# Section 1

# AVR Studio User Guide

## 1.1 Introduction

Welcome to AVR® Studio from Atmel Corporation. AVR Studio is a Development Tool for the AVR family of microcontrollers. This manual describes the how to install and use AVR Studio.

AVR Studio enables the user to fully control execution of programs on the AVR In-Circuit Emulator. AVR Studio supports source level execution of Assembly programs assembled with the Atmel Corporation's AVR Assembler and C programs compiled with IAR Systems' ICCA90 C Compiler for the AVR microcontrollers.

AVR Studio runs under Microsoft® Windows® 95 and Microsoft Windows NT®.

## 1.2 Installing AVR Studio

**In order to install AVR Studio under Windows 95 and Windows NT 4.0:**

1. Insert the diskette labeled "AVR Studio Diskette" 1 in drive A:
2. Press the "Start" button on the Taskbar and select "Run"
3. Enter "A:SETUP" in the Open field and press the OK button
4. Follow the instructions in the Setup program

**In order to install AVR Studio under Windows NT 3.51:**

1. Insert the diskette labeled "AVR Studio Diskette 1" in drive A:
2. Select "Run" from the "File" menu
3. Enter "A:SETUP" in the Command Line field and press the "OK" button
4. Follow the instructions in the Setup program

**Installing AVR Studio from WEB**

1. Connect to www.atmel.com and locate ASTUDIO.EXE in the AVR Software section
2. Download ASTUDIO.EXE to a temporary directory
3. Run ASTUDIO.EXE from your local disk. This will extract the setup program
4. Run SETUP.EXE
5. Follow the instructions in the Setup program

Once AVR Studio has been installed, it can be started by double clicking the AVR Studio icon. If an emulator is the desired execution target, remember to connect and power on the AVR In-Circuit Emulator before starting AVR Studio.

## 1.3 Description

This section gives a brief description of the main features of AVR Studio. In order to execute a program using AVR Studio, it must first be compiled with IAR Systems' C Compiler or assembled with Atmel's AVR Assembler to generate an object file which can be read by AVR Studio.

An example of what AVR Studio may look like during execution of a program is shown below. In addition to the Source window, AVR Studio defines a number of other windows which can be used for inspecting the different resources on the microcontroller.

The key window in AVR Studio is the Source window. When an object file is opened, the Source window is automatically created. The Source window displays the code currently being executed on the execution target, and the text marker is always placed on the next statement to be executed.

By default, it is assumed that execution is done on source level, so if source information exists, the program will start up in source level mode. In addition to source level execution of both C and Assembly programs, AVR Studio can also view and execute programs on a disassembly level. The user can toggle between source and disassembly mode when execution of the program is stopped.

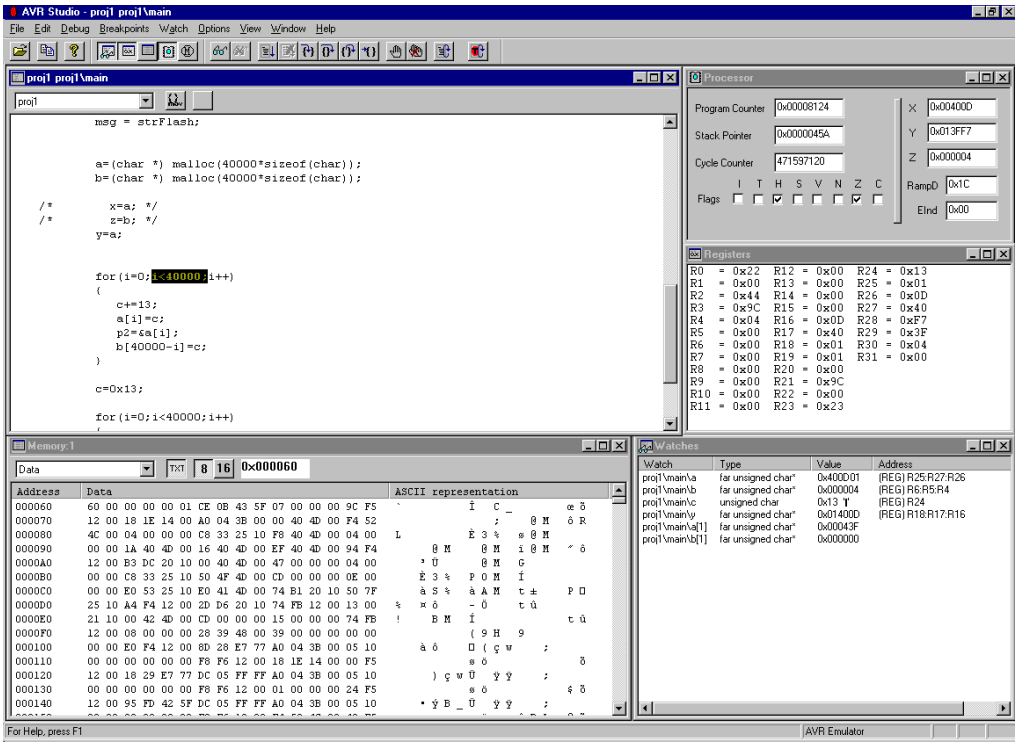
All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until that statement is reached, stop the execution, and reset the execution target. In addition, the user can have an unlimited number of code breakpoints, and every breakpoint can be defined as enabled or disabled. The breakpoints are remembered between sessions.

The Source window gives information about the control flow of the program. In addition, AVR Studio offers a number of other windows which enables the user to have full control of the status of every element in the execution target. The available windows are:

- Watch window: Displays the values of defined symbols. In the Watch window, the user can watch the values of variables in a C program.
- Register window: Displays the contents of the register file. The registers can be modified when the execution is stopped.
- Memory windows: Displays the contents of the Program Memory, Data Memory or I/O Memory. The memories can be viewed as hexadecimal values or as ASCII characters. The memory contents can be modified when the execution is stopped.
- Message window: Displays messages with timestamps from AVR Studio to the user.
- Processor window: Displays vital information about the execution target, including Program Counter, Stack Pointer, Status Register, Cycle Counter, X&Y&Z pointer, RampD register and Eind register. These parameters can be modified when the execution is stopped.

The first time an object file is being executed, the user needs to set up the windows which are convenient for observing the execution of the program, thereby tailoring the information on the screen to the specific project. The next time that object file is loaded, the setup is automatically reconstructed

The different windows will be described more carefully in the next chapter.




## 1.4 AVR Studio Windows

### 1.4.1 Source window

The Source window is the main window in an AVR Studio session. It is created when an object file is opened, and is present throughout the session. If the Source window is closed, the session is terminated.

The Source window displays the code which is being executed. An example of a Source window is given below.

The next instruction to be executed is always marked by AVR Studio. If the marker is moved by the user, this next statement can still be identified since the previously marked text becomes blue.

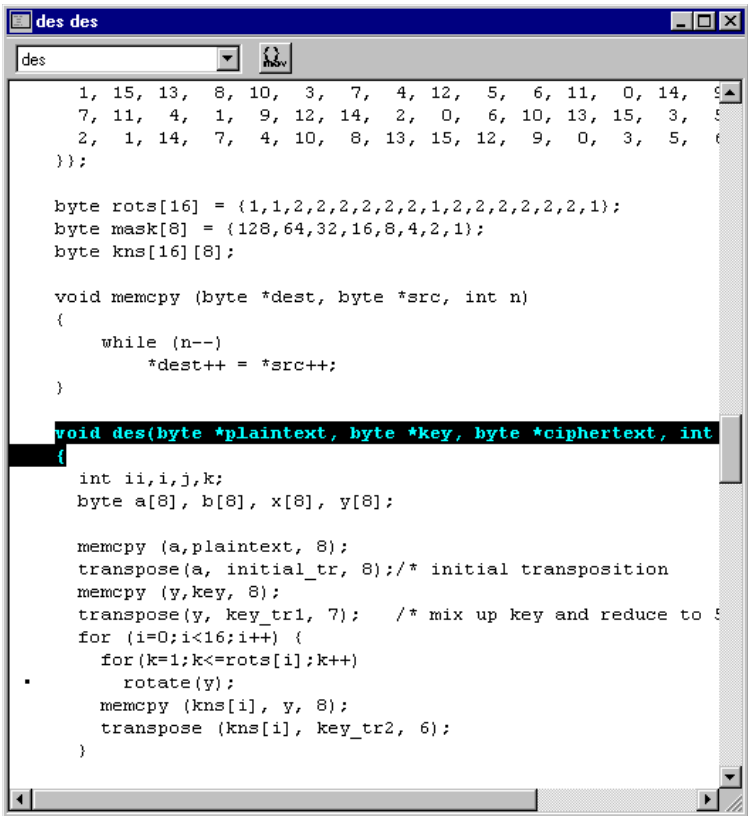
A breakpoint is identified in the Source window as a dot to the left of the statement where the breakpoint is set.

If the cursor is placed on a statement and a Run to Cursor command is issued, the program will execute until it reaches the instruction where the cursor is placed. Breakpoints are set in a similar way: the cursor is placed on a statement, and a Toggle Breakpoint command is issued. If a breakpoint was already set on the statement, the breakpoint will be removed. If no breakpoint was set on the statement, a breakpoint is inserted.

An object file can consist of several modules. Only one module is displayed at a time, but the user can change to the other modules by selecting the module of interest in the selection box on the top left of the Source window. This is a useful feature for viewing and setting breakpoints in other modules than the one currently active.

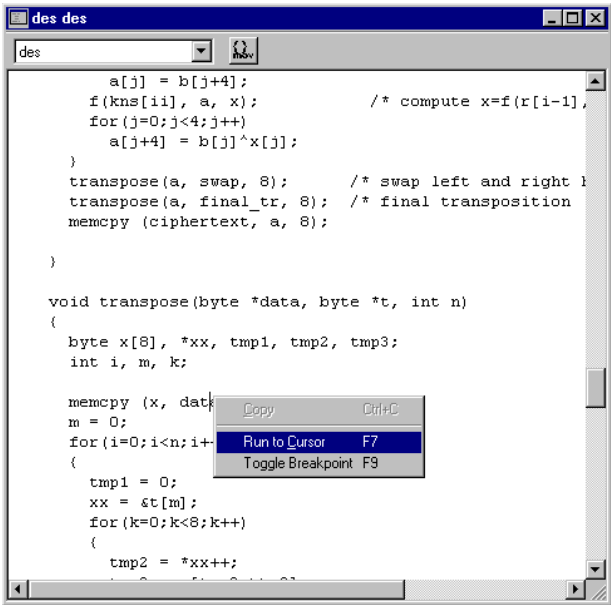
If the button to the right of the module selection box is pressed, the Source window switches between source level and disassembly level execution. When AVR Studio is in disassembly mode, all operations, such as Single stepping, is done on disassembly

level. In some cases, no source level information is available, for instance if an Intel-Hex file is selected as the object file. When no source level information is available, execution will be done on disassembly level..



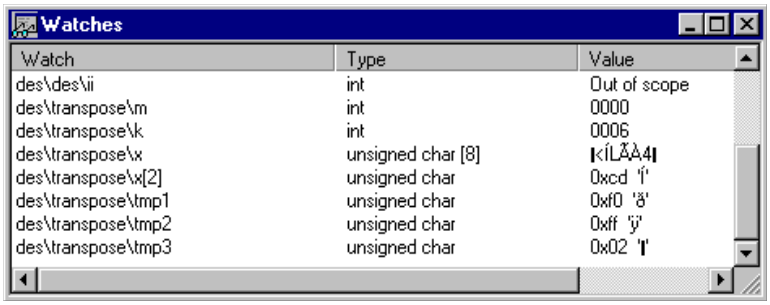

The Source window supports the Windows Clipboard. The user can select parts of (or all) the contents in the Source window and then copy it to the Windows Clipboard by selecting Copy from the Edit menu.

The Toggle breakpoint, Run to Cursor and the Copy functions are also available by pressing the right mouse button in the Source window. When the right mouse button is pressed, a menu appears on the screen:



### 1.4.2 Watch window

The Watch window can display the types and values of symbols like for instance variables in a C program. Since the *AVR* Assembler does not generate any symbol information, this window can only be used in a meaningful way when executing C programs. An example of a Watch window is given below.




The Watch window has three fields. The first field is the name of the symbol which is being watched. The next is the type of the symbol, and the third is the value of the symbol. By default, the Watch window is empty, i.e. all the symbols the user would like to watch have to be added to the Watch window. Once a symbol has been added, it is remembered also in subsequent executions of the programs. The added watches are also remembered if the Watch window is closed.

There are commands for adding watches, deleting watches and deleting all watches. A watch is added by giving an Add Watch command from the Watch menu or from the Debug toolbar. A watch can also be added by pressing the INS key if the Watch window is the active window. When an Add Watch command is issued, the user must enter the name of the symbol. The user can enter a symbol name with or without scope information.

*AVR* Studio will first search for the symbol as if it contains scope information. If no such symbol is found, *AVR* Studio appends the symbol name to the current scope, and searches for this new symbol. If no such symbol is found, the symbol is unbound, “???” appears in the type field, and the value field remains empty. If the symbol name is found, the symbol is bound, the symbol with scope information is displayed in the watch field, and the type and value fields are filled out. Every time execution stops, AVR Studio tries to bind unbound symbols using the current scope.

It is not possible to have floating symbols. Once a symbol is bound, it remains bound. The watches are remembered between sessions. Whether or not the symbol has been bound is a part of this information. If the program enters a scope where a bound symbol is not visible, the value field changes to “Out of scope”.

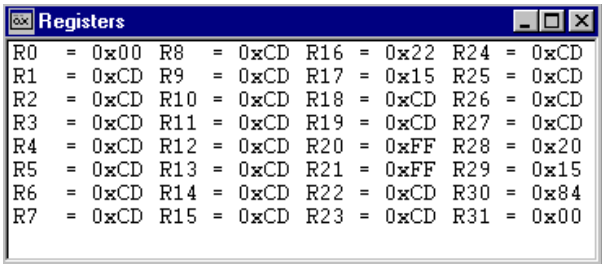
In order to delete a watch, the symbol name must first be clicked on using the left mouse button. When a symbol has been marked this way, AVR Studio accepts the Delete Watch command from the Watch menu. If the Watch window is the currently active window, the marked symbol can also be deleted by pressing the DEL key.

The Watch window can be used for watching C arrays and structs as well as simple variables. The syntax is the same as in C (use braces (“[“ and “]”) for arrays and dot (“.”) for structs). When watching arrays, variables can be used for dynamically indexing the arrays. It is for example possible to watch “my_array[i]” if i is an integer in the same scope as the array “my_array”.

There can only be one Watch window active at a time. The watched symbols (with scope information) are remembered between sessions. The Watch window can also be toggled on and off, and the watches are also remembered if the Watch window is toggled off.

### 1.4.3 Register window

The Register window displays the contents of the 32 registers in the AVR register file. An example of the Register window is given below.

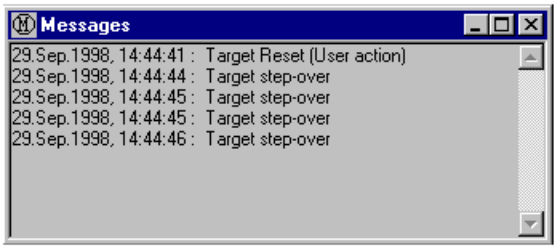



When the Register window is resized, the contents is reorganized in order to best fit the shape of the window.

The values in the Register window can be changed when the execution is stopped. In order to change the contents of a register, first make sure the execution is stopped. Then place the cursor on the register to change, and double-click the left mouse-button. The register can then be changed. Type in the new contents in hexadecimal form. Finally, press the Enter key to confirm or the ESC key to cancel the change. Only one Register window can be active at a time.

### 1.4.4 Message window

The Message window displays messages with timestamps from AVR Studio to the user. When a Reset command is issued, the contents of the Message window is cleared. An example of a Message window is given below.

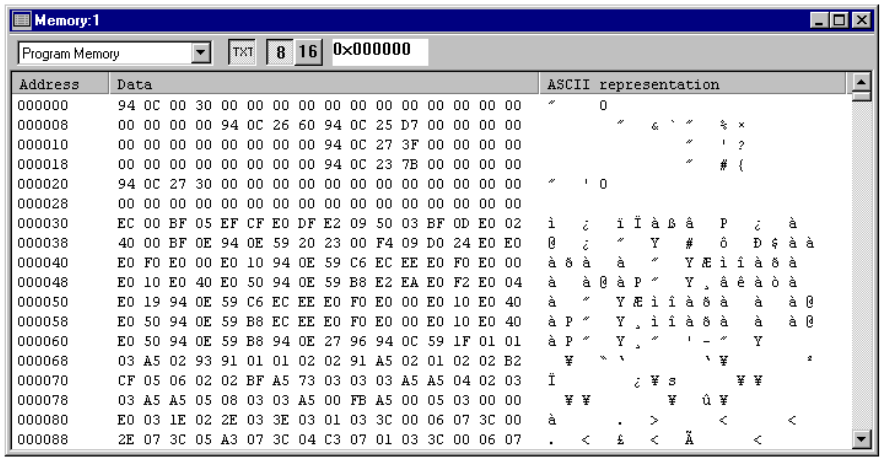



The contents in the Message window is remembered also when the Message window is toggled off and then on again. Only one Message window can be active at a time.

### 1.4.5 Memory window

The Memory window enables the user to inspect and modify the contents of the various memories present in the execution target. The same window is used to view all memory types. The Memory window can be used to view Data memory, Program memory and I/O memory.

The user can have several concurrent Memory windows. An example of a Memory window is shown below.



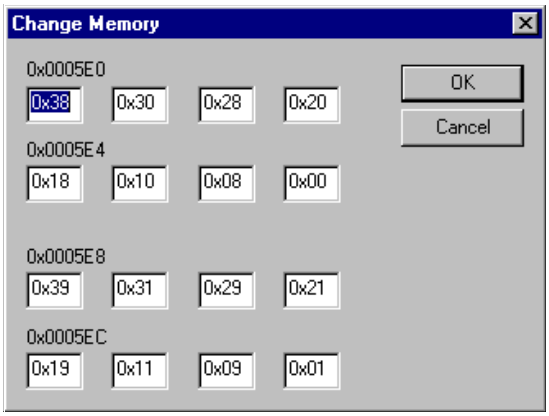
Which Memory type to view can be changed in the memory selection box at the top left of the Memory window. When a new Memory window is created, Data memory is the default memory type. AVR Studio not only keeps track over where the Memory windows are placed, but also which memory type it is displaying, and also the formatting status of the Window.

A hexadecimal representation of the addresses and the contents of the memory is always displayed. In addition, the user can view the memory contents as ASCII characters. The user also has the option to group the hexadecimal representation into 16 bit groups in stead of 8 bit groups. When viewing Program memory, it is the Word address which is displayed in the address column, and the LSB is listed before the MSB in the data column.

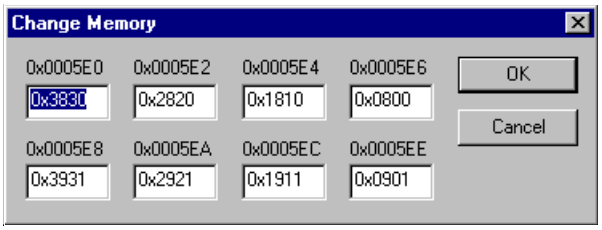
### 1.4.6 Modifying memory

The user can modify the contents of the memories by issuing a double click on the line containing the item(s) to be changed. When a line in the Memory view is doubleclicked, a Window appears on the screen. If memory is viewed in 8 bit groups, the modifications are done on 8 bit groups and when memory is viewed as 16 bit groups, the modifications are done on 16 bit groups.

When operating on 8 bit(data) groups, the following Window appears:



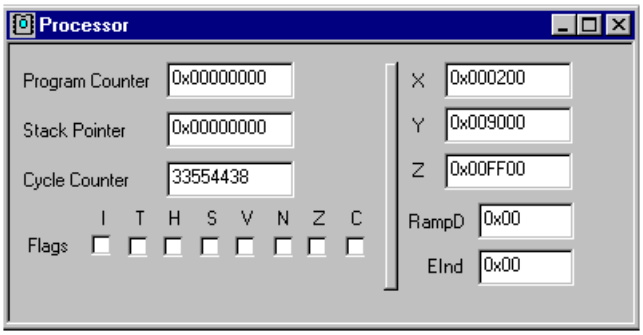

When operating on 16 bit#(program) groups, the following Window appears:




The operation is the same in the two cases. If the Cancel button is pressed, no update is done even if the user has edited one or more of the values. If the OK button is pressed, the Memory is updated if one or more of the values are changed.

### 1.4.7 Processor window

The Processor window contains vital information about the execution target. An example of a Processor window is shown below.




The Program Counter indicates the address of the next instruction to be executed. The Program Counter is displayed in hexadecimal form, and can be changed when the execution is stopped. When the Program Counter is changed, the current instruction is discarded. After the Program Counter is changed, the user must press the Single step function to jump to the desired address.

The Stack Pointer holds the current value of the Stack Pointer which is placed in the I/O area. If the Target has a Hardware stack instead of an SRAM based stack, this is indicated in the Stack Pointer field. The Stack Pointer value can be changed when the execution is stopped.

The Cycle Counter gives information about the number of clock cycles elapsed since last reset. It is not implemented in early releases of the V3 ICE.

The Flags is a display of the current value of the Status register. When the execution is stopped, these bits can be changed by clicking on the flags to change. A checked flag indicates that the flag is set (the corresponding bit in the Status register has the value 1).

The X, Y, Z, RampD and Eind registers holds the current value of the registers. RampD updates correct only when enabled in the options menu.

Only one Processor window can be active at a time.

## 1.5 Commands

AVR Studio incorporates a number of different commands. The commands can be given in various ways: through menu selections, toolbar buttons and by keyboard shortcuts. This section describes the available commands, and how they are invoked.

### 1.5.1 Administrative

#### 1.5.1.1 Opening files

When Open is selected from the File menu, a file selection dialog appears on the screen (note that AVR assumes the file extension .OBJ, so by default, only files with this extension are listed). The user must then select the object file to execute. Currently, the AVR Studio supports the following formats:

- IAR UBROF
- AVR Object Files generated by the Atmel AVR Assembler
- Intel-Hex

AVR Studio automatically detects the format of the object file. The four most recently used files are also available under the File menu and can be selected for loading directly.

When opening the file, AVR Studio looks for a file with the same filename as the file selected but with the extension AVD. This is a file AVR Studio generates when a file is closed, and it contains information about the project, including window placement. If the AVD project file is not found, only a Source window is created.

The AVD file also contains information regarding breakpoints. Breakpoints defined in the previous session are reinserted unless the object file is newer than the project file. In the latter case, the breakpoints are discarded.If source level information is available, the program is executed until the first source statement is reached.

**1.5.1.2 Closing files**

When Close is selected from the File menu, all the windows in a session are closed. AVR Studio also writes a file in the same directory as the object file, containing project information. The file has the same name as the object file, but has the extension AVD.

**1.5.1.3 Copying text**

The user can mark text in the Source window and transfer this to the Windows Clipboard by selecting Copy from the Edit menu.

**1.5.2 Execution Control**

Execution commands are used for controlling the execution of a program. All execution commands are available through menus, shortcuts and the Debug toolbar.

**1.5.2.1 Go**

The Go command in the Debug menu starts (or resumes) execution of the program. The program will be executed until it is stopped (user action) or a breakpoint is encountered. The Go command is only available when the execution is stopped. *Shortcut: F5*

**1.5.2.2 Break**

The Break command in the Debug menu stops the execution of the program. When the execution is stopped, all information in all windows are updated. The Break command is only available when a program is executing. *Shortcut: CTRL-F5*

**1.5.2.3 Trace Into**

The Trace Into command in the Debug menu executes one instruction. When AVR Studio is in source mode, one source level instruction is executed, and when in disassembly level, one assembly level instruction is executed. After the Trace Into is completed, all information in all windows are updated. *Shortcut: F11*

**1.5.2.4 Step Over**

The Step Over command in the Debug menu executes one instruction. If the instruction contains a function call/subroutine call, the function/subroutine is executed as well. After the Step Over is completed, all information in all windows are updated. *Shortcut: F10*

**1.5.2.5 Step Out**

The Step Out command in the Debug menu executes until the current function has completed. If a user breakpoint is encountered during Step Over, execution is halted. If a Step Out command is issued when the program is on the top level, the program will continue executing until it reaches a breakpoint or it is stopped by the user. After the Step Out command is completed, all information in all windows are updated. *Shortcut: SHIFT+F11.*

**1.5.2.6 Run to Cursor**

The Run to Cursor command in the Debug menu executes until the program has reached the instruction indicated by the cursor in the Source window. If a user breakpoint is encountered during a Run to Cursor command, execution is not halted. If the instruction indicated by the cursor is never reached, the program executes until it is stopped by the user. After the Run to Cursor command is completed, all information in all windows are updated. *Shortcut: F7*

**1.5.2.7 Reset**

The Reset command performs a Reset of the emulator. If a program is executing when the command is issued, execution will be stopped. If the user is in source level mode, the program will, after the Reset is completed, execute until it reaches the first source statement. After the Reset is completed, all information in all windows are updated. *Shortcut: SHIFT+F5*

**1.5.2.8 Target Reset**

The Target Reset command performs a Reset of the target platform. If a program is executing when the command is issued, execution will be stopped. If the user is in source level mode, the program will, after the Reset is completed, execute until it reaches the first source statement. After the Reset is completed, all information in all windows are updated.

## 1.6 Watches

When executing at source level, the Watch window can be used for watching symbols. When executing object files generated by the Atmel AVR Assembler, no symbol information is present so the Watch window can not be used for displaying any information.

### 1.6.1 Adding watches

In order to insert a new watch, the user must select Add Watch from the Watch window, or press the Add Watch button on the Debug toolbar. If the Watch window is not present when the Add Watch command is given, the Watch window is created, and already defined watches are reinserted (if any).

If the Watch window is the active window, a new watch can also be added by pressing the INS key.

### 1.6.2 Deleting watches

The user can delete a watch by first marking the symbol to be deleted in the Watch window and then give a Delete Watch command from the Watch menu or from the Debug toolbar. Selecting a watch is done by moving the mouse pointer to the name of the watch and pressing the left mouse button.

If the Watch window is the active window, a marked symbol can also be deleted by pressing the DEL key.

#### 1.6.2.1 Deleting all watches

The Delete all watches command is available from the Watch menu. When this command is issued, all defined watches are removed from the Watch window.

## 1.7 Breakpoints

The user can set an unlimited number of code breakpoints. The breakpoints are remembered between sessions unless a new object file has been generated. If the object file is newer than the project file, the breakpoints are discarded.

When a breakpoint is set on a location, the breakpoint is indicated by a dot on the left side of the instruction.

### 1.7.1 Toggle Breakpoint

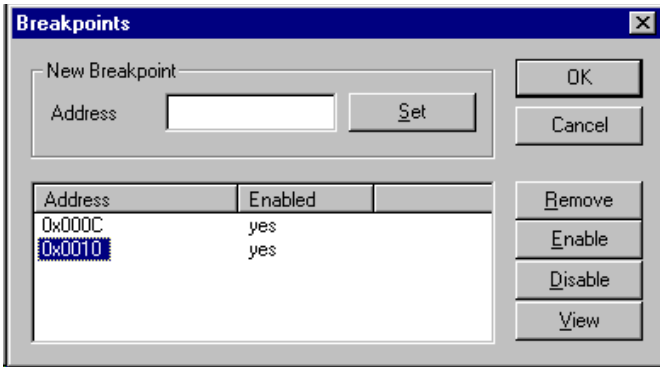
The Toggle Breakpoint command toggles the breakpoint status for the instruction where the cursor is placed. Note that this function is only available when the source window is the active view.

#### 1.7.1.1 Clear all breakpoints

This function clears all defined breakpoints, including breakpoints which have been disabled.

#### 1.7.1.2 Show list

When Show list is selected, the following dialog appears on the screen:




In the Breakpoints dialog, the user can inspect existing breakpoints, add a new breakpoint, remove a breakpoint, enable/disable breakpoints and view (goto) breakpoint.

## 1.8 Toolbars

AVR Studio contains three different toolbars described below. The toolbars can be individually removed and/or reinserted if desired by unchecking/checking them in the View → Toolbars menu.

### 1.8.1 The General toolbar

The General toolbar contains buttons for standard Windows commands. The General toolbar has the following buttons:

 Open File

 Copy

 Help

### 1.8.2 The Debug toolbar

The Debug toolbar contains buttons for execution control and Watch window control. The Debug toolbar has the following buttons:

 Add Watch

 Delete Watch

 Go

 Stop

 Trace Into

 Step Over

 Step Out

 Run to Cursor

 Toggle Breakpoint

 Clear all Breakpoints

 Reset

 Reset

 Target Reset

### 1.8.3 The Views toolbar

The Views toolbar contains buttons for enabling and disabling the most commonly used windows and for adding Memory windows. The Views toolbar has the following buttons:

 Toggle Watch window

 Toggle Register window

 Add Memory window

 Toggle Processor window

 Toggle Message window

## 1.9 Shortcut Summary

The following shortcuts are defined in AVR Studio:

**Table 1.** Shortcuts

| Command | Shortcut |
|---|---|
| Toggle Register window | Alt+0 |
| Toggle Watch window | Alt+1 |
| Toggle Message window | Alt+2 |
| Toggle Processor window | Alt+3 |
| Add Memory window | Alt+4 |
| Show Breakpoints List | Ctrl+B |
| Copy to Clipboard | Ctrl+C |
| Open File | Ctrl+O |
| Help | F1 |
| Run | F5 |
| Break | Ctrl+F5 |
| Reset | Shift+F5 |
| Run to Cursor | F7 |
| Toggle Breakpoint | F9 |
| Step Over | F10 |
| Trace Into | F11 |
| Step Out | Shift+F11 |

## 1.10 Execution Target

AVR Studio can be targeted towards an V3 In-Circuit Emulator. When the user opens a file, AVR Studio automatically detects whether an Emulator is present and available on one of the systems serial ports. If an Emulator is found, it is selected as the execution target.

If a V3 ICE is available in the system, it is automatically selected as the execution target. The Emulator must be connected through a serial port. If an Emulator is present in the system but can not be identified, close the file, reset the Emulator and try once more.

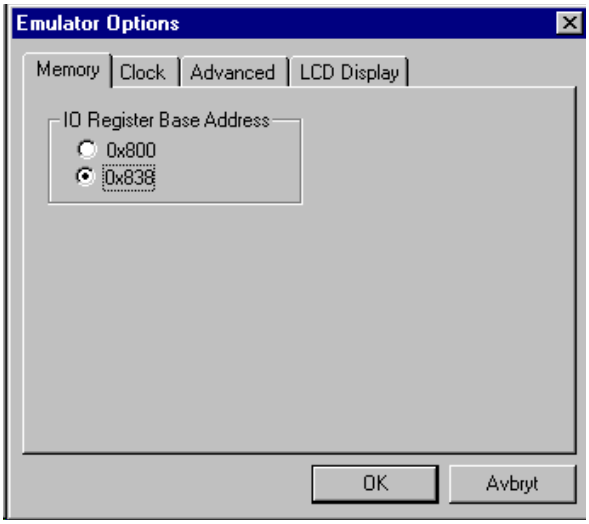
### 1.10.1 Emulator options

The emulator options will be displayed when a new project is started, or it can be selected from menu→options→emulator options. Settings will be remembered between sessions.

The following window will be displayed. Four sheets are available: Memory, Clock, Advanced and LCD display (LCD display is currently not used).

#### 1.10.1.1 Memory options sheet

Select IO Register Base Address



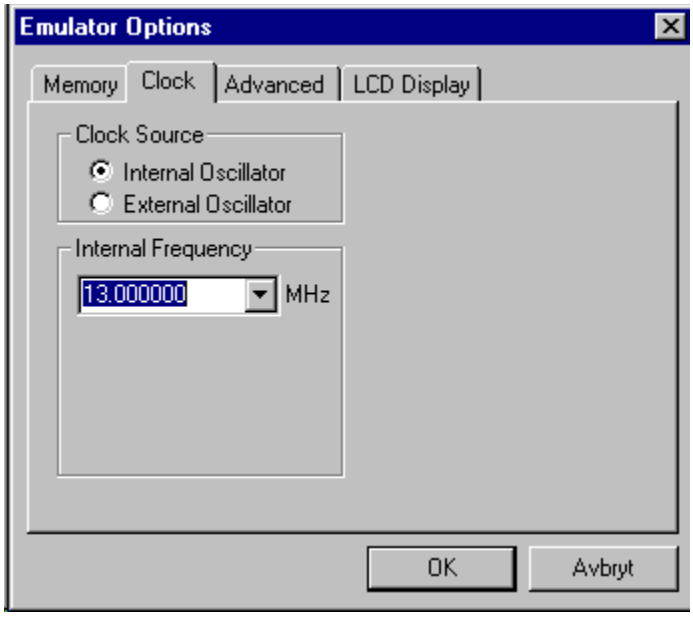

Default setting is 0x838.

**Table 2.** I/O Memory

| | RsCPUSub1 | RsCPUSub0 |
|---|---|---|
| 0x800 | 0x801 | 0x800 |
| 0x838 | 0x839 | 0x838 |

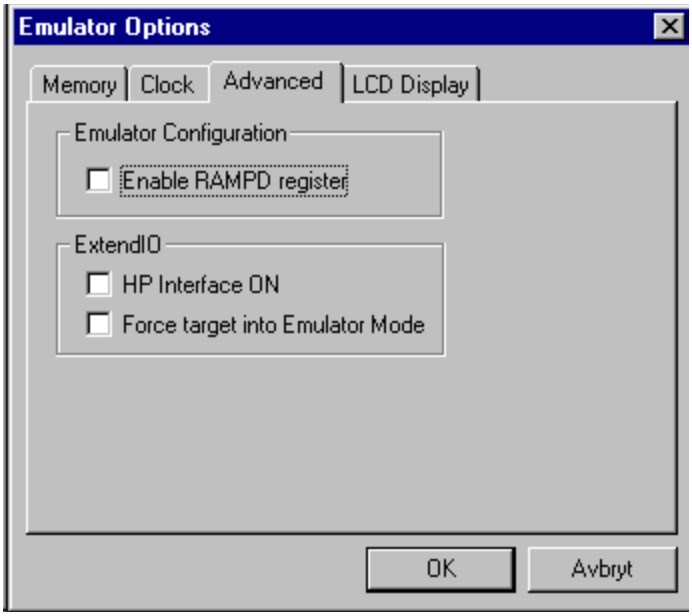
### 1.10.2 Clock options sheet

The user can select whether the Emulator should be clocked from the on-board programmable clock circuit, or if it should be clocked from en external source. If the Internal Oscillator is set as clock source, the user can select a frequency between 400#kHz and 20 MHz. The user can either select typical frequency from the list, or enter a custom frequency. Note that nor all frequencies can be exactly generated. The actual frequency is printed in the Message window. The speed of the Emulator is remembered between sessions.




Default setting is internal oscillator at 13 MHz.

### 1.10.3 Advanced options sheet




Toggle RAMPD register

Toggle HP interface ON/OFF

Force target into emulator mode

Default settings are OFF for all the features.

## Atmel Headquarters

***Corporate Headquarters***
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

***Europe***
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686677
FAX (44) 1276-686697

***Asia***
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road
Tsimshatsui East
Kowloon, Hong Kong
TEL (852) 27219778
FAX (852) 27221369

***Japan***
Atmel Japan K.K.
Tonetsu Shinkawa Bldg., 9F
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

## Atmel Operations

***Atmel Colorado Springs***
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

***Atmel Rousset***
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4 42 53 60 00
FAX (33) 4 42 53 60 01

***Fax-on-Demand***
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

***e-mail***
literature@atmel.com

***Web Site***
http://www.atmel.com

***BBS***
1-(408) 436-4309