German-Russian Institute of Advanced Technologies

TU-Ilmenau (Germany) and KNTRU-KAI (Kazan, Russia)

Guidelines for laboratory work №2 of the subject

«Computer systems»

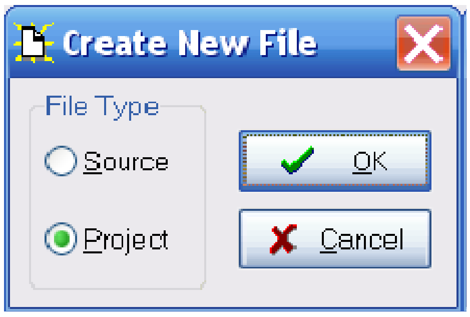
«CodeVisionAVR C Compiler Introduction and Exercises»

Kazan 2014

CodeVisionAVR C Compiler Introduction and Exercises

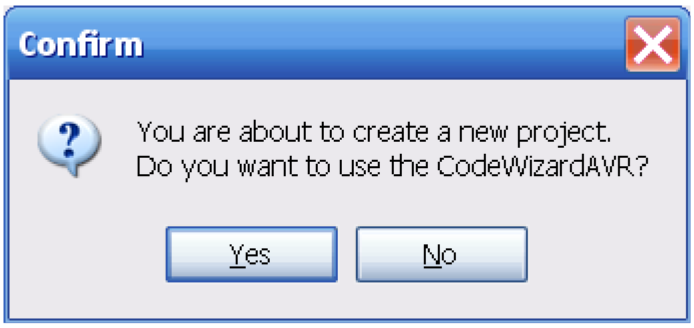
Creating a New Project

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.



New Project Dialog.

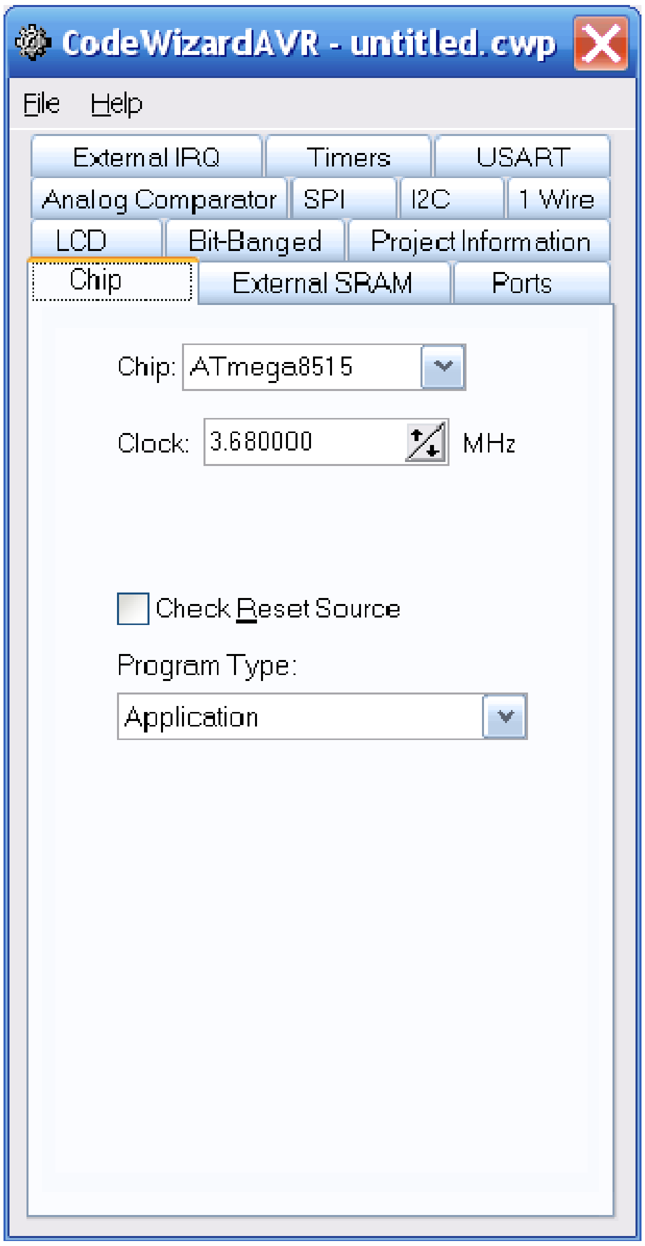
Select "Project", press "OK", and the dialog window will be displayed.


Confirm

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

Confirmation Dialog.

Press "Yes" to use the CodeWizardAVR Automatic Program Generator, and the dialog window will open.



Chip: ATmega8515

Clock: 3.680000 MHz

☐ Check Reset Source

Program Type: Application

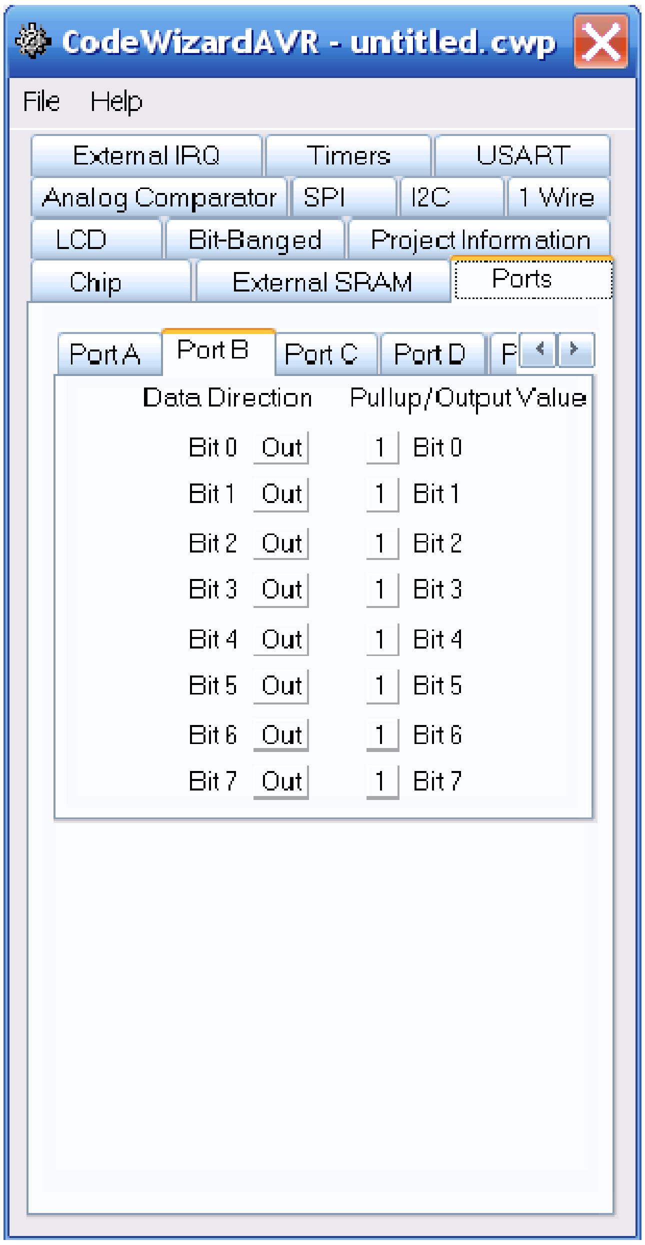
Chip Settings for CodeWizardAVR.

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

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.

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.

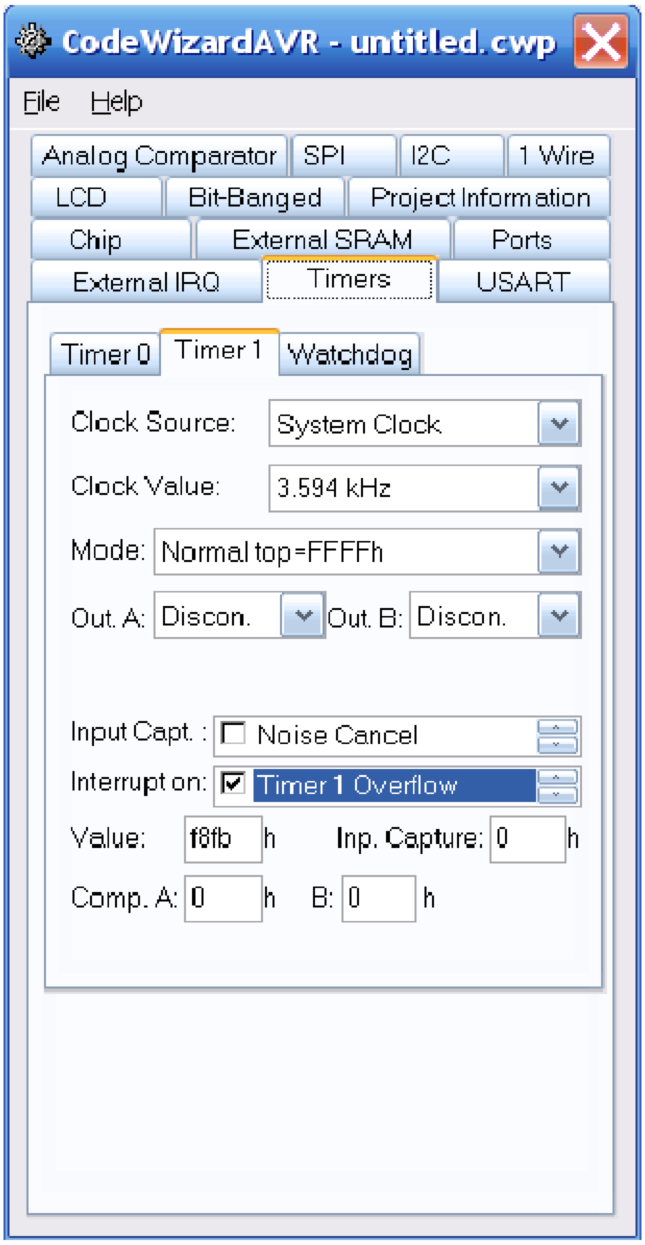


Data Direction		Pullup/Output Value	
Bit 0	Out	1	Bit 0
Bit 1	Out	1	Bit 1
Bit 2	Out	1	Bit 2
Bit 3	Out	1	Bit 3
Bit 4	Out	1	Bit 4
Bit 5	Out	1	Bit 5
Bit 6	Out	1	Bit 6
Bit 7	Out	1	Bit 7

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

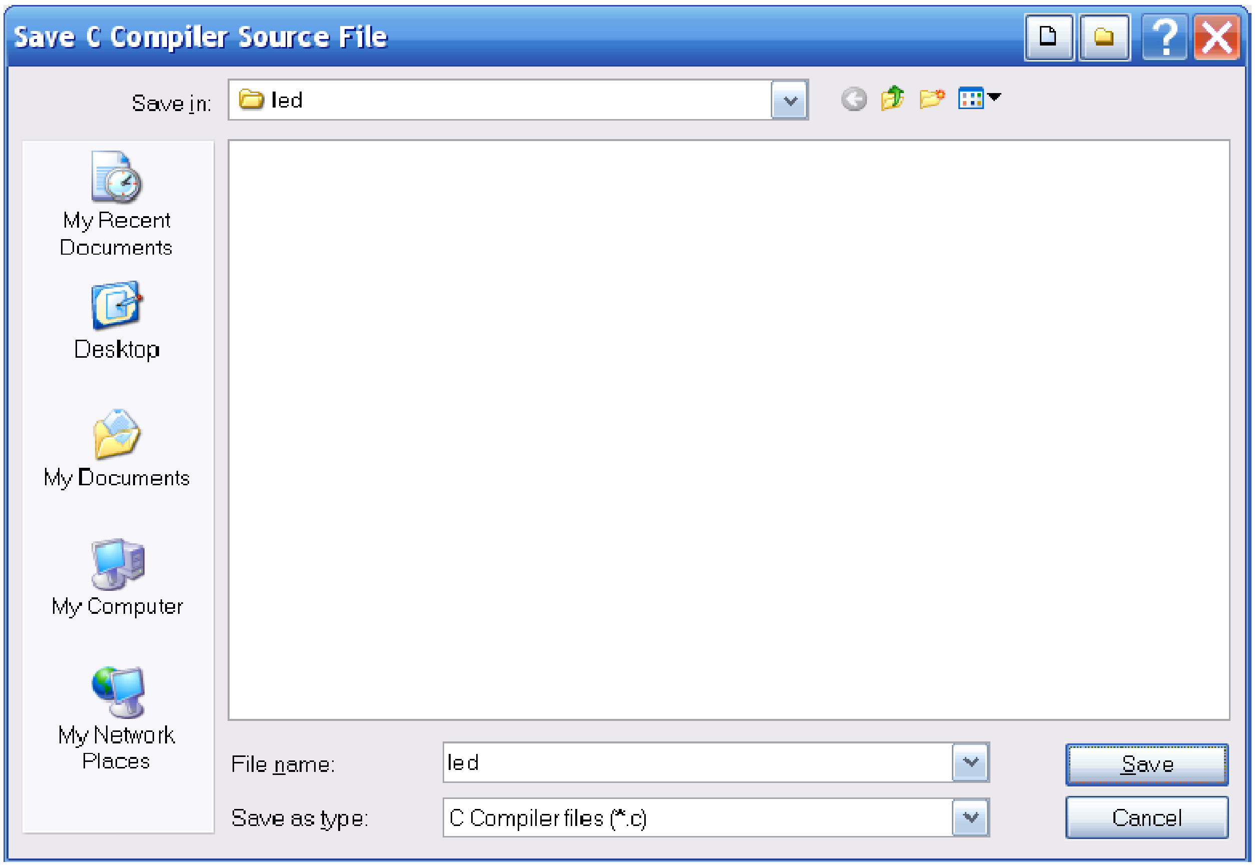
For this project, we want to configure Timer1 to generate overflow interrupts as shown in Figure 6-1.

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.



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

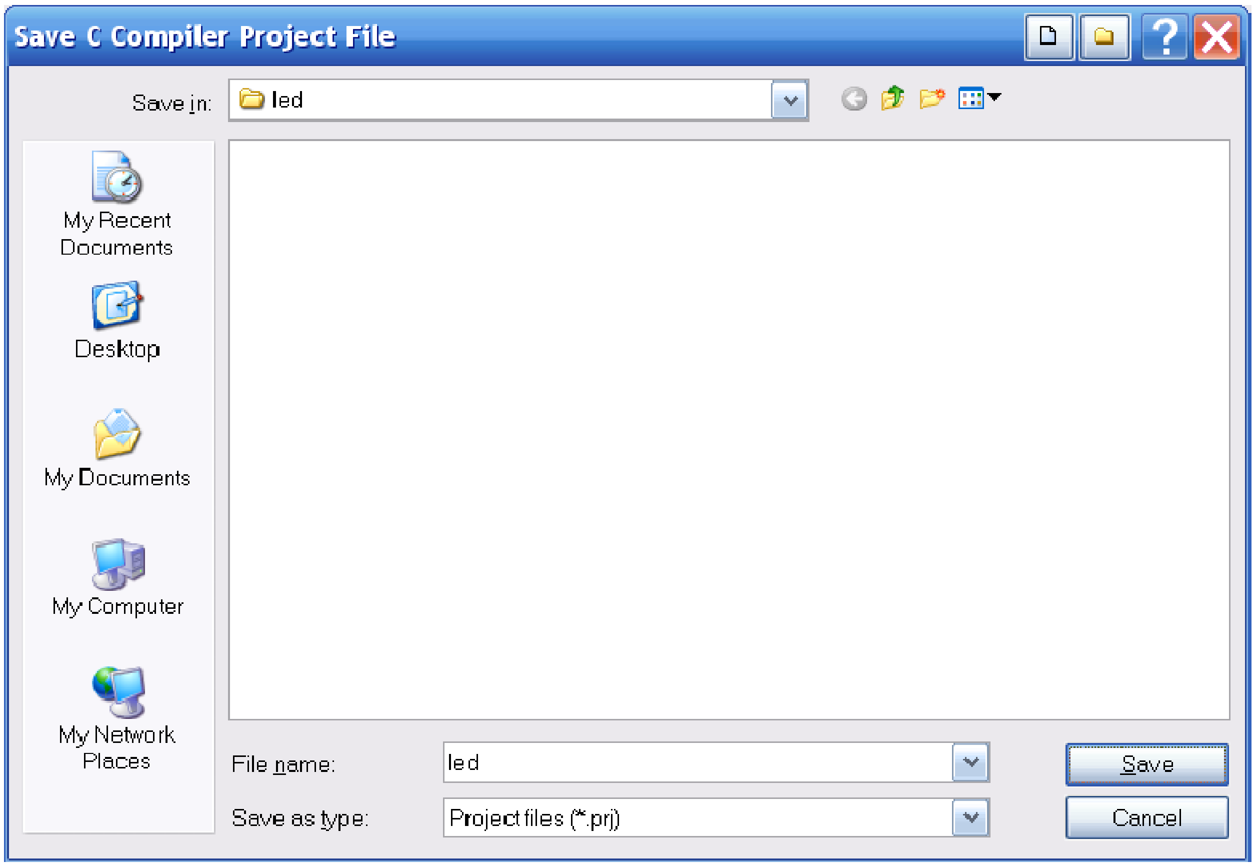
By selecting the “File→Generate, Save and Exit” menu option, the CodeWizardAVR will generate a skeleton C program with, in this case, Port B and Timer1 Overflow Interrupt set up correctly. A dialog window for saving the source code will then open.



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 will open.


Save C Compiler Project File

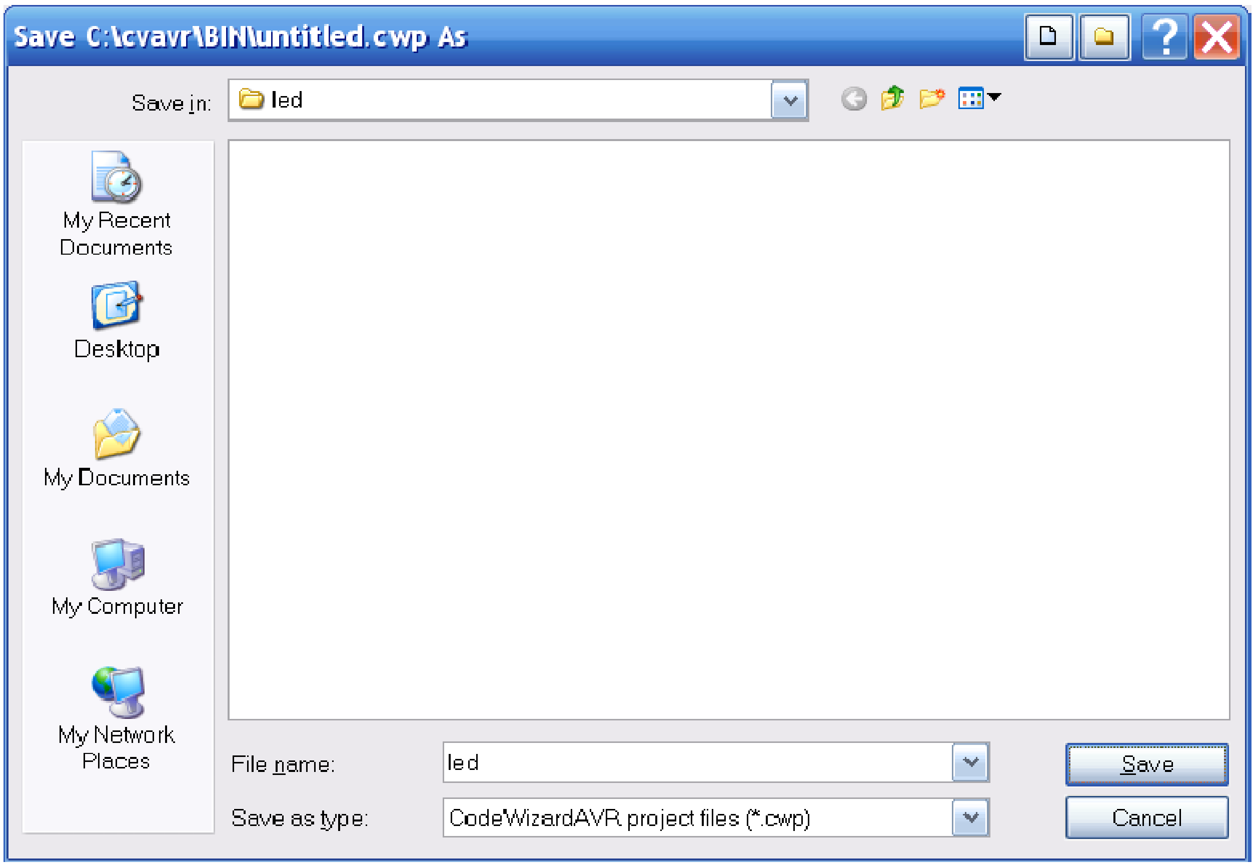
Save in: led

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. 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.



File name: led

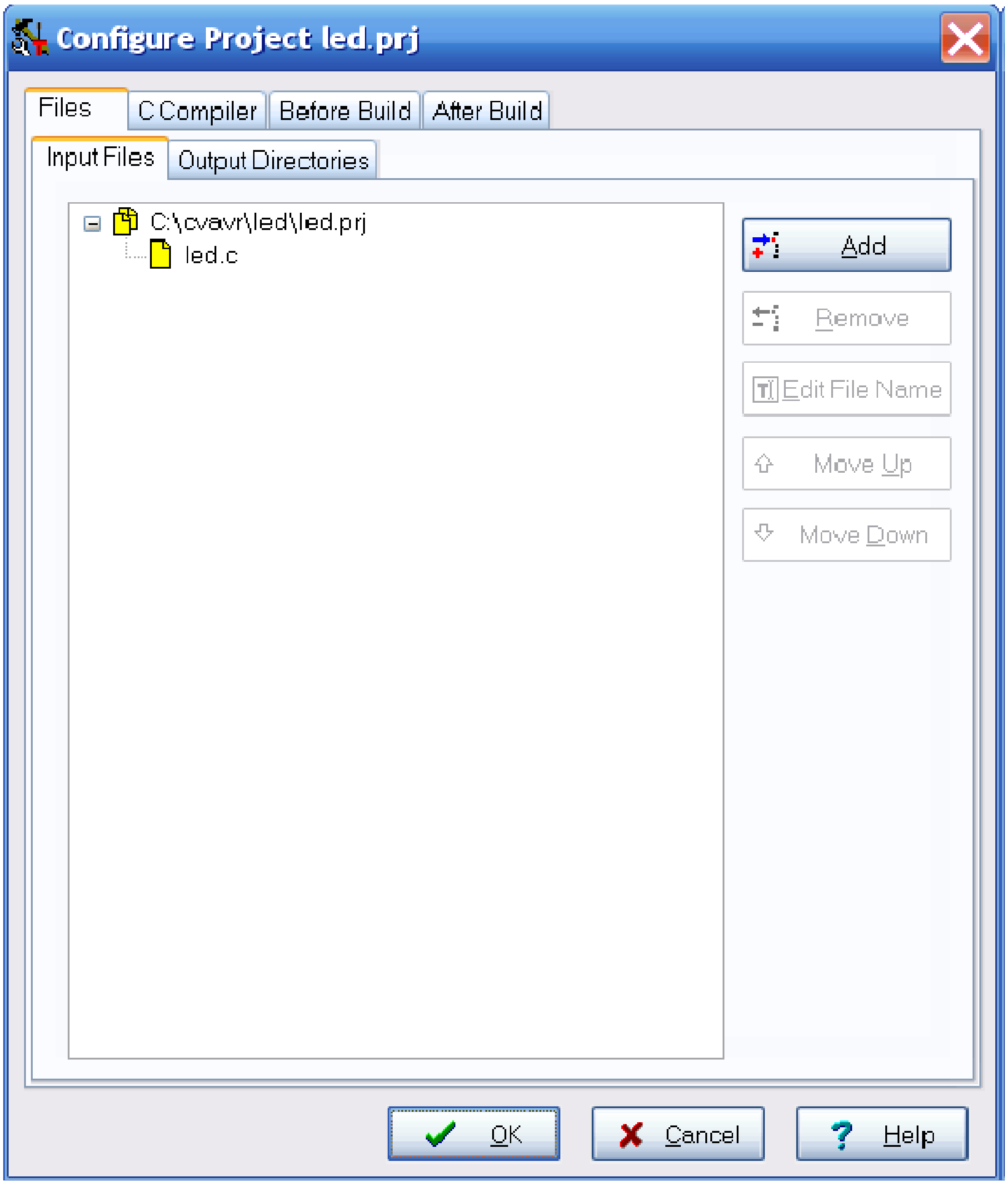
Save as type: CodeWizardAVR project files (*.cwp)

Specify the file name "led.cwp" and press the "Save" button.

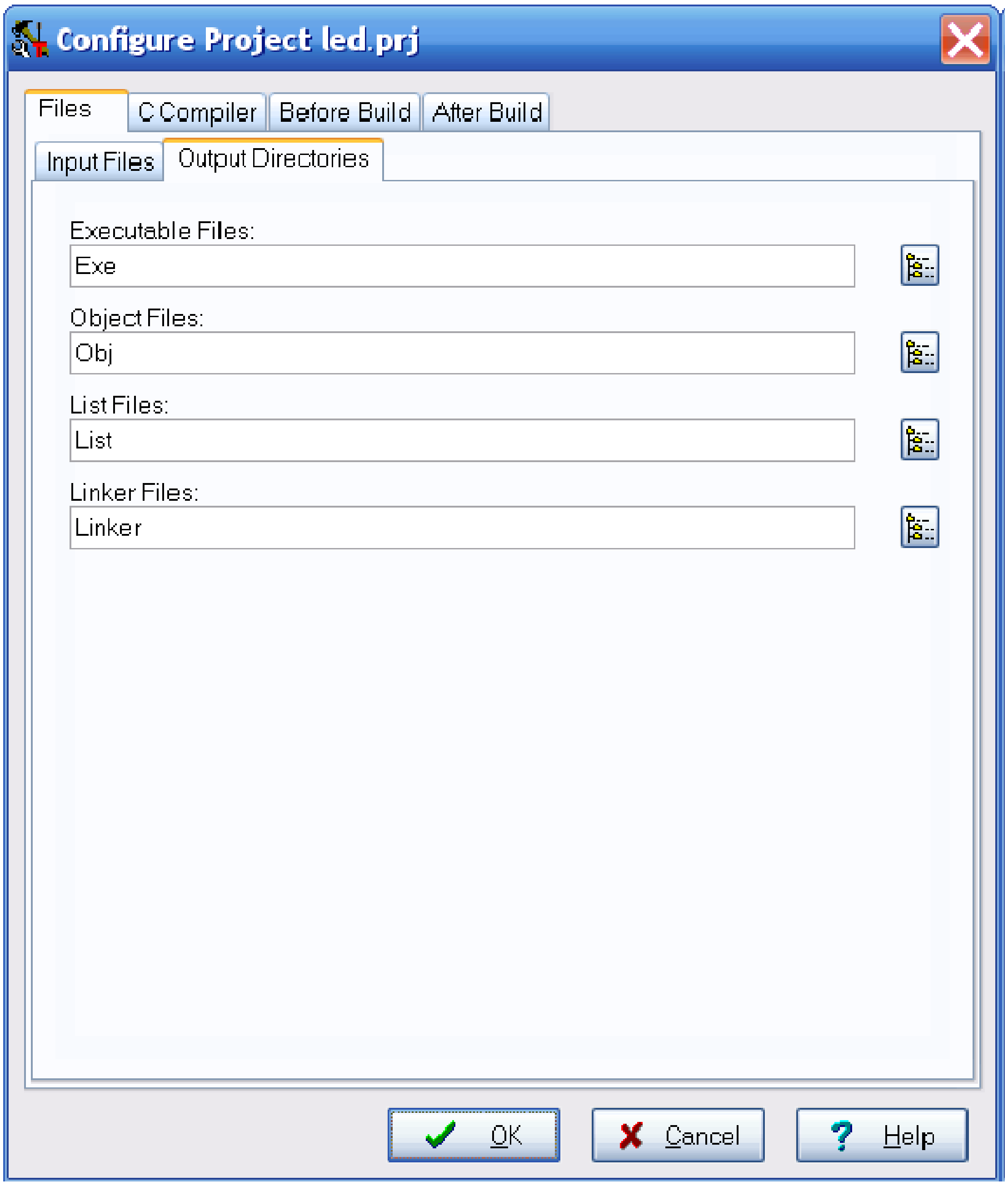
The "led.c" source file will now automatically be opened, and we may start editing the code produced by the CodeWizardAVR. In this example project, only the interrupt handler code needs to be amended to manage the LEDs. This is shown below. The small bit of code that was added is shown with bold font,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

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.



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.



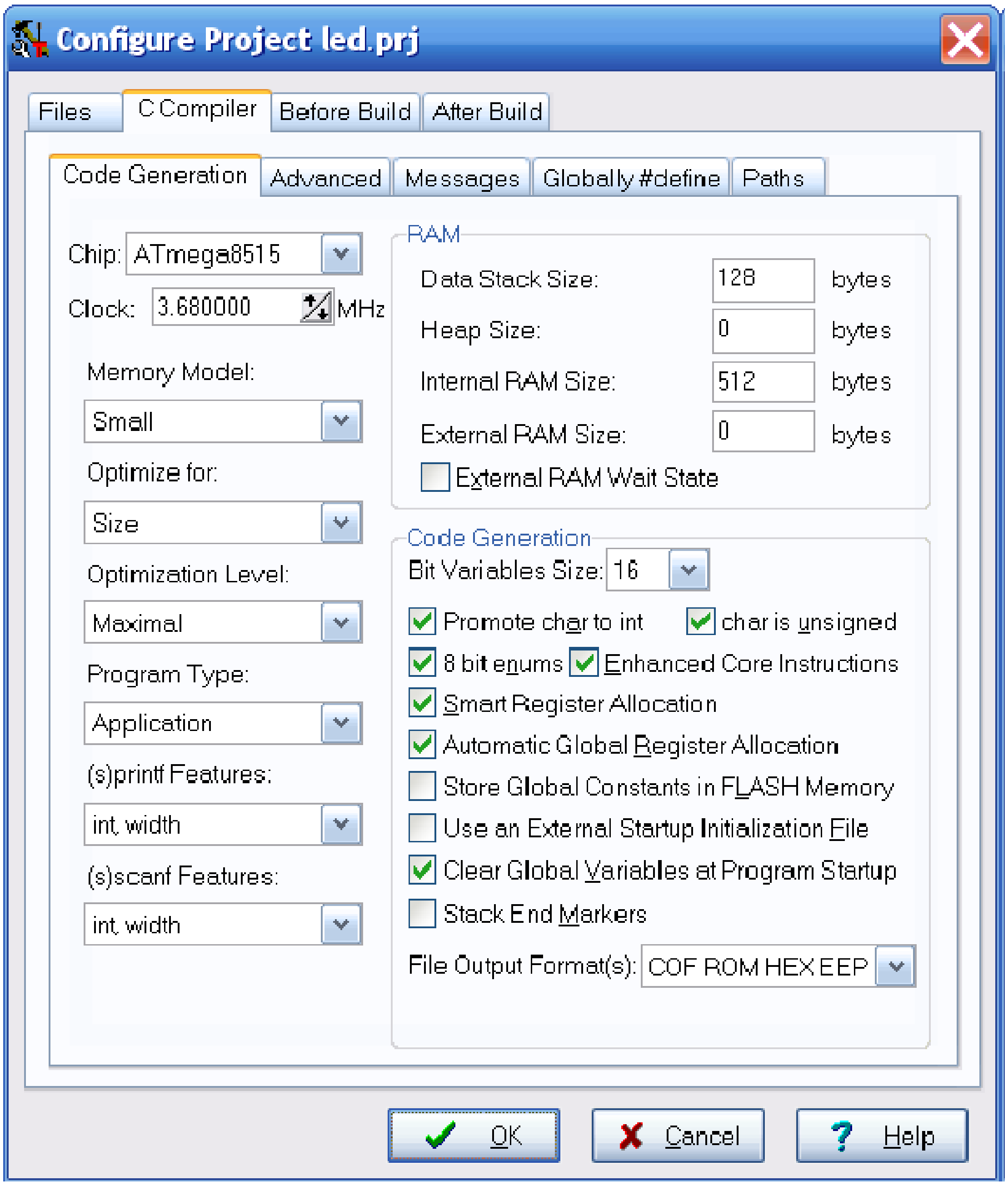
Executable Files: Exe

Object Files: Obj

List Files: List

Linker Files: Linker

The “Output Directories” tab, 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.



Configure Project led.prj

Files | C Compiler | Before Build | After Build

Code Generation | Advanced | Messages | Globally #define | Paths

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

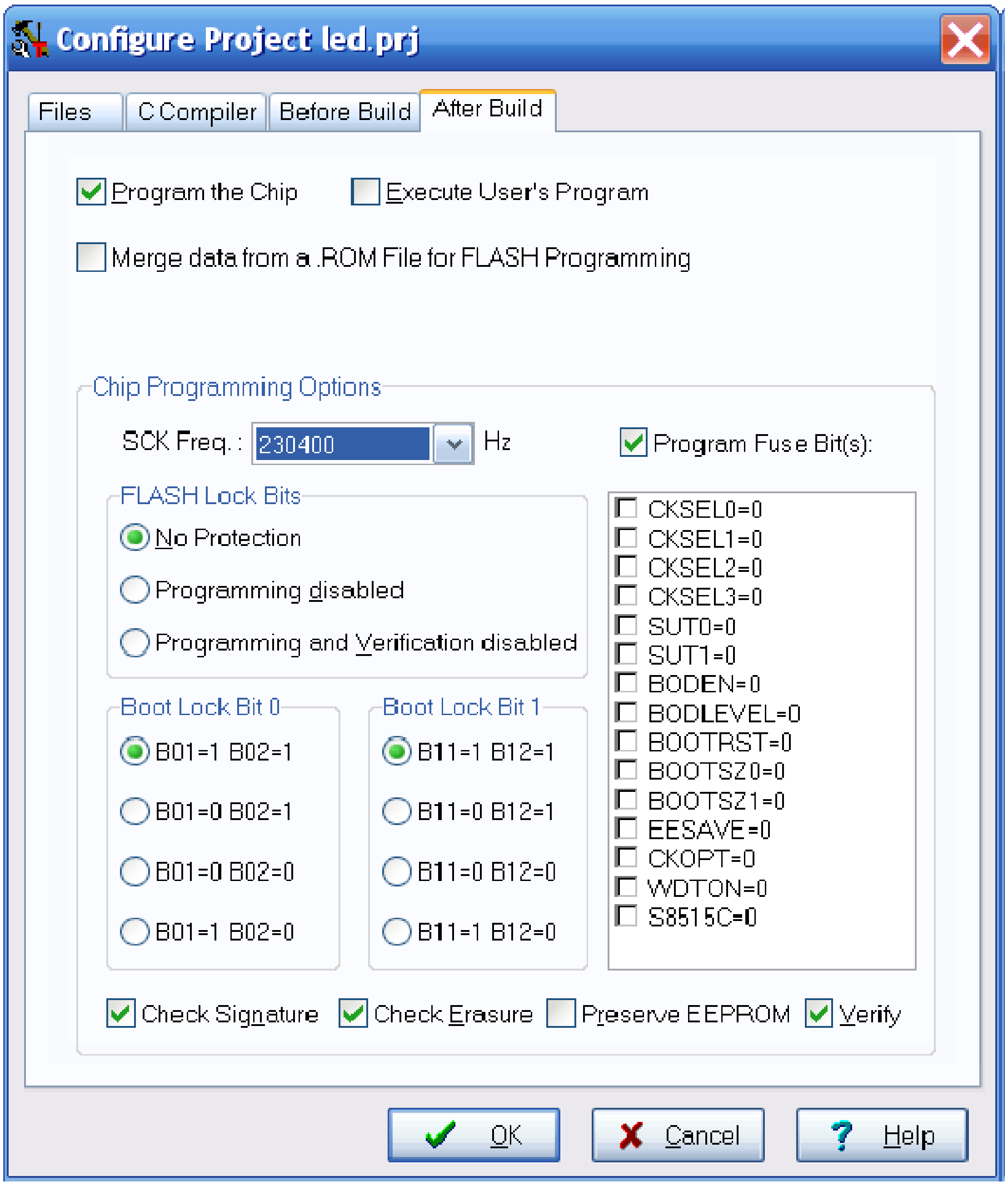
OK | Cancel | Help

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

On the “After Build” tab, 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.)


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

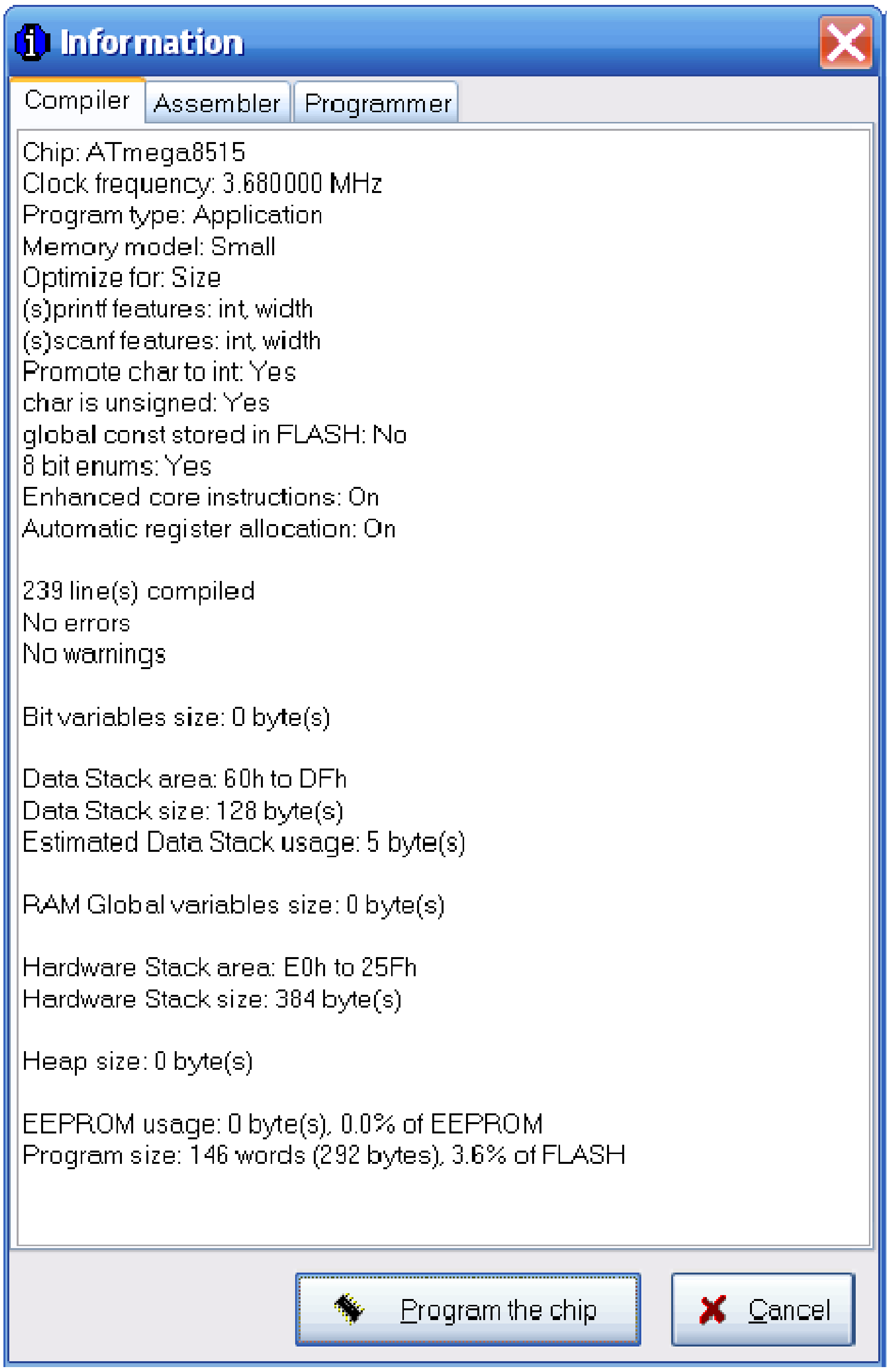
Boot Lock Bit 0
- [● B01=1 B02=1]
- B01=0 B02=1
- B01=0 B02=0
- B01=1 B02=0

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

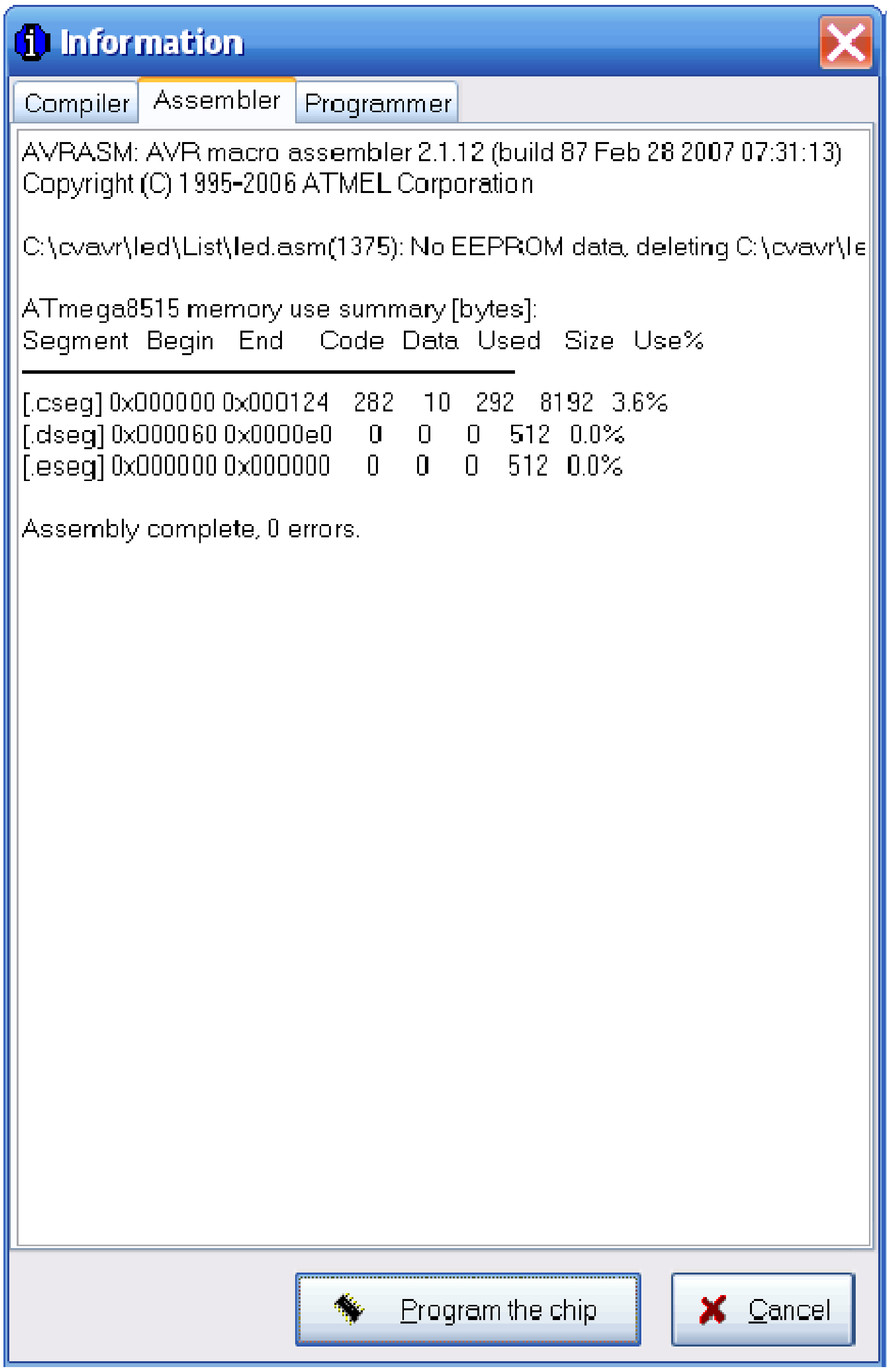
[✓] Check Signature [✓] Check Erasure [ ] Preserve EEPROM [✓] Verify

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 will be displayed.



```
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.



```
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.

Selecting the “Programmer” tab displays the value of the Chip Programming Counter. 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.

TASKS

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.

CONTROL QUESTIONS

1. How to create a new project?

2. How to build a project?

3. How to configure a project?

4. How to compile a project?

REPORT FORM

German-Russian Institute of Advanced Technologies

TU-Ilmenau (Germany) and KNTRU-KAI (Kazan, Russia)

Laboratory work 2

« CodeVisionAVR C Compiler Introduction and Exercises»

Student: ____________________

Teacher: ____________________

Kazan 2014