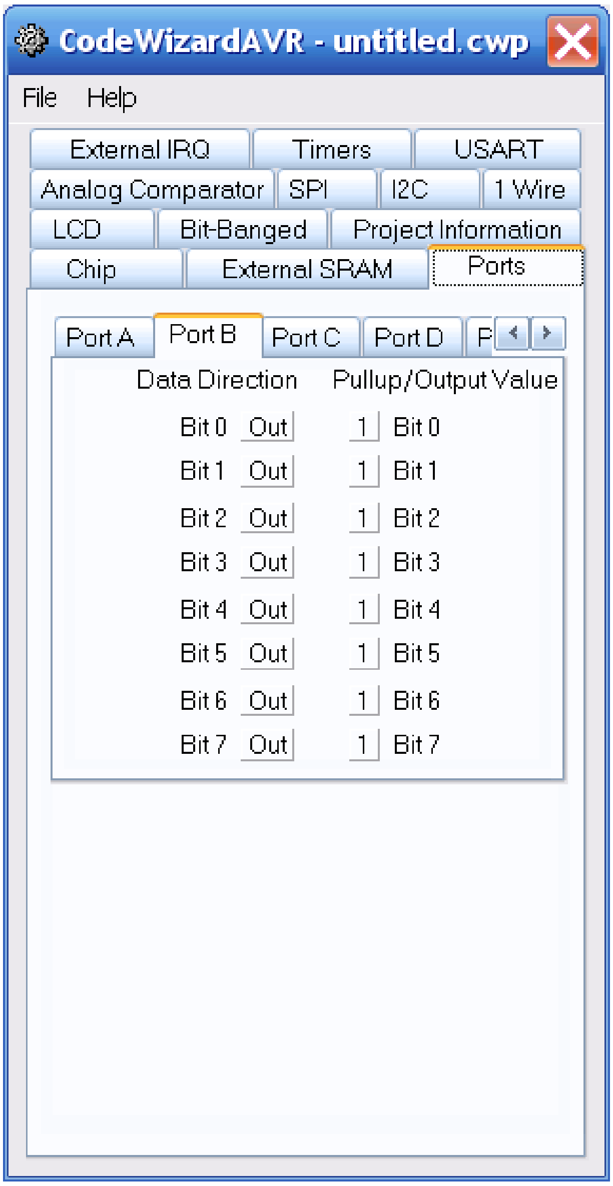
Program Type: Application

5 Configuring the Input/Output Ports

Select the “Ports” tab to determine how the I/O ports are to be initialized for the target system.

The default setting is to have the ports for all the target systems set as inputs (Data Direction bits to be all 1s) in their Tri-state mode. However, for this example project, we want to set Port B (by selecting the Port B tab) to be output only. This is done by setting all the Data Direction bits to Out (by clicking on them). We also set the Output Values to be all 1s, which will cause the LEDs on the STK500 to initially be turned off. The resulting settings window is shown in Figure 5-1.

Figure 5-1. Port Settings for CodeWizardAVR.



Data Direction		Pullup/Output Value	
Bit 0	Out	1	Bit 0
Bit 1	Out	1	Bit 1
Bit 2	Out	1	Bit 2
Bit 3	Out	1	Bit 3
Bit 4	Out	1	Bit 4
Bit 5	Out	1	Bit 5
Bit 6	Out	1	Bit 6
Bit 7	Out	1	Bit 7

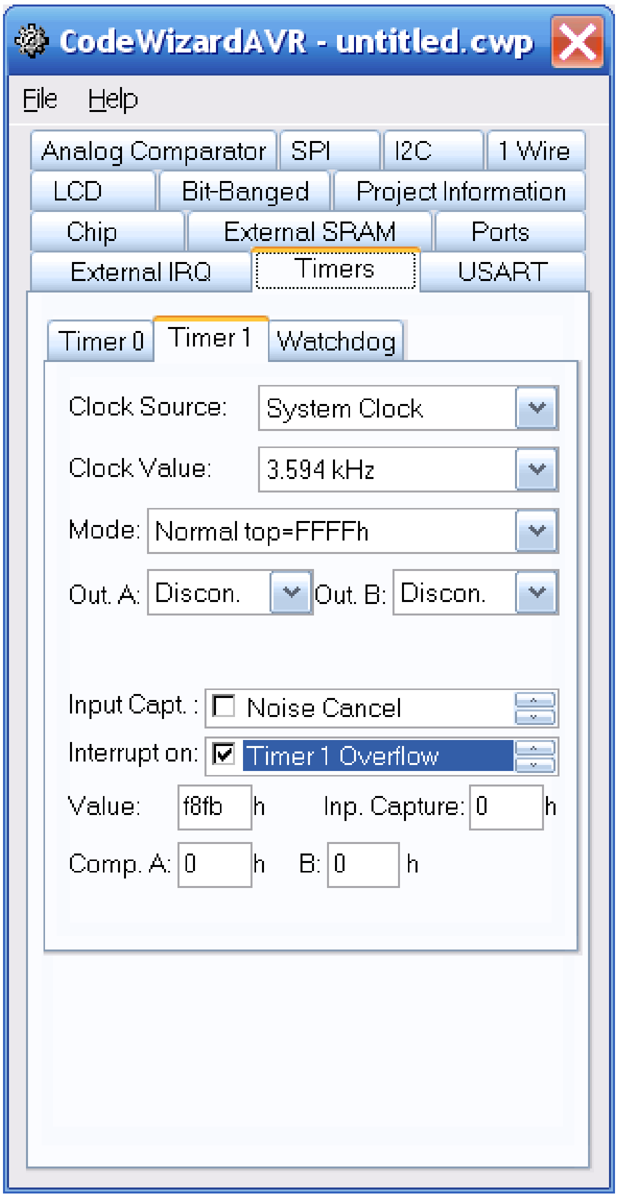
6 Configuring Timer1

Select the “Timers” tab to set up the behaviour of the timers.

For this project, we want to configure Timer1 to generate overflow interrupts as shown in Figure 6-1.

We have selected a clock rate of 3.594 kHz, which is the system clock of 3.68 MHz divided by 1024. The timer is set to operate in the default “Normal Top=FFFFh” mode and to generate interrupts on overflow. To be able to update the LEDs twice per second, we need to reinitialize the Timer1 value to 0x10000-(3594/2) = 0xF8FB on every overflow.

Figure 6-1. Timer Settings for CodeWizardAVR.



CodeWizardAVR - untitled.cwp

Clock Source: System Clock

Clock Value: 3.594 kHz

Mode: Normal top=FFFFh

Out. A: Discon. Out. B: Discon.

Input Capt. : □ Noise Cancel

Interrupt on: ☑ Timer 1 Overflow

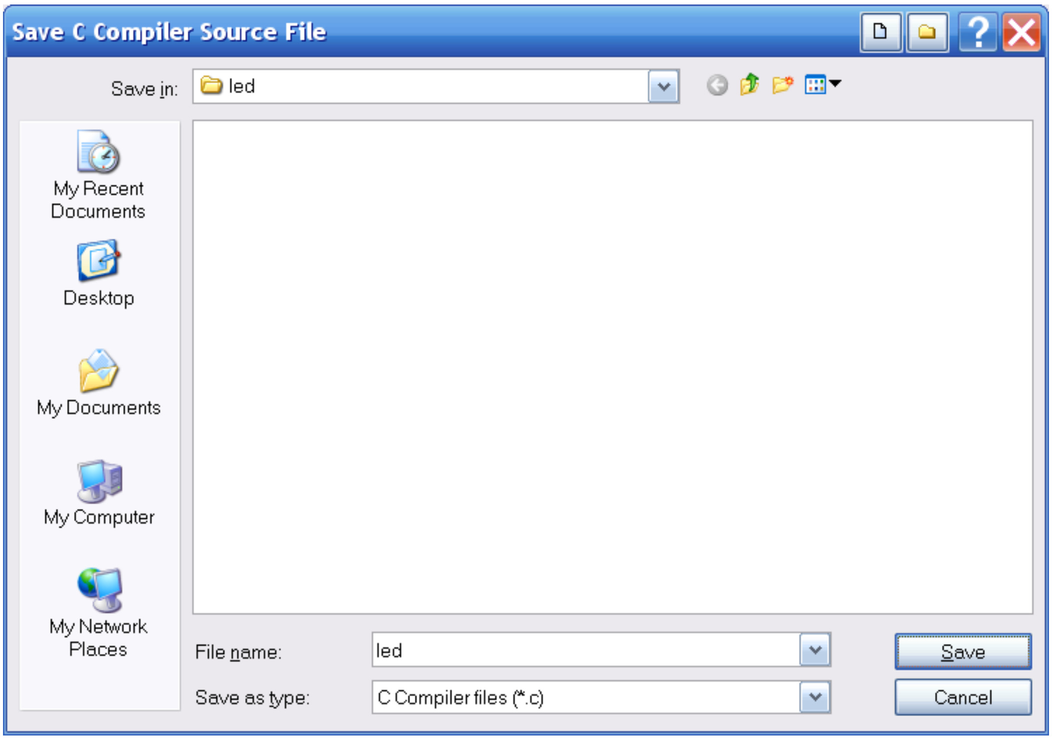
Value: f8fb h Inp. Capture: 0 h

Comp. A: 0 h B: 0 h

7 Completing the Project

By selecting the “File→Generate, Save and Exit” menu option, the CodeWizardAVR will generate a skeleton C program with, in this case, Port B and Timer1 Overflow Interrupt set up correctly. A dialog window for saving the source code, shown in Figure 7-1, will then open.

Figure 7-1. Save Source File Dialog.



Save C Compiler Source File

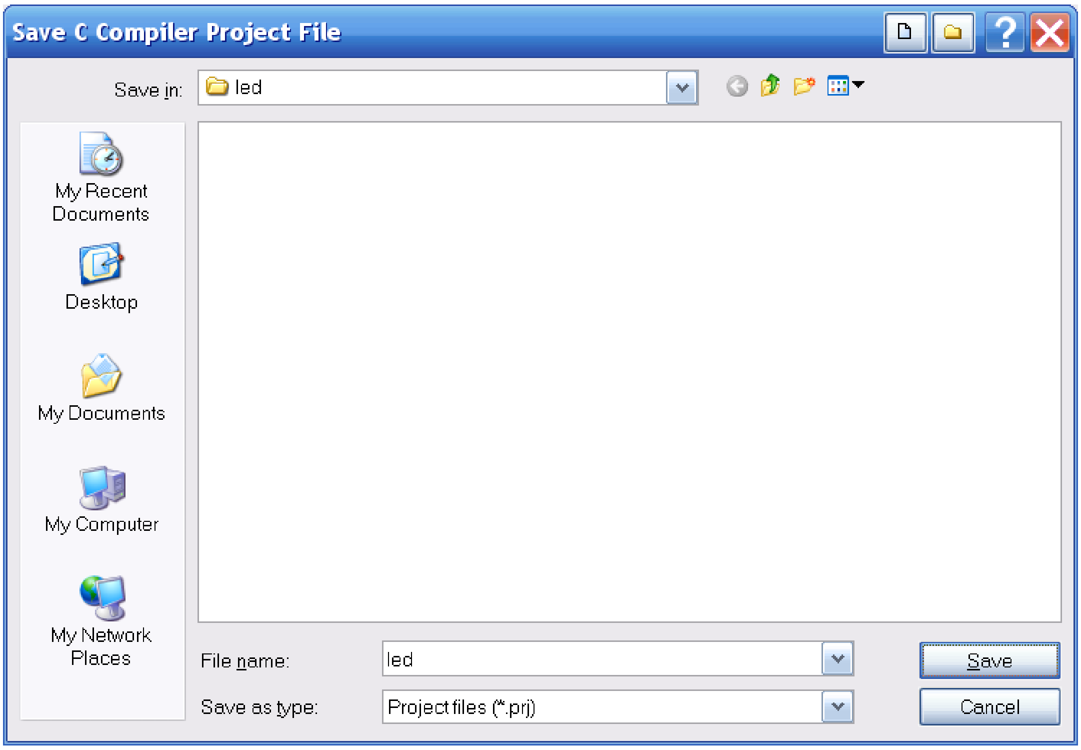
Save in: led

File name: led

Save as type: C Compiler files (*.c)

Create a new folder named “C:\cvavr\led” to hold all the files of our sample project. Open this directory, enter the file name of the C source file, “led.c”, and press the “Save” button. A dialog window for saving the project file, shown in Figure 7-2, will open.

Figure 7-2. Save Project Dialog.



Save in: led

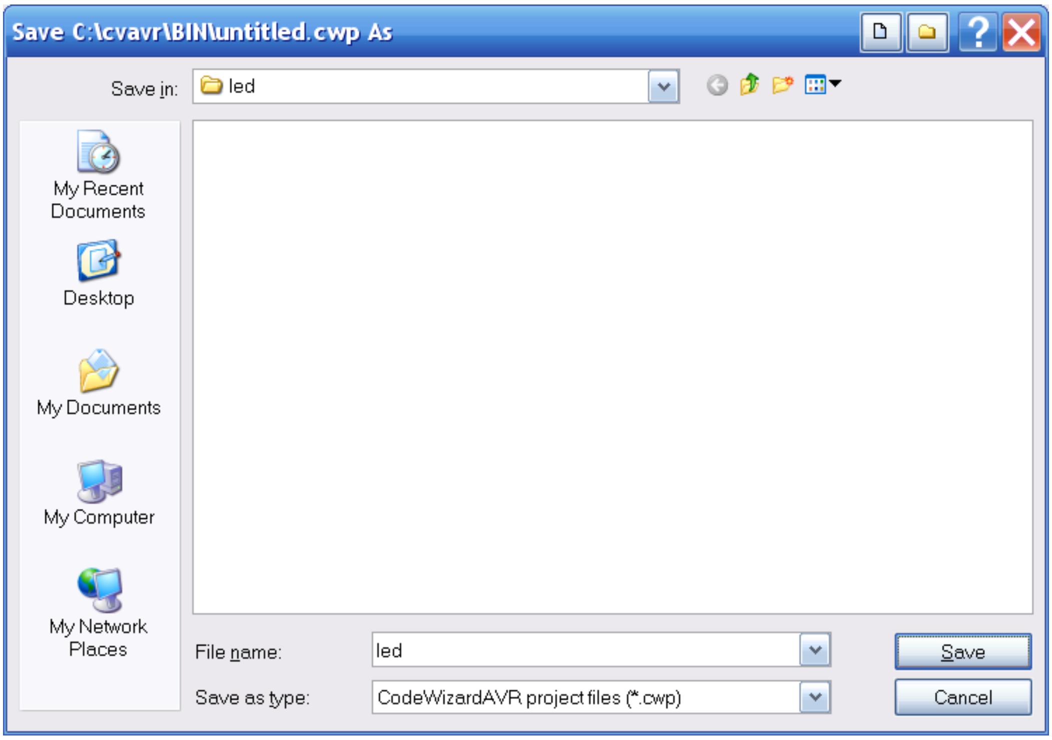
File name: led

Save as type: Project files (*.prj)

Here, specify the file name for the project, “led.prj”, and save it in the same folder as the C file (“C:\cvavr\led”).

Finally, we will be prompted to save the CodeWizardAVR project file, as shown in Figure 7-3. Saving all the CodeWizardAVR peripherals configuration in the “led.cwp” project file will allow us to reuse some of our initialization code in future projects.

Figure 7-3. Save CodeWizardAVR Project Dialog.



Save C:\cvavr\BIN\untitled.cwp As

Save in: led

File name: led

Save as type: CodeWizardAVR project files (*.cwp)

Specify the file name “led.cwp” and press the “Save” button.

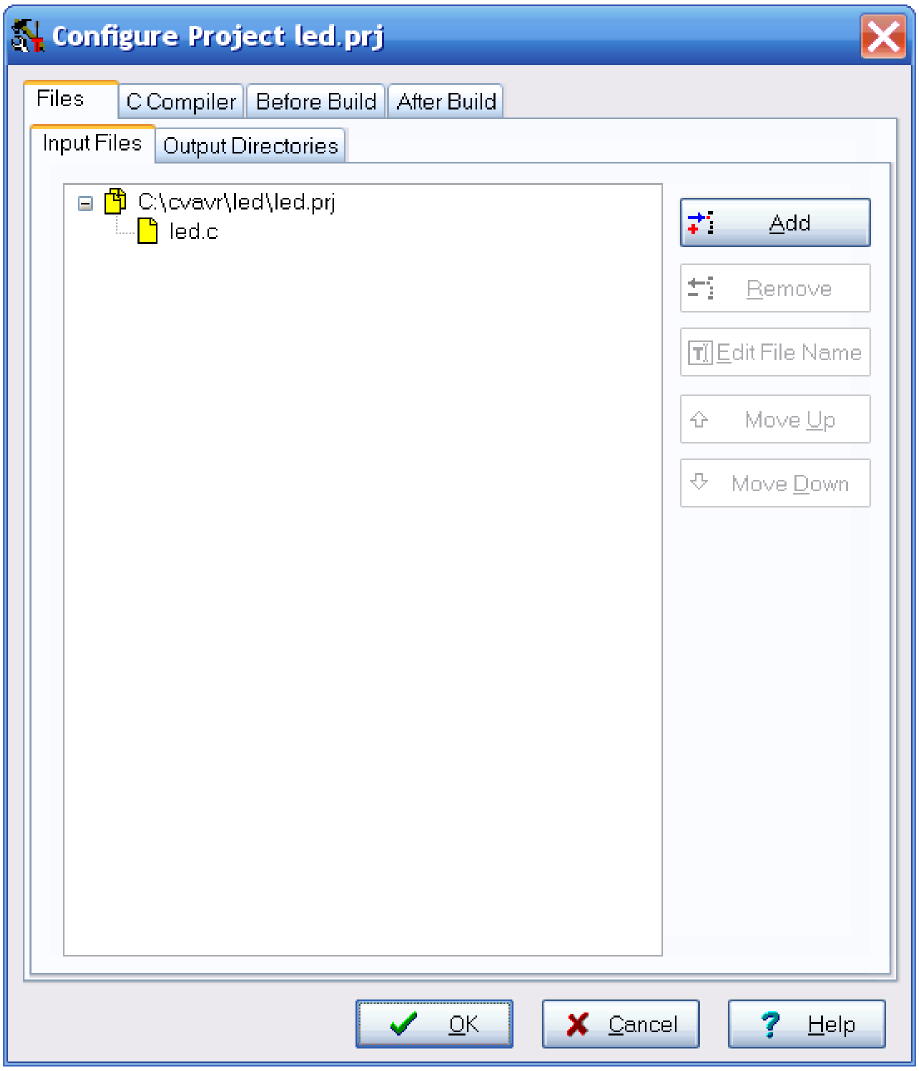
The “led.c” source file will now automatically be opened, and we may start editing the code produced by the CodeWizardAVR. In this example project, only the interrupt handler code needs to be amended to manage the LEDs. This is shown below. The small bit of code that was added is shown with bold font, while the remainder was supplied by the CodeWizardAVR.

```
// the LED 0 on PORTB will be ON
unsigned char led_status=0xFE;

// Timer 1 overflow interrupt service routine interrupt [TIM1_OVF]
void timer1_ovf_isr(void) {
    // Reinitialize Timer 1 value
    TCNT1H=0xF8;
    TCNT1L=0xFB;
    // Place your code here
    // move the LED
    led_status<<=1;
    led_status|=1;
    if (led_status==0xFF) led_status=0xFE;
    // turn ON the appropriate LED
    PORTB=led_status;
}
```

8 Viewing or Modifying the Project Configuration

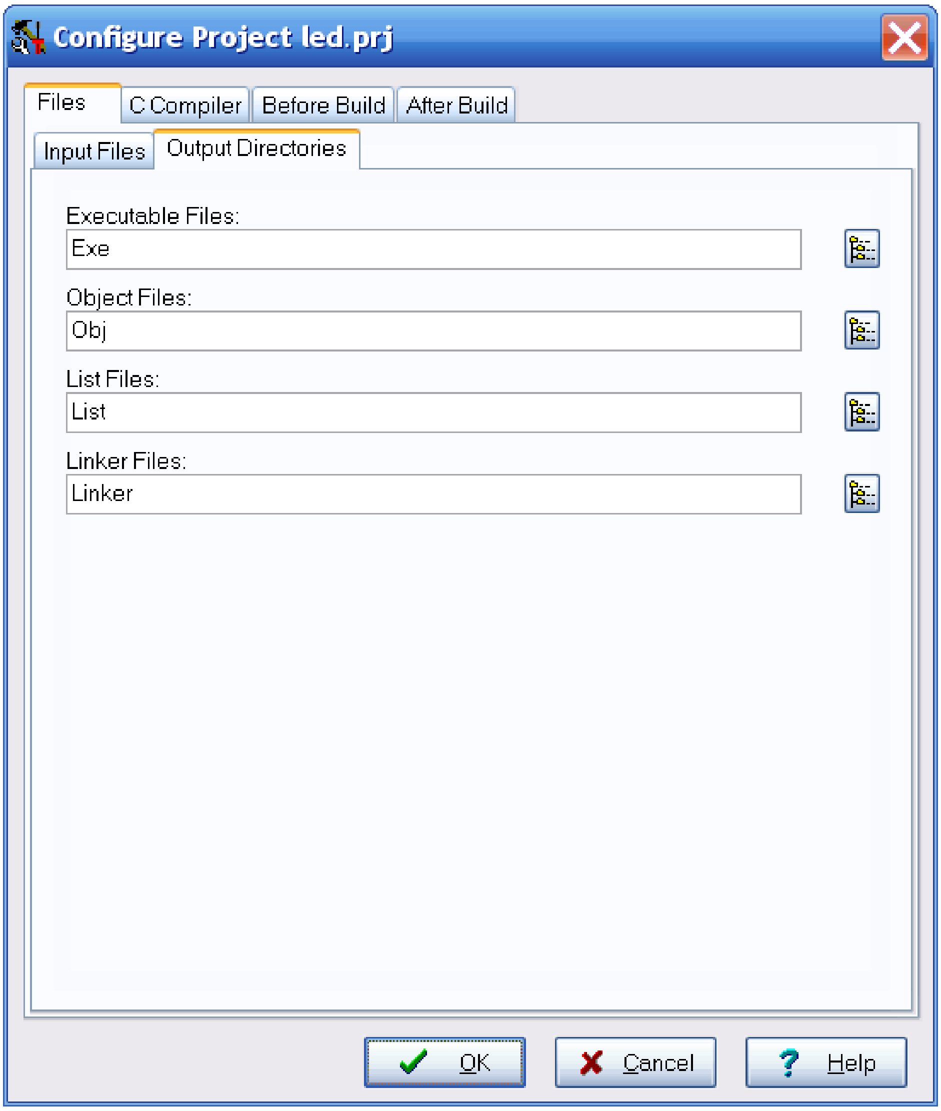
At any time, the project configuration may be changed using the “Project→Configure” menu option or by pressing the toolbar button. This will open the dialog window shown in Figure 8-1.

Figure 8-1. Project Input Files Configuration Tab.



To add or remove files from the project, select the “Files” tab and click the “Add” button, or select a file in the project file tree and click the “Remove” button. If you wish to rename a file in the project, select it in the project file tree and press the “Edit File Name” button.

Figure 8-2. Project Output Directories Configuration Tab.



Executable Files: Exe

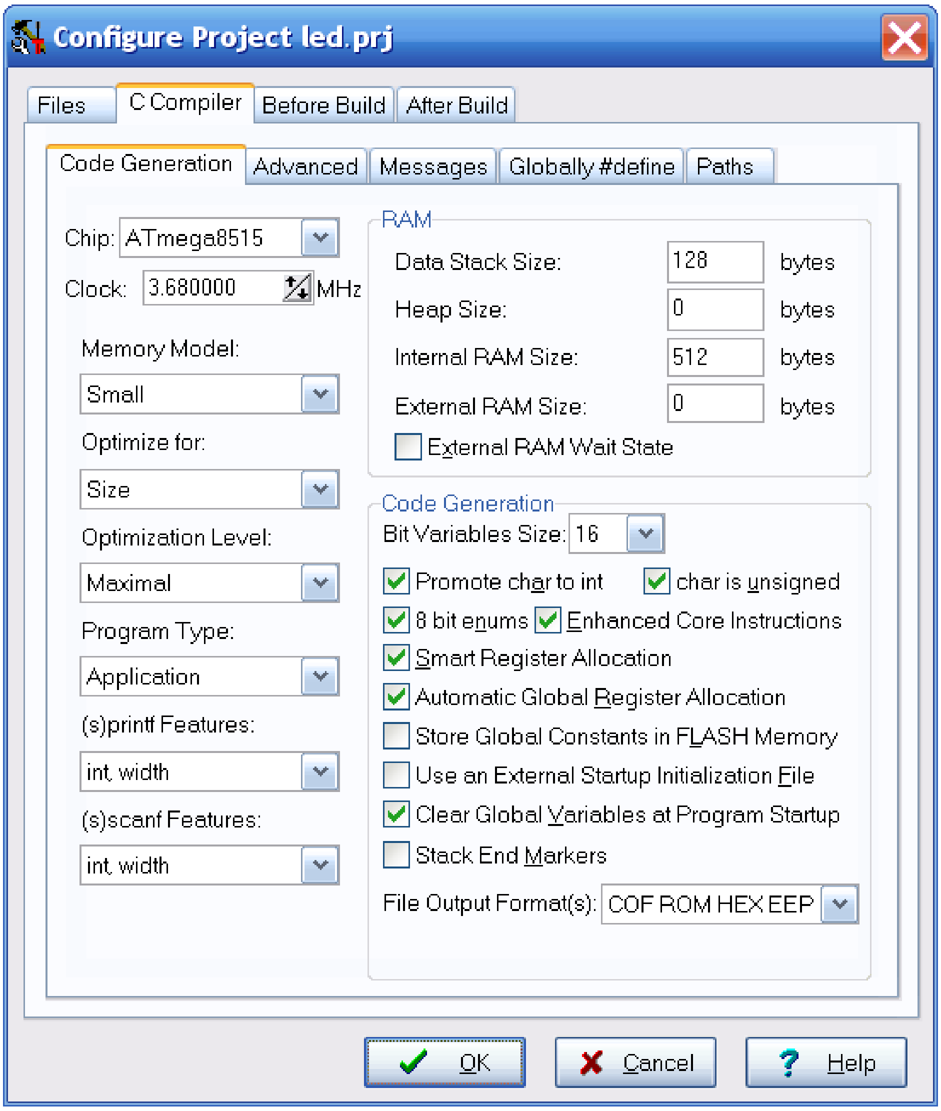
Object Files: Obj

List Files: List

Linker Files: Linker

The “Output Directories” tab, shown in Figure 8-2, allows you to specify in which directories the compiler shall place the files resulting from the Build process. With these default settings, the executable for this example project, “led.hex”, will be located in the directory “C:\cvavr\led\Exe” after a successful build.

Figure 8-3. Project C Compiler Configuration Tab.



Chip: ATmega8515

Clock: 3.680000 MHz

Memory Model: Small

Optimize for: Size

Optimization Level: Maximal

Program Type: Application

(s)printf Features: int, width

(s)scanf Features: int, width

RAM

Data Stack Size: 128 bytes

Heap Size: 0 bytes

Internal RAM Size: 512 bytes

External RAM Size: 0 bytes

□ External RAM Wait State

Code Generation

Bit Variables Size: 16

☑ Promote char to int ☑ char is unsigned

☑ 8 bit enums ☑ Enhanced Core Instructions

☑ Smart Register Allocation

☑ Automatic Global Register Allocation

□ Store Global Constants in FLASH Memory

□ Use an External Startup Initialization File

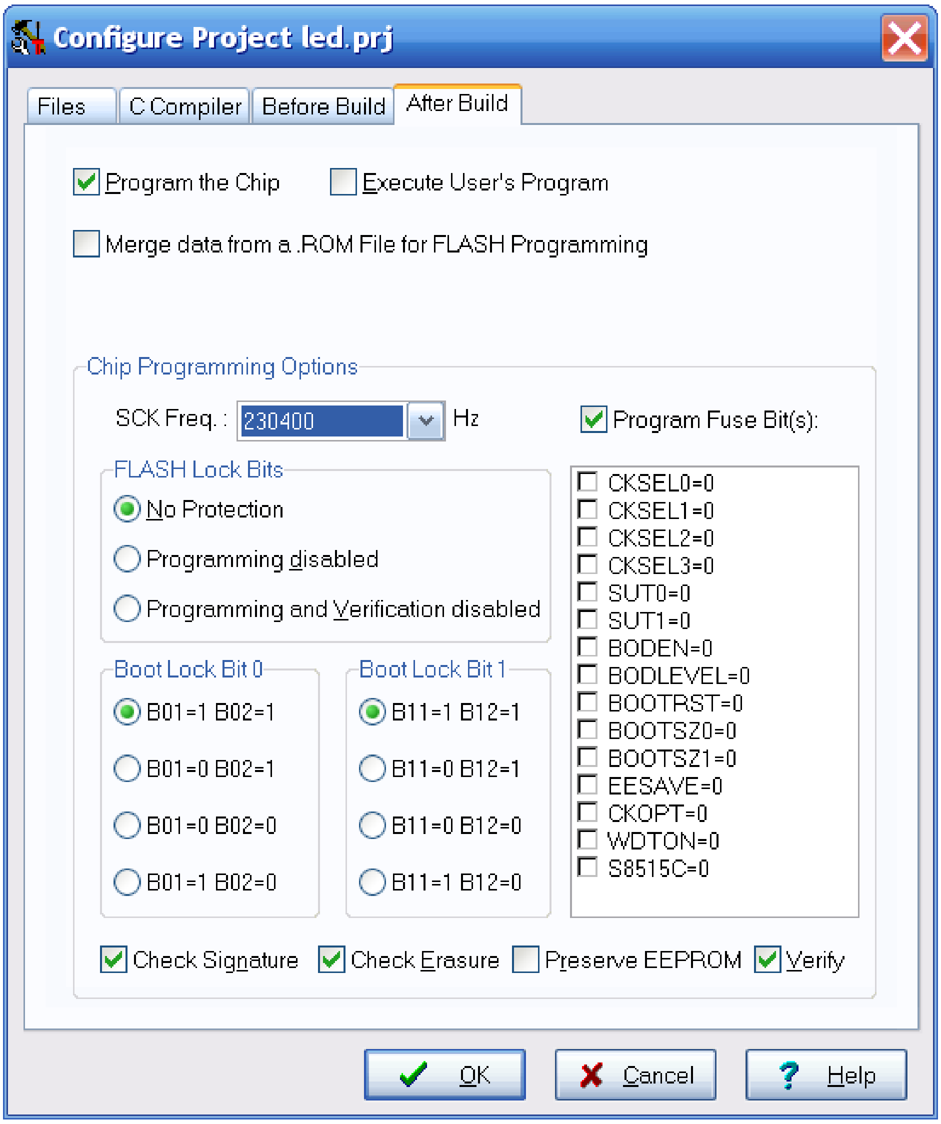
☑ Clear Global Variables at Program Startup

□ Stack End Markers

File Output Format(s): COF ROM HEX EEP

To change the target microcontroller, the clock rate or the various compiler options select the “C Compiler” tab. The dialog window shown in Figure 8-3 opens, and the configuration may be altered.

Figure 8-4. Project After Build Configuration Tab.



Configure Project led.prj

Files | C Compiler | Before Build | After Build

☑ Program the Chip ☐ Execute User's Program

☐ Merge data from a .ROM File for FLASH Programming

Chip Programming Options

SCK Freq. : 230400 Hz

☑ Program Fuse Bit(s):

FLASH Lock Bits

● No Protection
○ Programming disabled
○ Programming and Verification disabled

Boot Lock Bit 0

● B01=1 B02=1
○ B01=0 B02=1
○ B01=0 B02=0
○ B01=1 B02=0

Boot Lock Bit 1

● B11=1 B12=1
○ B11=0 B12=1
○ B11=0 B12=0
○ B11=1 B12=0

☐ CKSEL0=0
☐ CKSEL1=0
☐ CKSEL2=0
☐ CKSEL3=0
☐ SUT0=0
☐ SUT1=0
☐ BODEN=0
☐ BODLEVEL=0
☐ BOOTRST=0
☐ BOOTSZ0=0
☐ BOOTSZ1=0
☐ EESAVE=0
☐ CKOPT=0
☐ WDTON=0
☐ S8515C=0

☑ Check Signature ☑ Check Erasure ☐ Preserve EEPROM ☑ Verify

On the “After Build” tab, shown in Figure 8-4, various actions to be taken after the Build process has completed, may be selected. For the purposes of this example, the “Program the Chip” option must be checked to enable automatic programming of the AVR chip.

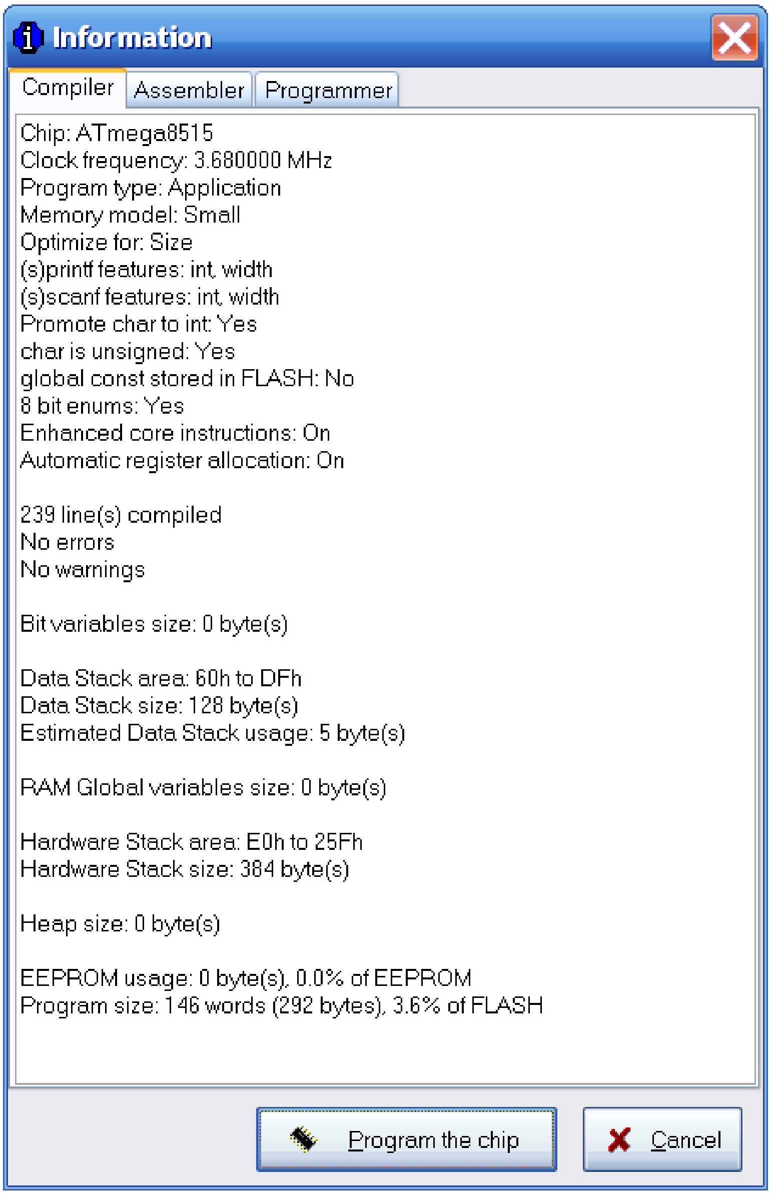
It is important to set the SCK Freq. value to 230400 Hz, so that ATmega8515 chips that come from the factory with the CKSEL3..0 fuse bits set to use the internal 1MHz oscillator, can be successfully programmed.

Please note that the CKSEL3..0 fuse bits will be set to 1111 so that the external 3.68 MHz clock, supplied by the STK500, will be used. (The CKSEL3..0=0 checkboxes should not be checked, or the fuse bits will be programmed to 0.)

9 Building the Project

The “Project” pull-down menu has the “Build” option. Click on it or on the button on the toolbar. When this process is completed, the Information window, shown in Figure 9-1, will be displayed.

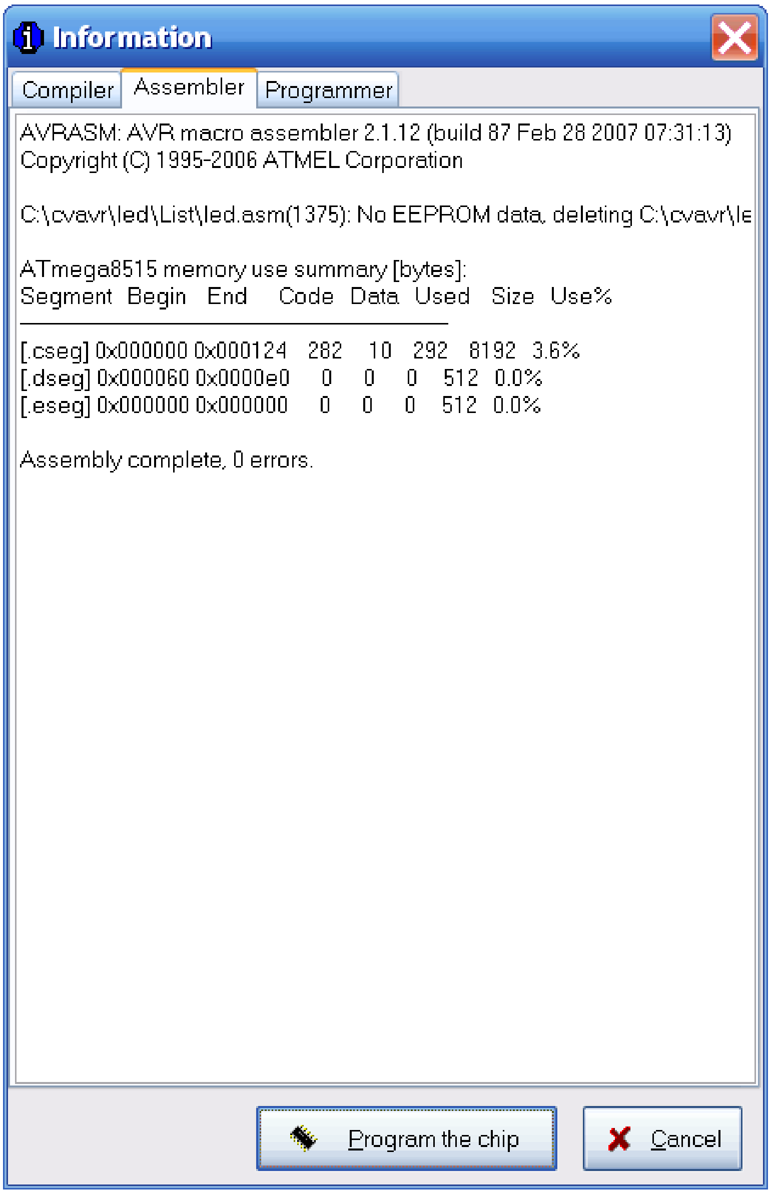
Figure 9-1. Information Window, Compiler Tab.



```
Chip: ATmega8515
Clock frequency: 3.680000 MHz
Program type: Application
Memory model: Small
Optimize for: Size
(s)printf features: int, width
(s)scanf features: int, width
Promote char to int: Yes
char is unsigned: Yes
global const stored in FLASH: No
8 bit enums: Yes
Enhanced core instructions: On
Automatic register allocation: On

239 line(s) compiled
No errors
No warnings

Bit variables size: 0 byte(s)

Data Stack area: 60h to DFh
Data Stack size: 128 byte(s)
Estimated Data Stack usage: 5 byte(s)

RAM Global variables size: 0 byte(s)

Hardware Stack area: E0h to 25Fh
Hardware Stack size: 384 byte(s)

Heap size: 0 byte(s)

EEPROM usage: 0 byte(s), 0.0% of EEPROM
Program size: 146 words (292 bytes), 3.6% of FLASH
```

This window shows the RAM, EEPROM and FLASH memory usage.

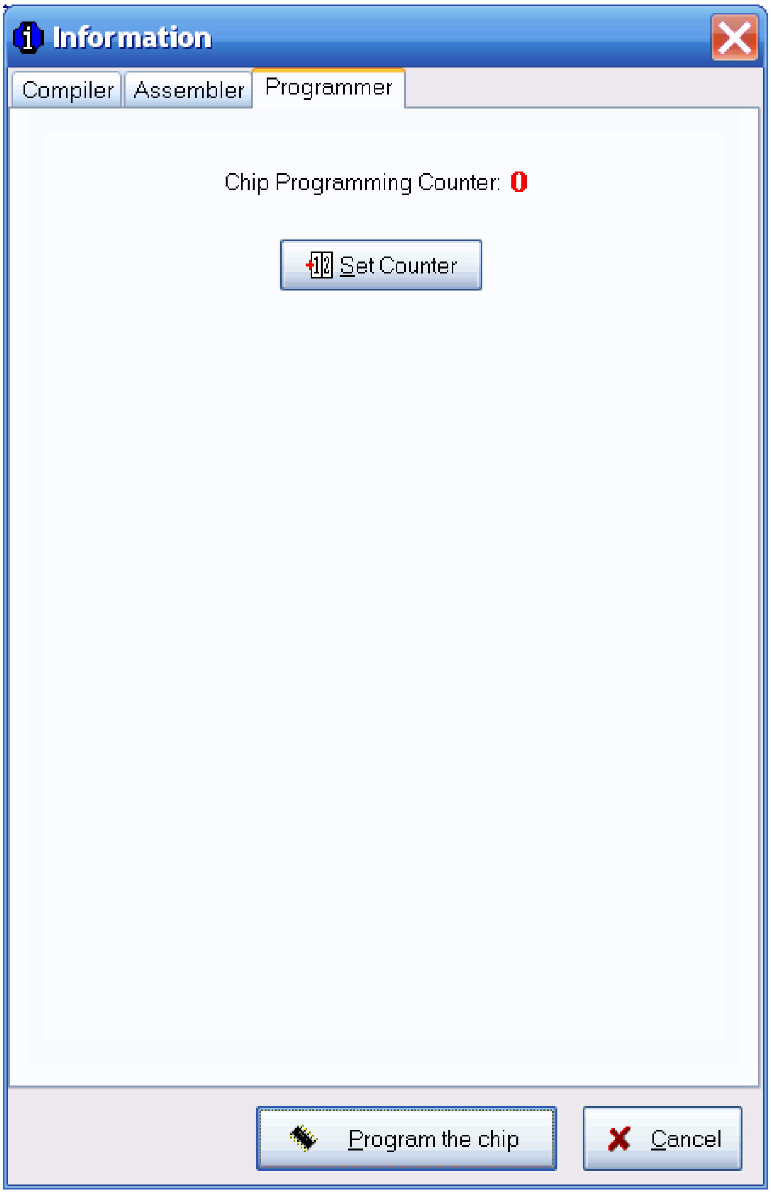
Figure 9-2. Information Window, Assembler Tab.



```
AVRASM: AVR macro assembler 2.1.12 (build 87 Feb 28 2007 07:31:13)
Copyright (C) 1995-2006 ATMEL Corporation

C:\cvavr\led\List\led.asm(1375): No EEPROM data, deleting C:\cvavr\le

ATmega8515 memory use summary [bytes]:
Segment  Begin    End      Code  Data  Used   Size  Use%
---------------------------------------------------------
[.cseg] 0x000000 0x000124   282    10   292   8192  3.6%
[.dseg] 0x000060 0x0000e0     0     0     0    512  0.0%
[.eseg] 0x000000 0x000000     0     0     0    512  0.0%

Assembly complete, 0 errors.
```

If the “Assembler” tab is clicked, the assembly results are displayed as shown in Figure 9-2.

Figure 9-3. Information Window, Programmer Tab.



Chip Programming Counter: 0

Selecting the “Programmer” tab displays the value of the Chip Programming Counter, as shown in Figure 9-3. Pressing the “Set Counter” button will initialize this counter.

If the Build process was successful, power-up the STK500 starter kit and press the “Program the chip” button to start the automatic chip programming. After the programming process is complete, the code will start to execute in the target microcontroller on the STK500 starter kit.

10 Short Reference

10.1 Preparations

1. Install the CodeVisionAVR C Compiler
2. Install the Atmel AVR Studio Debugger
3. Install the Atmel STK500 Starter Kit
4. Configure the STK500 Programmer Support in the CodeVisionAVR IDE by selecting: Settings→Programmer
   AVR Chip Programmer Type: STK500 + corresponding communication port
5. Configure the AVR Studio Support in the CodeVisionAVR IDE by selecting: Settings→Debugger
   “C:\Program Files\Atmel\AVRTools\AVR Studio4\AvrStudio.exe”

10.2 Getting Started

1. Create a new project by selecting:
   File→New→Select Project
2. Specify that the CodeWizardAVR will be used for producing the C source and project files: Use the CodeWizard?→Yes
3. In the CodeWizardAVR window specify the chip type and clock frequency:
   Chip→Chip: ATmega8515, Clock: 3.86MHz
4. Configure the I/O Ports: Ports→Port B→
   Data Direction: all Outputs, Output Value: all 1's
5. Configure Timer1: Timers→Timer1→
   Clock Value: 3.594kHz, Interrupt on: Timer1 Overflow, Value: F8FB hexadecimal
6. Generate the C source, C project and CodeWizardAVR project files by selecting:
   File→Generate, Save and Exit→
   Create new directory: “C:\cvavr\led”→
   Save: “led.c”, Save: “led.prj” , Save: “led.cwp”
7. Edit the C source code
8. View or Modify the Project Configuration by selecting
   Project→Configure→
   After Build→Program the Chip → SCK Frequency: 230400Hz
9. Compile the program by selecting:
   Project→ Build
10. Automatically program the ATmega8515 chip on the STK500 starter kit:
    Apply power→Information→Program chip.

Headquarters

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

International

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

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel 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

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
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. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.