WinAVR C Tutorial (within AVR Studio)

(for Version 4)

Updated - 4 September, 2008

Author: Peter Sutton

This tutorial will demonstrate how to use the C compiler for the AVR. If WinAVR is installed, the functionality of the C compiler is available within AVR Studio. This tutorial will demonstrate the same program functionality and activities as the AVR Studio (i.e. Assembly Language) Tutorial.

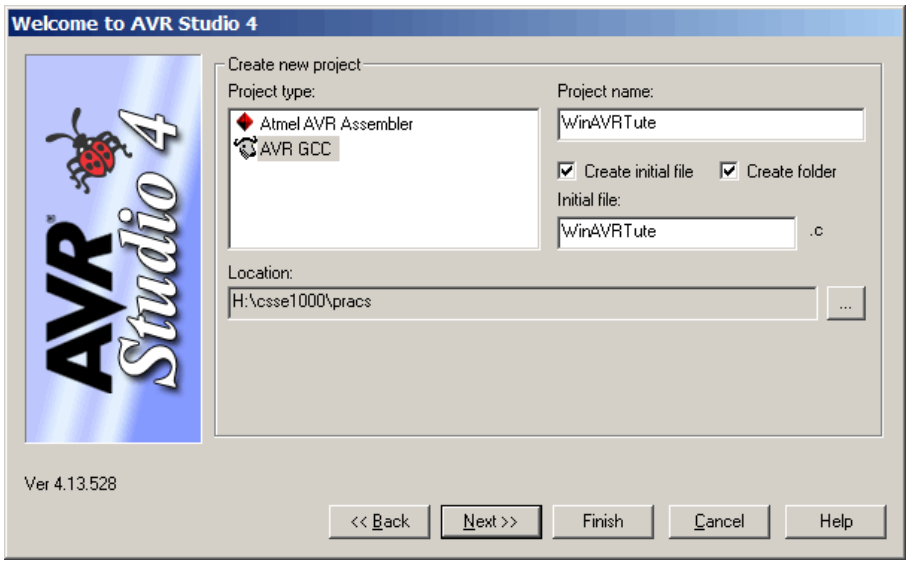
Contents

Creating a New Project

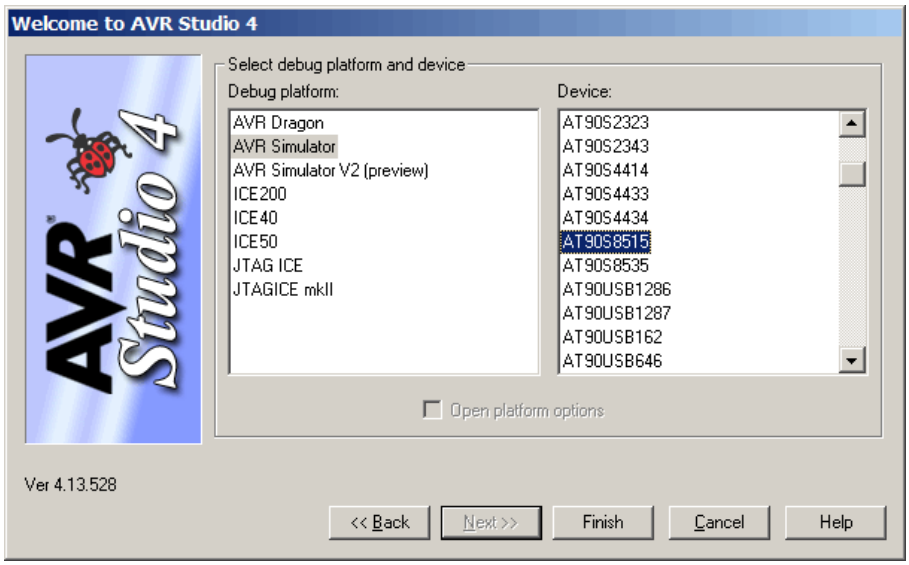
- Start the AVR Studio program by clicking on:

  ```
  Start->Programs->ATMEL AVR Tools->AVR Studio 4
  ```

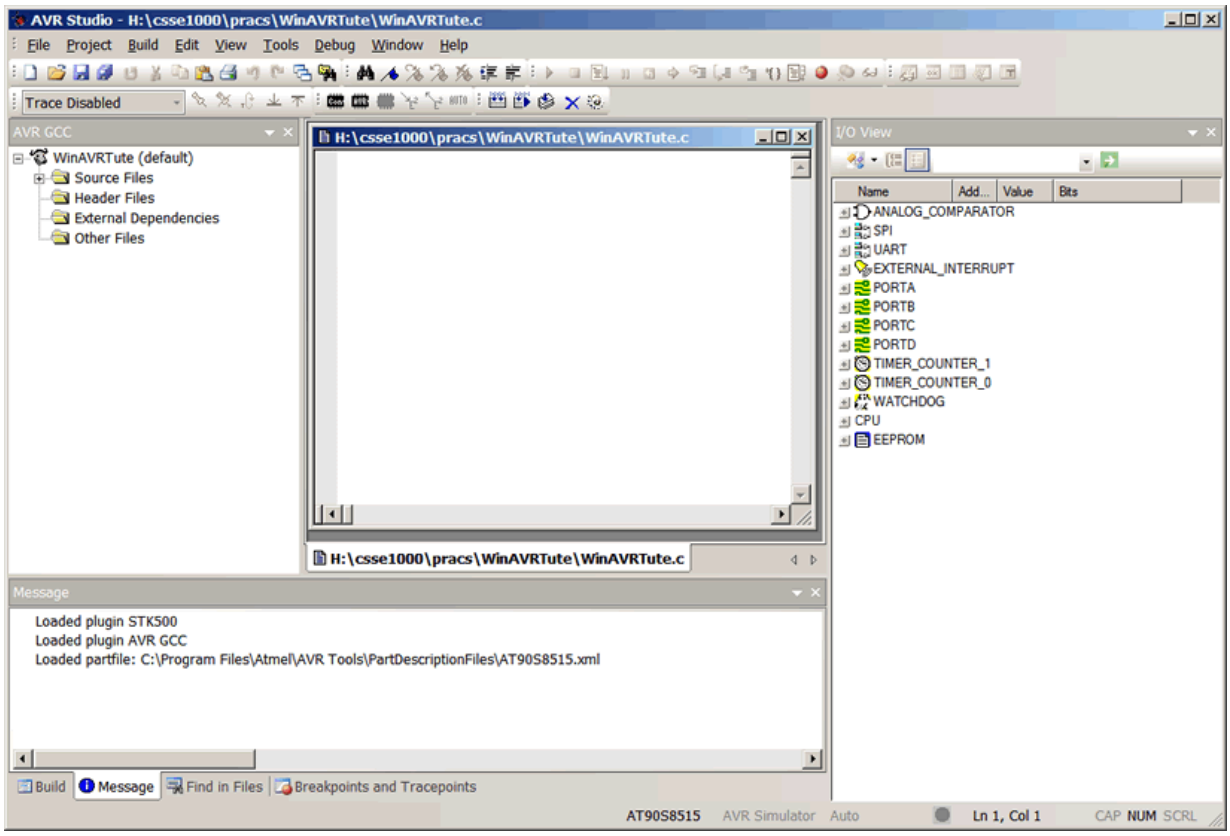
- Once the program has started, choose to create a New Project.
- Choose an appropriate location for the project (e.g. H:\csse1000\pracs)
- Choose "AVR GCC" as the Project Type. Type a project name (e.g. WinAVRTute) and choose to create an initial file and to create a folder for the project:



Click "Next>>" and choose the debug platform and device as shown:



Click "Finish". The main AVR Studio Window should then look something like the following:



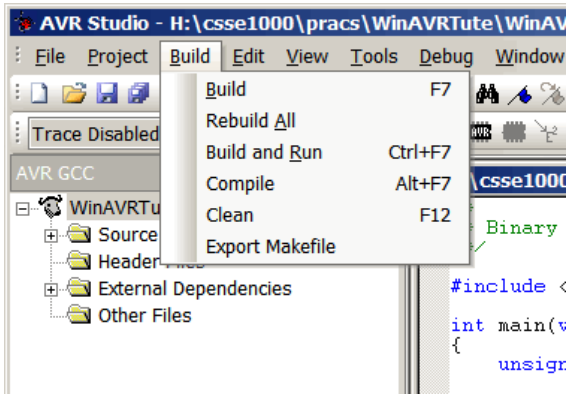
Editing and Building the C file

We have added a new but empty file to our project. The next step is to fill this file with our code. Manually enter the following code. This code peforms exactly the same functionality as the sample assembly language program in the AVR Studio (Assembly Language) Tutorial.

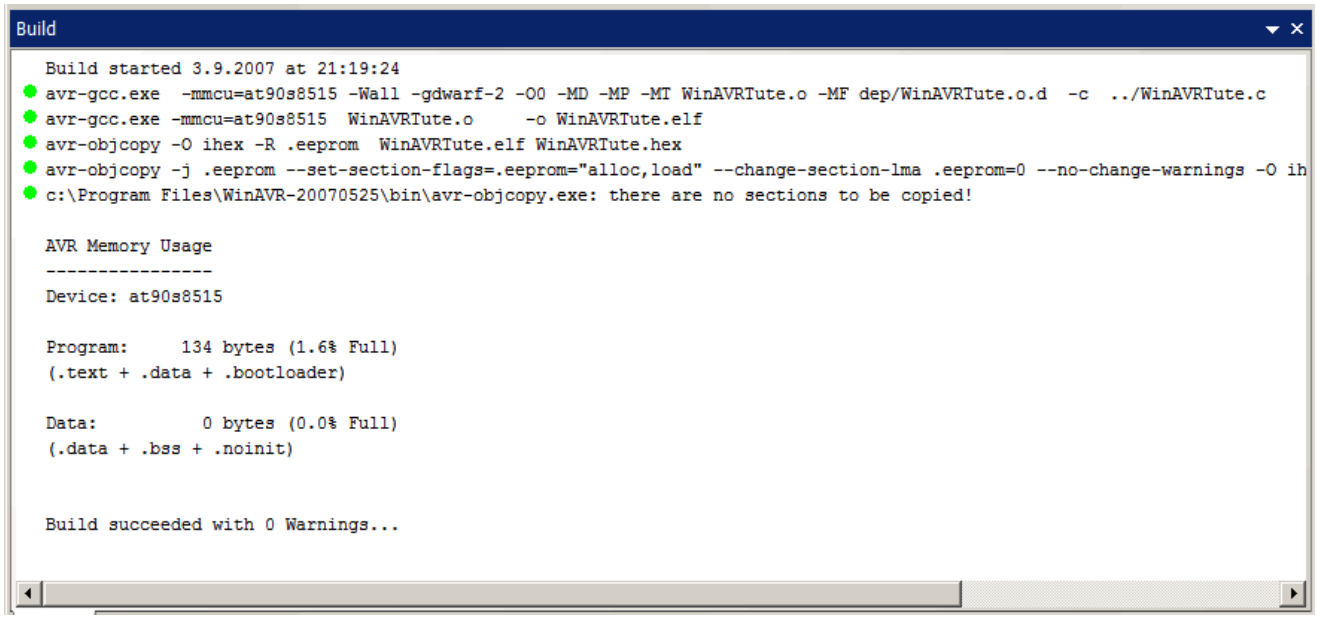
H:\csse1000\pracs\WinAVRTute\WinAVRTute.c *

```
/*
 * Binary counter for the WinAVR C Tutorial.
 */

#include <avr/io.h>

int main(void)
{
    unsigned char value;

    /* Set PORT B direction to output*/
    DDRB = 0xFF;

    /* Initialise value */
    value = 0xFF;

    /* Run forever - "for(;;)" is the same as "while(1)" */
    for (;;) {
        /* Write value to Port B */
        PORTB = value;

        /* Increment value */
        value++;
    }
}
```

Save the file and choose "Build" from the "Build" menu (or press <F7>):



The result of building the project will be shown in the "Build" pane and will be something like:

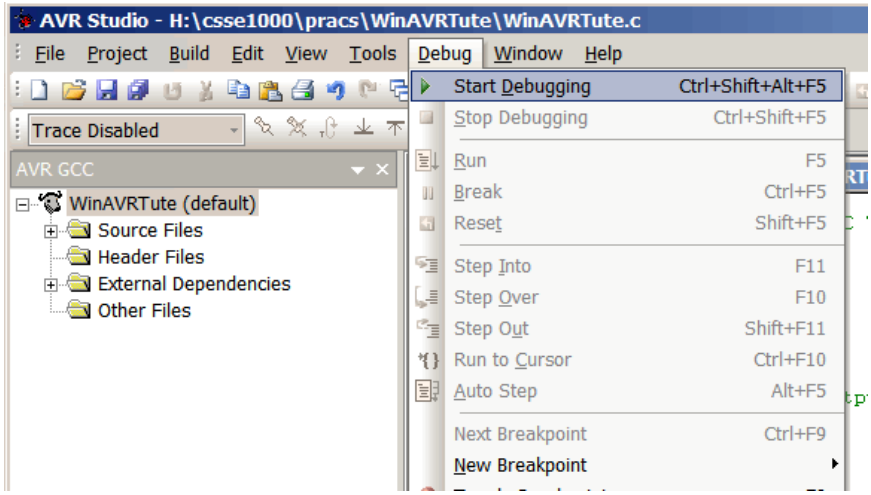


From this window we can see that the code is 134 bytes (compared with 20 bytes for the assembly language version) [NOTE: the result you get may be different, depending on the compiler version and optmisation settings.], and that the build was completed with no errors. We are now ready to simulate the code.

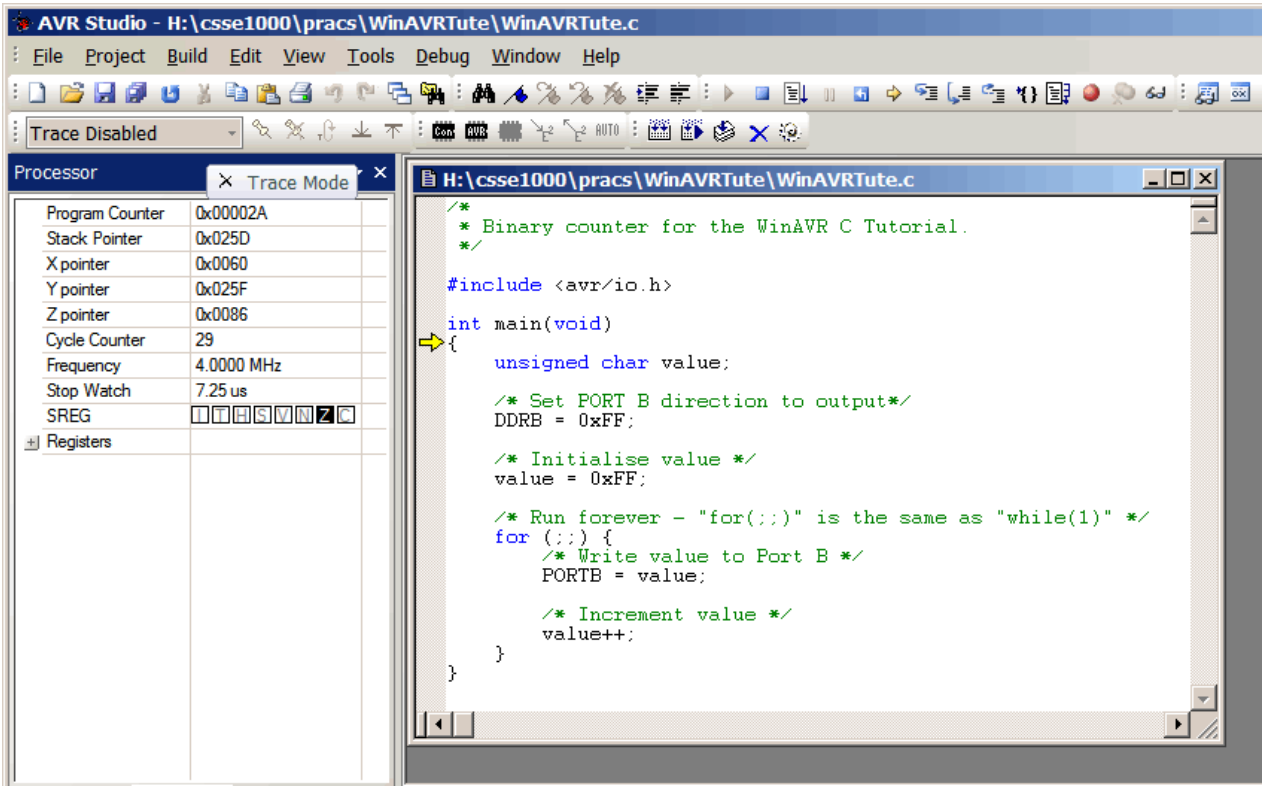
Back to Contents

Simulating the Code

We have generated the files necessary to simulate the code. To start running the code, select "Start Debugging" from the "Debug" menu:

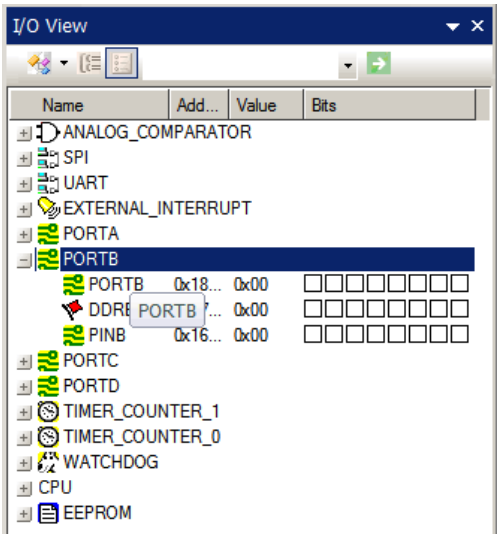


In the code editor, you'll now see a yellow right-arrow in the left margin. This arrow indicates the current code position:



If the "Processor" pane is not shown, display it. (Right click in the toolbar area and choose to display the "Processor" pane.) You will notice (as shown above) that even though we haven't started stepping through the code yet that the 29 clock cycles have already been executed and the program counter is up to 2A (hexadecimal). Unseen initialisation code has run before main() is reached. This code performs functions like setting the stack pointer, something which had to be done explicitly in the assembly languge program.

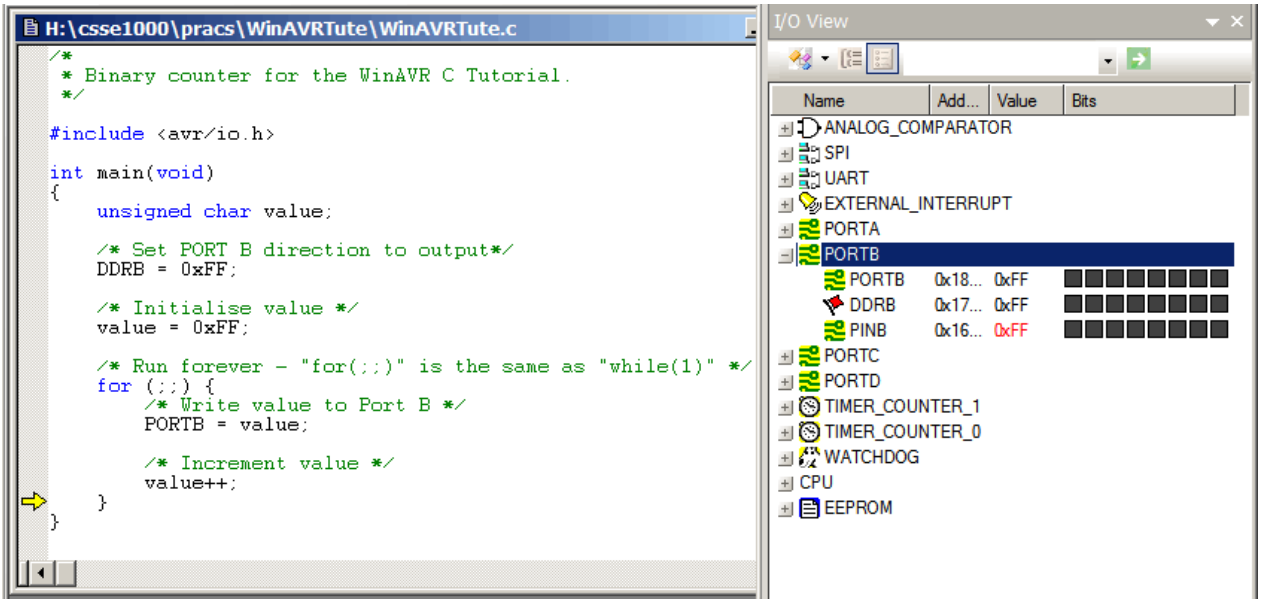
Before you step through the code, make sure you expand the I/O View so that you can see the Port B registers:



Single Stepping the Program

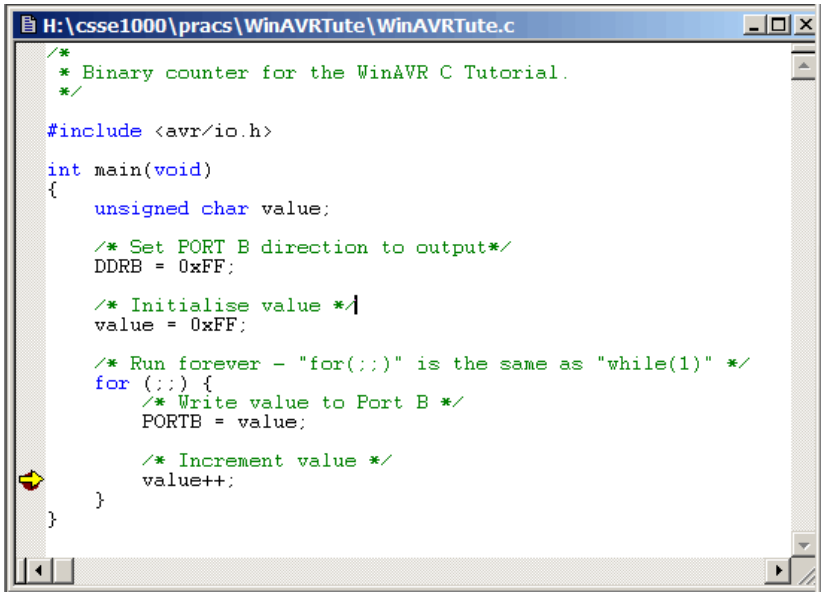
As in the assembly language simulation, there are two commands to single step through the code. These are "Step Over" <F10> and "Step Into" <F11>. The difference between these commands is that "Step Over" does not trace into functions. Since our example does not contain any functions, there is no difference between the operation of these commands in this example.

Now single step down to the last line of code (closing brace of the for loop) by repeatedly pressing the <F11> key or by selecting "Step Into" from the "Debug" menu. Notice how the colour changes from black to red on the registers that change value. This makes it easier to identify which registers change value on each instruction. Continue pressing the <F11> key and see how the binary value in Port B is increased. (It will roll over from value 255 to 0, then to 1, 2, etc.)



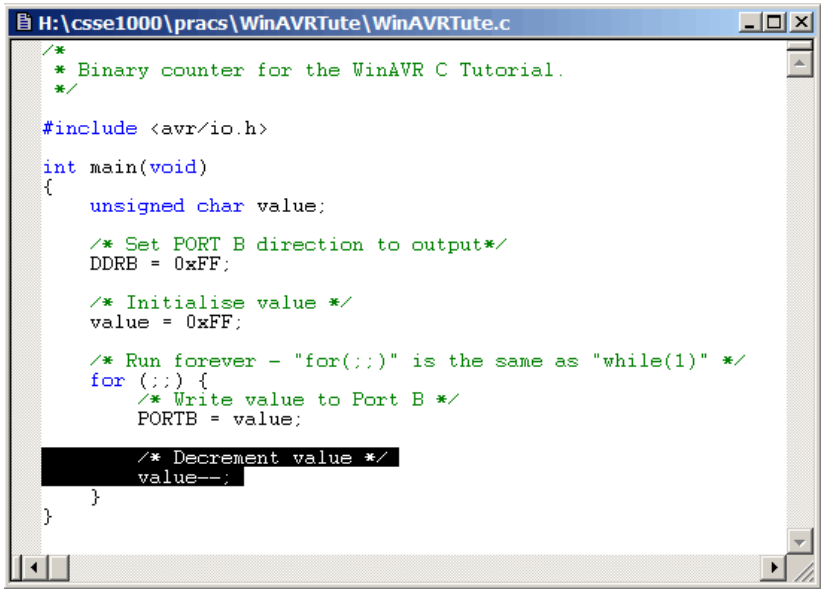
Setting Breakpoints

You can set breakpoints in C code just like you can in assembly language. Set a break point on the last statement (value++;). A red circle will appear in the left margin of the source view menu as shown. By pressing <F5> or "Run" from the "Debug" menu the program will start running and break (stop) at the statement with the breakpoint.

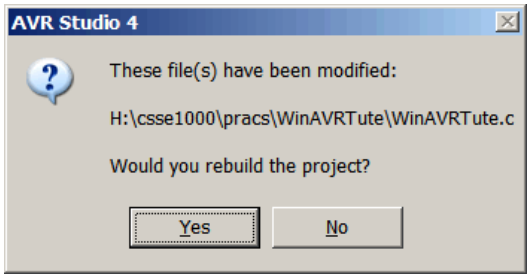


Modifying the Code

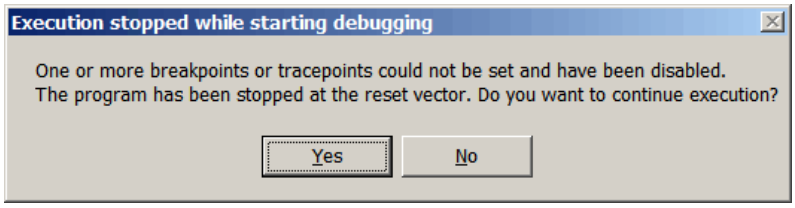
Now we want the program to count *down* instead of up. To make this change we'll have to edit the source code. Place the cursor in the source view, and change the "value++" statement to "value--". (You should also change the comment just above this.)



If you now press <F5> (Run) the following dialog box will appear. This box indicates that one of the source files has been changed, and that the project should be rebuilt. Press "Yes".



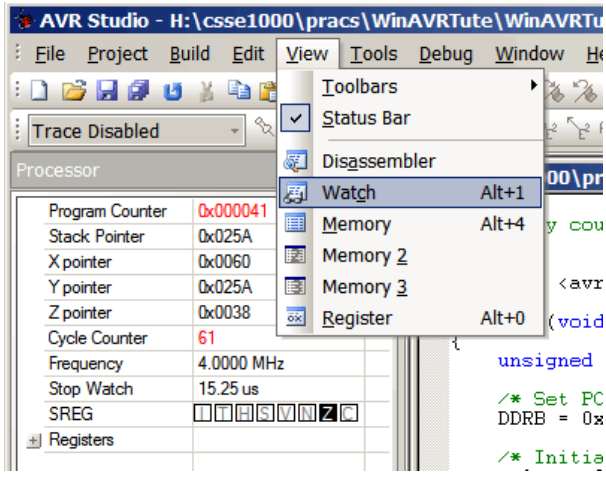
The project will now be rebuilt. You may get the following message, since the code at the breakpoint has changed. Choose "Yes" to continue execution.



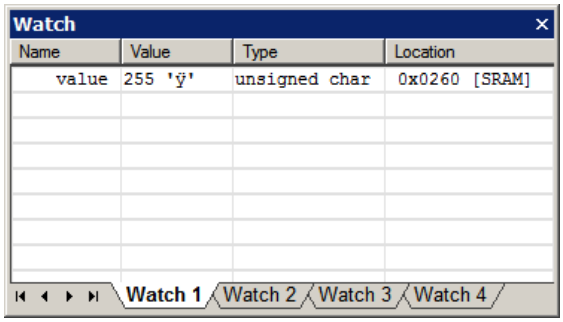
The yellow pointer will be reset to the start of the main() function.

Opening the Watch View

Open the Watch window by selecting "Watch" from the "View" menu:



When debugging assembly language programs, we could watch register values. When debuggging C programs, we can watch specific variables. Type "value" into the first "Name" box. As you step through the program you will see the value of the variable change. (The location shown is the memory address (in hexadecimal) at which this variable is stored. In some cases variables will be located in RAM, and in other cases they might be located in a general purpose register.



Saving the Project

Before exiting AVR Studio we will save our project. AVR Studio will remember where the views are placed and will use this setting when opening the project later. To save the project select "Save Project" from the "Project" menu.

Back to Contents