# AVR Studio Tutorial

(for Version 4 )

*Updated - 25 August, 2007*

Author: Len Payne, Modified for AVR Studio 4 by Peter Sutton

## Contents

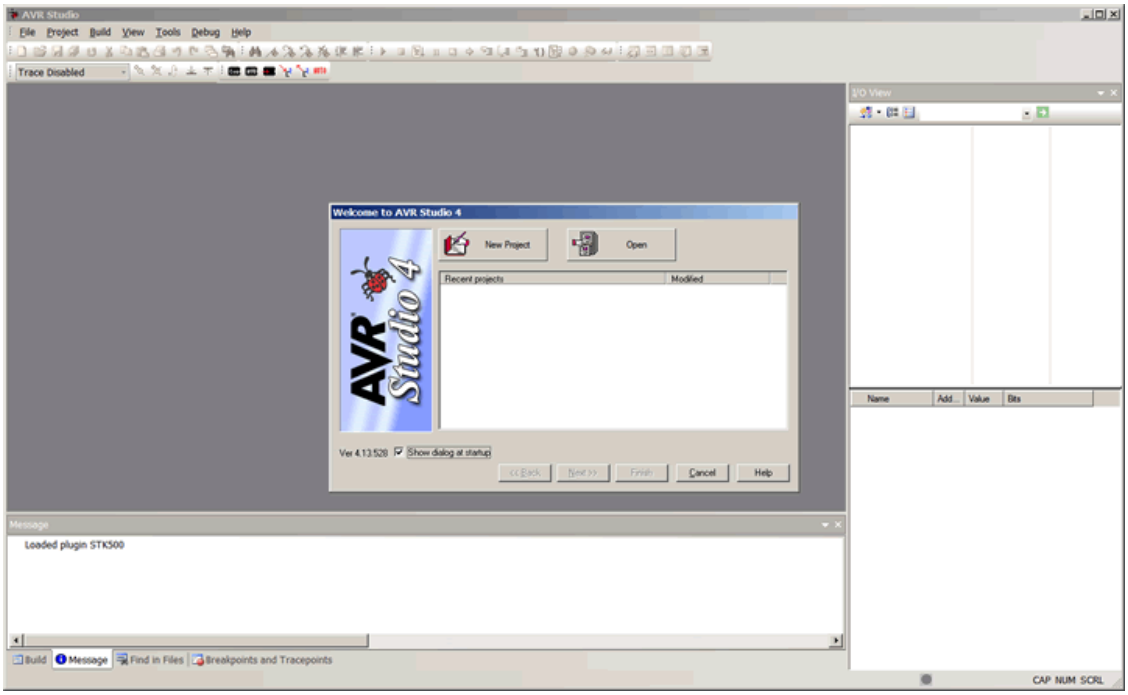


## Starting AVR Studio

- Start the AVR Studio program by clicking on:

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

- Once the program has started, you will be looking at a screen like this:

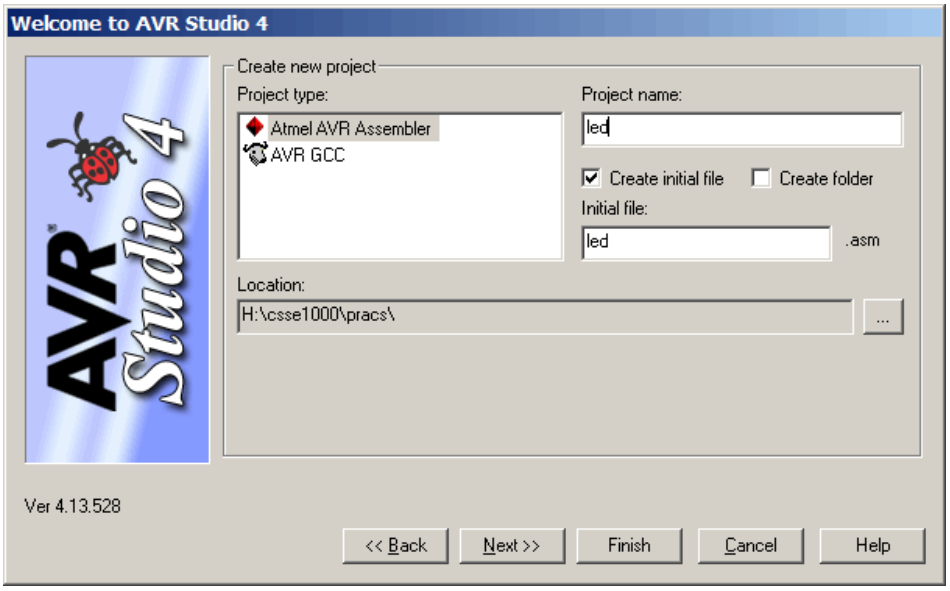



Back to Contents

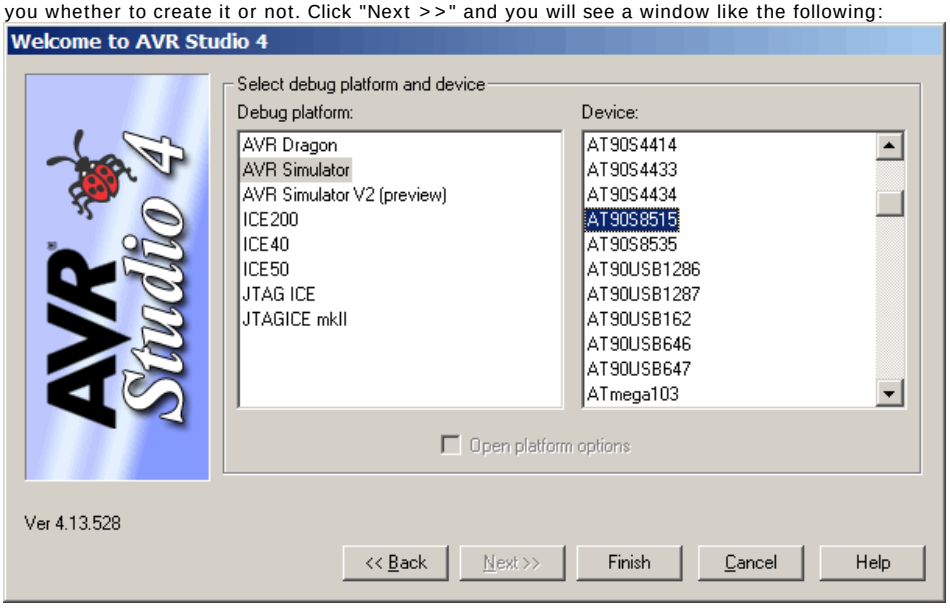
## Creating a New Project

In this tutorial we will make a simple program that increases the value of one of the PORT registers, making a binary counter.

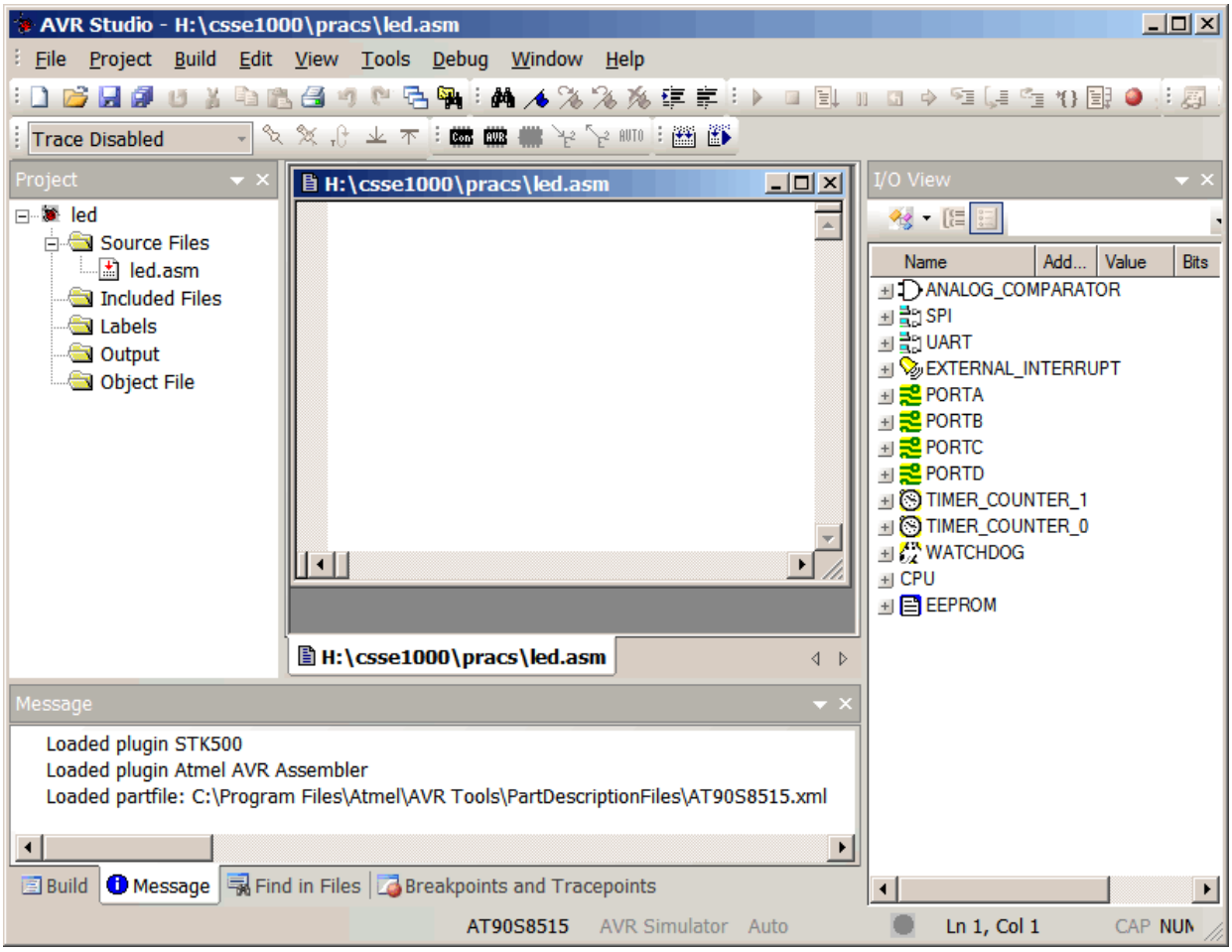
To create a new project, click on "New Project" on the Welcome Screen or go to the "Project" menu and select "New". The dialog box shown in the next figure appears.



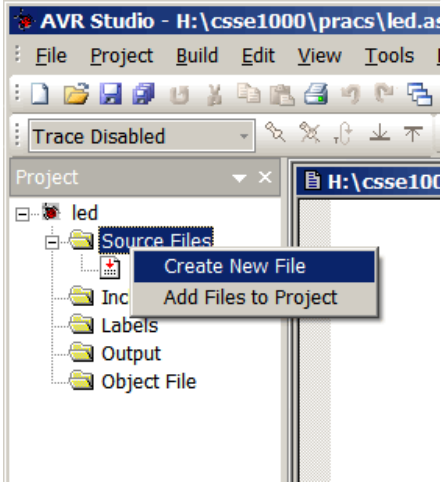

In this dialog box you should select "Atmel AVR Assembler" and enter the project name. We choose the name "led" here, but this could of course be an arbitrary name. Next you'll have to select the project location. This is the location where AVR Studio will store all files associated with the project. We have used the location H:\csse1000\pracs\ as the folder. If the folder does not exist, AVR Studio will ask you whether to create it or not. Click "Next >>" and you will see a window like the following:




Select "AVR Simulator" as the debug platform and "AT90S8515" as the device to use, then click "Finish". The main AVR window will now look something like this:




We can now enter the Assembly code of the program. (AVR Studio has opened an editor window called led.asm). Alternatively you can add or create other assembly language files by right clicking on "Source Files" in the Project window :




The led.asm file is automatically marked as "Assembler Entry File" (the small red arrow on the icon next to the filename shows this). This indicates that the code from this file will be assembled (and executed) first. Only one file can be marked as an entry file.

## Editing the Assembler file

We have now added a new but empty file to our project. The next step is to fill this file with our code. The file is initially empty and you'll have to manually enter the following code: (or you may Copy and Paste the code below directly into the editor window.)

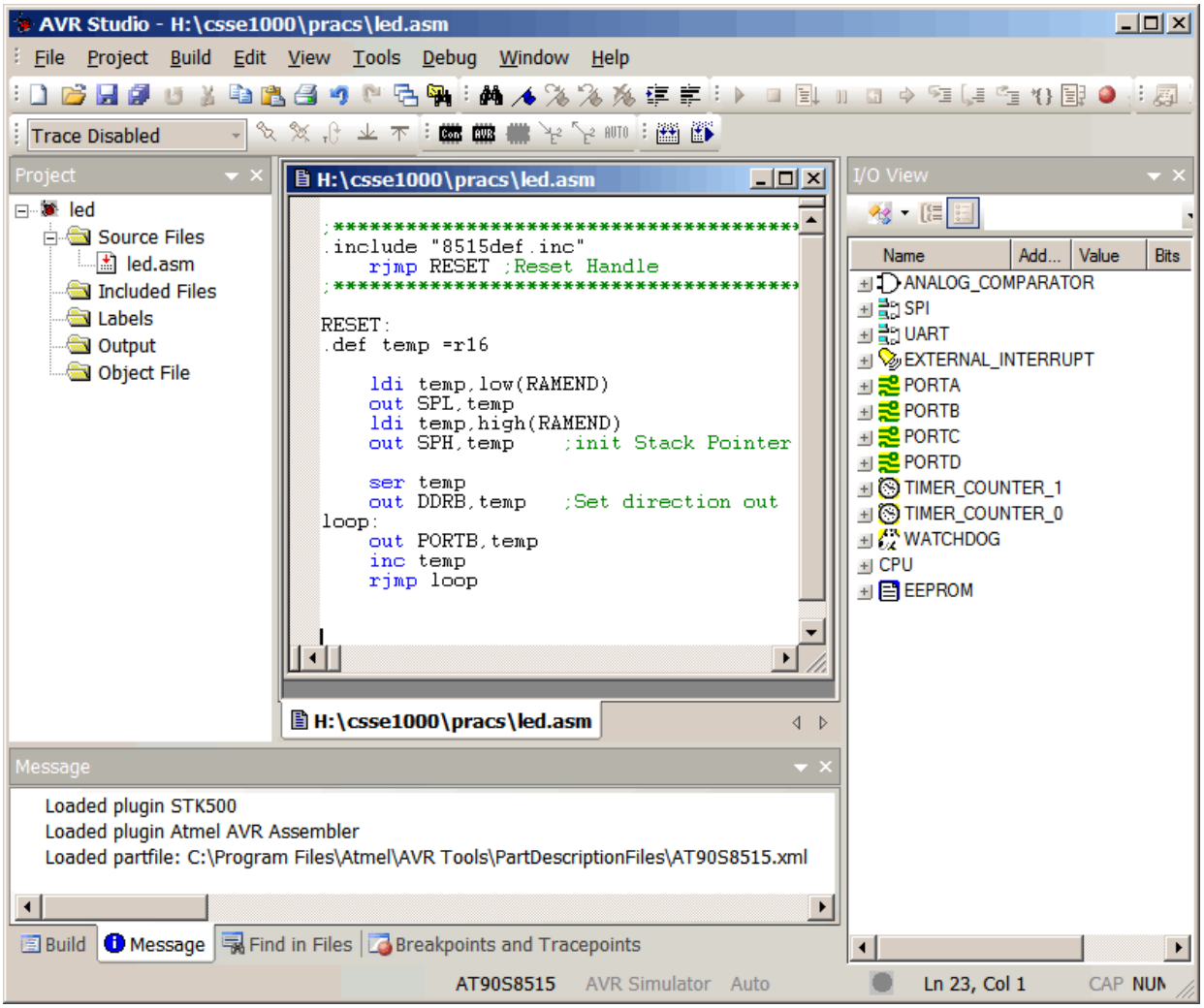
```
;***************************************************
.include "8515def.inc"
        rjmp RESET ;Reset Handle
;***************************************************

RESET:
.def temp =r16

        ldi temp,low(RAMEND)
        out SPL,temp
        ldi temp,high(RAMEND)
        out SPH,temp    ;init Stack Pointer
```

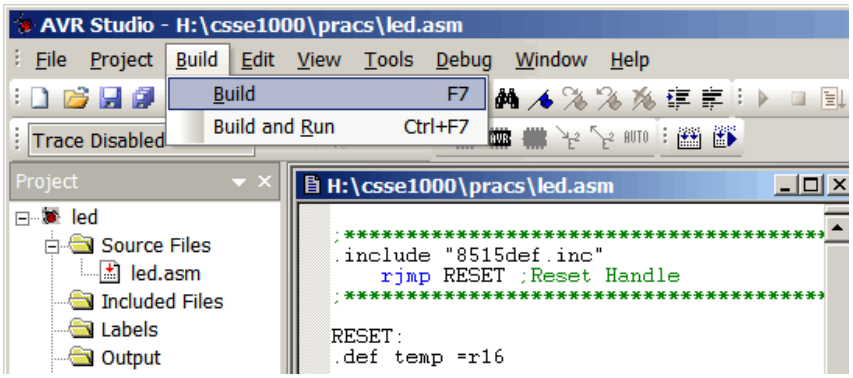
```
        ser temp
        out DDRB,temp   ;Set direction out
loop:
        out PORTB,temp
        inc temp
        rjmp loop
```

Save the file. (Select "Save" from the "File" menu.) The AVR Studio window should now look something like the following picture:

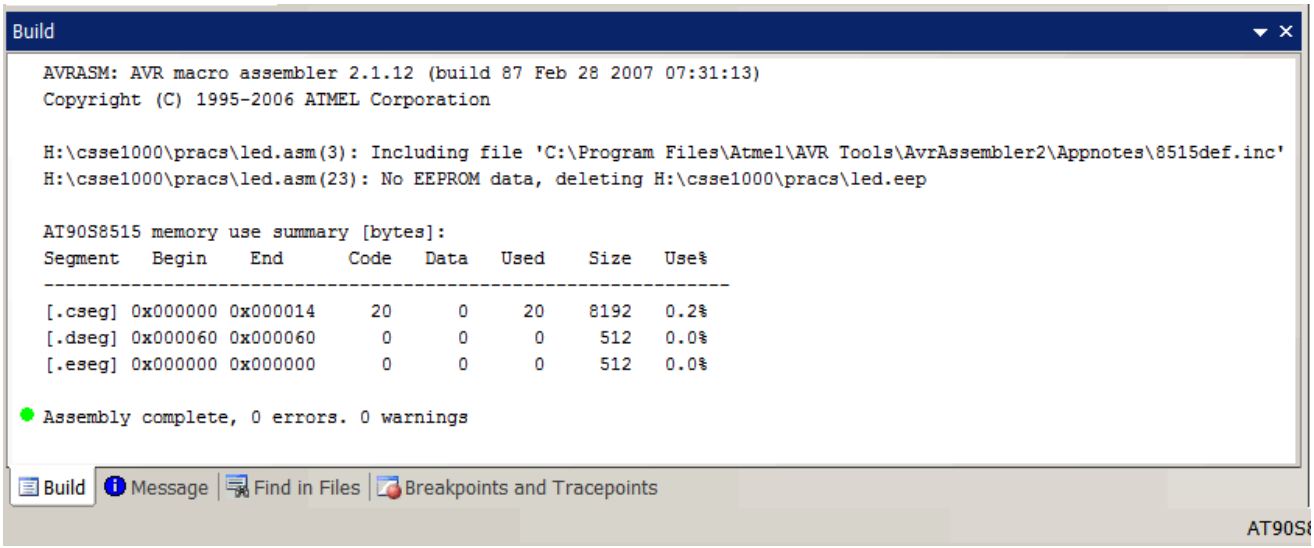



## Assemble the Source Code

To assemble the code, we need to build the project. Select "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 20 bytes, and that assembly was completed with no errors.

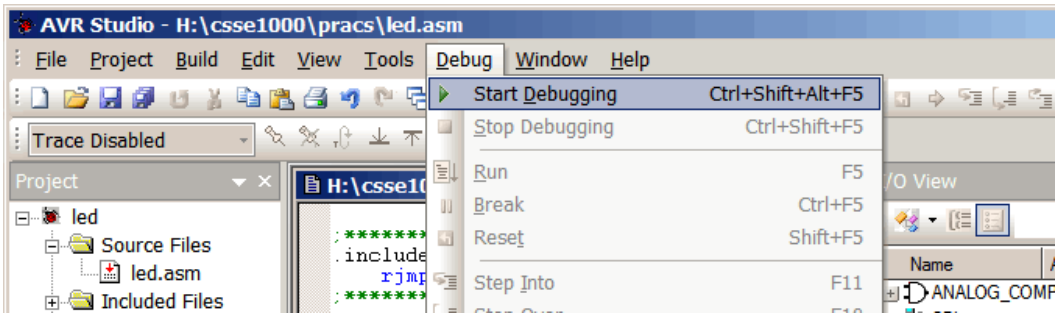
We are now ready to advance to the next step, which is running the code in simulator mode.

Back to Contents

---

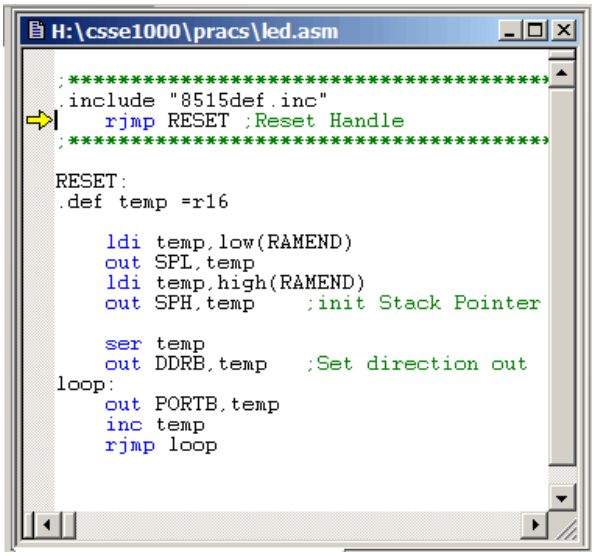
## Simulating the Code

At this point we have generated the files needed to simulate the code. To start running the code, select "Start Debugging" from the "Debug" menu:

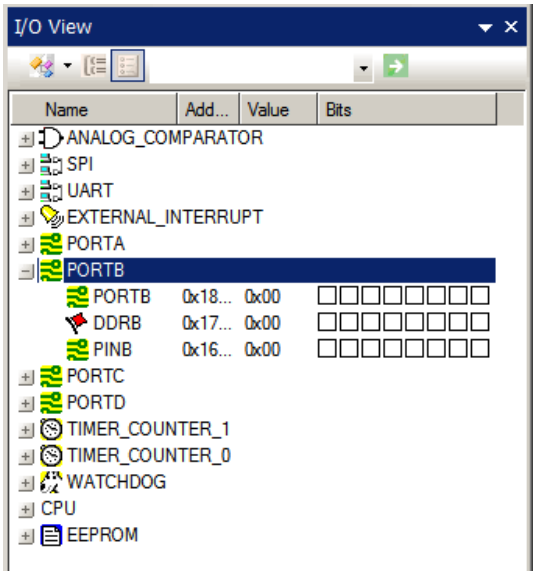



### Instruction Pointer

Now take a look in the editor view, you'll see that a yellow right-arrow has appeared in the left margin of the code. This arrow indicates the position of the program counter. In other words, it points to the next instruction to be executed.




We want to set the I/O View so that we can have a closer look at what is happening on the Port B registers during program execution. In the "I/O View" Window, click on the + symbol next to "PORTB". Similarly, you can expand other views if desired.

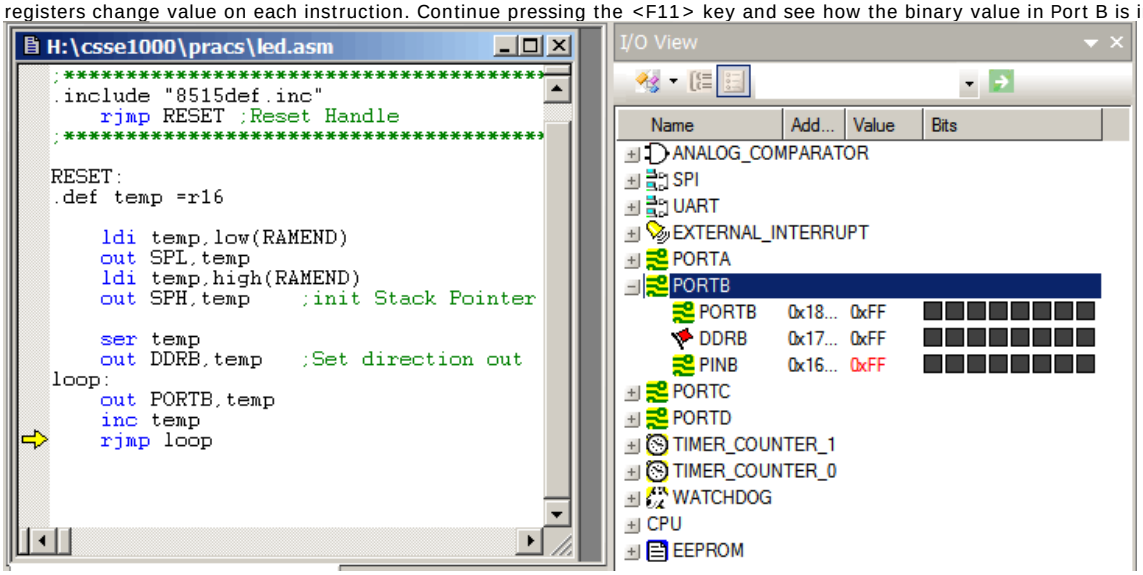



This shows all registers associated with Port B, these are: Port B Data register (PORTB), Data Direction (DDRB) and Input Pins (PINB). As shown each bit in the registers are represented by a checkbox. A logical 'zero' (0) is represented by an empty checkbox and a logical 'one' (1) is represented by a filled-in checkbox. These checkboxes will be updated during program execution, and show the current state of every bit. You may also set and clear these bits by clicking on the appropriate checkbox at any time during the program execution. You can also monitor the hexadecimal equivalent representation.

## Single Stepping the Program

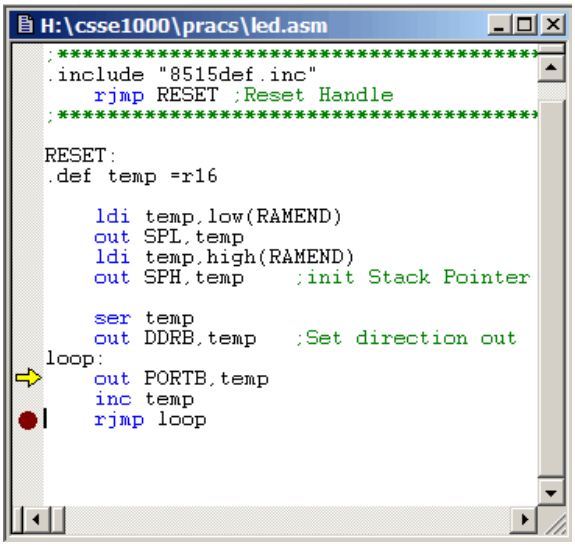
There are two commands to single step through the code. These are "Step Over" <F10> and "Step Into" <F11>. The difference between these commands is that "Step Over" does not trace into subroutines. Since our example does not contain any subroutines, there is no difference between the operation of these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

Now single step down to the last line of code (rjmp 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.



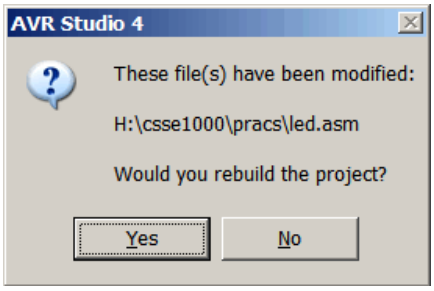

## Setting Breakpoints

Breakpoints are a method of halting execution flow. By adding a breakpoint in the assembly code we can run run the program at full speed, and it will be stopped at the line with the breakpoint. By now you have noticed that you have to press <F11> three times to go through the loop once. We will add a breakpoint at the rjmp loop instruction to show how this can be used to speed up the debug process. Place the cursor on the rjmp loop instruction in the source view window and press <F9> (or the "Toggle Breakpoint" in the "Debug" menu or right-click on the line of code and select "Toggle Breakpoint"). A red circle will appear in the left margin of the source view window as shown. By pressing <F5> or "Run" from the "Debug" menu the program will start running and break (stop) at the instruction with the breakpoint.




## Modifying the Code

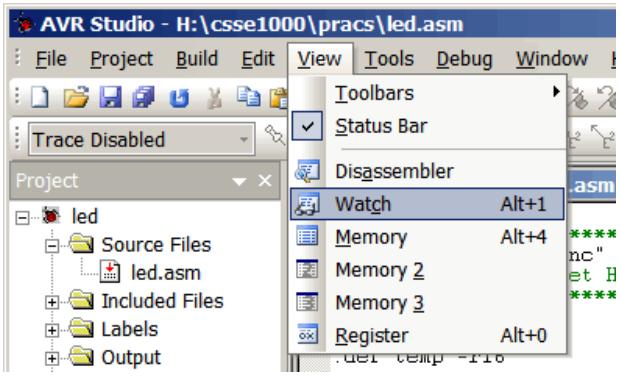
Now we want the program to count *down* instead of up. To make this change we'll have to edit the source code. Place the cursor in the source view, and change the inc to a dec instruction. If you now press <F5> (Run) the following dialog box will appear. This box indicates that one of the source files has been changed, and that the project should be rebuilt. Press "Yes".



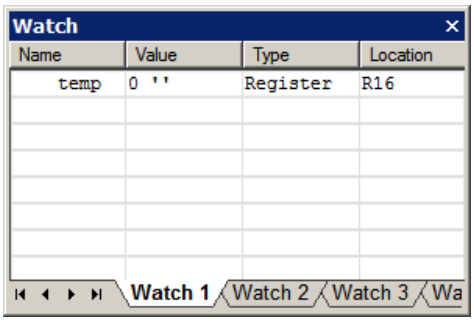

The Project will now be rebuilt, and the instruction pointer will start at the first line of code. Notice how the breakpoint is remembered.

## Opening the Watch View

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

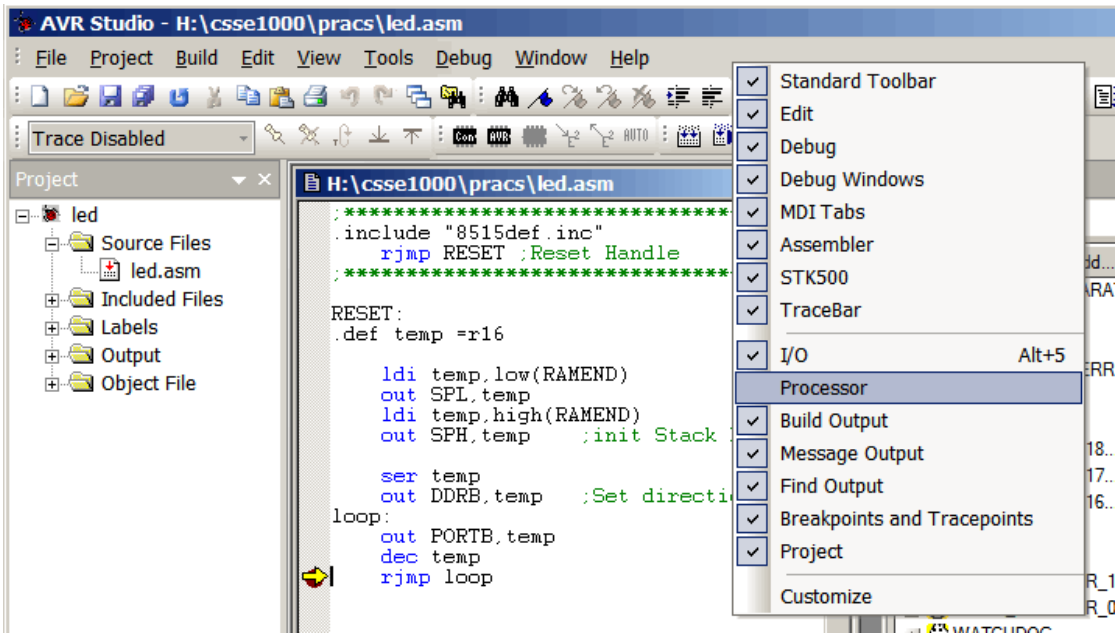



Variables that are defined (by the .def directive) can be placed in the Watch view. The only defined variable in this code is the temp variable. Right-click the "Watches" window and select "Add Item". Type in the variable name temp at the cursor and then press the Enter key. As we continue to run through the program the temp variable will constantly be updated during program execution.

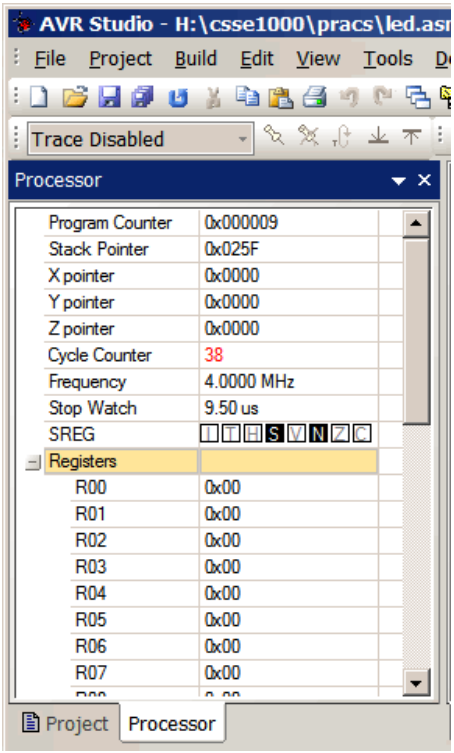



## Setting up the Processor View

Now we will examine the Processor view. Open this view by right clicking in the toolbar area and select "Processor":




This view shows processor specific information like the value of the Program Counter. In this view you will also find a Cycle Counter and a StopWatch. These are very useful if you wish to measure the length of a loop or how much time a specific subroutine uses. We will not use this view directly in this example, but it provides a lot of useful information during debugging of a project. You can also view the contents of the individual registers.




## 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