AVR033: Getting Started with the CodeVisionAVR C Compiler

Features

• Installing and Configuring CodeVisionAVR to Work with the Atmel STK500 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

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 AT90S8515 microcontroller on the STK500 starter kit.

Preparation

Install the CodeVisionAVR C Compiler in the default directory: C:\cvavr.
Install the Atmel AVR Studio debugger in the default directory:
   C:\Program Files\Atmel\AVR Studio.

The demonstration program to be developed in the next few pages requires an Atmel AT90S8515 microcontroller and the STK500 starter kit.
Set up the starter kit according to the instructions in the STK500 User Guide.
Make sure the power is off and insert the AT90S8515 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. This will allow displaying the state of AT90S8515’s PORTB outputs.
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.

In order to use this feature, one supplementary setting must be done:
Open the CodeVisionAVR IDE and select the “Settings|Programmer” menu option. The dialog window as shown in Figure 1 will open.
Figure 1. Programmer Settings

Make sure to select as Chip Programmer Type the Atmel STK500 AVR and the corresponding Communication Port that is used with the STK500 starter kit.

Then press the “STK500.EXE Directory” button in order to specify the location of the stk500.exe command line utility supplied with AVR Studio.

The dialog window as shown in Figure 2 will open.

Figure 2. Directory Selection

Select the “c:\Program Files\Atmel\AVR Studio\STK500” directory and press the “OK” button.

Then press once again the “OK” button in order to save the Programmer Settings.

In order to be able to invoke the AVR Studio debugger from within the CodeVisionAVR IDE one final setting must be done.
Select the “Settings|Debugger” menu option. The dialog window as shown in Figure 3 will open.

**Figure 3. Debugger Settings**

Enter “C:\Program Files\Atmel\AVR Studio\AvrStudio.exe” and press the “OK” button.

Creating a New Project

In order to create a new project, select the “File|New” menu option or press the toolbar button.

The window shown in Figure 4 will be displayed.

**Figure 4. New Project Window**

Select “Project” and press “OK”.

Then the window shown in Figure 5 will be displayed.

**Figure 5. Confirmation**

Press “Yes” to use the CodeWizardAVR Automatic Program Generator.
Using the CodeWizardAVR Automatic Program Generator

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

Figure 6. Selections

The window shown in Figure 6 opens and, for this example project, we shall select the AT90S8515 microcontroller and set the clock rate to 3.68 MHz since that is the clock on the STK500 starter kit.
Configuring the Input/Output Ports

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

Figure 7. I/O Ports Initialization

The default setting is to have the ports for all the target systems to be inputs (Data Direction bits to be all 1s) in their Tri-state mode.

For this exercise, we want to set Port B (by selecting the Port B tab) to be all outputs and we do this by setting all the Data Direction bits to 0 (by clicking on them). We also set the Output Values to be all 1s since this corresponds to the LEDs on the STK500 being off.
Configuring Timer1

For this project, we want to configure Timer1 to generate overflow interrupts. We select the Timers tab and then select the Timer1 tab resulting in Figure 8.

**Figure 8. Timer Tab**

Set the options as shown in Figure 8. We have selected a clock rate of 3.594 kHz (the system clock of 3.68 MHz divided by 1024).

The timer is set to operate in the default “Output Compare” mode and to generate interrupts on overflow.

To obtain the frequency of LED movement of two per second we need to reinitialize the Timer1 value to $0x10000-(3594/2) = 0xF8FB$ on every overflow.

Completing the Project

By selecting the File|Generate, Save and Exit menu option the CodeWizard will generate a skeleton C program with, in this case, the Port B and Timer1 Overflow Interrupt set up correctly.

The dialog window shown in Figure 9 will appear.

**Figure 9. Save Source File Dialog Box**
By pressing the button, a new directory C:\cvavr\led must be created. It will hold all the files of our sample project.

Then we must specify the File name of the C source file: led.c and press the “Save” button.

A new dialog window will open. This is shown in Figure 10.

**Figure 10. File Name Specification**

Here, we must specify the File name led.prj, as the project name and put it in the same folder: C:\cvavr\led.

Finally, we will be prompted to save the CodeWizard project file, as shown in Figure 11.

**Figure 11. File Save Prompt**

We must specify the File name as led.cwp and press the “Save” button.

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.

The led.c source file is now automatically opened and available.

One can then start editing the code produced by the CodeWizardAVR.

The source listing is given on Appendix A of this application note.

In this example, only the interrupt handler code needs to be amended to manage the LED display.

The small bit of code that was added is shown with bold font, the remainder was supplied by the CodeWizardAVR.
Viewing or Modifying the Project Configuration

At any time, a project configuration may be changed using the Project|Configure menu option or by pressing the toolbar button.

The dialog window shown in Figure 12 will open.

**Figure 12.** Configure Window Dialog Box

![Configure Window Dialog Box](image)

To add, respectively remove, files from the project select the “Files” tab and use the “Add”, respectively “Remove” buttons.

To change the target microcontroller, the clock rate or the various compiler options select the “C Compiler” tab.

The dialog box shown in Figure 13 opens and the configuration may be altered.

**Figure 13.** C Compiler Configuration

![C Compiler Configuration](image)
We may also select whether we wish to automatically program the target microprocessor after the Make or not.

This is chosen by selecting the “After Make” tab, which gives us the next window, shown in Figure 14.

**Figure 14. After Make Configuration**

For the purposes of this example, “Program the Chip” option must be checked.

This will enable automatic programming of the AVR chip after the Make is complete.
Making the Project

The “Project” Pull-down menu gives the Make option. Click on it or on the button on the toolbar.

After a successful compile and assembly, the Information window will be displayed as shown in Figure 15.

Figure 15. Information Window

This window shows how the compiler used the RAM memory.

If the Assembler tab is clicked, the Assembler window shows the size of the assembled code as shown in Figure 16.
Figure 16. Assembler Information

Selecting the Programmer tab displays the value of the Chip Programming Counter. Pressing the Set Counter button can initialize this counter.
If the Make process was successful, then power-up the STK500 starter kit and press the Program 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.

Short Reference

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→Specify STK500.EXE Directory: C:\Program Files\Atmel\AVR Studio\STK500→Communication Port
5. Configure the AVR Studio Support in the CodeVisionAVR IDE by selecting: Settings→Debugger→Enter: C:\Program Files\AtmelAVR Studio
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: AT90S8515→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→Val: 0xF8FB
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 Make→Program the Chip
9. Compile the program by selecting:
   Project→Make
10. Automatically program the AT90S8515 chip on the STK500 starter kit:
    Apply power→Information→Program.

Appendix A - The Source Code

/******************************
This program was produced by the
CodeWizardAVR V1.0.1.8c Standard
Automatic Program Generator
© Copyright 1998-2001
Pavel Haiduc, HP InfoTech S.R.L.
http://infotech.ir.ro
e-mail: hpinfotech@xnet.ro, hpinfotech@xmail.ro

Project :
Version :
Date :
Author :
Company :
Comments :

Chip type : AT90S8515
Clock frequency : 3.680000 MHz
Memory model : Small
Internal SRAM size : 512
External SRAM size : 0
Data Stack size : 128
*******************************/

#include <90s8515.h>
// 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’s 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 LED
    PORTB=led_status;
}

void main(void)
{
    // Input/Output Ports initialization
    // Port A
    PORTA=0x00;
    DDRA=0x00;

    // Port B
    PORTB=0xFF;
    DDRB=0xFF;

    // Port C
    PORTC=0x00;
    DDRC=0x00;

    // Port D
    PORTD=0x00;
    DDRD=0x00;

    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: Timer 0 Stopped
    // Mode: Output Compare
    // OC0 output: Disconnected
    TCCR0=0x00;
    TCNT0=0x00;

    // Timer/Counter 1 initialization
    // Clock source: System Clock
    // Clock value: 3.594 kHz
// Mode: Output Compare
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x05;
TCNT1H=0xF8;
TCNT1L=0xFB;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x80;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;

// Global enable interrupts
#asm("sei")

// the rest is done by TIMER1 overflow interrupts
while (1);
}