

Getting started programming the Atmel ATmega32 in assembly using AVRStudio

by
Dr.-Ing. Joerg Mossbrucker
EECS Department
Milwaukee School of Engineering
August 20, 2005

Modified by:

Professor Barnekow
March 9, 2007

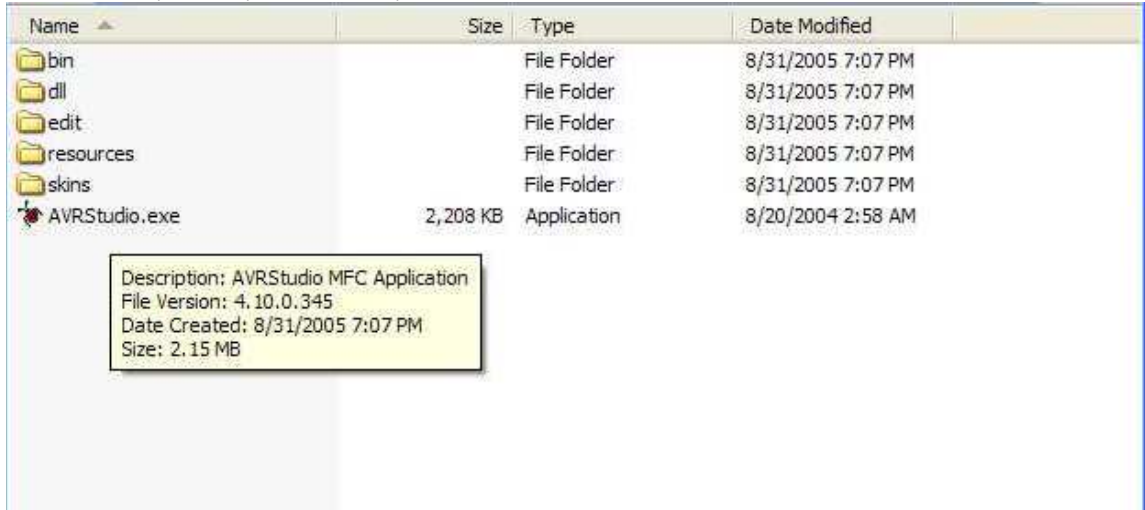
Dr. Taylor
March 12, 2007

This manual covers the following:

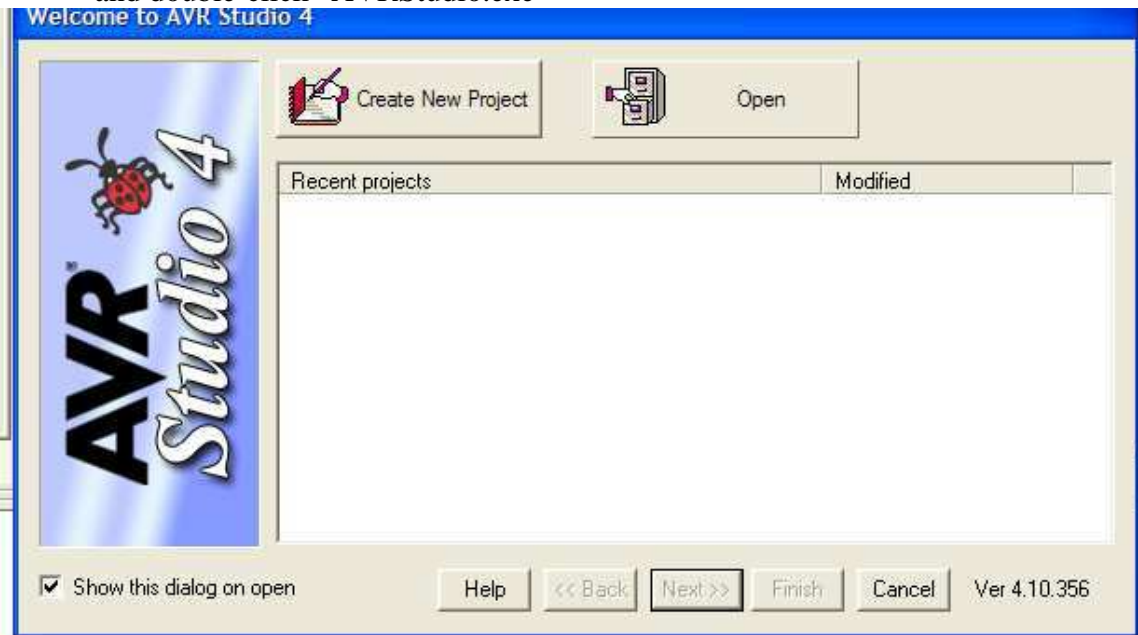
- How to create and code a new project
- How to build and simulate the tutorial
- How to download and execute the tutorial on the hardware platform

1 New Project

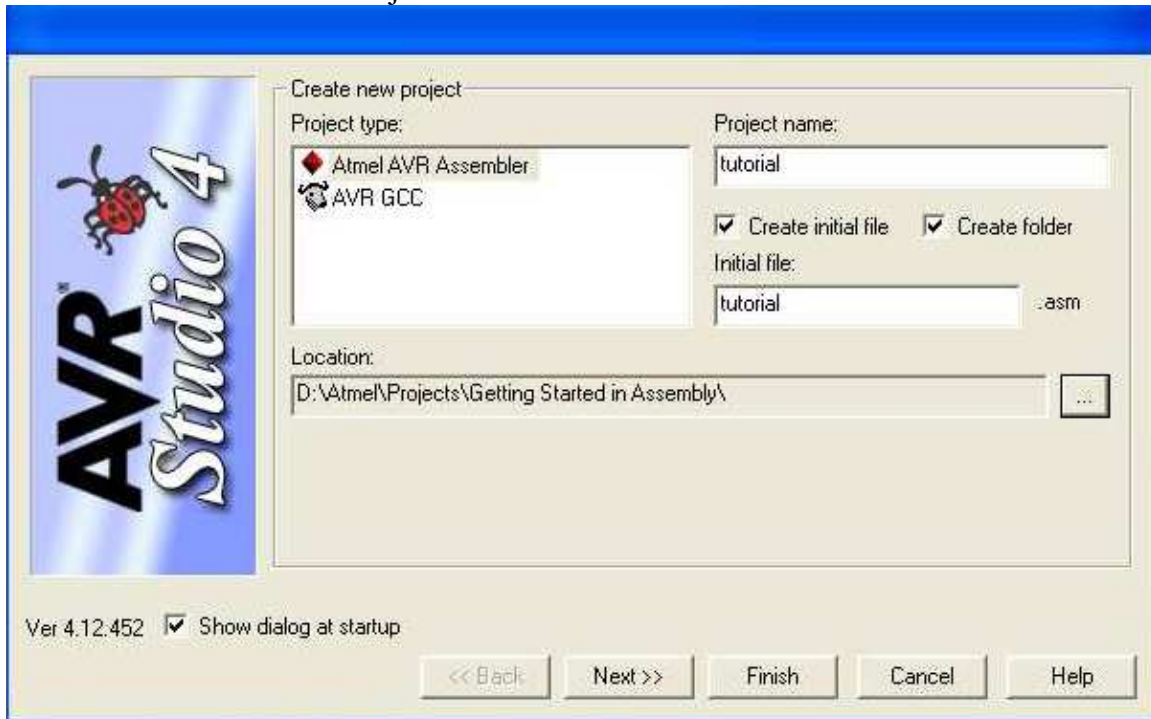
- Go to "D:\Atmel\AVRTools\AvrStudio4"



- and double-click "AVRStudio.exe"



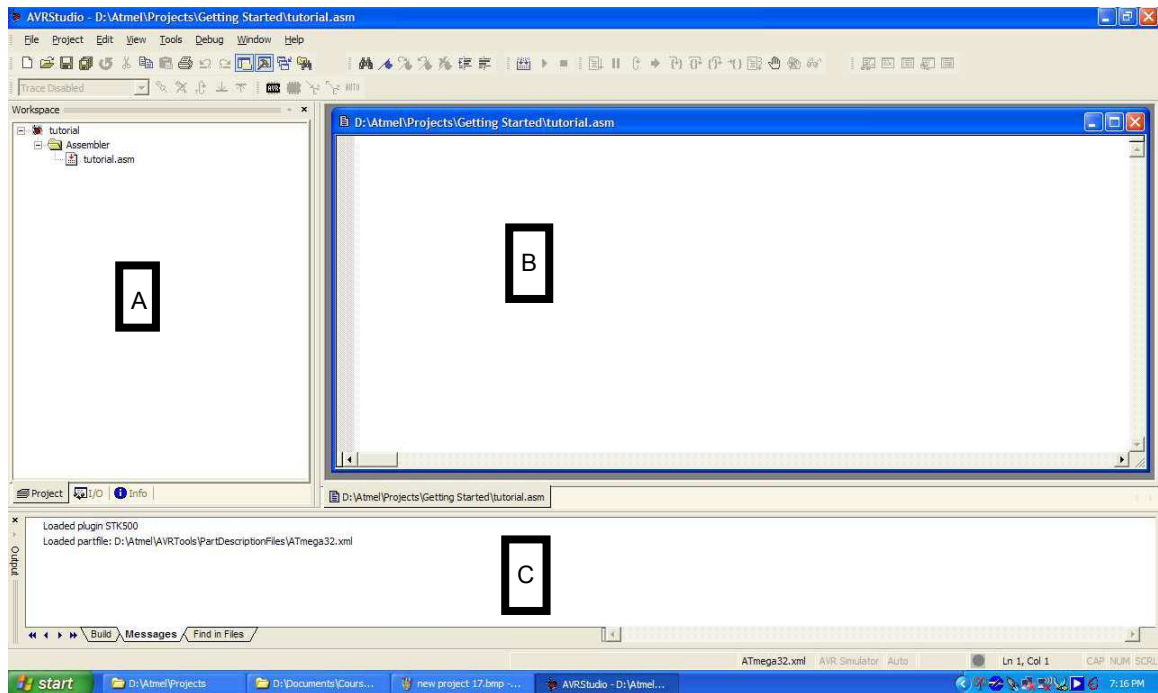
- click “Create New Project”



- and fill out the dialog box as shown above.; make sure that the Location is set as above and that “Create initial file” is checked. Click “Next>>”



- and select “AVR Simulator” and “ATmega32”. Click “Finish”. This closes the dialog box.



AVRStudio is an integrated development environment. The user interface consists of the following areas:

A: The workspace; this area will either show the structure of the current project or the simulator

B: The source code window(s)

C: The status window

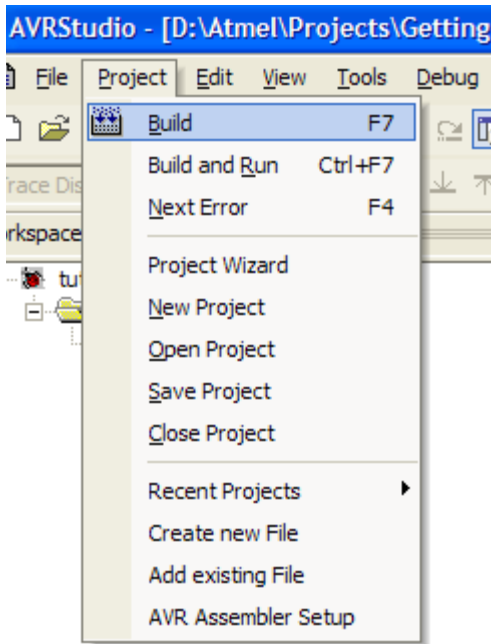
Please note that depending on the configuration of AVRStudio many more windows might appear.

Click on the source code window (B) and enter the following assembly program:

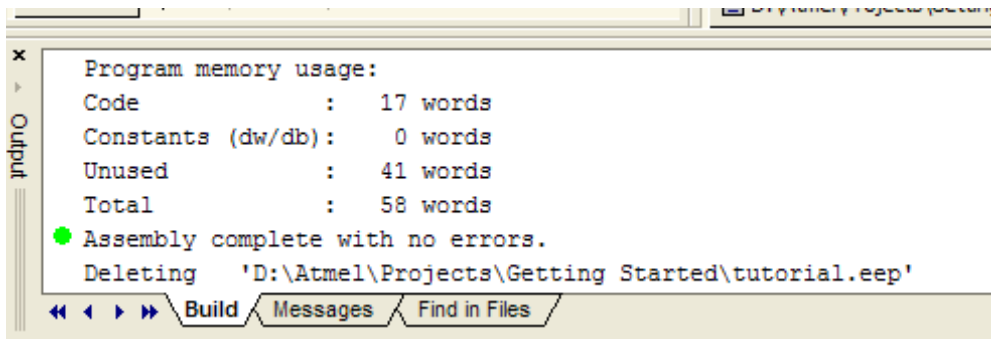
```
1 ; Copyright (c) 2003 Joerg Mossbrucker <mossbruc@msoe.edu>
2 ;
3 ; All Rights Reserved.
4 ;
5 ; Redistribution and use in source and binary forms, with or without
6 ; modification, are permitted provided that the following conditions
7 ; are met:
8 ;
9 ; 1. Redistributions of source code must retain the above copyright
10 ; notice, this list of conditions and the following disclaimer.
11 ; 2. Redistributions in binary form must reproduce the above copyright
12 ; notice, this list of conditions and the following disclaimer in the
13 ; documentation and/or other materials provided with the distribution.
14 ;
15 ;
16 ; ASMTutorial1.asm
17 ; This is a tutorial file, implementing a binary counter on PORTB
18 ; Note: this program is neither a efficient nor a small program, it
19 ; is intended to show some functionality of the assembler/simulator
20 ; Copyright (c) 2004 Joerg Mossbrucker <mossbruc@msoe.edu>
21 ;
22 ;
23 ;
24 .include "m32def.inc" ; Include the appropriate register definition file
25 ;
26 .equ Zero = 0
27 .equ Time = 255 ; This is used in a timing loop
28 ;
29 .def Counter = r21 ; Use register 21 (r21) as the counter register
30 ; We then can use Counter as a variable in the
31 ; simulator just like a variable in SRAM
32 ;
33 .dseg ; DATA SEGMENT
34 .org 0x60 ; Set SRAM address to hex 60, this is right after I/O
35 Data: .byte 1 ; reserve one byte in SRAM (for the counter)
36 ;
37 .cseg ; CODE SEGMENT
38 .org 0 ; Set the RESET vector to
39 rjmp Reset ; the beginning of our code
40 ;
41 .org 0x2A ; Set Program Counter to hex 2A, end of IJT
42 Reset:
43 ldi r20,0xff ; Load immediate 0xff into r20
44 out DDRB,r20 ; Output that on DDRB, i.e. set PORTB to output
45 ;
46 ldi r20,Zero ; Load r20 with 0
47 sts Data,r20 ; Store r20 in Data
48 out PORTB,r20 ; Out r20 on PORTB
49 ;
50 start: ldi Counter,time ; Load Counter with immediate time
51 ;
52 loop: dec Counter ; Waiting loop: count down to 0 by decrementing Counter
53 brne loop ; Branch if not equal back to loop
54 ;
55 lds r20,Data ; Load r20 with Data
56 inc r20 ; Increment r20
57 sts Data,r20 ; Store r20 into Data
58 out PORTB,r20 ; Out r20 on PORTB
59 ;
60 rjmp start ; Jump back to start
```

Once entered, save your work by selecting File -> Save All

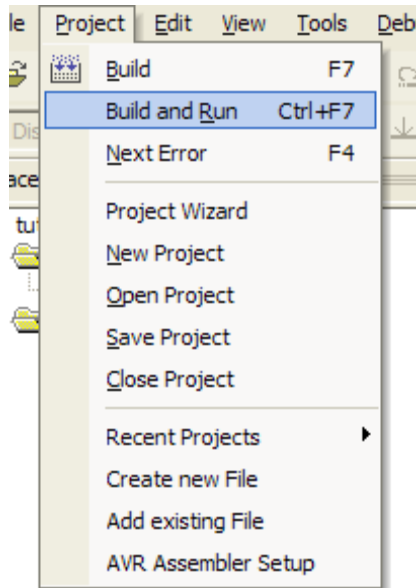
The source file is in AVR assembly language and needs to be assembled into an executable. Click “Project->Build”



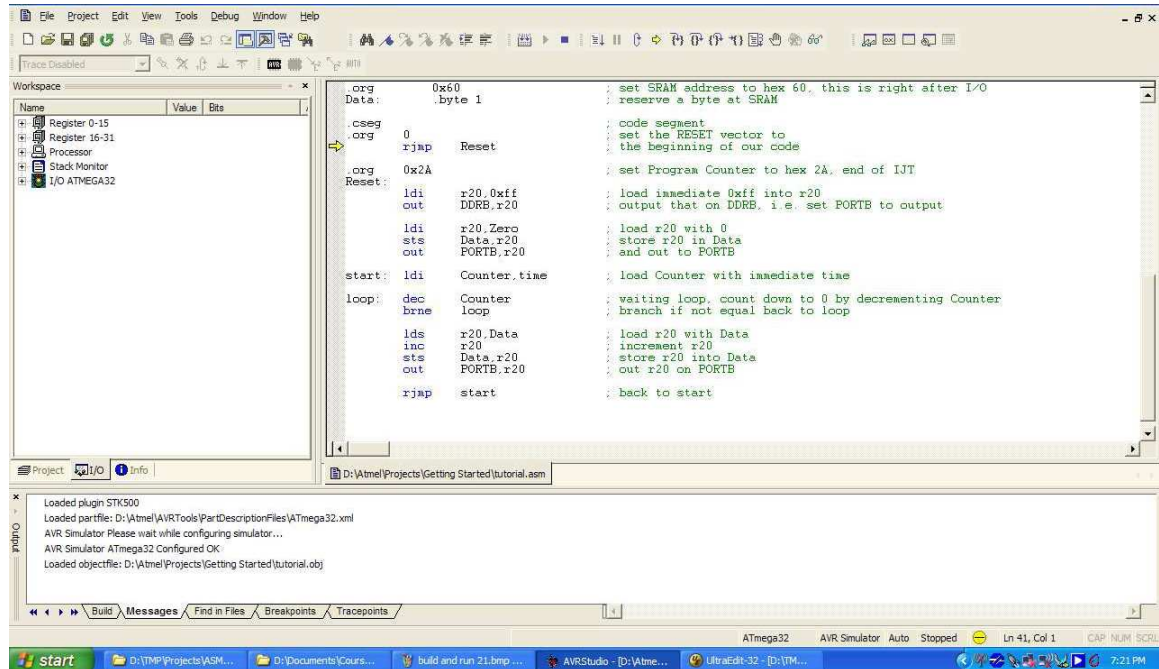
The status area will show if the assembly process was successful



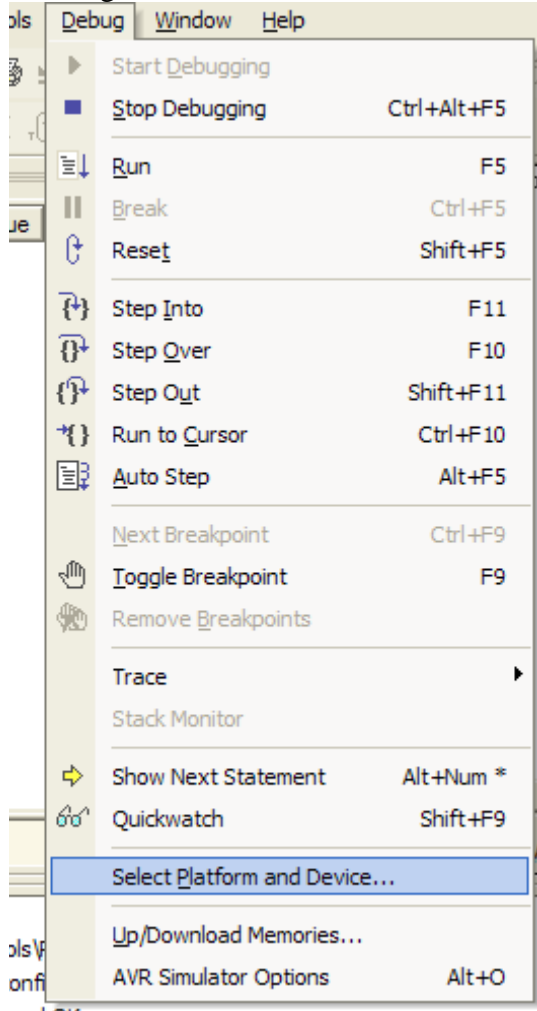
To start the simulator click “Project-> Build and Run”



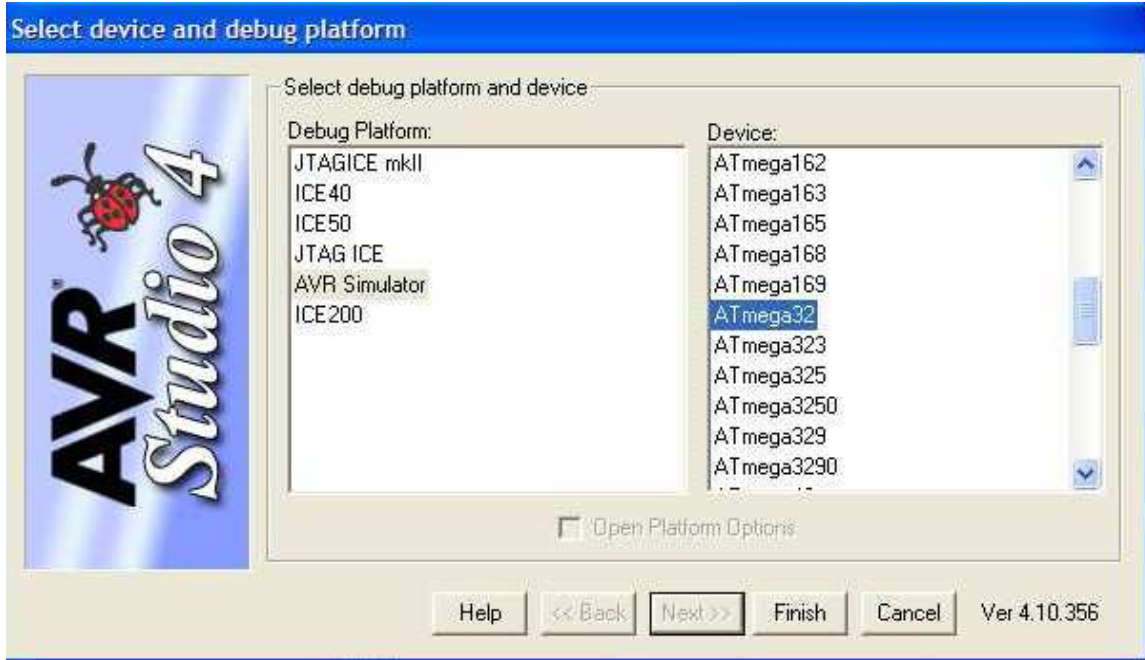
This assembles and starts the simulator



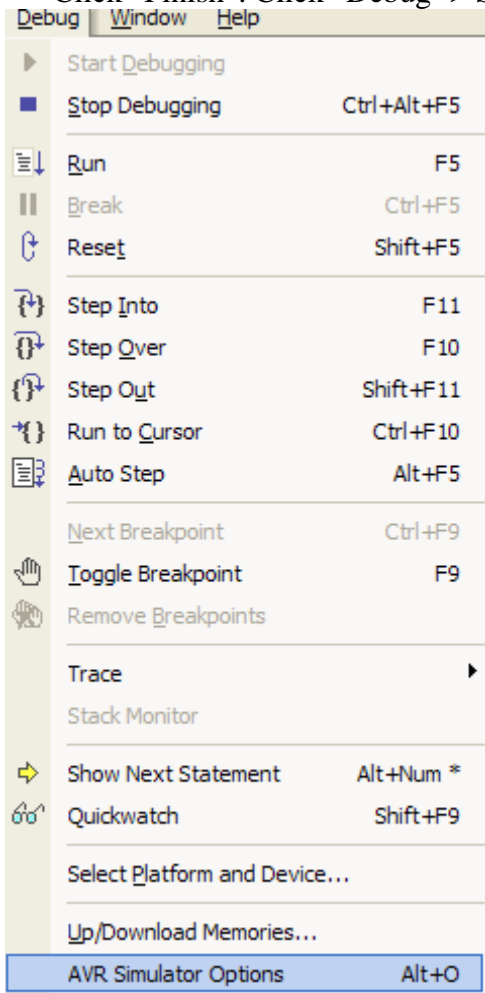
The simulator is capable of simulating a variety of AVR microcontroller. Click “Debug -> Select Platform and Device”



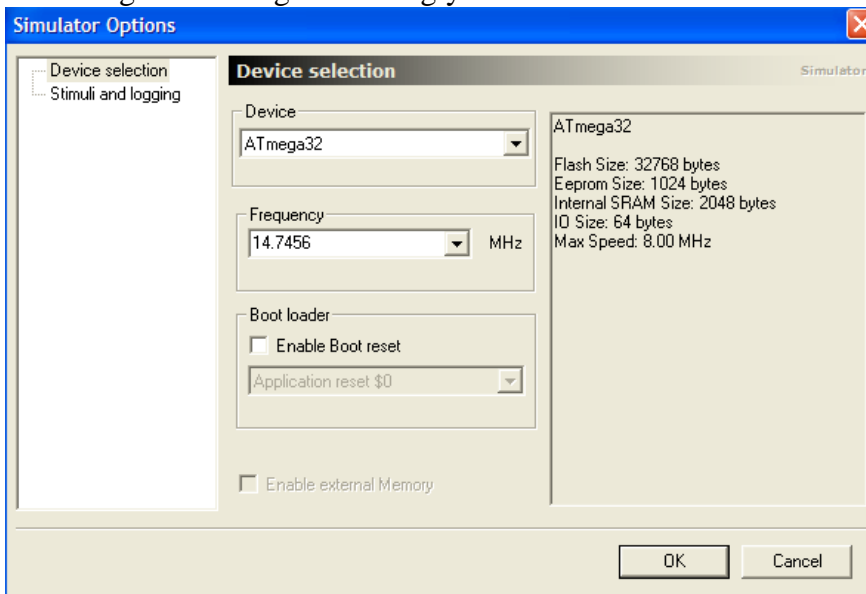
and select the appropriate microcontroller



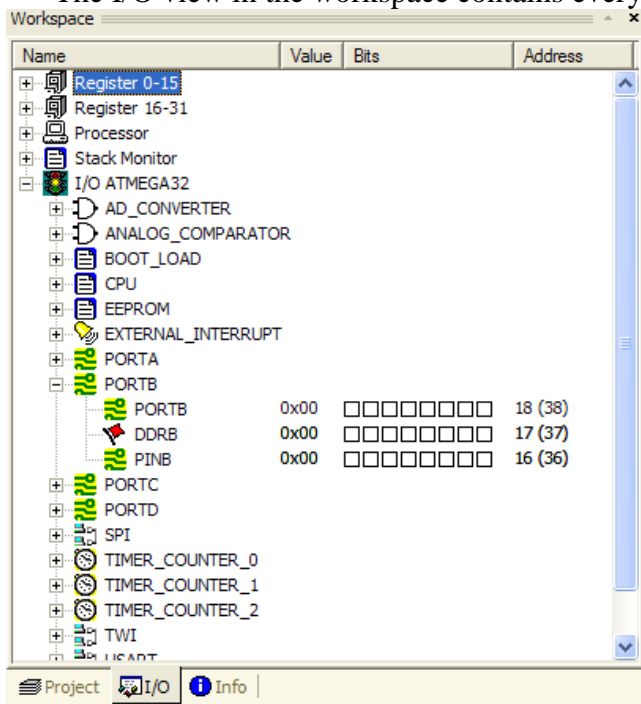
- Click "Finish". Click "Debug -> Simulator Options"



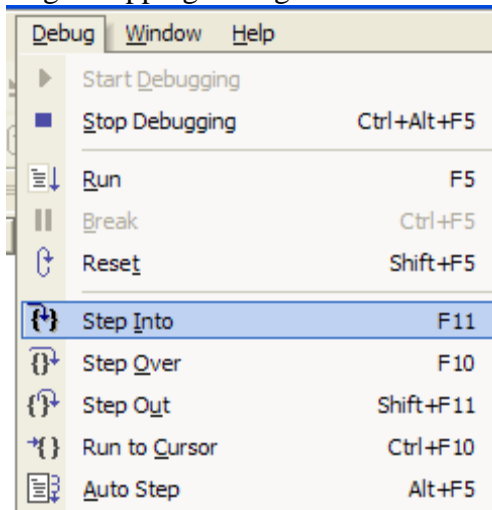
and change the settings accordingly



The I/O view in the workspace contains every register of the selected microcontroller.



The debug menu contains every command for debugging using the simulator, such as single stepping through the code



The I/O view reflects the changes of every step in the program

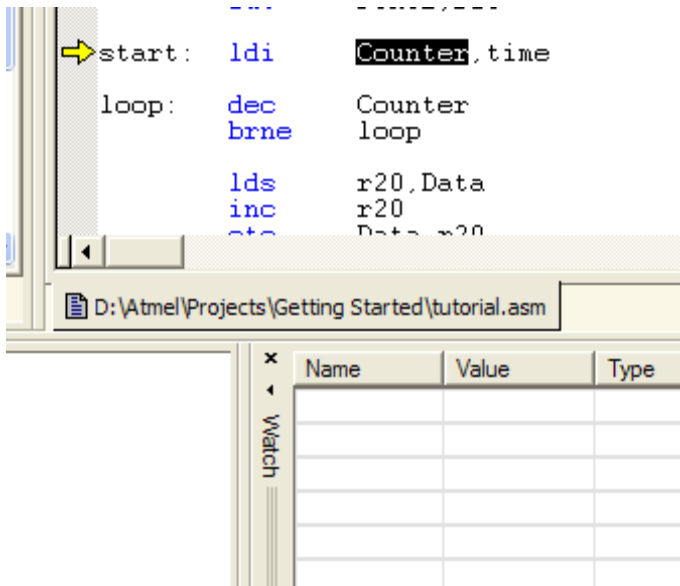
The screenshot displays the I/O view of an AVR microcontroller and its assembly code. The I/O view on the left shows the state of various hardware modules. The assembly code on the right shows the program's execution flow.

| Module | Value | Hex | Address |
|------------------|-------|-----------------|---------|
| PORTB | 0x00 | □□□□□□□□ | 18 (38) |
| DDR B | 0xFF | ■ ■ ■ ■ ■ ■ ■ ■ | 17 (37) |
| PINB | 0x00 | □□□□□□□□ | 16 (36) |

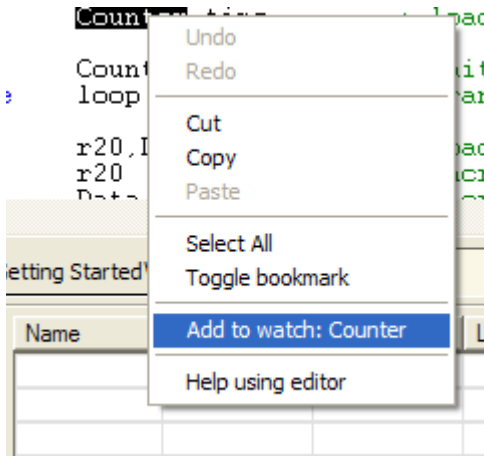
```

.org 0x2A
Reset:
    ldi r20,0xff
    out DDRB,r20
    ldi r20,Zero
    sts Data,r20
    out PORTB,r20
start: ldi Counter,time
loop:  dec Counter
      brne loop
      lds r20,Data
      inc r20
      sts Data,r20
      out PORTB,r20
      rjmp start
  
```

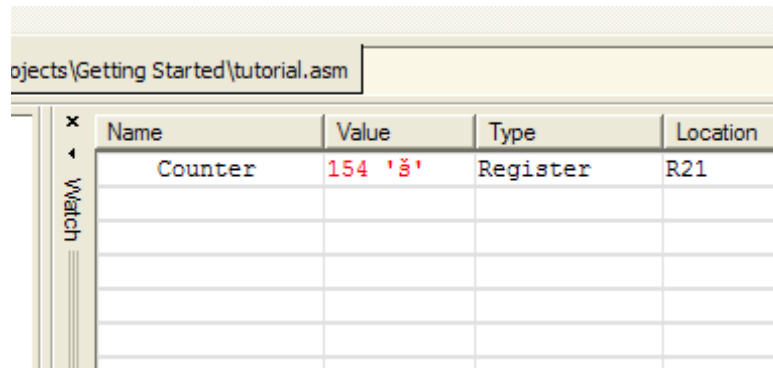
By enabling the Watch view (click “View -> Watch”) variables can be traced throughout the program by simply selecting the variable name



right-click and select “Add to Watch: Counter”



which adds the variable to the watch list

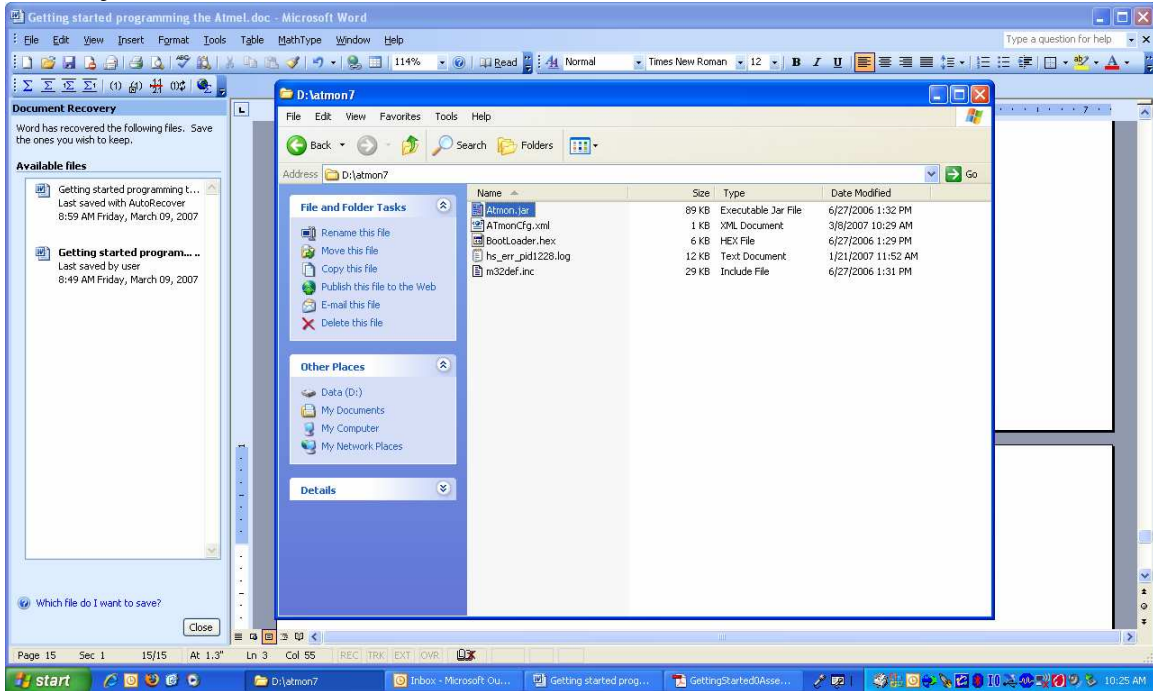


The image shows a screenshot of a debugger's watch window. At the top, there is a text box containing the file path "objects\Getting Started\tutorial.asm". Below this is a table with the following columns: Name, Value, Type, and Location. The first row contains the data: Counter, 154 '8', Register, and R21. The table has five rows in total, with the first row containing data and the others being empty. On the left side of the table, there is a vertical label "Watch" and a small "x" icon.

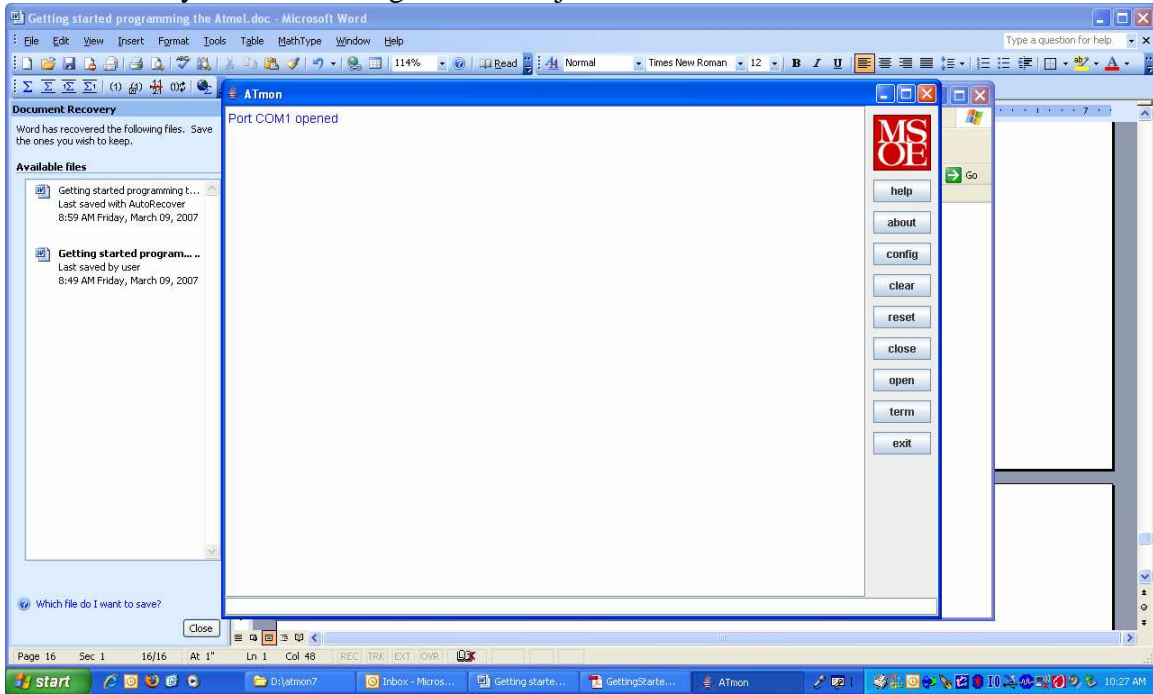
| Name | Value | Type | Location |
|---------|---------|----------|----------|
| Counter | 154 '8' | Register | R21 |
| | | | |
| | | | |
| | | | |
| | | | |

Downloading using Atmon

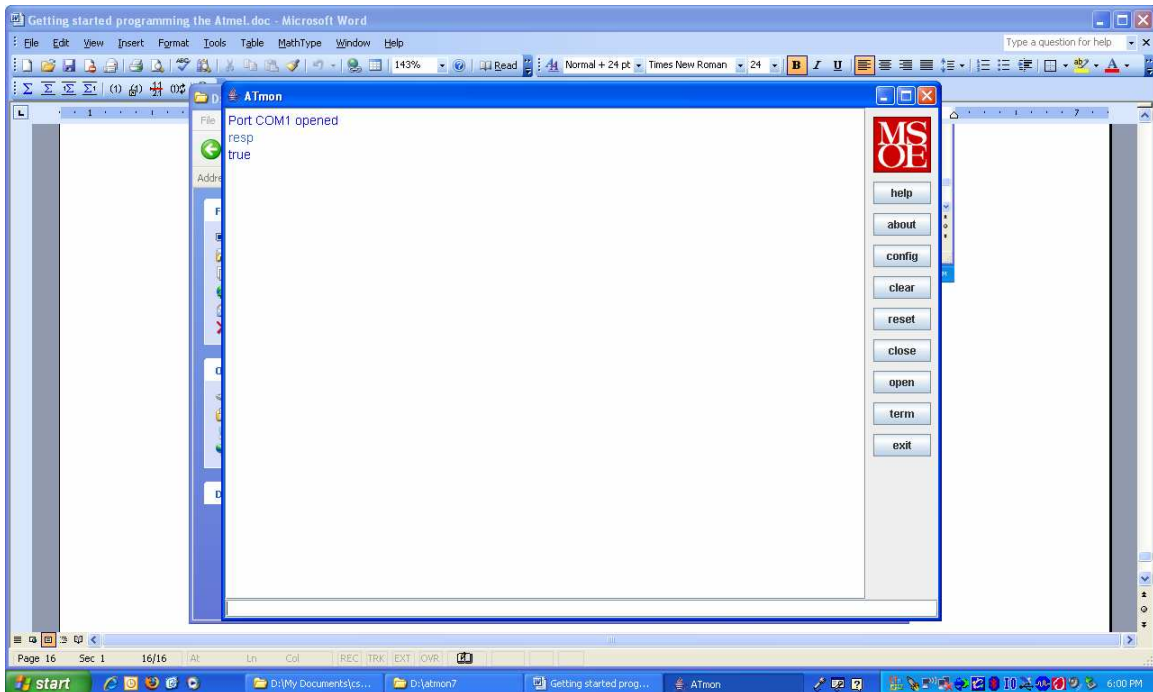
Open the directory in which you stored the Atmon tools and select the file called Atmon.jar



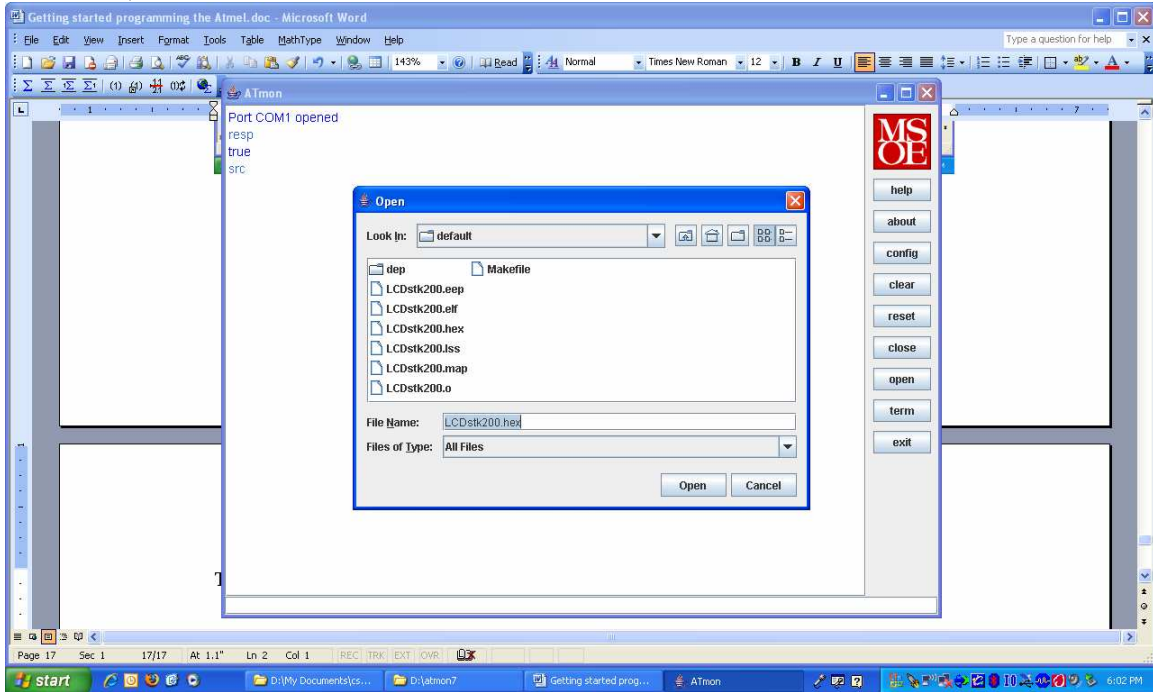
Run Atmon by double-clicking the Atmon.jar file



Test for connectivity by typing **resp** on the command line. You should get the response "true"



To download a program, type **src** on the command line and select the desired file (.hex extension)



At the command prompt type **wr v** to write and verify. You will see a progress bar that indicates that the write and verify is taking place. If there are no errors in the process, your application will immediately start executing.