

# **DTU8 Debugger User's Manual**

---

Program Development Support Software

ISSUE DATE: Dec. 2012

## NOTICE

No copying or reproduction of this document, in part or in whole, is permitted without the consent of LAPIS Semiconductor Co., Ltd.

The content specified herein is subject to change for improvement without notice.

The content specified herein is for the purpose of introducing LAPIS Semiconductor's products (hereinafter "Products"). If you wish to use any such Product, please be sure to refer to the specifications, which can be obtained from LAPIS Semiconductor upon request.

Examples of application circuits, circuit constants and any other information contained herein illustrate the standard usage and operations of the Products. The peripheral conditions must be taken into account when designing circuits for mass production.

Great care was taken in ensuring the accuracy of the information specified in this document. However, should you incur any damage arising from any inaccuracy or misprint of such information, LAPIS Semiconductor shall bear no responsibility for such damage.

The technical information specified herein is intended only to show the typical functions of and examples of application circuits for the Products. LAPIS Semiconductor does not grant you, explicitly or implicitly, any license to use or exercise intellectual property or other rights held by LAPIS Semiconductor and other parties. LAPIS Semiconductor shall bear no responsibility whatsoever for any dispute arising from the use of such technical information.

The Products specified in this document are intended to be used with general-use electronic equipment or devices (such as audio visual equipment, office-automation equipment, communication devices, electronic appliances and amusement devices).

The Products specified in this document are not designed to be radiation tolerant.

While LAPIS Semiconductor always makes efforts to enhance the quality and reliability of its Products, a Product may fail or malfunction for a variety of reasons.

Please be sure to implement in your equipment using the Products safety measures to guard against the possibility of physical injury, fire or any other damage caused in the event of the failure of any Product, such as derating, redundancy, fire control and fail-safe designs. LAPIS Semiconductor shall bear no responsibility whatsoever for your use of any Product outside of the prescribed scope or not in accordance with the instruction manual.

The Products are not designed or manufactured to be used with any equipment, device or system which requires an extremely high level of reliability the failure or malfunction of which may result in a direct threat to human life or create a risk of human injury (such as a medical instrument, transportation equipment, aerospace machinery, nuclear-reactor controller, fuel-controller or other safety device). LAPIS Semiconductor shall bear no responsibility in any way for use of any of the Products for the above special purposes. If a Product is intended to be used for any such special purpose, please contact a ROHM sales representative before purchasing.

If you intend to export or ship overseas any Product or technology specified herein that may be controlled under the Foreign Exchange and the Foreign Trade Law, you will be required to obtain a license or permit under the Law.

Windows is a registered trademark of Microsoft Corporation (USA) in USA and other countries, and other product names and company names are trademarks or registered trademarks.

Copyright 2008 - 2012 LAPIS Semiconductor Co., Ltd.

# CONTENTS

|  |               |
|--|---------------|
| <b>1 . INTRODUCTION .....</b>                                      | <b>6</b>      |
| <b>1.1 About This Product .....</b>                                | <b>6</b>      |
| 1.1.1 Features .....   | 8             |
| 1.1.2 Features in Dr.U8 ICE / Dr.U16 ICE / Dr.ICE Mode .....       | 9             |
| 1.1.3 Features in Simulation Mode .....                            | 9             |
| <b>1.2 Menus in Each Mode .....</b>                                | <b>9</b>      |
| <b>1.3 Manual Organization .....</b>                               | <b>11</b>     |
| <b>1.4 Notation .....</b>  | <b>12</b>     |
| <br><b>2 . BEFORE YOU BEGIN .....</b>                              | <br><b>13</b> |
| <b>2.1 System Requirements .....</b>                               | <b>13</b>     |
| <b>2.2 Supported File Formats .....</b>                            | <b>14</b>     |
| <br><b>3 . PREPARING TO DEBUG .....</b>                            | <br><b>16</b> |
| <b>3.1 Checking Hardware Environment .....</b>                     | <b>16</b>     |
| <b>3.2 Command Line Options .....</b>                              | <b>16</b>     |
| 3.2.1 Communications Parameters Dialog Box .....                   | 17            |
| 3.2.2 Confirmation Dialog Displayed at Startup in uEASE Mode ..... | 18            |
| <b>3.3 Checking and Modifying Memory Map .....</b>                 | <b>18</b>     |
| <b>3.4 Program Code Files .....</b>                                | <b>20</b>     |
| 3.4.1 Reading in Program Code .....                                | 20            |
| 3.4.2 Automatic Reloading .....                                    | 22            |
| 3.4.3 Important Notes .....  | 22            |
| <b>3.5 Project Files .....</b>                                     | <b>25</b>     |
| <br><b>4 . BASIC OPERATION .....</b>                               | <br><b>27</b> |
| <b>4.1 Screens and Menus .....</b>                                 | <b>27</b>     |
| <b>4.2 Windows .....</b>   | <b>29</b>     |
| 4.2.1 Opening Windows .....  | 29            |
| 4.2.2 Replicating Windows .....                                    | 30            |
| 4.2.3 Double-Clicking in a Window .....                            | 30            |
| <b>4.3 Miscellaneous .....</b>                                     | <b>31</b>     |
| 4.3.1 Tool Bar .....   | 31            |
| 4.3.2 Shortcut Keys .....  | 31            |
| 4.3.3 Miscellaneous .....  | 31            |

## CONTENTS

|  |               |
|--|---------------|
| <b>5 . SOURCE LEVEL DEBUGGING AND SYMBOLIC DEBUGGING .....</b>   | <b>33</b>     |
| <b>5.1 Source Window .....</b>                                   | <b>33</b>     |
| 5.1.1 Display .....  | 33            |
| 5.1.2 Displaying Values of Variables on Source Window .....      | 34            |
| 5.1.3 Search Order for Source Code Files .....                   | 35            |
| 5.1.4 Displaying a Different File .....                          | 35            |
| 5.1.5 Properties .....   | 37            |
| <b>5.2 Disassembly Window .....</b>                              | <b>37</b>     |
| 5.2.1 Display .....  | 37            |
| 5.2.2 Searching for Character String in Disassembly Window ..... | 38            |
| 5.2.3 Properties .....   | 40            |
| <b>5.3 Symbolic debugging .....</b>                              | <b>40</b>     |
| <b>5.4 Links Between Windows .....</b>                           | <b>41</b>     |
| <b>5.5 PC Tracking .....</b>                                     | <b>42</b>     |
| <br><b>6 . EMULATION .....</b>                                   | <br><b>43</b> |
| <b>6.1 Execution Modes .....</b>                                 | <b>43</b>     |
| 6.1.1 Continuous Execution (Emulation Execution) .....           | 43            |
| 6.1.2 Continuous Execution (Simulation Execution) .....          | 43            |
| 6.1.3 Execute to cursor .....                                    | 43            |
| 6.1.4 Reset and run .....  | 43            |
| 6.1.5 Step in .....  | 44            |
| 6.1.6 Step over .....  | 44            |
| 6.1.7 Step out .....   | 45            |
| 6.1.8 Step Execution at C Source Level .....                     | 45            |
| <b>6.2 Resets .....</b>  | <b>46</b>     |
| <b>6.3 Functionality Available During Emulation .....</b>        | <b>47</b>     |
| 6.3.1 Windows .....  | 47            |
| 6.3.2 Menus .....  | 48            |
| <b>6.4 Measuring Execution Times .....</b>                       | <b>49</b>     |
| <b>6.5 Exiting and Reconnecting during Emulation .....</b>       | <b>51</b>     |
| <b>6.6 Cycle Counter .....</b>                                   | <b>52</b>     |
| <b>6.7 Interrupt Simulation .....</b>                            | <b>53</b>     |
| 6.7.1 Interrupt Window .....                                     | 53            |
| <br><b>7 . BREAK FUNCTIONS .....</b>                             | <br><b>56</b> |
| <b>7.1 Breakpoints .....</b>                                     | <b>56</b>     |
| 7.1.1 Setting Breakpoints .....                                  | 56            |
| 7.1.2 Listing Breakpoints .....                                  | 58            |

|            |   |           |
|------------|---|-----------|
| 7.1.3      | Enabling/Disabling All Breakpoints.....                   | 59        |
| 7.1.4      | Breakpoints for C Source Code Lines.....                  | 59        |
| 7.1.5      | Notes on Setting Breakpoints.....                         | 60        |
| <b>7.2</b> | <b>Forced Break.....</b>                                  | <b>60</b> |
| <b>7.3</b> | <b>Break Condition Settings.....</b>                      | <b>61</b> |
| <b>8</b>   | <b>. REFERENCING/MODIFYING REGISTERS AND MEMORY .....</b> | <b>63</b> |
| 8.1        | Program Counter (PC).....                                 | 64        |
| 8.2        | Registers .....   | 65        |
| 8.3        | Special Function Registers (SFRs) .....                   | 66        |
| 8.4        | Memory.....   | 67        |
| 8.5        | Saving and Comparing Program Code .....                   | 71        |
| 8.5.1      | Saving Program Code.....                                  | 71        |
| 8.5.2      | Comparing Program Code .....                              | 72        |
| 8.6        | Saving and Restoring Data Regions .....                   | 72        |
| 8.6.1      | Saving Data Region.....                                   | 72        |
| 8.6.2      | Restoring Data Regions.....                               | 73        |
| 8.7        | Realtime RAM monitor .....                                | 74        |
| <b>9</b>   | <b>. TRACING FUNCTION .....</b>                           | <b>76</b> |
| 9.1        | Trace Window .....  | 76        |
| 9.2        | Mode of Display .....                                     | 77        |
| 9.3        | Trace Counter .....                                       | 78        |
| 9.4        | Storing Trace Data.....                                   | 79        |
| 9.5        | Searching for Trace Data .....                            | 81        |
| 9.6        | Clearing Trace Memory .....                               | 82        |
| 9.7        | Trace Trigger.....  | 83        |
| 9.8        | Others .....  | 84        |
| 9.8.1      | Jumping to Source Window /Disassembly Window .....        | 84        |
|            | Jumping to Specified Line .....                           | 84        |
| 9.8.2      | 84  |           |
| <b>10</b>  | <b>. WATCH FUNCTION .....</b>                             | <b>85</b> |
| 10.1       | Overview.....   | 85        |
| 10.2       | Adding Items .....  | 86        |
| 10.3       | Changing Data on Watch Window.....                        | 88        |
| 10.4       | Saving Watch Item List .....                              | 89        |
| 10.5       | Loading Watch Item List .....                             | 89        |
| 10.6       | Editing Watch Item List.....                              | 90        |

## CONTENTS

|   |            |
|---|------------|
| <b>10.7 Real-Time Watch .....</b>   | <b>90</b>  |
| 10.7.1 Real-Time Watch Display Options .....                              | 91         |
| <b>11 . OTHER FUNCTIONS .....</b>   | <b>93</b>  |
| <b>11.1 Coverage Function .....</b>                                       | <b>93</b>  |
| 11.1.1 Displaying the Coverage Rate .....                                 | 94         |
| <b>11.2 Macro Function .....</b>  | <b>95</b>  |
| 11.2.1 Overview .....   | 95         |
| 11.2.2 Executing a Single Script Command .....                            | 97         |
| <b>11.3 Log Function .....</b>  | <b>97</b>  |
| 11.3.1 Using Log Window .....   | 97         |
| <b>11.4 Customizing Various Windows.....</b>                              | <b>98</b>  |
| <b>11.5 On-Line Help.....</b>   | <b>101</b> |
| <b>11.6 Displaying Version Data .....</b>                                 | <b>102</b> |
| <b>11.7 Firmware Update Function.....</b>                                 | <b>103</b> |
| <b>11.8 Target Device change Function.....</b>                            | <b>105</b> |
| <b>11.9 Realtime LCD monitor Function .....</b>                           | <b>105</b> |
| 11.9.1 How to use the Realtime LCD monitor.....                           | 105        |
| <b>12 . APPENDIX .....</b>  | <b>107</b> |
| <b>12.1 Symbolic Debugging and Input Formats .....</b>                    | <b>107</b> |
| 12.1.1 Symbols .....  | 107        |
| 12.1.2 Expressions.....   | 108        |
| 12.1.3 Value Lists.....   | 109        |
| <b>12.2 Watch Item Candidates .....</b>                                   | <b>110</b> |
| 12.2.1 Adding Local Variables.....  | 110        |
| 12.2.2 Adding Global Variables .....                                      | 110        |
| 12.2.3 Adding Function Parameters .....                                   | 110        |
| 12.2.4 Displaying C Variable Types.....                                   | 111        |
| 12.2.5 Mixing C and ASM.....  | 111        |
| 12.2.6 Variable Assigned to Register .....                                | 111        |
| <b>12.3 Debugging Optimized C Source Code .....</b>                       | <b>113</b> |
| <b>12.4 Macro Script References .....</b>                                 | <b>114</b> |
| 12.4.1 Script Commands.....   | 114        |
| 12.4.2 Script Commands for Projects .....                                 | 116        |
| 12.4.3 Script Commands for Referencing/Changing CPU Contents .....        | 117        |
| 12.4.4 Script Commands for Referencing/Changing C Variables .....         | 118        |
| 12.4.5 Script Commands for Code Memory.....                               | 118        |
| 12.4.6 Script Commands for Data Memory.....                               | 120        |
| 12.4.7 Script Commands for Memory in Physical Segments 1 and Higher ..... | 122        |

|   |            |
|---|------------|
| 12.4.8 Script Commands for Emulation.....   | 123        |
| 12.4.9 Script Commands for Resetting .....  | 124        |
| 12.4.10Script Commands for Breaks .....   | 124        |
| 12.4.11Script Commands for Tracing .....  | 125        |
| 12.4.12Script Commands for Performance and Coverage.....  | 126        |
| 12.4.13Script Commands for Macros.....  | 127        |
| 12.4.14Script Commands for Symbols.....   | 128        |
| 12.4.15Other Script Commands .....  | 128        |
| <b>12.5 Errors in Emulator Connection .....</b>   | <b>130</b> |
| 12.5.1 Errors Common to When Dr.U8 ICE / Dr.U16 ICE / Dr.ICE Connected and When<br>uEASE Connected..... | 130        |
| 12.5.2 Errors in uEASE Connection.....  | 133        |
| 12.5.3 Errors in Dr.U8 ICE / Dr.U16 ICE / Dr.ICE Connection.....  | 135        |

# 1. Introduction

---

## 1.1 About This Product

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The DTU8 debugger for evaluating and debugging programs created with the CCU8 compiler and MACU8 assembler packages is part of the support environment for developing application programs for 8-bit RISC processors based on the LAPIS Semiconductor nX-U8/100 core and 16-bit RISC processors based on the LAPIS Semiconductor nX-U16/100 core.

The DTU8 debugger supports the following mode.

“Dr.U8 ICE mode,” which enables a debug function by connecting the debugger to the Dr.U8 ICE in-circuit emulator.

“Dr.U16 ICE mode,” which enables a debug function by connecting the debugger to the Dr.U16 ICE in-circuit emulator.

“Dr.ICE mode,” which enables a debug function by connecting the debugger to the Dr.610XXX in-circuit emulator.

“uEASE mode,” which enables a debug function by connecting the debugger to the uEASE on-chip debug emulator.

“nanoEASE mode,” which enables a debug function by connecting the debugger to the nanoEASE on-chip debug emulator.

“Simulation mode,” which enables a simulation debug function by using the simulation engine of the nX-U8/100 core inside the debugger.

Using these functions, the debugger supports evaluating and debugging embedded application programs for microcontrollers.

This manual uses the following signs in order to a) separately give an explanation concerning Dr.U8 ICE mode only, Dr.U16 ICE mode only, Dr.ICE mode only, uEASE mode only, and simulation mode only, and b) give a common explanation of all of these modes, for convenience of explanation.

**Dr.U8 ICE**

Explains the content that only relates to Dr.U8 ICE mode.

**Dr.U16 ICE**

Explains the content that only relates to Dr.U16 ICE mode.

**Dr.ICE**

Explains the content that only relates to Dr.ICE mode.

**uEASE**

Explains the content that only relates to uEASE mode.

**nanoEASE**

Explains the content that only relates to nanoEASE mode.



## 1.1 About This Product

**Simulate**

Explains the content that only relates to simulation mode.

**Dr.U8 ICE**  
**uEASE**  
**Simulate**

Explains the common content of Dr.U8 ICE mode, uEASE mode and Simulation mode.

## 1. Introduction

### 1.1.1 Features

The debugger incorporates the following advanced features to create an easy-to-use interactive debugging environment.

#### ■ **Windows Xp/Vista/7 support.**

This 32-bit application provides a powerful debugging environment.

#### ■ **Source Level and Symbolic Debugging**

Source level debugging allows the user to set breakpoints in C and assembly language source code, execute source code in steps, etc. It also allows the user to use function names, label symbols, and constants defined in the source code in place of numerical values.

Source window links with the Disassemble and Trace windows allow the user to evaluate operation at the machine instruction level.

#### ■ **Powerful Watch Functions**

These powerful watch functions provide complete support for C language structures, unions, and pointer types.

At the assembly language level, these functions support all nX-U8 core addressing modes--including register indirect addressing with base.

#### ■ **Project Files**

Saving settings in project files allows the user to quickly roll back to a preceding configuration or rapidly switch between alternate configurations.

#### ■ **Exiting and Reconnecting during Emulation**

The user can exit and reload the debugger without disrupting emulation--allowing aging tests to run for days unattended, for example.

#### ■ **Macros**

These are text files containing batch processing scripts for automating repetitive tasks and purely mechanical sequences over extended periods.

#### ■ **Other Functions**

Command for comparing program files and emulator memory

Convenient built-in assembler for patching programs

Command for saving internal RAM data

On-line help

### 1.1.2 Features in Dr.U8 ICE / Dr.U16 ICE / Dr.ICE Mode

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE

The debugger has the following features in Dr.U8 ICE, Dr.U16 ICE, Dr.ICE mode:

#### ■ **Exiting and Reconnecting during Emulation**

The user can exit the debugger in the middle of emulation and can reconnect it to the emulator without disrupting emulation.

This is convenient for aging tests to run for days unattended, for example.

### 1.1.3 Features in Simulation Mode

Simulate

The debugger has the following features in simulation mode:

#### ■ **Simulating the nX-U8/100 core**

The simulation engine of the debugger allows simulation of the instructions of the nX-U8/100 core. Also, stand-alone use of the debugger allows program execution, which eliminates the need to connect an in-circuit emulator.

## 1.2 Menus in Each Mode

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
Simulate

The DTU8 debugger is used as part of various types of emulation systems. The DTU8 debugger is designed to automatically change menus and dialog boxes according to the emulation system connected and the user need not be aware of the differences when using the DTU8 debugger.

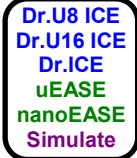
This manual describes the DTU8 debugger's functions in general; however, some functions are not supported depending on emulation system. Unsupported functions are not displayed on the menu. See below for the menu list for each mode. Of the functions described in this manual, skip those that are not available for the target mode.

| Menu | Menu command             | Dr.U8 ICE | Dr.U16 ICE | Dr.ICE | uEASE | nano EASE | Simulate |
|------|--------------------------|-----------|------------|--------|-------|-----------|----------|
| File | Open project             | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Save project             | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Save project as          | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Load program file        | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Save program to file     | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Compare program and file | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Load RAM data file       | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Save RAM data to file    | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Select source file       | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Exit                     | ●         | ●          | ●      | ●     | ●         | ●        |
| Edit | Undo                     | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Cut                      | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Copy                     | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Paste                    | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Delete                   | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Select all               | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Find                     | ●         | ●          | ●      | ●     | ●         | ●        |
|      | Replace                  | ●         | ●          | ●      | ●     | ●         | ●        |

## 1. Introduction

| Menu   | Menu command                             | Dr.U8<br>ICE | Dr.U16<br>ICE | Dr.ICE | uEASE | nano<br>EASE | Simulate |
|--------|--|--------------|---------------|--------|-------|--------------|----------|
| View   | Source                                   | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Disassembly                              | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Status                                   | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Register                                 | ●            | ●             | ●      | ●     | ●            | ●        |
|        | SFR                                      | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Code                                     | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Data                                     | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Memory in physical segments 1 and higher | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Trace                                    | ●            | ●             | ●      | —     | —            | ●        |
|        | Watch                                    | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Realtime RAM monitor                     | ●            | ●             | —      | —     | —            | —        |
|        | Log                                      | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Interrupt                                | —            | —             | —      | —     | —            | ●        |
|        | Tool bar                                 | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Status bar                               | ●            | ●             | ●      | ●     | ●            | ●        |
| Run    | Reset                                    | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Change program counter                   | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Device Reset                             | —            | —             | —      | ●     | ●            | —        |
|        | Execute program                          | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Execute to cursor position               | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Reset and run                            | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Force break                              | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Step over                                | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Step in                                  | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Step out                                 | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Toggle (set/remove) breakpoint           | ●            | ●             | ●      | ●     | ●            | ●        |
|        | List Breakpoints                         | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Clear all breakpoints                    | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Validate breakpoints                     | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Break conditions                         | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Trace trigger set                        | ●            | ●             | —      | —     | —            | —        |
|        | Add sync out point                       | —            | ●             | ●      | —     | —            | —        |
|        | List sync out points                     | —            | ●             | ●      | —     | —            | —        |
|        | Clear all sync out points                | —            | ●             | ●      | —     | —            | —        |
|        | Reset sync out signal                    | —            | ●             | ●      | —     | —            | —        |
| Tool   | Run macro                                | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Stop macro                               | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Script command                           | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Symbol list                              | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Add watch item                           | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Realtime LCD monitor                     | ●            | ●             | —      | —     | —            | —        |
|        | Operation settings                       | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Environment settings                     | ●            | ●             | ●      | ●     | ●            | ●        |
| Window | Close active window                      | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Close all                                | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Replicate window                         | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Cascade windows                          | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Tile vertically                          | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Tile horizontally                        | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Update                                   | ●            | ●             | ●      | ●     | ●            | ●        |
| Help   | Help topic                               | ●            | ●             | ●      | ●     | ●            | ●        |
|        | System information                       | ●            | ●             | ●      | ●     | ●            | ●        |
|        | Update Firmware                          | ●            | ●             | —      | ●     | ●            | —        |
|        | Change target device of ICE              | ●            | ●             | —      | —     | —            | —        |
|        | Version information                      | ●            | ●             | ●      | ●     | ●            | ●        |

## 1.3 Manual Organization



This manual consists of the following Chapters.<sup>1</sup>

### ■ **Chapter 1 Introduction**

This is the Chapter that you are now reading.

### ■ **Chapter 2 Before You Begin**

This Chapter describes the debugger's operating environment requirements and installation procedure.

### ■ **Chapter 3 Preparing to Debug**

This Chapter describes loading the debugger, procedures for executing programs, and the contents of project files.

### ■ **Chapter 4 Basic Operation**

This describes basic menu and window operations for the DTU8 debugger.

### ■ **Chapter 5. Source Level and Symbolic Debugging**

This Chapter gives the procedures for source level debugging in Source windows and symbolic debugging using expressions based on symbols and numeric values.

### ■ **Chapter 6 Emulation**

This Chapter describes emulation and closely related functions.

### ■ **Chapter 7 Break Functions**

This Chapter describes the use of break conditions to automatically suspend emulation.

### ■ **Chapter 8 Referencing/Modifying Registers and Memory**

This Chapter gives the procedures for viewing the contents of memory locations, the program counter, and other registers and for changing those contents.

### ■ **Chapter 9 Watch Function**

This Chapter gives the procedures for adding registers, data memory addresses, variables in the program source code, and other items to the Watch window for automatic monitoring by the debugger.

### ■ **Chapter 10 Trace Function**

This Chapter describes a tracing function used to check program execution progress.

---

<sup>1</sup> The user's manual is explained for every chapter according to function as a tutorial. Please use an online help to look for a function from a menu and a dialog box. Moreover, the on-line document explaining the matter for which the publication to a manual or an online help is not of use, a special mention matter, etc. is also attached.

## 1. Introduction

### ■ *Chapter 11 Other Functions*

This Chapter describes such additional capabilities as macros, logging, and on-line help.

### ■ *Chapter 12 Appendix*

This describes input formats for symbols and expressions, Watch window items, and macro script commands.

## 1.4 Notation

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

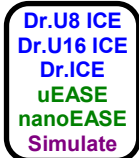
Individual keystrokes are represented by the lettering on the corresponding keys; combinations, by such representations joined with plus signs. The notation Shift+F1, for example, represents the combination produced by holding down a Shift key and hitting the F1 key.

This document does not use icons or any other indications requiring explanation.

## 2. Before You Begin

---

### 2.1 System Requirements



The DTU8 debugger assumes the following environment.

#### ■ **Development Languages**

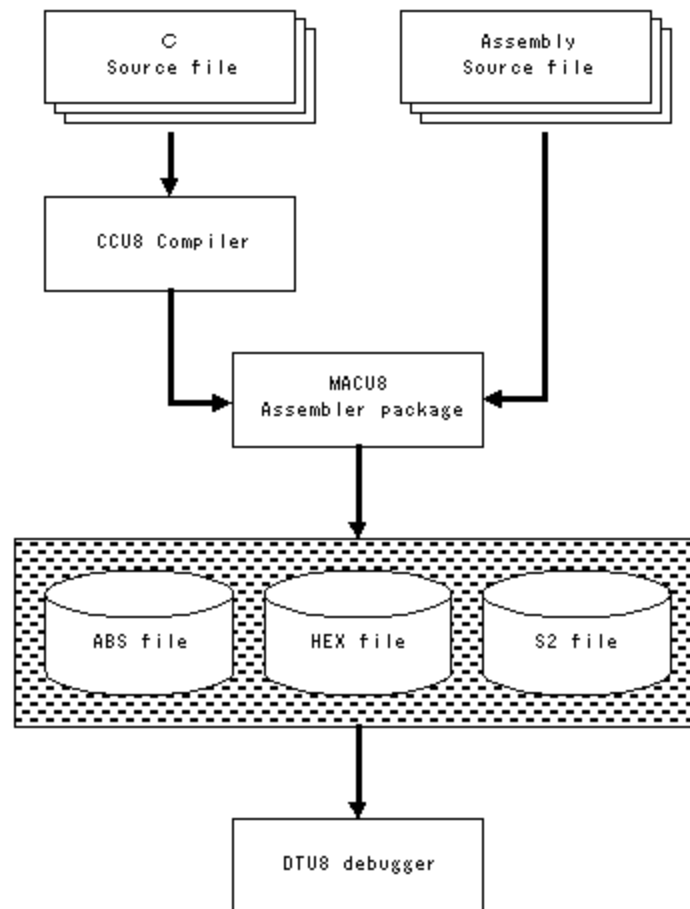
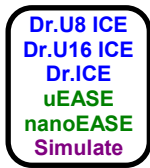
CCU8 Ver. 1.00 or later  
RASU8 Ver. 1.00 or later

#### ■ **Operating Environment**

- Operating system: A 32-bit version of Windows XP, Windows Vista or Windows 7, and 64-bit version of Windows Vista or Windows 7.
- Hardware environment supporting the above operating system
- Graphic adapter and display supporting at least SVGA (800 × 600) resolution
- Minimum of 20 megabytes of free hard disk space
- CD-ROM drive
- Mouse
- USB port

## 2. Before You Begin

### 2.2 Supported File Formats



The DTU8 debugger supports the following CCU8 and RASU8 output file formats: U8 absolute object format, and Intel HEX format, Motorola S2 format.

#### ■ **U8 Absolute Object File (Extension: .ABS)**

Original binary format, this file format provides both user program code (non-relocatable, executable code) and debugging information. It is therefore the CCU8 and RASU8 output file format to specify for displaying source files in debugger Source windows and debugging the program in those windows.

#### ■ **Intel HEX File (Extension: .HEX)**

This general-purpose ASCII format is frequently the format required by commercial PROM writers. One limitation is that it does not support addresses over one megabyte.

CCU8 and RASU8 extend this format to include symbol tables. Because this extension follows guidelines within the original Intel HEX format specifications, however, the files remain compatible with a wide range of existing utilities and PROM writers.

#### ■ **Motorola S2 File (Extension: .S)**

This general-purpose ASCII format is also widely supported by commercial PROM writers. One advantage is that it supports addresses over one megabyte.



## 2.2 Supported File Formats

This format stores user defined symbol tables in a separate file with the same base name and the extension `.SYM`. Note, however, that this symbol table is incomplete, listing only public symbols and not local symbols.

## 3. Preparing to Debug

---

### 3.1 Checking Hardware Environment

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE

Before using the debugger in Dr.U8 ICE, Dr.U16 ICE, Dr.ICE or uEASE mode, check all connections to the emulator and user application circuit as well as all hardware settings. This checklist should contain at least the following items.

- Power supply to the emulator
- Operating frequency (supplied clock)
- Target device settings
- Connections to user application circuit and user cable
- Connections to probe cable

The above checklist covers just the bare minimum. Expand it to include any additional hardware environment settings specified in the manual included with the emulator.

Before running the debugger, always power up the emulator and make sure that it is ready to connect to the DTU8 debugger.

Before turning off the power to the emulator, always exit the debugger.<sup>2</sup>

### 3.2 Command Line Options

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

There are three ways to load the debugger.

- Select an icon in the start menu that the *nX-U8 Development Tool Setup* program automatically created under [Start] → [U8 Tools] → [nX-U8] → [DTU8 debugger].
- Double-click the DTU8 icon or shortcut in Explorer.
- Enter the command line under [Start] → [Run].

The debugger has the following command line syntax.

DTU8.exe [*load\_file*]  
where *load\_file* is the name of the program or project file to load.

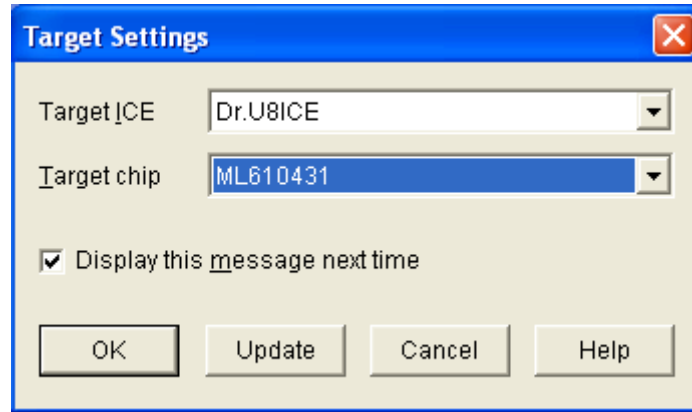
---

<sup>2</sup> Otherwise, the message that the debugger sends the emulator when it shuts down triggers a communications error

### 3.2.1 Communications Parameters Dialog Box

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The first time that you run the debugger, it displays the following dialog box for specifying the communications settings.



From the “Target chip” pull-down list, select the model to be used.

From the [Target ICE] pull-down list, select the target ICE from among the following:

| Target ICE      | Description  |
|-----------------|--|
| Dr.U8 ICE       | Select this when connecting the debugger with the Dr.U8 ICE in-circuit emulator.   |
| Dr.U16 ICE      | Select this when connecting the debugger with the Dr.U16 ICE in-circuit emulator.  |
| uEASE           | Select this when connecting the debugger with the uEASE on-chip debug emulator.    |
| nanoEASE        | Select this when connecting the debugger with the nanoEASE on-chip debug emulator. |
| SimU8 Simulator | Select this when activating the debugger in simulation mode.                       |
| Dr.ICE          | Select this when connecting the debugger with the Dr.610XXX in-circuit emulator.   |

Deselecting the Display this next time check box skips this dialog box the next time that the debugger loads. The debugger immediately links to the emulator and then displays the dialog box.

Note that all settings other than Debugging setup and Communications link also appear on the Communications parameters tab of the Tools menu *Environment settings* menu command dialog box.

Pushing the Cancel button aborts the debugger.

If the Update button is pushed, the debugger will be started in updating mode. In this mode, updating a firmware of Dr.U8 ICE and uEASE is performed, and change of a target device of Dr.U8 ICE can be made. Please refer to “11.7 Firmware Update Function” for the updating method of firmware. Please refer to “11.8 Target Device Change Function” for the change method of a target device.

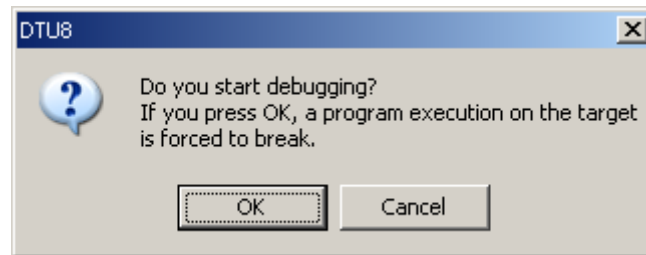
### 3. Preparing to Debug

#### 3.2.2 Confirmation Dialog Displayed at Startup in uEASE/nanoEASE Mode

uEASE  
nanoEASE

When uEASE or nanoEASE is selected for the target ICE in the Communications parameter dialog box, the following dialog box is displayed.

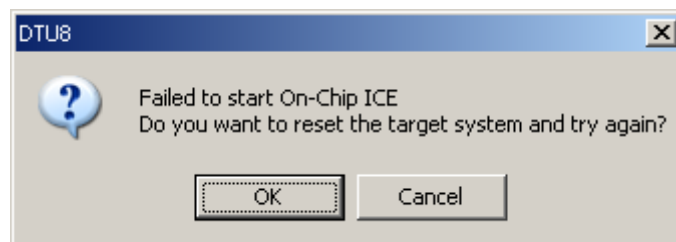
Immediately after uEASE or nanoEASE is connected to the PC and target LSI with a cable, the target LSI may be executing a program. For this reason a confirmation dialog box will be displayed that asks the user whether to start debugging in that state.



When the OK button is clicked on the dialog box, uEASE activates the on-chip ICE block on the target LSI, and if the target LSI is executing a program, uEASE forces a break in program execution.

Clicking the Cancel button exits the debugger.

If the activation of the on-chip ICE block or the processing of a forced break fails, the following dialog box is displayed:



Clicking the OK button resets the target LSI to restart the on-chip ICE. Clicking the Cancel button exits the debugger.

When you run the debugger, the Security ID checking dialog box may be displayed.

If this dialog box is displayed, select "Initialize Flash memory and security setting" and click OK.

### 3.3 Checking and Modifying Memory Map

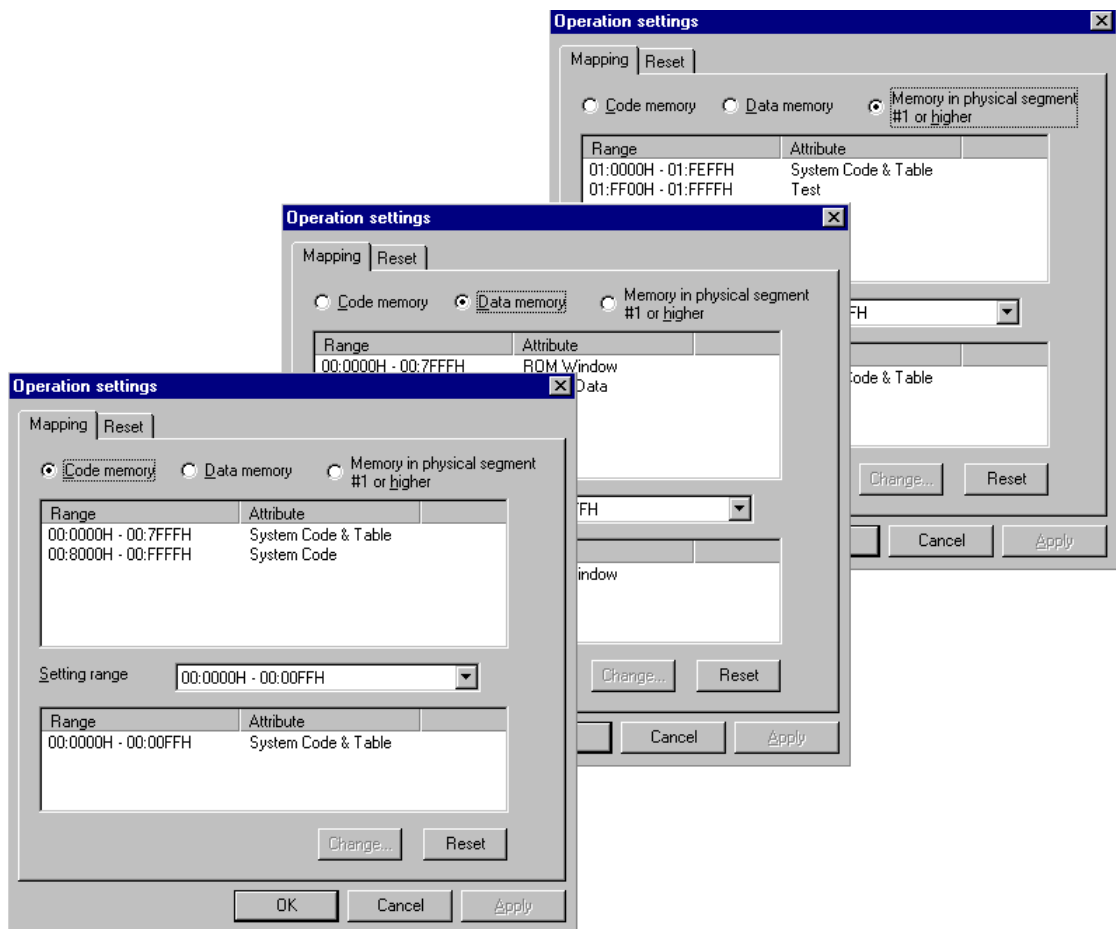
Dr.ICE  
Simulate

The memory map assigns memory types to portions of the nX-U8 core address space.

Selecting the Tools menu *Operating settings* menu command opens the dialog box with a Mapping tab displaying the memory maps for checking and modifying the assignments for code memory, data memory, and physical segments 1 and higher.

### 3.3 Checking and Modifying Memory Map

When immediately after activation in simulation mode, these settings specify the default memory maps for the target microcontroller or, if a project file is read in, the settings saved in that project file.

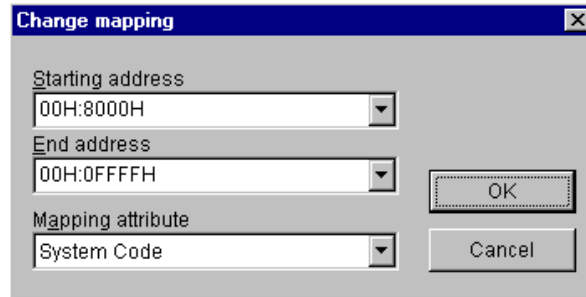


The mappings available depend on the emulator settings and the target microcontroller. For the address ranges and attributes available, refer to the manual included with the emulator. The following are examples of when the developer might wish to modify these settings.

- Designating unused address regions as N/A, having them trigger a ROM or RAM N/A area access break, and thus making it easier to detect when the program has run out of control
- Temporarily expanding the built-in ROM and RAM regions to run the program without worrying about optimization of the program code or memory used
- Using system memory inside the emulator as a substitute for external memory in the target system--particularly useful in evaluating external ROM code (system code or system code & table) during the initial stages of development

Changing a memory map involves specifying the memory type (code memory, data memory, or physical segments 1 and higher), selecting the region to modify, selecting the closest address range from the detailed list for that region, and pressing the Modify... button.

### 3. Preparing to Debug



Specify the address range and mapping attribute in the Change mapping attribute dialog box that appears. Pressing the OK button adds the specifications to the list displayed on the Mapping tab in the Tools menu *Operating settings* menu command dialog box. Note, however, that actually switching the emulator to the new mapping requires closing the dialog box with the OK button or pressing the Update button.

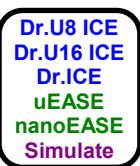
## 3.4 Program Code Files

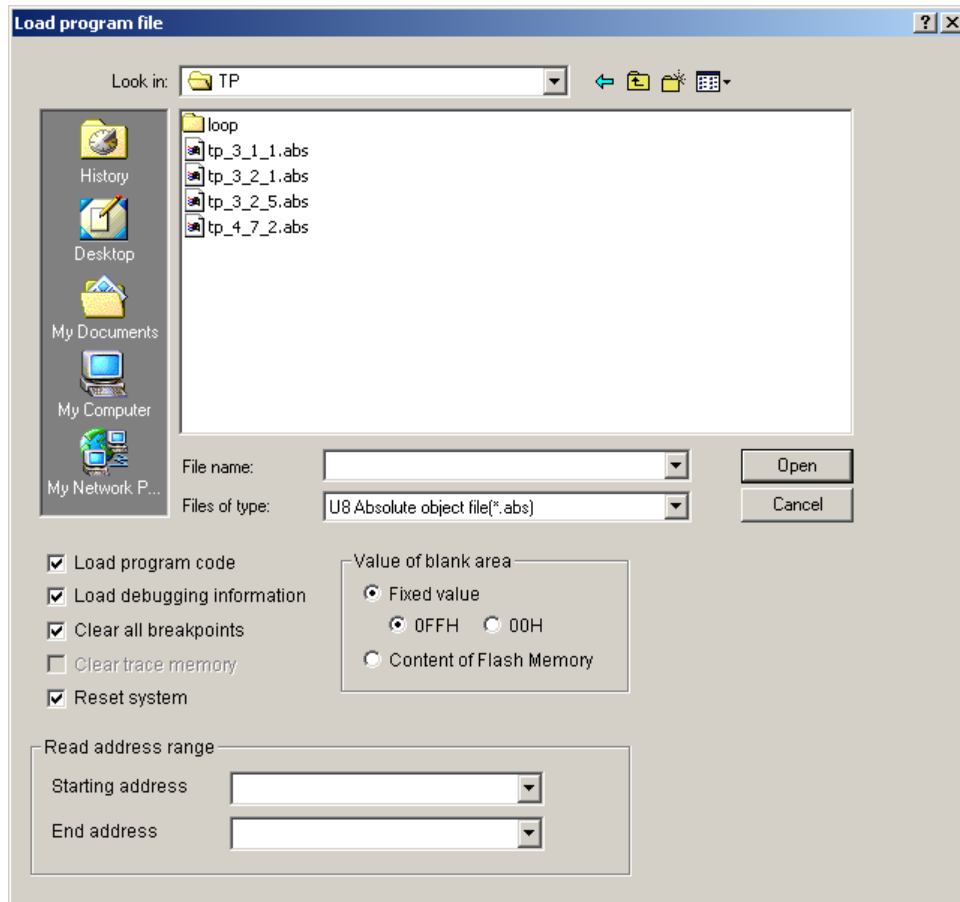
### 3.4.1 Reading in Program Code

The *Load program file...* command on the File menu is for loading the program code to debug.

Although the debugger supports other file formats, we recommend using exclusively the absolute object file (.ABS) file format during development. It is the only one supporting source level debugging.

Note that specifying a directory in the Program files field on the Directory tab of the Tools menu *Environment settings* menu command dialog box eliminates the need to specify the path when reading in absolute object files.





#### ■ **Load program code**

Select this to load the executable code from the selected file.

#### ■ **Load debugging information**

Select this to load the source line data and symbol table from the selected file.

Source level debugging requires selecting both this check box and the preceding one and then reading in an absolute object file containing debugging information.

Reading in new debugging information clears any debugging information already in memory.

#### ■ **Clear all breakpoints**

Select this to clear all existing breakpoints after loading the program code. In most cases, these have no relevance for the new program code.

#### ■ **Clear trace memory**

Select this to clear trace memory after loading the program code. In most cases, the trace memory has no relevance for the new program code.

#### ■ **Reset target chip**

Select this to reset the target chip and initialize the program counter and other registers after loading the program code.

### 3. Preparing to Debug

#### ■ Value of blank area

If a blank area is found upon loading of a program code, the blank area is padded with the value specified by Value of blank area.

[Fixed Data] When this is selected, only 0FFH or 00H can be specified for the Value of blank area. All the area (memory in physical segments 1 or higher is included) of program memories other than the program code contained in a program file are filled up with a fixed data.

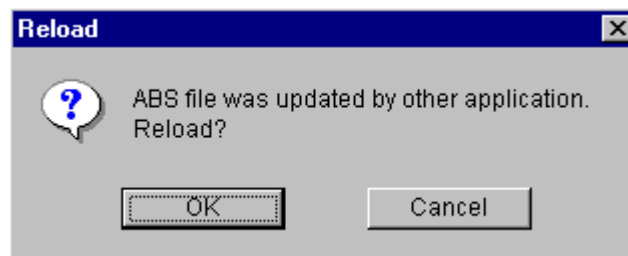
[Content of Flash Memory] When this is selected, the current contents of the flash memory are used as the filling data.

#### ■ Address Range

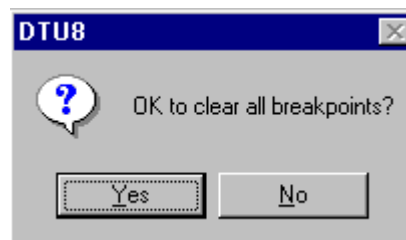
At the bottom of the dialog box, there are two list boxes for limiting the address range--to read in an updated ROM table or to reload certain subroutines, for example. Specify the first and last addresses in the desired range.

### 3.4.2 Automatic Reloading

Switching to the debugger from another application triggers a check that displays the following confirmation message if the time stamp on the last program file read in with debugging information has changed in the interim.



Pressing the Yes button to reload the file displays the following confirmation message if there are breakpoint settings.



These two functions help link the debugger with the object code build tool.

### 3.4.3 Important Notes

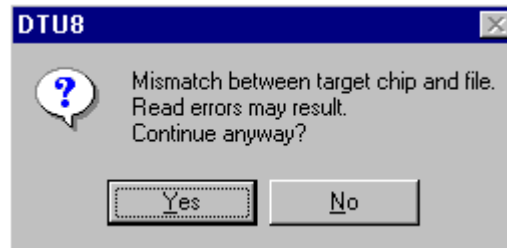
#### ■ Load Regions

Loads are limited to program code and NVDATA code and to memory mapping regions with either the system code attribute (System Code, System Code & Table, or System Code & Data) or the NVDATA attribute. Attempting to read code into other regions aborts the load operation with an error message.



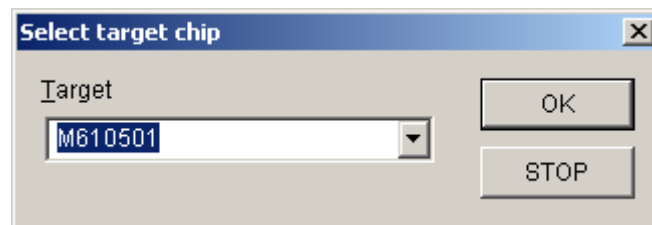
### ■ *Device-Specific Information*

The DTU8 debugger displays the following warning message when the emulator does not match the device specified in the absolute object (.ABS) file with the CCU8 /T command line option or with TYPE directives in the assembly language source code.



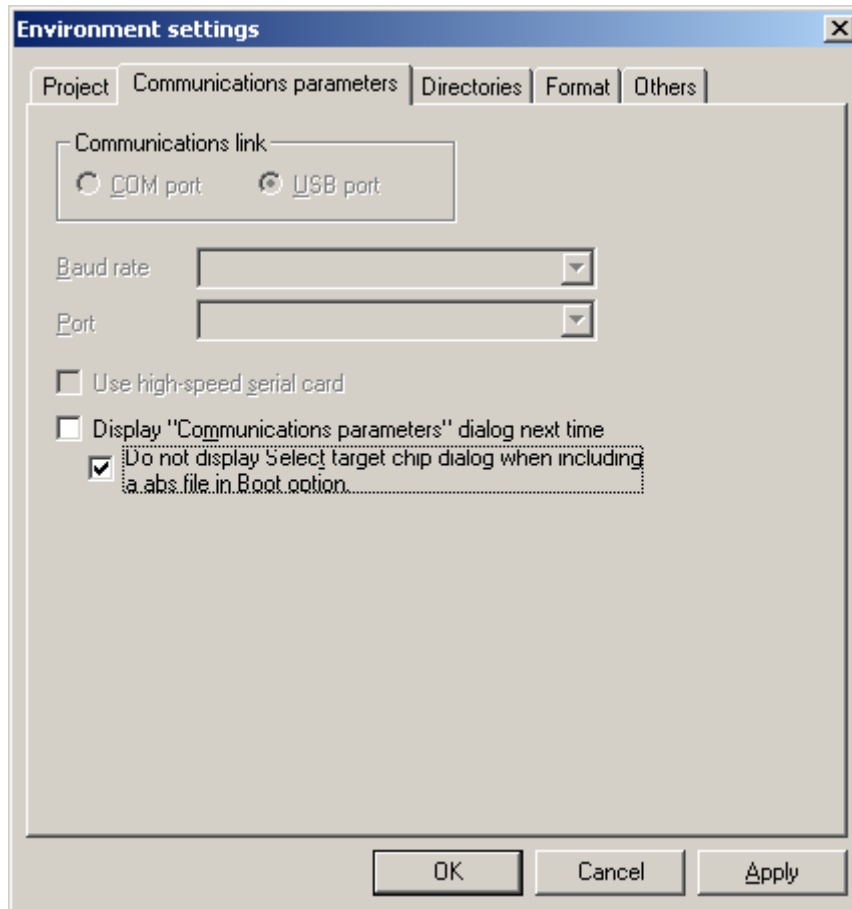
### ■ *Settings for not displaying Select target chip dialog box*

In Dr.U8 ICE, Dr.U16 ICE, Dr.ICE, uEASE or nanoEASE mode, when the 'Display "Communications parameters" dialog box next time' check box is not selected in the Tools menu *Environment settings* menu command dialog box and then the DTU8 debugger is run, the following dialog box appear:



Settings can be so made that this dialog box will not appear only when you run the debugger by specifying an absolute object file (.ABS). To make this dialog box not appear, configure settings on the Communications parameters tab of the Tools menu *Environment settings* menu command dialog box, as shown below.

### 3. Preparing to Debug



The Select target chip dialog box mentioned above can be made not to appear by running the debugger with the 'Display "Communications parameters" dialog box next time' check box deselected and the Do not display Select target chip dialog when including a abs file in Boot option check box checked.

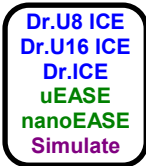
#### ■ **Breakpoints After a Reload**

The general rule is to always clear breakpoints after reloading a program because there is no guarantee that the breakpoint addresses maintained by the DTU8 debugger still match the appropriate source code lines.

The following, however, are examples of situations where the developer might chose to deliberately ignore this rule.

- The rebuild changes only options that do not affect code generation.
- All breakpoints are at the start of absolute segments, and modifications do not affect those addresses.
- The modifications do not change table data or other execution addresses.
- The developer reserves the right to determine whether the addresses still match based on the breakpoint list.

## 3.5 Project Files



Taking full advantage of the DTU8 debugger's many capabilities requires configuring a multitude of settings. Saving these settings in a project file eliminates the need to repeat this setup process each time.

Note, however, that project files save only settings. They do not save the contents of ROM, RAM, or registers.

Saving different sets of settings in separate project files also allows the developer to easily and rapidly switch between alternate debugging configurations.

The DTU8 debugger uses project files according to the given target ICE. Therefore, project files having a different target ICE cannot be read. Consequently, a project file saved in uEASE mode cannot be read in Dr.U8 ICE mode or a project file saved in Dr.U8 ICE mode cannot be read in uEASE mode.

Project files contain the following settings.

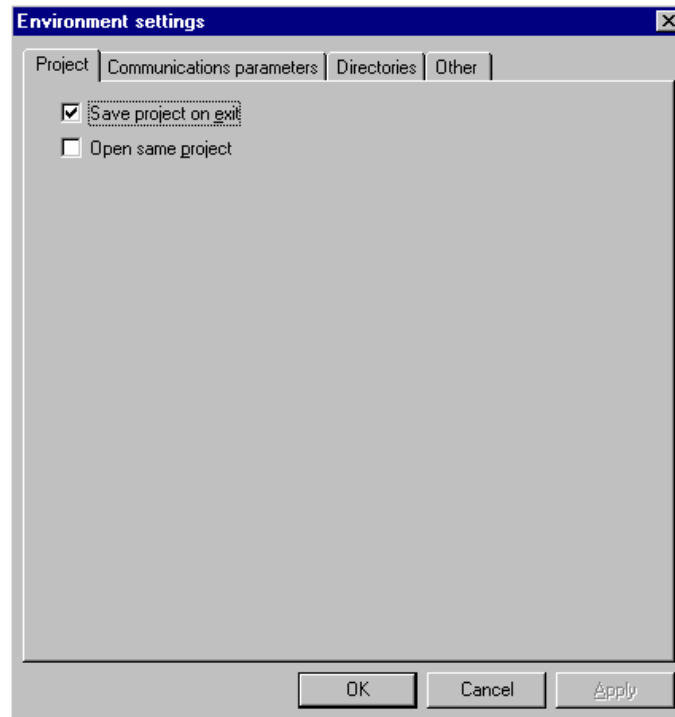
- Target microcontroller information
- Name of the last program file loaded into the debugger
- Operation settings property sheets
- Environment settings from the Tools menu
- Desktop state
- Communications parameters
- Breakpoints
- Sync out points
- Break settings
- Watch settings

Project files have the extension .PDT.

### 3. Preparing to Debug

#### ■ **Project Tab**

Selecting the Save project upon exit check box automatically saves any updated settings in the project file when the DTU8 debugger exits.



Selecting the Reload next time check box automatically reloads the same project files the next time that the DTU8 debugger runs.

#### ■ **Reading In Project Files**

Choosing the File menu *Open project* menu command and selecting a project file updates debugger settings from the ones saved to the file.

Note that specifying a directory in the Project files field on the Directory tab of the Tools menu *Environment settings* menu command dialog box eliminates the need to specify the path to the project file in the above step.

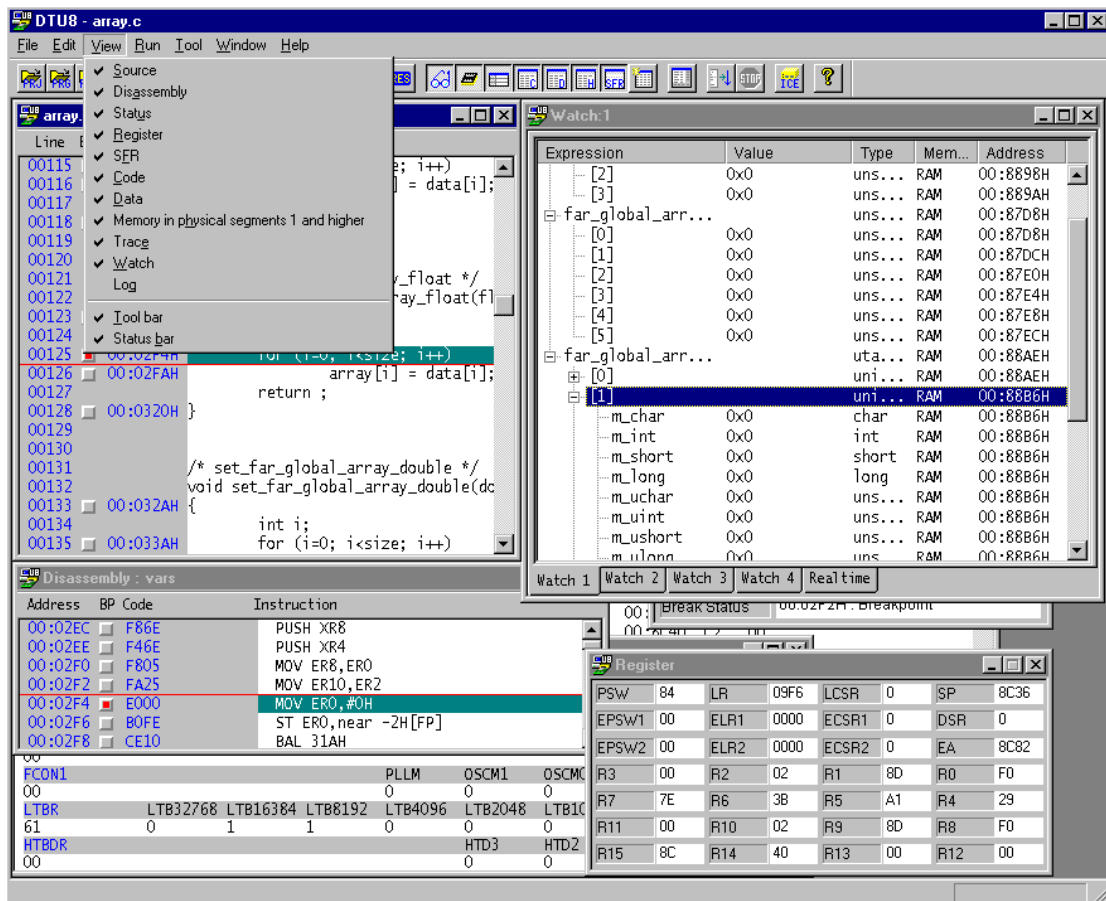
## 4. Basic Operation

### 4.1 Screens and Menus

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The DTU8 debugger features a graphical user interface (GUI) controlling the operation of all functions with menus, windows, and dialog boxes.

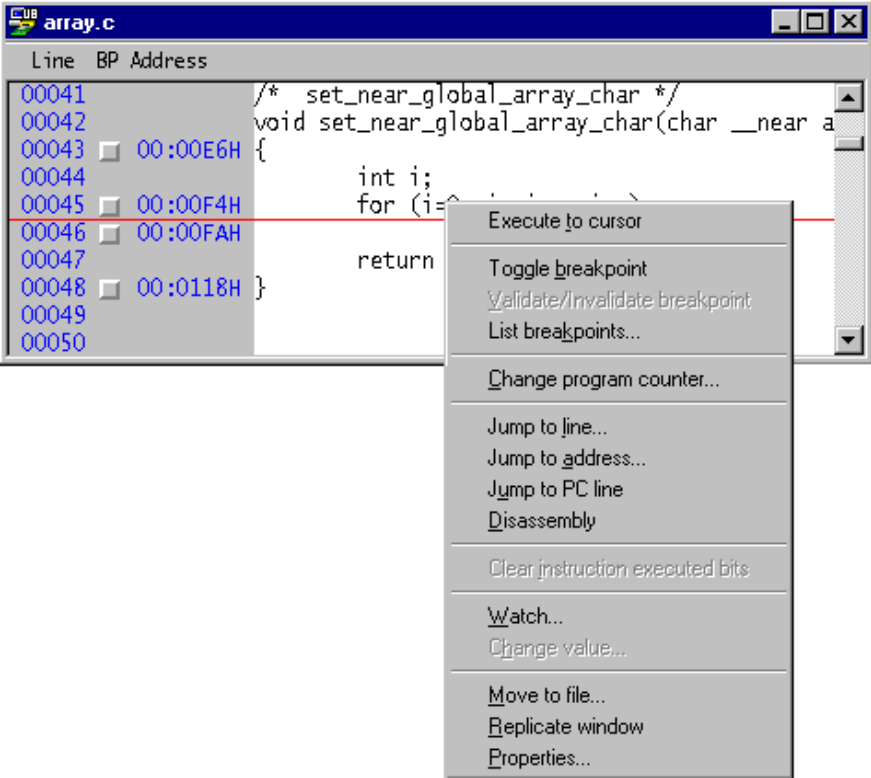
Note: Controlling basic emulator operation uses a command line interface for entering script commands one at a time with the Tools menu Execute script command menu command dialog box.



As the above Figure shows, a broad variety of window types are available for debugging. All functions are available using menu commands and window operations. The main menu and tool bar provide access to frequently used commands and ones that are not limited to specific windows.

Right-clicking on a window's display region opens a context menu specifically for that window type. The following Figure shows the one for a Source window.

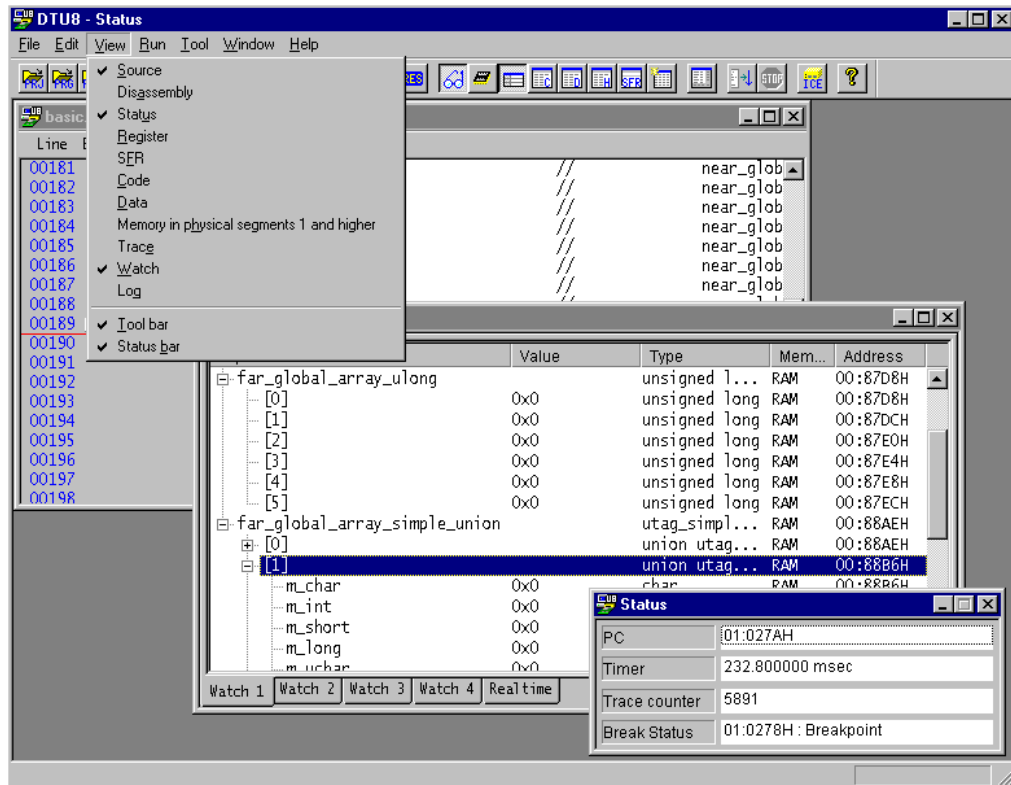
4. Basic Operation



## 4.2 Windows

### 4.2.1 Opening Windows

To open a window, use the corresponding View menu command.



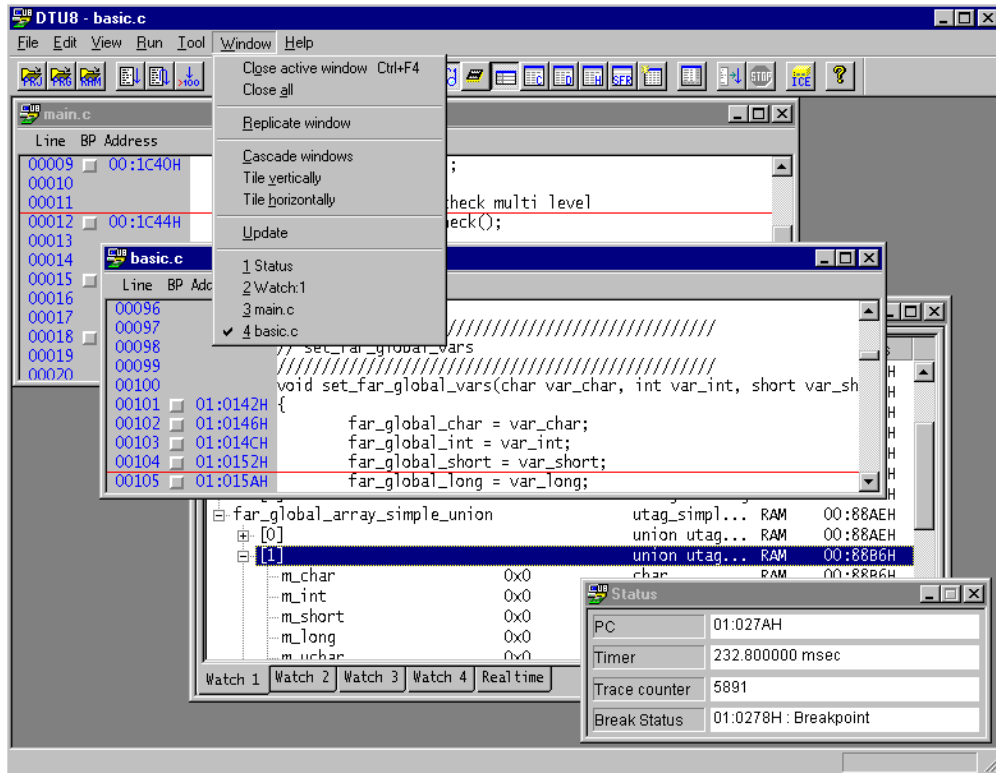
The check marks down the left side of the View menu indicate the windows that are currently open. Choosing such an entry erases the check mark and closes the corresponding window.

Source, Disassemble, SFR, Trace, Watch, and Log windows have their own cursors with distinctive shapes. Certain menu commands affect only the item under the cursor. The Run menu *Toggle breakpoint* menu command, for example, toggles a breakpoint at the address corresponding to the line under the cursor position in a Source/Disassembly window.

## 4. Basic Operation

### 4.2.2 Replicating Windows

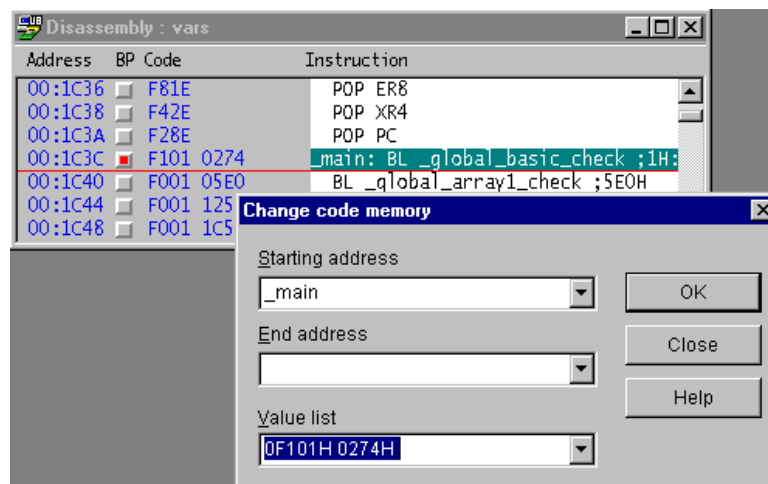
The developer can replicate and then simultaneously access the following types of windows: Source, Disassemble, SFR, Code memory, Data memory, and Physical segments 1 and higher.



The above Figure shows how the Window menu lists all windows currently open.

### 4.2.3 Double-Clicking in a Window

Double-clicking in the display portion of a window displays the dialog box appropriate for the object under the cursor position. The following Figure shows, for example, how double-clicking in the Disassembly window Code field opens the dialog box for modifying code memory.





Note: For further details, refer to Windows in the on-line help.

## 4.3 Miscellaneous

### 4.3.1 Tool Bar

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The standard layout has a tool bar immediately under the menu bar at the top of the application window. This is detachable. The user can drag it anywhere on the screen.



This tool bar contains buttons accessing major, frequently used functions. Moving the mouse over the tool bar provides a tool tip for the function under the cursor. For further details, refer to the on-line help.

### 4.3.2 Shortcut Keys

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Major, frequently used functions have shortcut keys assigned to them. These appear to the right of the command on the menu. A list of these shortcut keys appears in the on-line help.

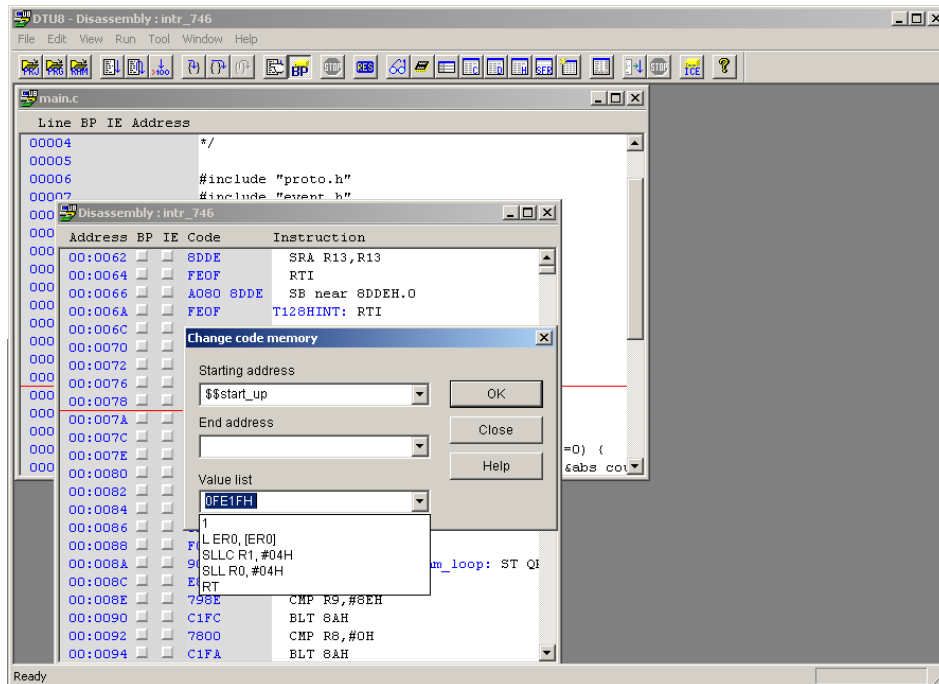
### 4.3.3 Miscellaneous

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

#### ■ *Input History*

Most dialog box input fields feature lists retaining previous entries that eliminate the need for repeated typing. Note, however, that the debugger discards these lists when it exits.

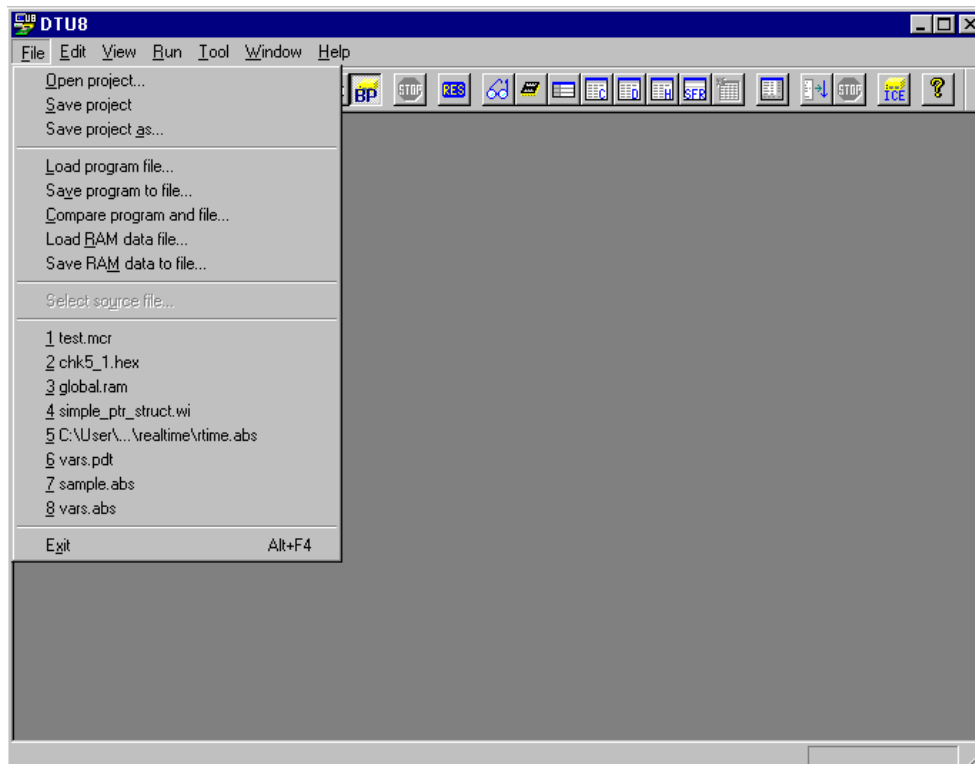
## 4. Basic Operation



### ■ File History

Specifying a file with the debugger adds its name to the most recently used (MRU) section of the File menu for faster access to the file if subsequently needed.

Note: This list skips the path for files in the current directory, the one containing the absolute object (.ABS) file.



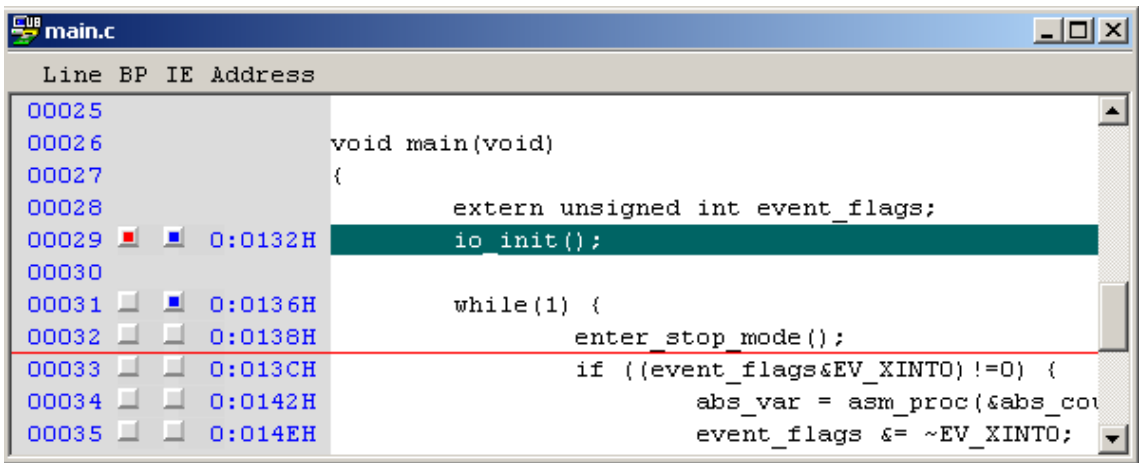
# 5. Source Level Debugging and Symbolic Debugging

## 5.1 Source Window

### 5.1.1 Display

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The debugger aids basic debugging operation by displaying the source code in a Source window. The *Source* menu command on the View menu opens the Source window.



- [Line] Line number in the source code file
- [BP] Check box for enabling/disabling breakpoint on line
- [IE] Instruction executed bits (simulation mode only)
- [Address] Code address

The Source window indicates the line with the current program counter with a green bar and the line under the cursor with an underline.

Note: Source windows do not have a horizontal scroll bar, but the left arrow, right arrow, Home, and End keys produce horizontal scrolling as necessary.

Note that it is perfectly valid to select text in a Source window, copy it to the Windows clipboard, and paste it into a dialog box input field.

## 5. Source Level Debugging and Symbolic Debugging

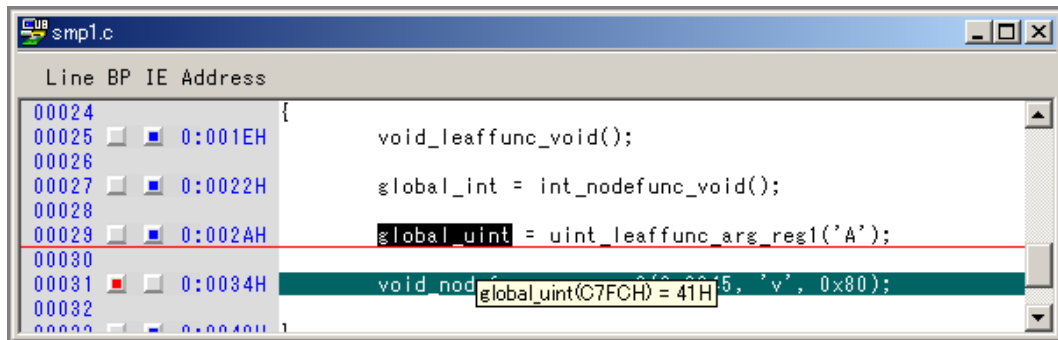
### 5.1.2 Displaying Values of Variables on Source Window

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

A Source window is provided with a function that displays the address and value of a variable selected on a Source window with a balloon message.

Highlight a desired variable name (that is, put it in a “selected” state) on a Source window and then move the mouse cursor over the variable name, and the address and value of the variable will be displayed as a balloon message.

Shown below is an example of a balloon message for the address (0C7FCH) and value (41H) of the variable “global\_uint”.



If the highlighted variable is a name of an array, structure, or union, the balloon message goes in the following format:

variable name (address) = {...}

This function displays the following variables or expressions in a balloon message:

- Signed or unsigned simple *char*, *int*, *long*, or *enum* type variable
- Array with an arbitrary subscript (such as array[10])
- Member of a structure or union that uses member operators (such as struct.member)
- Pointer expression that is a pointer to a specific address (such as struct->member, \*pointer, and \*(val+2))
- SFR<sup>3</sup>

To display a local variable or a global variable declared as static, the current PC must be within the scope where the selected variable is defined.

<sup>3</sup> There may be an SFR whose status changes when referenced. So check the hardware manual of the target device beforehand and then use the balloon message function.

### 5.1.3 Search Order for Source Code Files

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Displaying source code files in Source windows requires reading in an absolute object (.ABS) file containing source level debugging information. For further details on the absolute object file format, see Chapter 2 "Before Loading the Debugger."

The compiler and assembler save the source code file names in their debugging information output when they generate object files. The debugger uses this information from the absolute object (.ABS) file to open source code files in Source windows.

The debugger uses the following search strategy.

#### ■ File name with absolute path

The debugger first tries to open the file specified by that absolute path.

If that file does not exist, the debugger looks for a file with the same name in the directory specified by the Source code file field on the Tools menu *Environment settings* menu command Directory tab.

#### ■ File name with relative path

The debugger first tries to open the file using the current directory, the one containing the absolute object (.ABS) file, as the starting point for the relative reference.

If that file does not exist, the debugger tries using as the starting point the directory specified by the Source code file field on the Tools menu *Environment settings* menu command Directory tab.

Finally, the debugger looks for a file with the same name in the directory specified by the Source code file field on the Tools menu *Environment settings* menu command Directory tab.

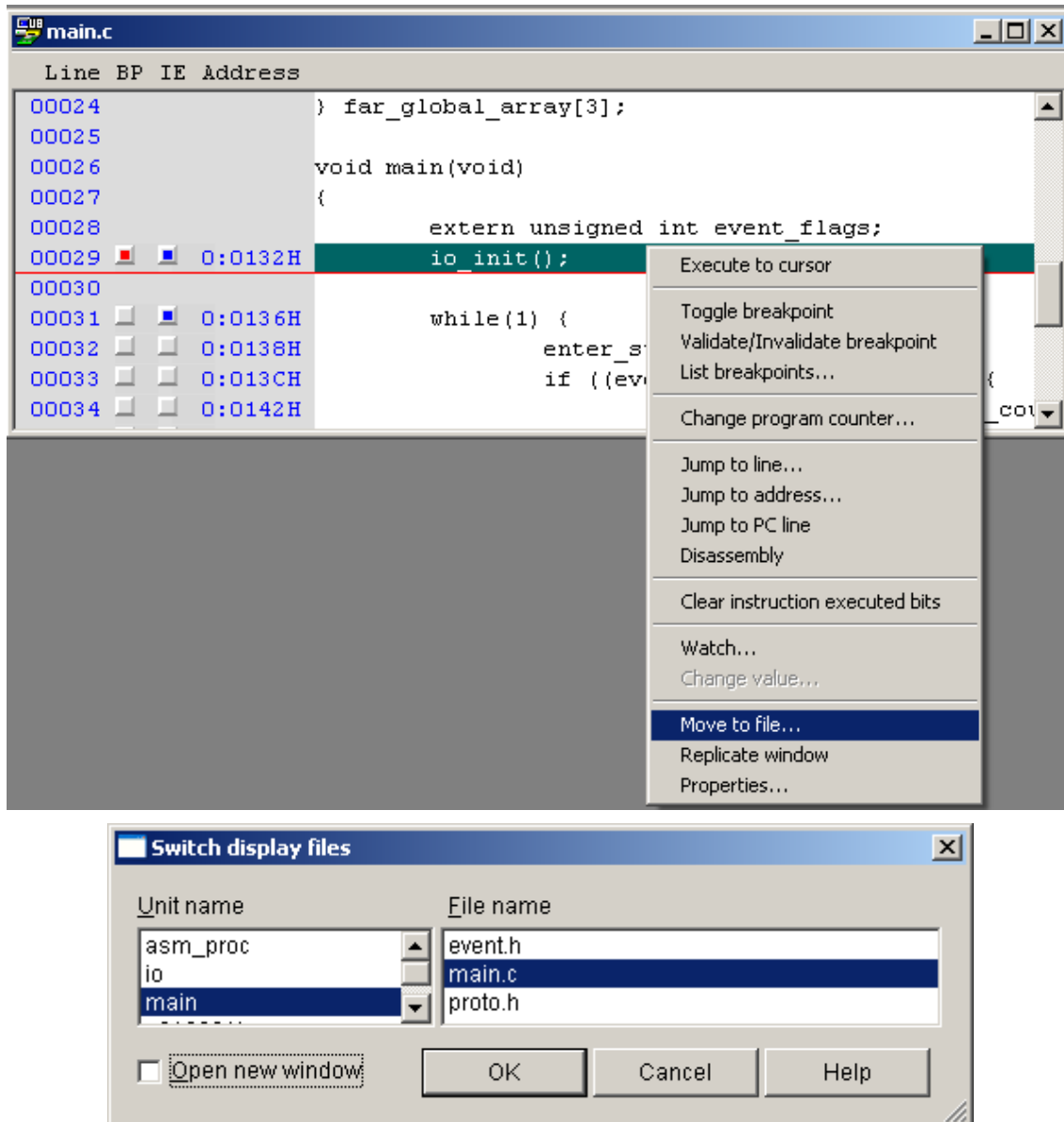
### 5.1.4 Displaying a Different File

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The first Source window opened after reading in an absolute object file displays either the source code file containing main() or the one containing the current program counter (PC).

If the program has multiple source code files, right-clicking inside the Source window and choosing Move to another file displays the list of files available.

## 5. Source Level Debugging and Symbolic Debugging



On the left side of the Switch display files dialog box, the units (units of compilation and assembly) that compose the program to be loaded are listed. Select one of them, then select a file you want to display from among the files listed to the right that compose the selected unit.<sup>4</sup>

Selecting the Open new window check box on the Switch display files dialog box displays the selected file on a new window.

The Switch display files dialog box can be displayed also by selecting File menu *Select source file* menu command. In this case, the source window can be opened by specifying the file you want to display even in a state in which no source window is being displayed.

In addition, a Source window context menu *Jump to address* menu command dialog box allows switching to the display of the source file including the specified address.

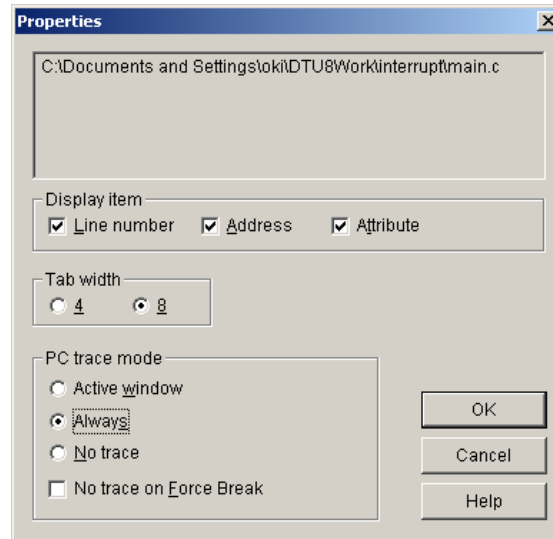
To close the Switch display files dialog box, click on the Cancel button on it or the Close button on the title bar.

<sup>4</sup> Although include files can be displayed on a source window, an include file including an execution code cannot be debugged.

### 5.1.5 Properties

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Selecting a Source window and choosing the *Properties...* command on the context menu displays the following dialog box.



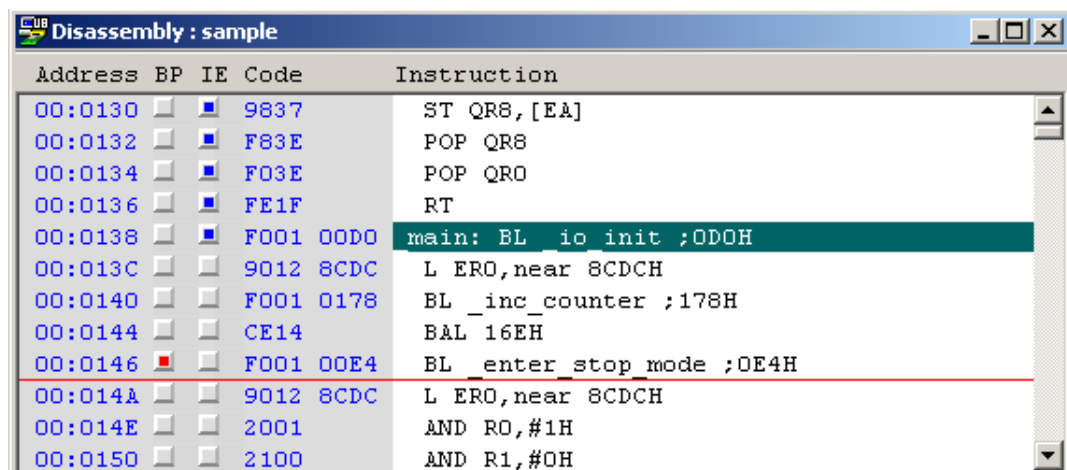
This dialog box gives the full file path, specifies the Source window display format, etc.

## 5.2 Disassembly Window

### 5.2.1 Display

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Disassembly windows show the results of disassembling emulator memory regions with the Code attribute for debugging at the machine instruction level.



[Address] Code address  
 [BP] Check box for enabling/disabling breakpoint on line  
 [IE] Instruction executed bits (not displayed unless supported)  
 [Code] Instruction code

## 5. Source Level Debugging and Symbolic Debugging

The Disassembly window indicates the line with the current program counter with a green bar and the line under the cursor with an underline.

Note: Disassembly windows do not have a horizontal scroll bar, but the left arrow, right arrow, Home, and End keys produce horizontal scrolling as necessary.

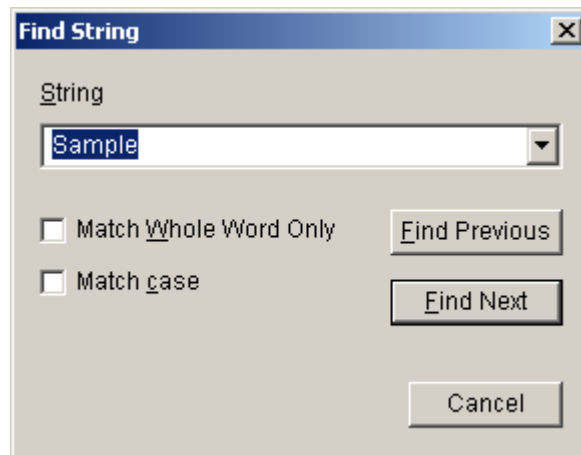
Note that it is perfectly valid to select text in a Disassembly window, copy it to the Windows clipboard, and paste it into a dialog box input field.

### 5.2.2 Searching for Character String in Disassembly Window

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

A character string that is displayed in the “Instruction” field on a Disassembly window can be searched for.

If the Edit menu *Find* menu command is selected while a Disassembly window is active, the following dialog box is displayed, enabling searching for a character string in a Disassembly window. Only the character strings displayed in the “Instruction” field in the Disassembly window can be searched for and “Address”, “BP”, “IE”, and “Code” fields are not searched.



#### ■ String entry field

Enter a character string you want to search for.

#### ■ Match Whole Word Only check box

Selecting this check box searches for the search string on a per-whole word basis. For example, if “rom is specified for the search string, neither “romwindow” nor “from” are searched for.

#### ■ Match Case check box

Selecting this check box searches for the search string while distinguishing between uppercase and lowercase letters. For example, if “abc” is specified for the search string, neither “ABC” nor “Abc” are searched for.

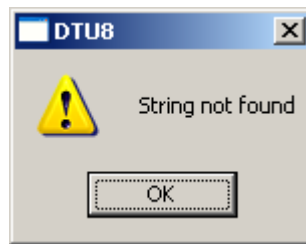


### ■ *Find Previous, Find Next buttons*

When either of these buttons is clicked, a search starts.

When the Find Next button is clicked, the target character string is searched for towards the end of the Disassembly window from the current cursor position. If the target string is not found through to the end of the window, the search continues from the beginning of the window to the entire window. If the target string is not found in the end, a “String not found” message is displayed.

When the Find Previous button is clicked, the target character string is searched for towards the top of the Disassembly window from the current cursor position.



Once a string search is performed with the Find Next or Find Previous button, the Find String dialog box closes. To perform a search in secession, press the F3 key (Find Next) or the Shift + F3 keys (Find Previous).

During a search, the following dialog box appears:



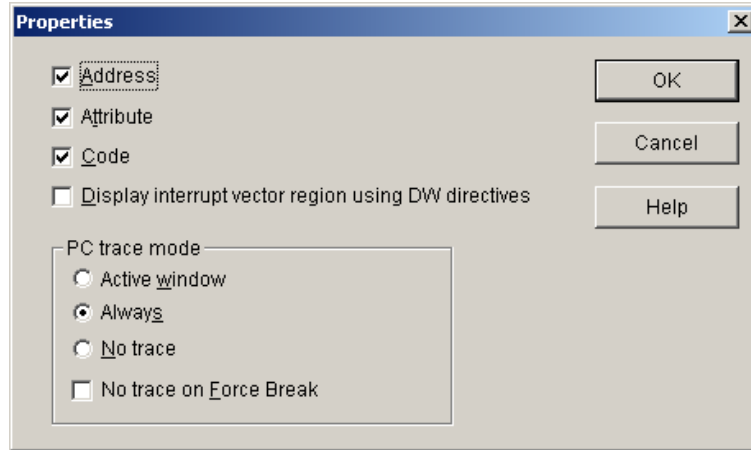
Clicking on the Stop button stops the search, in which case the cursor remains at the position before search.

## 5. Source Level Debugging and Symbolic Debugging

### 5.2.3 Properties

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Selecting a Disassembly window and choosing the *Properties...* command on the context menu displays the following dialog box.



This dialog box specifies the Disassembly window display format.

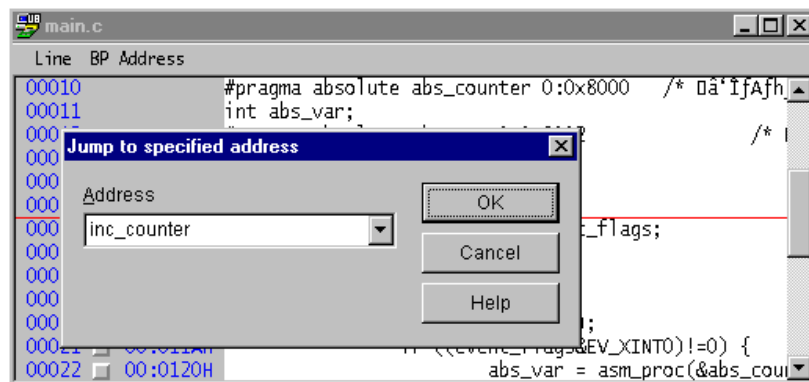
Selecting the Display interrupt vectors as DW check box displays addresses 0:0000h to 0:00FEh as words (DW) to help indicate that the contents represent interrupt address vectors.

## 5.3 Symbolic debugging

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

### ■ Substituting Symbols for Numerical Input

Input fields accepting numerical values also accept the symbols described in Appendix 11.1 "Symbols." An example is the dialog box for specifying the target address for moving the cursor for the *Jump to specified address* command on the Source window context menu.



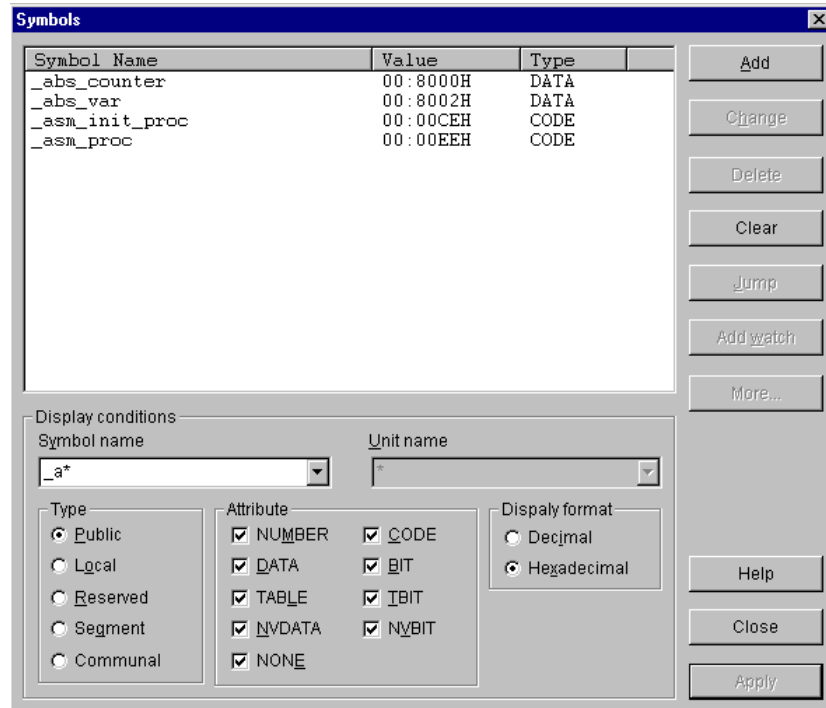
Specifying a C function name as a jump target switches the source code file containing that function and moves the cursor to the start of the function.

Note: Double-clicking on the source code in a Source window selects a word. Copying that word to the Windows clipboard with Ctrl+C permits pasting that text into the dialog box input box with Ctrl+V.

The above techniques apply to moving the cursor to a specific address in the following types of windows as well: Disassemble, Code memory, Data memory, and Physical segments 1 and higher.

### ■ Checking Symbols

The Tools menu *Symbol list...* menu command dialog box provides maintenance facilities for checking assembly language level symbols, changing their definitions, deleting them, and adding new ones.



Changing the display conditions in the lower half and pressing the Apply button limits the list to only symbols matching those conditions. Clicking on a column title (Symbol Name, Value, or Type) sorts the list entries by that column.

## 5.4 Links Between Windows

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The debugger provides mutual links between its many types of windows to make debugging that much easier.

### ■ Switching from Source to Disassemble Window

Selecting the Source window context menu *Jump to Disassembly window* menu command displays the address corresponding to the current cursor position in the Source window in a Disassembly window. Double-clicking the mouse on an address in the Source window produces the same results.

Nothing happens, however, if the source code line has no executable code.

### ■ Switching from Disassemble to Source Window

By the same token, selecting the Disassembly window context menu *Jump to Source window* menu command displays the source code line corresponding to the address at the current cursor position in the Disassembly window in a Source window. If there is no source code line containing that address, however, a Source window becomes active (comes to the front), but nothing else happens.

## 5. Source Level Debugging and Symbolic Debugging

### ■ **Adding Item to Watch Window**

Selecting a string in a Source window or Disassembly window, choosing the *Add watch item...* command on the context menu adds the selected string to the Watch window. For further details, see Chapter 11 "Watch Function".

### ■ **Switching from Trace Window (only a emulator system supporting tracing function)**

Choosing the *Link to source* command on the context menu switches from the Trace windows to the line corresponding to that code address: to the source code in the Source window.

## 5.5 PC Tracking

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Activating PC tracking on the Source and Disassembly window properties automatically updates the window cursor position each time that the program counter (PC) changes.

### ■ **Active window**

This setting produces PC tracking only when the window is active. Use it, for example, to track the PC in only the active one of multiple Source windows.

### ■ **Always**

This setting produces PC tracking regardless of the window's active state. The debugger uses it as the default for the first Source and Disassembly windows opened, producing simultaneous PC tracking in both windows.

### ■ **No trace**

This setting disables PC tracking. The debugger uses it as the default for replicated Source and Disassembly windows. The idea is to use only one Source window for tracking and leave the others as is for reference and modification purposes.

### ■ **No trace on Force Break**

This setting disables PC tracking upon a forced break regardless of the "PC trace mode" setting.

Selecting the No trace on Force Break check box disables PC tracking in Source and Disassembly windows only if a forced break occurs.

## 6. Emulation

---

### 6.1 Execution Modes

#### 6.1.1 Continuous Execution (Emulation Execution)

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE

Executing an emulator continuously with the debugger activated in Dr.U8 ICE, Dr.U16 ICE, Dr.ICE, uEASE or nanoEASE mode is called “Emulation Execution.”

This mode, which is the most basic execution mode for an emulation system, produces operation similar to that of the target microcontroller. To start continuous program execution, choose the Run menu Run program menu command. Execution continues until a break condition is satisfied or the user forces a break.

An indicator appears on the status bar while the program is running. Selecting the Display dialog box during emulation check box on the Other tab in the Tools menu Environment settings menu command dialog box, however, upgrades this to a dialog box giving more information.

#### 6.1.2 Continuous Execution (Simulation Execution)

Simulate

Executing the loaded program for each cycle by the debugger activated in simulation mode, is called “Simulation Execution.”

In this mode, the debugger analyzes the instruction codes of addresses of the PC.

To start continuous program execution, choose the [Run program] command from the [Run] menu. Execution continues until a break condition is satisfied or the user forces a break.

A message “Emulating...” appears on the status bar while the program is running. Selecting the [Display dialog box during emulation] check box on the [Other] tab in the Tools menu [Environment settings] menu command dialog box, however, upgrades this to a dialog box giving more information.

#### 6.1.3 Execute to cursor

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

This execution mode is equivalent to continuous execution except that execution stops at the cursor position in the currently active Source or Disassembly window. To access this mode, move the cursor to the target line in a Source or Disassembly window and choose the context menu *Execute to cursor position* menu command.

Note: Source windows provide this menu command only when the current line has executable code.

#### 6.1.4 Reset and run

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

This Run menu command resets the emulator and runs the program from its starting point. It represents a convenient way to start over from the beginning.

## 6. Emulation

### 6.1.5 Step in

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

This Run menu command produces single-step execution for a detailed analysis of program flow. The step size is a single C statement for C source code and a single machine instruction for all other situations.

If the current instruction is a C function call or subroutine call instruction, the cursor advances to the first executable line in the corresponding C function or assembly language subroutine.

An interrupt request immediately after the start of this execution mode stops execution at the entry point for the corresponding interrupt handler.

This execution mode is for examining program flow in detail.

This execution mode temporarily disables breakpoints because it automatically stops at each address, with or without a breakpoint setting.

Step in execution in a Source window switches to a Disassembly window if a function call, subroutine call, or interrupt branches into a region for which no debugging information is available. The same switching occurs when execution returns to such a region.

### 6.1.6 Step over

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

This Run menu command produces single-step execution that differs from step in execution in treating as a single step everything from the call to a C function or assembly language subroutine to the corresponding return.

If there is an interrupt during step over execution, the cursor advances to the first executable line after the return from the interrupt handler

This command skips over subroutines and interrupt processing, so is ideal for tracing mainline processing.

Note: Step over execution at the assembly language level skips over subroutines and interrupt handlers by temporarily switching to continuous execution through to the return address.

During step over execution, breakpoints are disabled temporarily; therefore, step over execution processing will continue without stop even if an address where a breakpoint has been set is passed.

An interrupt request during step over execution produces continuous execution through to the return from the interrupt handler. This skipping over subroutines and interrupt handlers allows the developer to concentrate on mainline processing.

Step over execution in a Source window similarly produces continuous execution when execution returns to a region for which no debugging information is available, preventing PC tracking.

### 6.1.7 Step out

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

To execute Step out, select *Run* menu *Step out* menu command.

This command executes the remaining instructions of the function located at the current program counter and then control returns to the caller of the function.<sup>5</sup>

This execution mode supports most breakpoints, but temporarily disables the conditions specified with the Run menu Address break menu command because it requires the Address break portion of the Break Condition Settings dialog box for its own use.

The Step out function is executed based on C debug information. Therefore, the function is disabled at an assembly source level that has no C debug information about Step out.

If a forced break is made or step execution is carried out on a disassembly window, the Step out function may be disabled. In this case, the Step out function can be enabled by making the C source window active and then carrying out step execution from the program counter line currently displayed to the next line.

This command can be selected from the Run menu only when the command can be executed. If it cannot be executed, the *Run* menu *Step out* menu command and the *Step out* button on the tool bar are grayed out and cannot be selected.

### 6.1.8 Step Execution at C Source Level

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

#### ■ Emulation Library or C Standard Library Calls

The compiler generates code calling emulation library functions when the C source code includes arithmetic expressions involving data types not supported by the nX-U8 core. For further details on types supported by the emulation libraries, refer to the CCU8 User's Manual.

Step in execution of such C source code in a Source window with PC tracking enabled therefore suddenly switches to a Disassembly window for the duration of such emulation library calls because emulation library functions do not provide debugging information. Likewise, a step in execution of the C source code that includes the C standard Library also switches to a Disassembly window suddenly.

Keeping track of whether the compiler uses emulation library calls or the C standard Library calls represents an unnecessary burden on the developer, so the Other tab in the Tools menu *Environment settings* menu command dialog box provides the C library is stepped over always in C source level debugging check box. Selecting this check box directs the debugger to automatically step over such calls during source level step in execution.

Selecting this check box change, however, the handling of interrupt requests, skipping the break at the entry point for the interrupt handler for normal step in execution. Execution continues on through to the beginning of the line following the one containing the emulation library or C standard Library call.

<sup>5</sup> An unexpected Step out operation may be caused depending on how the compiler performs optimization. For example, if a program is so written that f3() will be called at the end of f2(), a tail call optimization (optimization that uses a B instruction instead of a BL instruction) will be performed. In cases like this, if Step out is executed from f3(), which is a function in the lowest layer, control returns to the caller of f1(), not to f2().

## 6. Emulation

### ■ *Assembly Language Instructions in C Source Code*

The compiler treats the blocks of assembly language instructions between pairs of #asm and #endasm preprocessing directives,

#pragma asm and #pragma endasm pragmas, and \_\_asm and \_\_endasm keywords in the C source code as single lines of C source code.

## 6.2 Resets

To make a reset, select the [Reset] command on the [Run] menu.

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE

### ■ *Reset in Dr.U8 ICE, Dr.U16 ICE, Dr.ICE mode*

In Dr.U8 ICE, Dr.U16 ICE or Dr.ICE mode the Reset command resets the target microcontroller installed in the emulator.

uEASE  
nanoEASE

### ■ *Reset in uEASE, nanoEASE mode*

In uEASE or nanoEASE mode the Reset command resets the target microcontroller connected to uEASE or nanoEASE. When resetting succeeds, a “Target Reset is done.” message is displayed.

Simulate

### ■ *Reset in Simulation Mode*

In simulation mode the Reset command resets the simulation engine inside the debugger.

uEASE  
nanoEASE

### ■ *Reset by the RESET\_N pin*

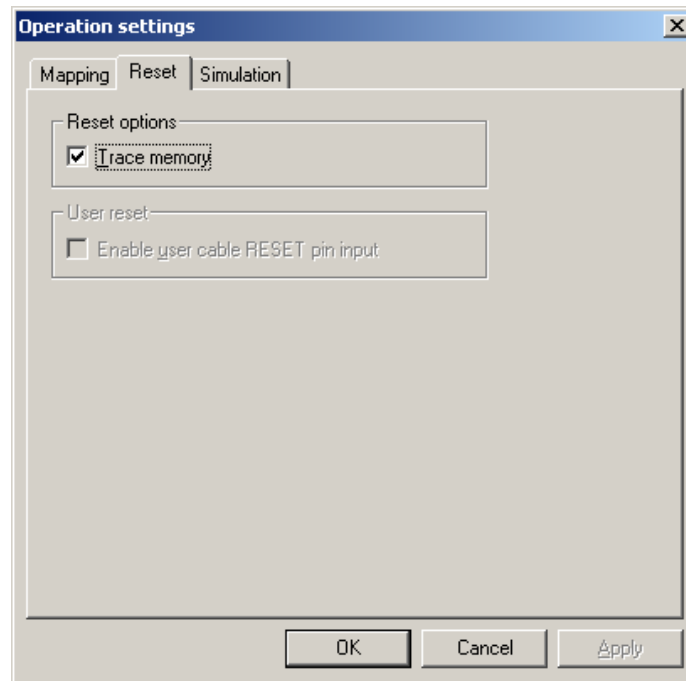
In uEASE or nanoEASE mode the [Device Reset] command on the [Run] menu resets the target microcontroller connected to uEASE or nanoEASE by the RESET\_N pin.



Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

### ■ **Reset Settings**

Some emulators add to the Tools menu *Operation settings...* menu command dialog box a Reset tab specifying emulator actions to take after a reset.



## 6.3 Functionality Available During Emulation

### 6.3.1 Windows

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The following window types are available during emulation. Other types can be made active or closed, but cannot be opened, scrolled, or resized.

#### ■ **Source Windows**

These display source code files. They also permit switching to another file.

#### ■ **Status Window**

This window, if enabled, displays the time elapsed since the start of emulation in the Timer field.

#### ■ **Log Window**

Emulation does not produce any log output. There are no limits on editing, saving, clearing, and other Log window operations.

#### ■ **Watch Window (only for emulators supporting real-time watch operation)**

This window tracks watch items added to the Realtime tab before starting emulation.

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

## 6. Emulation

Dr.U8 ICE  
Dr.U16 ICE

### ■ **Realtime RAM Monitor Window**

This window displays the contents of RAM changed by program execution on real time.

## 6.3.2 Menus

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The following menu commands are available during emulation.

### ■ **File menu Specify source code file**

This switches the Source window to another file.

### ■ **File menu Exit**

See Section 6.5 "Exiting and Reconnecting during Emulation."

### ■ **Edit menu**

This displays only the menu commands applicable to the active window.

### ■ **View menu**

This displays only the menu commands for available window types.

### ■ **Run menu Force break menu command**

This forces a break in emulation.

### ■ **Run menu Breakpoint list menu command**

This displays a list of the currently defined breakpoints, but does not allow modification.

### ■ **Run menu Break condition settings menu command**

This displays the currently specified break condition settings, but does not allow modification.

### ■ **Tools menu Symbol list menu command**

This displays a list of the currently defined symbols, but does not allow modification.

### ■ **Tools menu Operation settings menu command**

This displays a list of the current settings, but does not allow modification.

### ■ **Tools menu Environment settings menu command**

This displays a list of the current settings and allows modification of all settings except those for communications.

### ■ **Help menu**

All commands are available.

Dr.U8 ICE  
Dr.U16 ICE

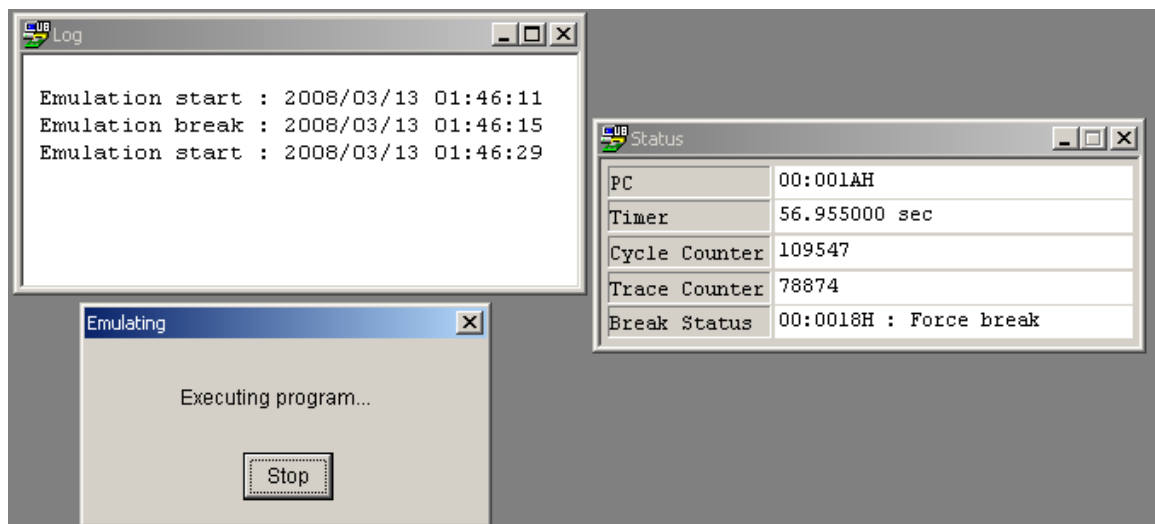
#### ■ Run menu Realtime LCD monitor menu command

This launches a LCD image check tool and displays the contents of LCD changed by program execution on real time.

## 6.4 Measuring Execution Times

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The DTU8 debugger supports the execution time measurement functions.



This figure shows an example of screen in Dr.U8 ICE mode (the content of the Status window is depending on the operation mode).

In Dr.U8 ICE, Dr.U16 ICE, Dr.ICE, uEASE or nanoEASE mode, the Status window shows an execution time from starting the emulation to the break in the "Timer" field. In simulation mode, the Status window shows an execution time from starting the simulation to the break in the "Timer" mode. Note that accuracy of those execution times or measureable range depends on the hardware of emulator. See emulator's manual for more detail about it.

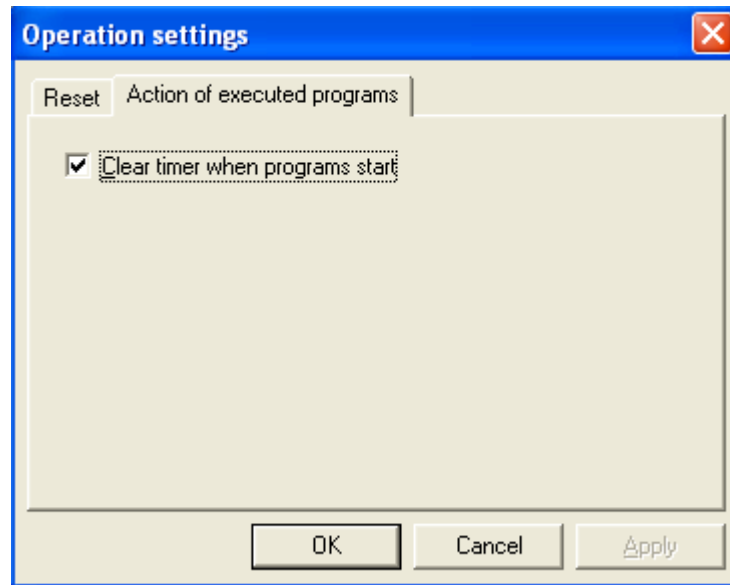
If "Log program starts and stops" check box is selected on the "Other" tab in a dialog box (select "Tools" menu → "Environment settings" menu, a start time and an end time for every emulation are time-stamped.

## 6. Emulation

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE

### ■ *Clearing of Execution Time*

It can be set up whether measurement time is cleared at the time of an emulation start with the [Action of executed programs] tab of a dialog opened by [Operation settings] of a [Tool] menu. A check of the check box [Clear timer when programs start] will clear measurement time at the time of an emulation start. However, when a [Reset and Run] command or a [Reset] command is executed, measurement time is certainly cleared.



## 6.5 Exiting and Reconnecting during Emulation



The user can exit the debugger during emulation. As long as the same development host is used, reloading the debugger automatically reconnects to the emulator--allowing the emulator to run aging tests and other user application circuit evaluations for hours or even days unattended, for example, while the user uses the development host for other applications or simply shuts it down.

Such exiting and reconnecting during emulation must, however, observe the following precautions.

### ■ **Reconnect with same debugger**

Attempting to reconnect to the emulator from another development host fails because, when exiting during emulation, the debugger saves the information necessary for reconnecting in registries as well as overwrite-saving window settings in the project file and initialize file automatically.

### ■ **Do not update program file**

The debugger reloads the program file when it reloads. Updating the program file and then reconnecting thus risks discrepancies between the program code versions in the emulator and the debugger.

### ■ **Restricted function at reconnect**

When emulation execution continues at the time of re-connection, in addition to the restriction in the usual emulation, the following functions are restricted until it takes a break.

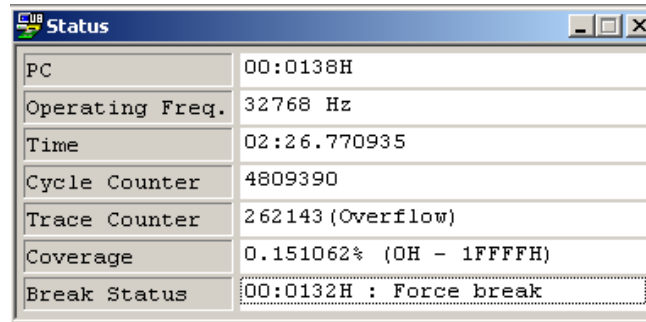
- A source window cannot be opened.
- A realtime watch function cannot be used.
- A setup of break condition cannot be checked.
- A realtime LCD monitor function cannot be used.

## 6. Emulation

### 6.6 Cycle Counter

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

The cycle counter is a 64-bit counter for counting instruction cycles during program execution. The cycle counter value is displayed in the [Cycle Counter] field in the Status window of the debugger.<sup>6</sup>



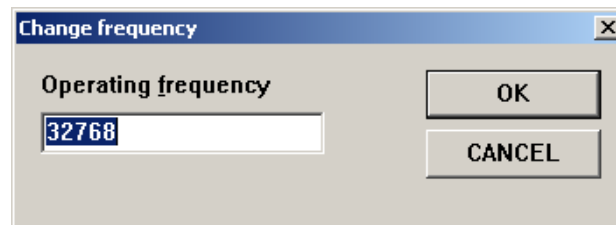
|                 |                         |
|-----------------|-------------------------|
| PC              | 00:0138H                |
| Operating Freq. | 32768 Hz                |
| Time            | 02:26.770935            |
| Cycle Counter   | 4809390                 |
| Trace Counter   | 262143 (Overflow)       |
| Coverage        | 0.151062% (0H - 1FFFFH) |
| Break Status    | 00:0132H : Force break  |

Simulate

The [Time] value displayed in the Status window during simulation mode is calculated based on this cycle counter value and the operating frequency displayed in the [Operating Freq] field.

The operating frequency in the [Operating Frequency] field can be changed either by the [Set Frequency] menu command of the context menu of the Status window or on the [Simulation] tab of the dialog box opened by the [Operation settings] menu command of the Tool menu.

Selecting [Set Frequency] of the context menu of the Status window displays the following dialog box:



Change frequency

Operating frequency

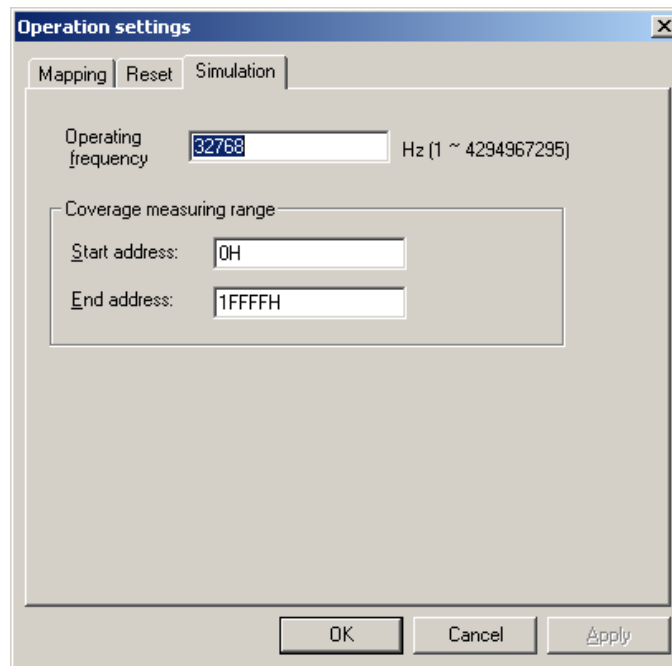
32768

OK

CANCEL

<sup>6</sup> The cycle counter can be used only in simulation, Dr.U8 ICE, Dr.U16 ICE and Dr.ICE mode (except when the Dr.610XXX in-circuit emulator of an old type is connected). The bit length of the cycle counter is 64 bits in simulation mode and 32 bits in Dr.U8 ICE, Dr.U16 ICE and Dr.ICE mode. If the cycle counter is not supported, the [Cycle Counter] field is not displayed.

Selecting the [Simulation] tab of the dialog box opened by the [Operation settings] menu command of the Tool menu displays the following dialog box:



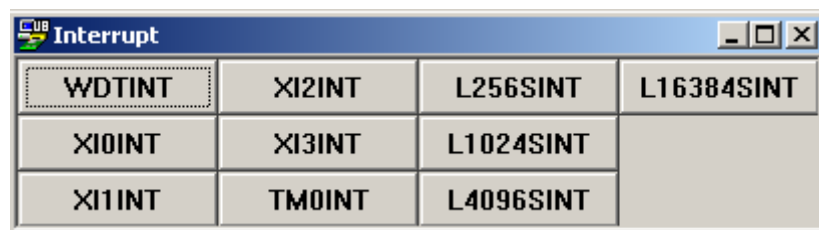
## 6.7 Interrupt Simulation

### Simulate

Interrupt simulation is supported in simulation mode. An interrupt can be generated at an arbitrary point using the Interrupt window.

### 6.7.1 Interrupt Window

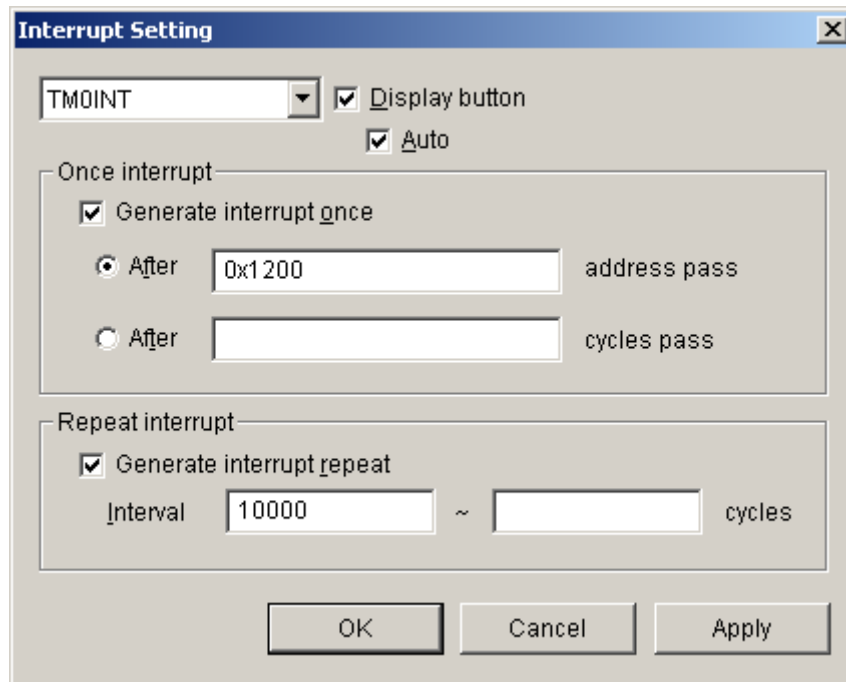
Selecting the View menu *Interrupt* menu command displays the Interrupt window.



The interrupt sources that the target MCU has are indicated on the Interrupt window as buttons (hereafter called source buttons). By clicking on one of these buttons, the corresponding interrupt request can be generated.

## 6. Emulation

Various settings can be made for these interrupt source buttons. Configure settings for each interrupt source button on the Interrupt Setting dialog box, which is displayed by selecting the *Interrupt settings* menu command on the context menu displayed by right-clicking on the interrupt source button you want to configure.



The Interrupt Setting dialog box always opens with the interrupt of the interrupt source button that has been right-clicked selected, allowing the user to change interrupt settings.

Clicking on the OK button reflects the interrupt settings being displayed into the Interrupt window and then closes it.

Clicking on the Cancel button closes the Interrupt window without reflecting the interrupt settings being displayed into the Interrupt window.

Clicking on the Apply button reflects the interrupt settings being displayed into the Interrupt window; however, the dialog box is not closed. The user can configure settings for another interrupt source continuously by selecting the type of interrupt source from the pull-down list at the top-left corner of the Interrupt Setting dialog box.

Each item for configuration on the dialog box is explained below.

### ■ **Display button**

If this check box is checked, the interrupt source buttons are displayed on the Interrupt window. In cases where the Auto check box is checked, no button can be put into a nondisplayed state.

### ■ **Auto**

If this check box is checked, the “Once interrupt” and “Repeat interrupt” items are enabled and an interrupt can be generated automatically according to the conditions that are set. When generating an interrupt automatically, either “Once interrupt” or “Repeat interrupt” or both must be set.



In addition, the color of the interrupt source button is changed to enable you to identify that automatic generation has been set for the interrupt source. The button color can be changed on the Format tab in the Environment settings dialog box.

### ■ **Once interrupt**

When the Generate interrupt once check box is checked, the items under the check box are enabled.

Specify whether an interrupt request is generated when the specified address is passed through or whether an interrupt request is generated when an instruction is executed for the specified cycle count using the current program position as 0.

When inputting an address, the address input range must be within the program area. If no address is input or the specified address is outside the input range, an error dialog box is displayed, prompting the user to re-input an address.

The number of execution cycles to be input must be within the range from 1 to 0xFFFFFFFFFFFFFFFF (decimal input is allowed). If no value is input or the specified number of cycles is outside the input range, an error dialog box is displayed, prompting the user to re-input an address.

When the specified interrupt occurs once, the color of the button is changed to the color that has been set as the one when automatic interrupt generation is disabled (Auto check box deselected), so that the user can know the termination of automatic generation.

### ■ **Repeat interrupt**

When the Repeat interrupt check box is checked, the items under the check box are enabled, enabling input of interval values for interrupt. The interval input range is from 1 to 0xFFFFFFFFFFFFFFFF (decimal input is allowed). If no value is input or the specified number of cycles is outside the input range, an error dialog box is displayed, prompting the user to re-input an interval value. If the number of cycles is input only in the left entry field, an interrupt request is generated always with that number of cycles specified. If a range is specified (using the right field also), DTU8 randomly determines an interval value at each interrupt.

The interrupt simulation starts incrementing the interval counter value using the current program position as 0. When the count value reaches the specified cycle count, an interrupt occurs. When an interrupt occurs, the simulator continues incrementing using the counter value as 0 and generates interrupts repeatedly.

## 7. Break Functions

### 7.1 Breakpoints

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The user can set a breakpoint at any code address. Execution then stops just before that address.

Emulation stops immediately before the emulator attempts to execute the machine instruction at an address with a valid breakpoint.

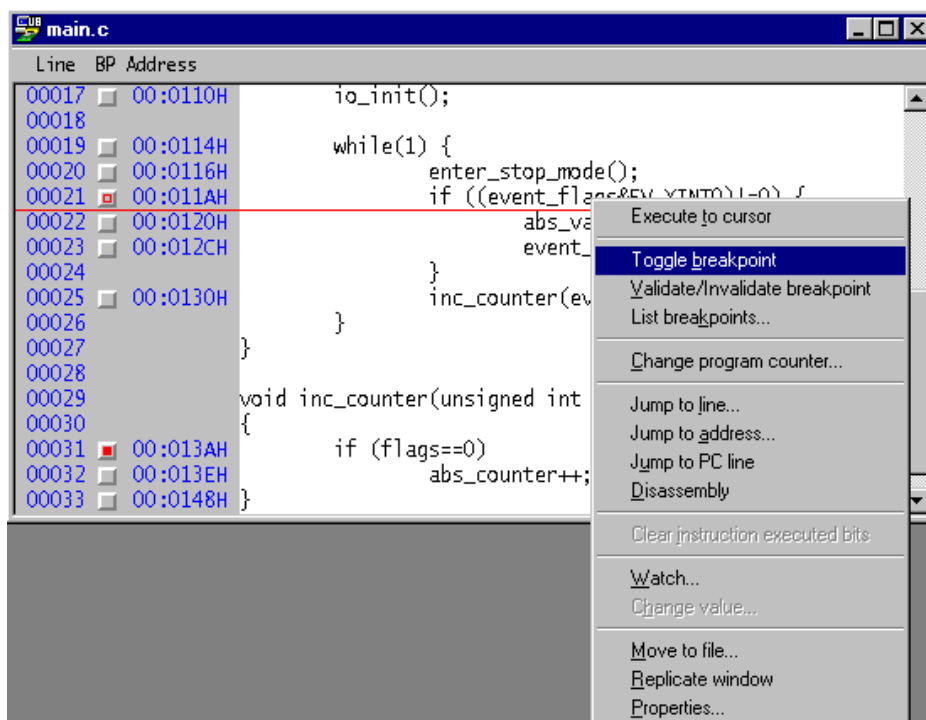
#### 7.1.1 Setting Breakpoints

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Source and Disassembly windows provide multiple ways to define, undefined, disable, and enable breakpoint settings.

##### ■ Window BP Buttons

Selecting the Attribute check box in the Properties dialog box accessible from the context menus for Source and Disassembly windows adds a column of BP buttons to the window.



These buttons indicate both the lines for which breakpoint settings are possible and the current settings for those lines. A BP button in a Source window, for example, indicates that the source code line has executable code and is thus a candidate for a breakpoint.

A solid red mark inside a BP button indicates a breakpoint at the corresponding address; a red square, a disabled one. A plain BP button with neither mark has no breakpoint defined.

Note: A red mark next to a C source code line indicates a breakpoint somewhere in the address range covered by that line.

Clicking on a plain BP button sets a breakpoint. Clicking on one with either red mark clears the setting.

### ■ **Window Context Menus**

Right-clicking on a line in a Source or Disassembly window and choosing *Toggle breakpoint* sets a breakpoint at the corresponding address if there is currently no breakpoint there and removes the breakpoint if there is one defined.

If the line has a breakpoint, choosing the context menu *Enable/disable breakpoint* menu command temporarily disables the breakpoint. Applying the same command to such a disabled breakpoint enables it once again.

### ■ **Run menu *Toggle breakpoint* menu command**

Moving the cursor to a line in a Source or Disassembly window and choosing the Run menu *Toggle breakpoint* menu command sets a breakpoint at the corresponding address if there is currently no breakpoint there and removes the breakpoint if there is one defined.

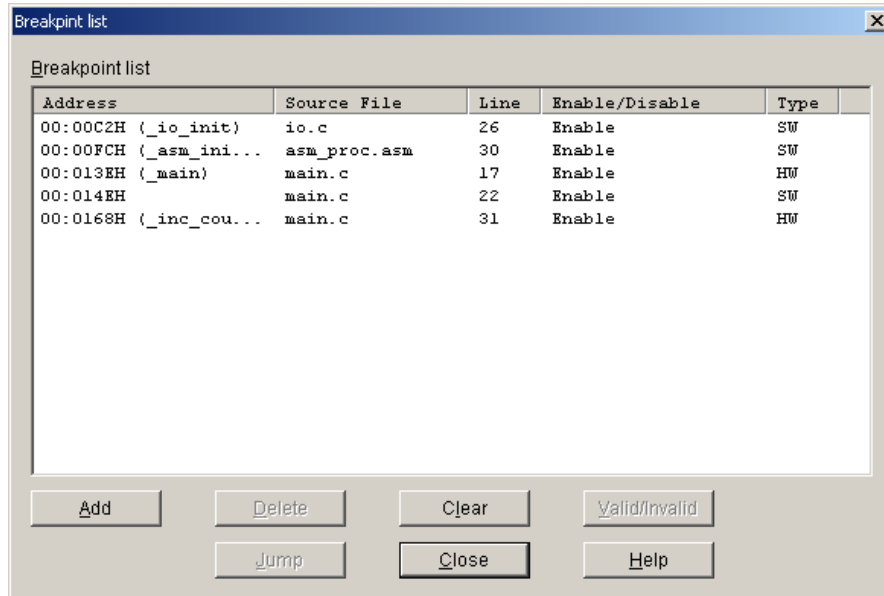
Note: This menu command has the shortcut key F9.

## 7. Break Functions

### 7.1.2 Listing Breakpoints

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Choosing the *Breakpoint list* menu command on the Run menu or Source/Disassembly window context menu displays a list of currently defined breakpoints.



The Add button displays the dialog box for directly specifying a breakpoint.

The Delete button deletes the currently selected breakpoint from the list.

The Clear button deletes all breakpoints from the list.

The Jump button moves the cursor to the line corresponding to the code address for the currently selected breakpoint: to the source code in the Source window, if available, or to the object code in the Disassembly window otherwise.

### 7.1.3 Enabling/Disabling All Breakpoints

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The debugger provides the Run menu *Enable breakpoint* menu command for toggling breakpoints on and off. The first use places a check mark before the command, indicating that breakpoints are valid. The second use erases the check mark, and the debugger ignores all breakpoint settings.

|                           |          |
|---------------------------|----------|
| Reset                     | Ctrl+F5  |
| Change program counter... |          |
| Execute program           | F5       |
| Execute to cursor         | F7       |
| Reset and run             |          |
| Force break               | Shift+F5 |
| Step over                 | F10      |
| Step in                   | F11      |
| Toggle breakpoint         | F9       |
| List breakpoints...       |          |
| Clear all breakpoints     |          |
| ✓ Validate breakpoints    |          |
| Break conditions...       |          |
| Add sync out point...     |          |
| List sync out points...   |          |
| Clear all sync out points |          |
| Reset sync out signal     |          |

The BP button on the tool bar provides rapid access to this toggle command. The button changes to a depressed icon when breakpoints are enabled.

### 7.1.4 Breakpoints for C Source Code Lines

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

#### ■ **Breakpoint at Start of Line**

Adding a breakpoint anywhere on a C source code line in a Source window sets a breakpoint at the starting address for that line, which in general represents multiple machine instructions.

#### ■ **BP field Icons**

A solid red mark inside a BP button next to a C source code line indicates that there is an enabled breakpoint somewhere in the address range covered by that line; a red square, that all such breakpoints are disabled.

#### ■ **Undefining, Disabling, and Enabling Breakpoints**

Undefining a breakpoint for a C source code line in a Source window undefines all breakpoints within the address range covered by that line. Disable and enable operations work the same way.

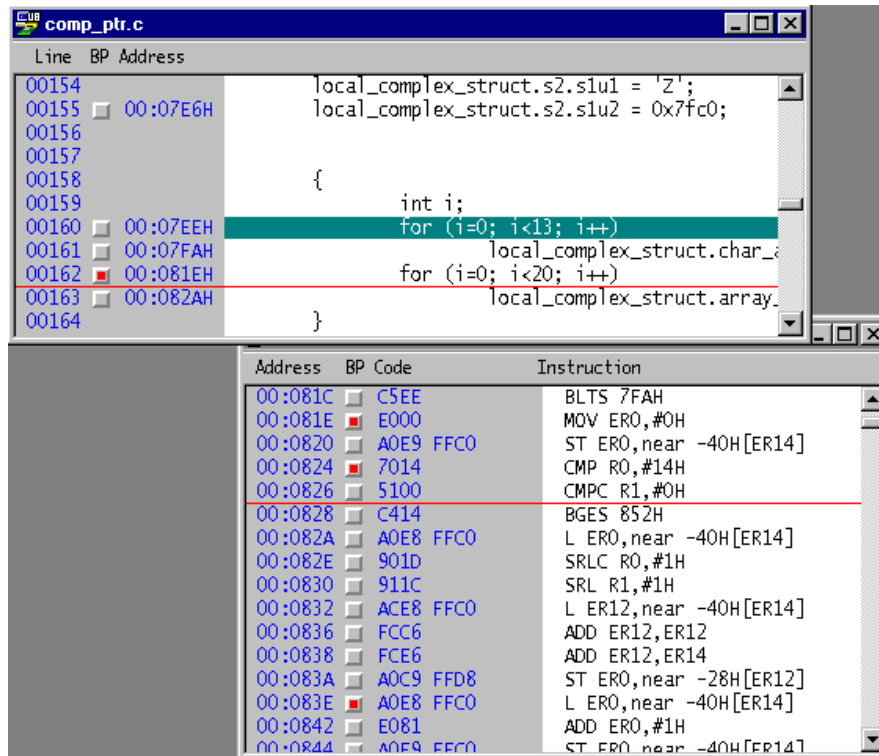
Note that using a Disassembly window or the breakpoint list to define a breakpoint anywhere other than the starting point for the line's address range produces the same BP button indicator next to the corresponding source code line in a Source window, but using the above undefine, disable, and enable operations requires caution.

#### ■ **C Source Code Splitting Multiple Regions**

A single line of C source code can produce machine language code occupying disjoint regions. Line 162 in the Source window in the following Figure, for example, produces three such code fragments. Setting a breakpoint in the Source window defines breakpoints for all three.

## 7. Break Functions

Note: In the case of uEASE or nanoEASE, if a line of C source code produces machine language code occupying disjoint regions, a breakpoint is set only at one address in the heading region.



### 7.1.5 Notes on Setting Breakpoints

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The debugger implements breakpoints by replacing the executable code at the corresponding addresses with software breaks. Data access to those addresses during execution can therefore yield unforeseen results.

#### ■ Vector Region Breakpoints

Setting a breakpoint in the vector table region (0:0h to 0:0FFh) in a Disassembly window does not yield proper results when a reset or interrupt request triggers access to that address vector.

#### ■ Table Region Breakpoints

Never set a breakpoint in a table data region containing data referenced during instruction execution because doing so interferes with the operation of data transfer instructions accessing that address. Data reads from that address do not return the proper value. Data writes to that address are doubly dangerous because they overwrite the software break, disabling the breakpoint, and because the debugger returns the contents to the original state if the emulation hits another breakpoint.

## 7.2 Forced Break

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

If there are no breakpoints or other break conditions specified, the program executes without a break. The debugger therefore provides the Run menu *Force break* menu command to force a break in program execution.

Selecting the Display dialog box during emulation check box on the Other tab in the Tools menu *Environment settings* menu command dialog box displays a dialog box with a Break button producing the same result.

## 7.3 Break Condition Settings

Choosing the Run menu *Break conditions* menu command displays the following dialog box:

**Break conditions**

Break ON/OFF

☐ Power down break ☒ PC match break ☐ Trace count brak

☒ ROM N/A area access ☒ RAM match break1 ☐ External break1

☒ RAM N/A area access ☒ RAM match break2 ☐ External break2

Trace count value  
  
 (3 - 262144)

External break1  
☐ L edge  
☒ H edge

External break2  
☐ L edge  
☒ H edge

PC match break

Break address  Pass count value  Count value

Set >> (1 - 65536)

RAM match break1

RAM address

RAM address mask

Access method

☒ Specify RAM data

RAM data  Data width

RAM data mask  Compare

☒ RAM match pass count

Match count (1 - 65536)

Count value

Set >>

OR

RAM match break2

RAM address

RAM address mask

Access method

☐ Specify RAM data

RAM data  Data width

RAM data mask  Compare

☐ RAM match pass count

Match count (1 - 65536)

Count value

Set >>

OK Cancel Help

The contents depend on the emulator connected. The debugger grays out break conditions that are not supported.

## 7. Break Functions

### ■ **Trace Count Break**

A break occurs when the trace counter value reaches the set value. If a trace count break is set, the counter will be decremented by as much amount as traced during program execution, and a break occurs when the counter reaches 0.

### ■ **Power Down Break**

A break occurs if the debugger enters power down mode (such as HALT and STOP modes) during program execution.

### ■ **Address Break**

Unlike the breakpoints set in the Source or Disassembly window, execution count can be specified for this break.

A break occurs if the instruction at the address specified in the [Break address] field is executed as many times as specified in [Pass count value]. When setting the count value for the first time, input the number of times to [Pass count value], then press the [ >> ] button.

### ■ **External Break**

A break occurs when the signal specified in the [External break conditions] field is input to the external break signal input probe connected to the emulator.

### ■ **ROM N/A area access**

A break occurs if access is made to a reserved area or a program unexecutable area. Always turn on (select) this check box for normal operation.

### ■ **RAM N/A area access**

A break occurs if access is made to a reserved area or an area that cannot be accessed by a data transfer instruction. Always turn on (select) this check box for normal operation.

### ■ **RAM Match Break**

A break occurs depending on the RAM address to be accessed during program execution.

Specify in the [RAM address] field the RAM address to be checked. If a value is specified in the [RAM address mask] field, the result produced by carrying out the bitwise AND between the RAM address that has been accessed during program execution and the value specified in the [RAM address mask] field will be the target RAM address.

Specify in the [Access method] field the access method to be used during program execution. Select from among [Read], [Write], and [Read/Write].

If the [Specify RAM data] check box is turned on (selected), a break occurs depending on the result of checking the specified RAM address. Specify in the [RAM data] field the data value you want to check; specify in the [Access unit] field the data size. In addition, if a mask value is specified in the [RAM data mask] field, the value will be applied to the data value to be checked. If a mask value is specified in the [RAM data mask] field, the result produced by carrying out the bitwise AND between the contents of the RAM address that matched the specified condition and the value specified in the [RAM data mask] field will be compared with the [RAM data] value.

Specify [Equal] or [Not Equal] for [Compare].



Up to two RAM match conditions can be set, and the two RAM match condition can be combined with an AND or OR condition. In cases where the OR condition is specified, a break occurs if either of the RAM match break conditions is satisfied. In cases where the AND condition is specified, a break occurs if both of the RAM match break conditions are satisfied.

### ■ ***Specifying RAM match count***

Depending on how the RAM match conditions are combined, pass count specification is available. Selecting the RAM match count check box enables the RAM match condition fulfillment count to be specified. A value being displayed in the [Count value] field indicates the count value set in the emulator. This value may have been updated if a break occurs under any other condition than those specified here. The Count value can be re-set using the [>>] button.

In the case of Dr.U8 ICE or Dr.U16 ICE mode, it is possible to specify a path count about each of two RAM matches.

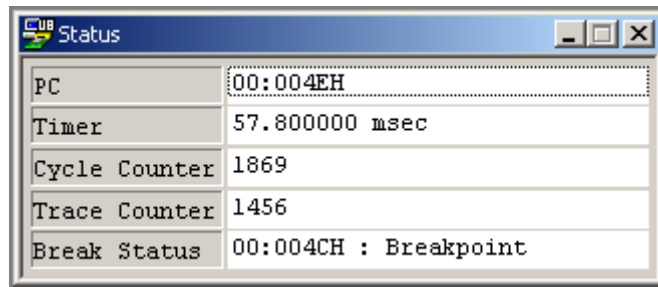
## **8. Referencing/Modifying Registers and Memory**

---

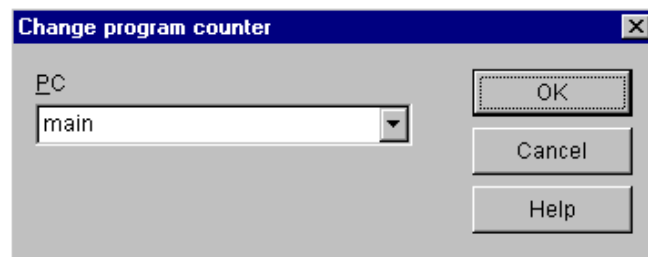
## 8. Referencing/Modifying Registers and Memory

### 8.1 Program Counter (PC)

The contents of the program counter appear in the Status window.



To change them, use the *Change program counter...* command on the Status window context menu, or double-click the current value in the Status window.



The program counter (PC) is eligible as a candidate for addition as a watch item to the Watch window.

The Change Program Counter dialog box is also accessible with the Run menu *Change program counter* menu command.

Note: The debugger refuses addresses mapped to a region marked N/A or TEST.

Source and Disassembly windows highlight the program counter line in green.

Choosing the *Change program counter* command on the Source or Disassembly window context menu displays the Change Program Counter dialog box with the address corresponding to the current cursor position as the default input value.

## 8.2 Registers

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The contents of the other registers appear in the Register window.

|       |    |      |      |       |    |     |      |
|-------|----|------|------|-------|----|-----|------|
| PSW   | 40 | LR   | 0138 | LCSR  | 0  | SP  | 8DDE |
| EPSW1 | 00 | ELR1 | 0000 | ECSR1 | 0  | DSR | 0    |
| EPSW2 | 00 | ELR2 | 0000 | ECSR2 | 0  | EA  | 8DE8 |
| R3    | 03 | R2   | 02   | R1    | 00 | R0  | 00   |
| R7    | 00 | R6   | 00   | R5    | 00 | R4  | 00   |
| R11   | 00 | R10  | 00   | R9    | 8E | R8  | 00   |
| R15   | 00 | R14  | 00   | R13   | 00 | R12 | 00   |

To change them, use the corresponding Change xxx register command on the Register window context menu or double-click the current value in the Register window.

All registers are eligible as candidates for addition as a watch items to the Watch window.

### ■ Customizing Register window

Only necessary registers can be displayed in the Register window.

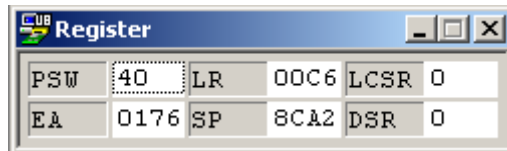
Using the Register window context menu, control registers, exception registers, and general-purpose registers can be displayed.



### ■ Displaying control registers

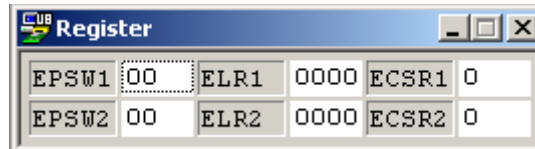
Checking *Display Control-registers* on the context menu displays the following register group in the Register window:

## 8. Referencing/Modifying Registers and Memory



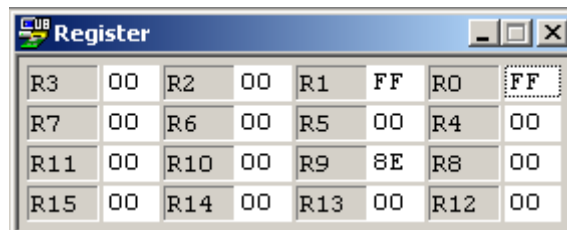
### ■ Displaying exception registers

Checking *Display Exceptional Registers* on the context menu displays the following register group in the Register window:



### ■ Displaying general-purpose registers

Checking *Display General Registers* on the context menu displays the following register group in the Register window:



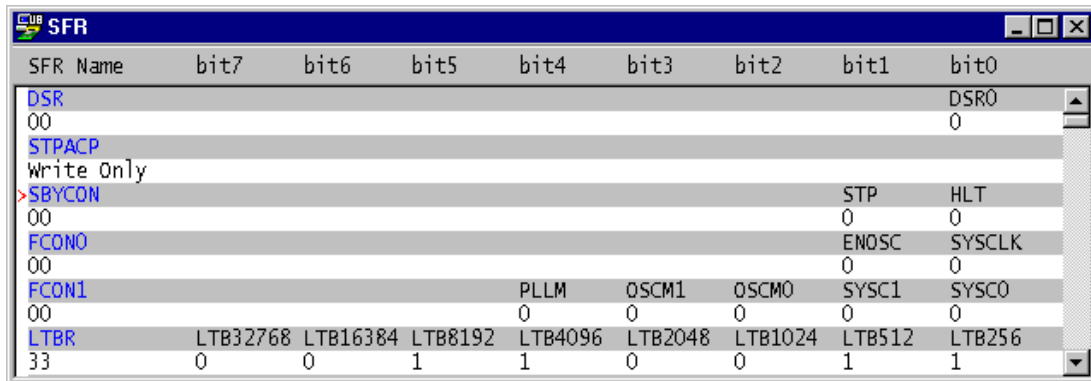
By combining these display register groups, only necessary registers can be displayed in the Register window.

## 8.3 Special Function Registers (SFRs)

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The contents of these registers appear in the SFR window. To change them, click the register line and use the *Change value* command on the SFR window context menu or double-click the current value in the SFR window. Note that double-clicking a bitwise SFR changes only the selected bit field.

Note: An emulation break triggers access to any external circuits connected to these SFRs. So do debugger screen updates while the SFR window is open.



| SFR Name   | bit7     | bit6     | bit5    | bit4    | bit3    | bit2    | bit1   | bit0    |
|------------|----------|----------|---------|---------|---------|---------|--------|---------|
| DSR        |          |          |         |         |         |         |        | DSR0    |
| 00         |          |          |         |         |         |         |        | 0       |
| STPACP     |          |          |         |         |         |         |        |         |
| Write Only |          |          |         |         |         |         |        |         |
| SBYCON     |          |          |         |         |         |         | STP    | HLT     |
| 00         |          |          |         |         |         |         | 0      | 0       |
| FCON0      |          |          |         |         |         |         | ENOSC  | SYSCCLK |
| 00         |          |          |         |         |         |         | 0      | 0       |
| FCON1      |          |          |         | PLLM    | OSCM1   | OSCM0   | SYSC1  | SYSC0   |
| 00         |          |          |         | 0       | 0       | 0       | 0      | 0       |
| LTBR       | LTB32768 | LTB16384 | LTB8192 | LTB4096 | LTB2048 | LTB1024 | LTB512 | LTB256  |
| 33         | 0        | 0        | 1       | 1       | 0       | 0       | 1      | 1       |

### ■ Searching SFR List

The *Find...* command on the SFR window context menu displays the dialog box for entering the desired SFR name.

### ■ Adding to Watch window

The *Add watch item...* command on the SFR window context menu adds the SFR at the current cursor position to the Watch window.

## 8.4 Memory

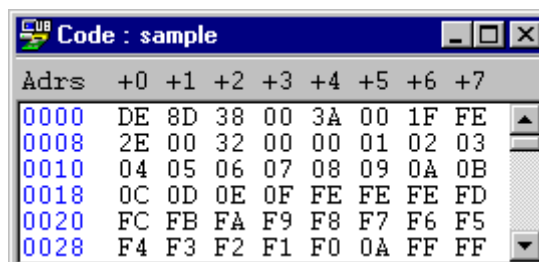
Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

### ■ Displaying Dump Windows

To display the contents of code memory, data memory, or memory in physical segments 1 and higher, use the corresponding command on the View menu. The resulting memory window is for displaying and changing the contents of the corresponding address space.

Note: Selecting the Update code memory also with window update check box on the Other tab in the Tools menu *Environment settings* menu command dialog box produces automatic updates of code memory window contents.

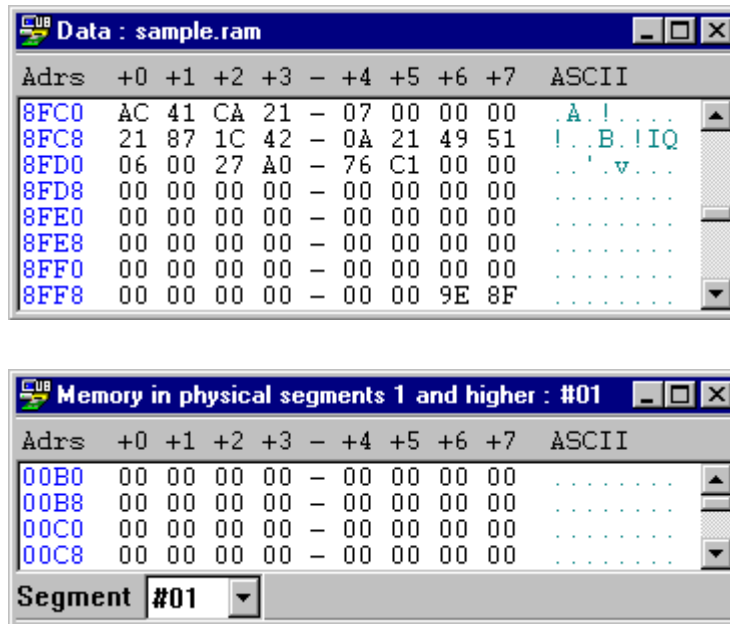
Note: Although the Disassembly window is more convenient for viewing instruction mnemonics and instruction code, the Code memory window is better for examining such areas as table data.



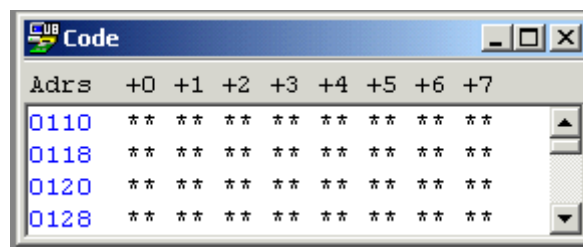
| Adrs | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 |
|------|----|----|----|----|----|----|----|----|
| 0000 | DE | 8D | 38 | 00 | 3A | 00 | 1F | FE |
| 0008 | 2E | 00 | 32 | 00 | 00 | 01 | 02 | 03 |
| 0010 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B |
| 0018 | 0C | 0D | 0E | 0F | FE | FE | FE | FD |
| 0020 | FC | FB | FA | F9 | F8 | F7 | F6 | F5 |
| 0028 | F4 | F3 | F2 | F1 | F0 | 0A | FF | FF |

Data memory windows fill the SFR address space with asterisks (\*) and do not allow modification of SFR contents.

## 8. Referencing/Modifying Registers and Memory

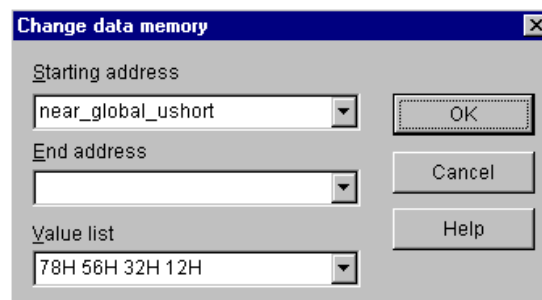


If an error occurs when the DTU8 display the windows of code memory, data memory, or memory in physical segments 1 and higher, the DTU8 issue an error message and fill the window with asterisks (\*).



### ■ Changing Data

The Change xxx memory command on the Dump window context menu opens the following dialog box for changing data.



The final list box accepts a list of numeric values separated by blanks, commas, or semicolons. For code memory and memory in physical segments 1 and higher, word data can be specified in the value list.

Selecting the Display confirmation message for code changes after loading debugging information check box on the Other tab in the Tools menu *Environment settings* menu command dialog box warns that modifying code memory contents threatens to introduce a discrepancy with the Source window contents.

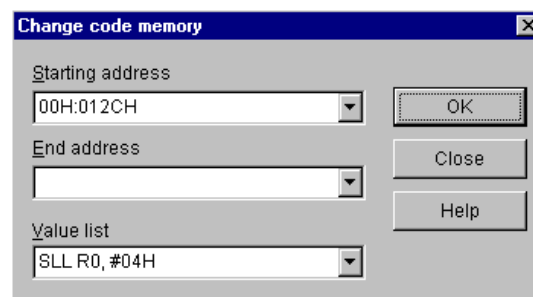
Only the Starting address is required. Specifying an end address repeats the specified value list as often as necessary to fill the specified range.<sup>7</sup>

When entering an address into the Starting address and End address entry fields in the Change physical segments 1 and higher memory dialog box, specify the target physical segment explicitly. For example, to specify address 2000H at physical segment 1, specify 01H:2000H.

Note: The debugger refuses addresses mapped to a region marked N/A or TEST.

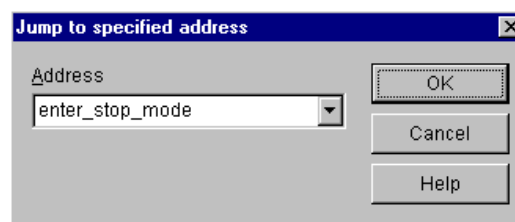
### ■ **Modifying Memory Using Assembly Language Instructions**

For code memory and memory in physical segments 1 and higher, the dialog box also accepts assembly language instructions. In this case, pushing the OK button does not close the dialog box, but updates the starting address ready for the next assembly language instruction.



### ■ **Jump to specified address**

The *Jump...* command on the Dump window context menu opens the following dialog box for specifying the jump target address.



### ■ **Data fill**

The *Fill...* command on the Dump window context menu opens the following dialog box for filling the address space with the same value.

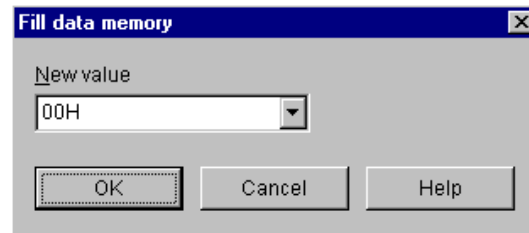
---

<sup>7</sup> If the specified starting address is an odd address and a value represented in units of words is specified in the value list, that starting address is corrected to an even address. For example, if the starting address is 2001H and 1234H is specified in the value list, the starting address will be corrected to 2000H, 34H will be written to 2000H, and 12H will be written to 2001H.

## 8. Referencing/Modifying Registers and Memory

Note: In uEASE or nanoEASE, the *Fill...* command on the Dump window context menu is only enabled for the Dump window for data memory. To make an equivalent operation possible for the Dump window for code memory or memory in physical segments 1 and higher, select the Change code memory command or Change memory in physical segments 1 and higher command on the Dump window context menu and then specify the starting address and end address of the target memory to change data.

The *Fill...* command is a convenient way to initialize data regions in data memory or current segment for a physical segment 1 and higher memory window.



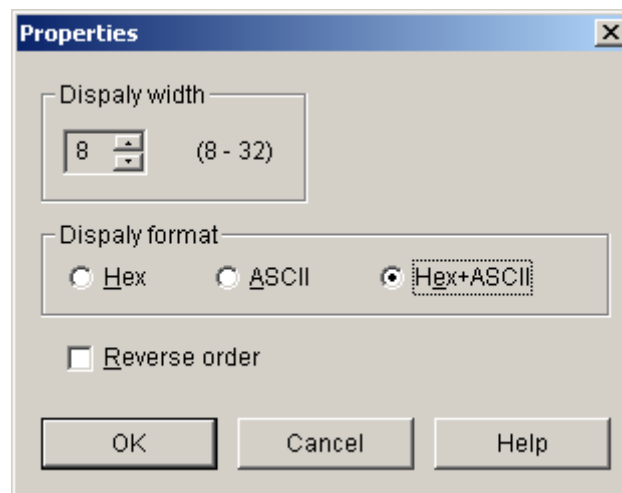
### ■ **Modifying Memory Using Source or Disassembly Window**

Selecting a variable or label in a Source or Disassembly window and choosing the context menu *Change value* menu command displays the appropriate dialog box: Change code memory, Change data memory, or Change physical segments 1 and higher memory. This dialog box automatically inserts the address corresponding to the selected variable or label as the default starting address, allowing the user to proceed immediately to entering the value list.

### ■ **Changing the number of data items displayed in each line in a memory window**

It is possible to change the number of data items to be displayed in each line in a Code memory window, a Data memory window, or a Memory in physical segments 1 and higher window.

Selecting the *Properties* command on the Code memory window, Data memory window, or Memory in physical segments 1 and higher window context menu displays the following dialog box:



### ■ **Display width entry field**

Specify the number of data items displayed in one line. An even number of 8 to 32 can be specified.



For example, if a value of 32 is specified for the display width of a Data memory window, data for 32 addresses is displayed in each line, as shown below.

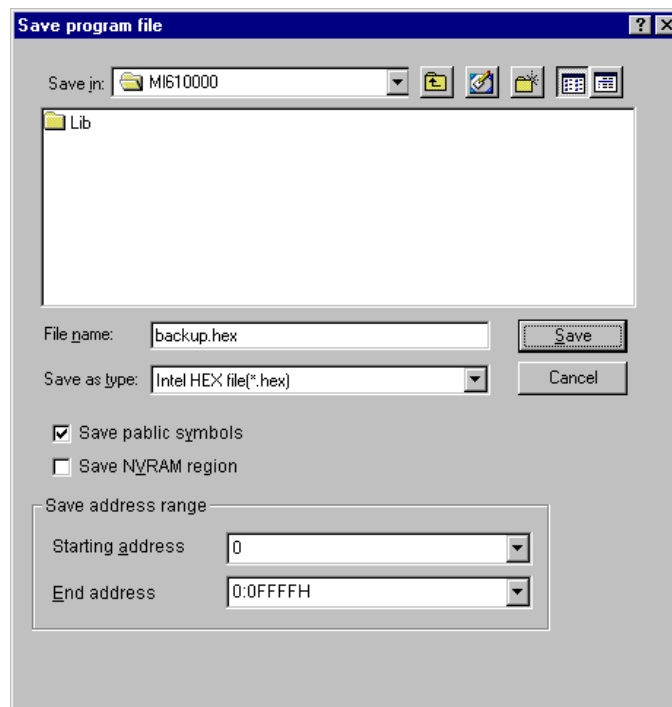
| Data |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |  |  |  |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--|--|--|
| Adrs | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 | +A | +B | +C | +D | +E | +F | +10 | +11 | +12 | +13 | +14 | +15 | +16 | +17 | +18 | +19 | +1A | +1B | +1C | +1D | +1E | +1F |  |  |  |
| 0000 | A2 | 8C | 52 | 00 | 54 | 00 | 1F | FE | 2E | 00 | 32 | 00 | 00 | 01 | 02 | 03 | 04  | 05  | 06  | 07  | 08  | 09  | 0A  | 0B  | 0C  | 0D  | 0E  | 0F  | FE  | FE  | FE  | FD  |  |  |  |
| 0020 | FC | FB | FA | F9 | F8 | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 | FF | FF | FF | 0F  | FE  | 6E  | F0  | 80  | A0  | DE  | 8C  | 12  | 90  | 10  | 81  | 12  | 92  | 00  | 81  |  |  |  |
| 0040 | 26 | F0 | 13 | 90 | 10 | 81 | 0A | 70 | 00 | 51 | 01 | C8 | FF | CE | 2E | F0 | 0F  | FE  | 01  | CE  | FF  | CE  | 00  | E0  | 00  | E2  | 00  | E4  | 00  | E6  | 00  | 08  |  |  |  |
| 0060 | 80 | 09 | 8A | F0 | 57 | 90 | 88 | E8 | 8E | 79 | FC | C1 | 00 | 78 | FA | C1 | 00  | 0A  | 0C  | F0  | 6C  | 01  | AF  | 90  | 52  | 90  | FF  | 70  | 02  | C8  | FF  | 71  |  |  |  |
| 0080 | 20 | C9 | AF | 90 | 52 | 92 | AF | 90 | 52 | 94 | AF | 90 | 50 | 96 | AF | 90 | 50  | 97  | 80  | E4  | F0  | C9  | 01  | A2  | 0B  | C8  | 01  | A4  | 09  | C8  | 6F  | 90  |  |  |  |
| 00A0 | 02 | 98 | 82 | E0 | 7F | 90 | 23 | 98 | 82 | E2 | FE | E4 | F8 | C8 | E3 | CE | 6F  | 90  | 00  | 98  | 81  | E0  | 7F  | 90  | 21  | 98  | 81  | E2  | FF  | E4  | F8  | C8  |  |  |  |
| 00C0 | DA | CE | 01 | F0 | 06 | 00 | 00 | F0 | 32 | 01 | CE | F8 | 01 | F0 | F0 | 00 | 00  | E0  | 13  | 90  | 10  | 81  | 90  | A0  | 10  | F0  | 08  | ED  | 8E  | F2  | 55  | 00  |  |  |  |
| 00E0 | AA | 01 | 11 | 90 | 08 | F0 | 11 | 91 | 08 | F0 | 90 | A0 | 09 | F0 | 1F | FE | CE  | F8  | 6E  | F4  | 7E  | F8  | 0C  | F0  | 0C  | 00  | 56  | 90  | 36  | 98  | 01  | F0  |  |  |  |
| 0100 | 18 | 01 | 0C | F0 | E0 | 8C | 57 | 90 | 37 | 98 | 3E | F8 | 2E | F4 | 8E | F2 | 02  | 90  | 4B  | 91  | 4A  | 90  | 1F  | FE  | 7E  | F0  | 7E  | F8  | 0C  | F0  | 1C  | 00  |  |  |  |
| 0120 | 56 | 90 | 36 | 98 | 0C | F0 | F0 | 8C | 57 | 90 | 37 | 98 | 3E | F8 | 3E | F0 | 1F  | FE  | 01  | F0  | CA  | 00  | 11  | CE  | 01  | F0  | DE  | 00  | 81  | A0  | DE  | 8C  |  |  |  |
| 0140 | 08 | C9 | 00 | 00 | 80 | 01 | 01 | F0 | 10 | 01 | 13 | 90 | 02 | 80 | 82 | A0 | DE  | 8C  | 12  | 90  | DE  | 8C  | 01  | F0  | 5C  | 01  | EE  | CE  | 05  | F0  | 05  | C8  |  |  |  |
| 0160 | 12 | 90 | 00 | 80 | 81 | E0 | 13 | 90 | 00 | 80 | 1F | FE | 00 | 00 | 00 | 00 | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  | 00  |  |  |  |
| 0180 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  |  |  |  |
| 01A0 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  |  |  |  |
| 01C0 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  | FF  |  |  |  |

## 8.5 Saving and Comparing Program Code

### 8.5.1 Saving Program Code

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The *Save program file...* command on the File menu is for saving the current program code to a disk file after adjusting ROM table parameters and applying patches in the debugger.



Selecting the Save public symbols check box saves the symbol table as well as the program code.

Selecting the Save NVRAM region check box saves the contents of nonvolatile data memory to a separate disk file--with extension .XNV for the Intel HEX format and .SNV for the Motorola S2 format.

## 8. Referencing/Modifying Registers and Memory

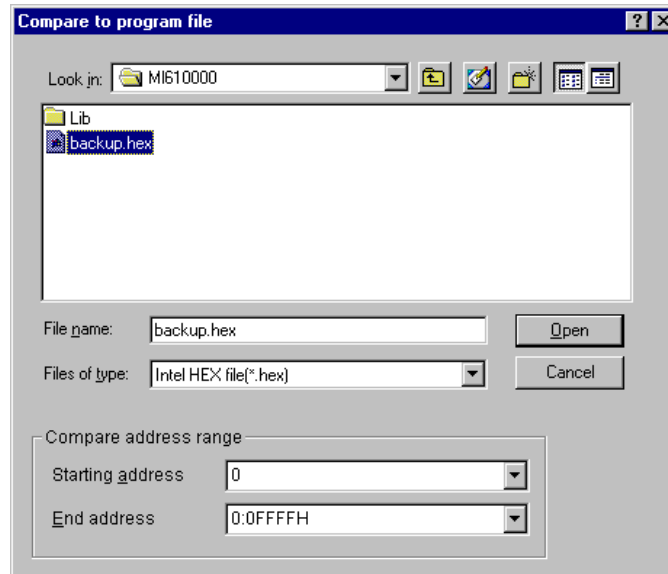
At the bottom of the dialog box are two list boxes for limiting the address range for the save operation.

### 8.5.2 Comparing Program Code

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The File menu *Compare to program file...* menu command displays the following dialog box for comparing the emulator's code region contents with the specified HEX file to confirm program loading and highlight any subsequent changes.

At the bottom of the dialog box are two list boxes for limiting the address range for the compare operation.



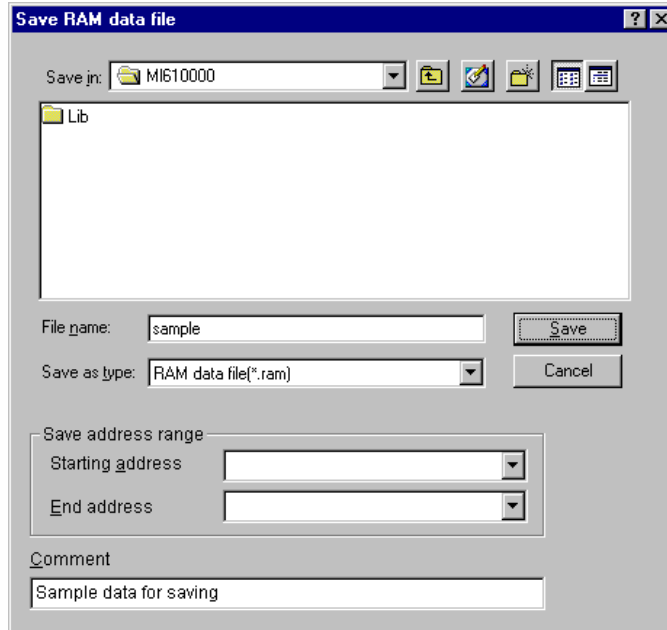
## 8.6 Saving and Restoring Data Regions

### 8.6.1 Saving Data Region

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The File menu *Save RAM data file...* menu command saves the current data memory contents to a file.

Note: Specifying a directory in the RAM data files field on the Directory tab in the Tools menu *Environment settings* menu command dialog box eliminates the need to specify the path each time.



### ■ **Saving a Portion of Data Memory**

Filling in the Starting address and End address input fields saves only the specified address range.

### ■ **Handling of Inaccessible Regions**

The save operation skips address regions that are marked N/A or are otherwise inaccessible.

### ■ **Handling of SFR Address Space**

The save operation treats the SFR address space as inaccessible, so skips it.

### ■ **Inserting Comments**

The save operation copies any text entered into the Comment field in the Save RAM Data to File dialog box to the first line of the target file as a comment. The File menu *Load RAM data file...* menu command displays this comment when the user selects a RAM data file.

## 8.6.2 Restoring Data Regions

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The File menu *Load RAM data file...* menu command displays the dialog box for restoring data memory contents from a file saved with the File menu *Save RAM data file...* menu command.

## 8. Referencing/Modifying Registers and Memory



### ■ Restoring a Portion of Data Memory

Filling in the Starting address and End address input fields restores only the specified address range.

### ■ Displaying Comments

Selecting a RAM data file displays its stored comment in the comment field.

## 8.7 Realtime RAM monitor

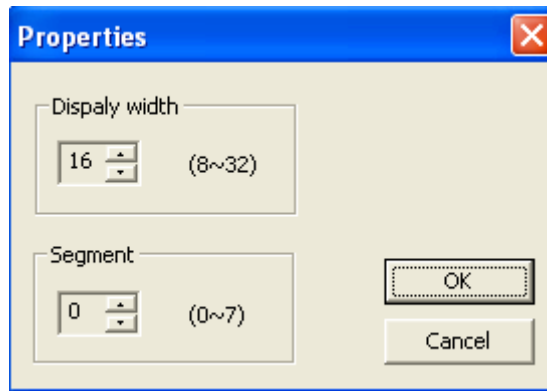
Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE

Maximum 16Kbyte of RAM within one segment can be monitored in realtime on the “Realtime RAM monitor” window. Select [View] menu → [Realtime RAM monitor] to activate the window.

| Adrs    | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 | +8 | +9 | +A | +B | +C | +D | +E | +F |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00:E720 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E730 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E740 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E750 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E760 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E770 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E780 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E790 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E7A0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E7B0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E7C0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E7D0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00:E7E0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 6F | 4E |
| 00:E7F0 | 00 | 00 | FF | FF | F8 | E7 | 00 | 00 | 45 | 00 | 23 | 01 | 00 | 00 | 00 | 00 |

**■ Change segment and the number of data per line**

The segment and the number of data per line can be specified. Select “Properties” on the context menu.

**■ Display width entry field**

Specify the number of data displayed in each line. An even number of 8 to 32 can be specified.

**■ Segment entry field**

Specify the segment displayed in realtime RAM monitor window. An error message shows up if a segment in which any RAM does not exist.

## 9. Tracing Function

The tracing function stores execution history into a memory called trace memory during the realtime emulation.

The tracing function is not available in some emulation systems, which is not highlighted in the “View” menu on the DTU8 debugger.

The trace memory has a capacity of 256K steps and when stored data exceeds 256K steps, data is overwritten in order of time from the oldest data to the latest. In the break mode when the program execution stops, the execution history can be checked tracing back to up to 256K steps.

In the case of Dr.U8 ICE mode, the trace is available in increments of one instruction. However, in the case of instructions which access to RAM, it displays the number of bytes of accessed data.

In the case of simulation mode, trace is available in increments of one cycle, it displays one step one cycle traces.

### 9.1 Trace Window

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

The Trace window displays the contents of the trace memory of the emulator.

To open the Trace window, select the View menu *Trace* menu command.

If the trace memory is empty in cases where a program is not running, for example, a blank window is opened.

On the Trace window, the oldest data traced is displayed in the uppermost line and the latest data traced is displayed in the bottom line. Labels that are displayed directly under the title bar in the Trace window indicate the names for each data types.

Data that has just been changed from its preceding state is displayed in red.

| Trace  |        |              |         |     |      |       |        |                |
|--------|--------|--------------|---------|-----|------|-------|--------|----------------|
| Loc    | PC     | PSW(CZSOMHE) | RAM     | Adr | Data | Probe | Source |                |
| 002860 | 0:0094 | 00 0000000   | 00:0007 | 00  | 00   | 00    | st     | er8, r7:[er2]  |
| 002861 | 0:0094 | 00 0000000   | 00:E7FC | 00  | 00   | 00    |        |                |
| 002862 | 0:0094 | 00 0000000   | 00:E7FD | 00  | 00   | 00    |        |                |
| 002863 | 0:0098 | 00 0000000   | 00:E7FD | 00  | 00   | 00    | add    | er2, #2        |
| 002864 | 0:009A | 20 0010000   | 00:E7FD | 00  | 00   | 00    | add    | er4, #-2       |
| 002865 | 0:009C | C4 1100010   | 00:E7FD | 00  | 00   | 00    | bnz    | init loop2     |
| 002866 | 0:009E | C4 1100010   | 00:E7FD | 00  | 00   | 00    | bal    | init loop      |
| 002867 | 0:0066 | C4 1100010   | 00:E7FD | 00  | 00   | 00    | l      | er0, r10:[ea+] |
| 002868 | 0:0066 | C4 1100010   | 00:0010 | FF  | 00   | 00    |        |                |
| 002869 | 0:006A | A4 1010010   | 00:0011 | FF  | 00   | 00    |        |                |
| 002870 | 0:006A | A4 1010010   | 00:0011 | FF  | 00   | 00    | cmp    | r0, #0ffh      |
| 002871 | 0:006C | 40 0100000   | 00:0011 | FF  | 00   | 00    | bne    | skip           |
| 002872 | 0:006E | 40 0100000   | 00:0011 | FF  | 00   | 00    | cmp    | r1, #0ffh      |
| 002873 | 0:0070 | 40 0100000   | 00:0011 | FF  | 00   | 00    | beq    | init end       |

### ■ **LOC**

Under this label is shown the information that indicates the location of the trace memory. The smaller the value, the older the data.

### ■ **PC**

Under this label is shown the value of the program counter that was in effect when it was traced.

### ■ **PSW (CZSOMHE)**

Under this label are shown the byte count, flags and filed of the PSW.

### ■ **RAM Adr**

Under this label is shown the data memory address that was in effect when an instruction that rewrites the contents of the data memory was executed.

### ■ **Data**

Under this label is shown the data that was written when an instruction that rewrites the contents of the data memory was executed.

In case of nX-U8 core, DTU8 displays the write data by byte value. In case of nX-U16 core, DTU8 displays the write data by byte or word value according to the size of write data.

### ■ **Probe**

Under this label is shown the data of the trace probe cables PROBE0–7 indicated by byte count. In the case of simulation mode, \*\* is displayed.

### ■ **Source / Disassembly**

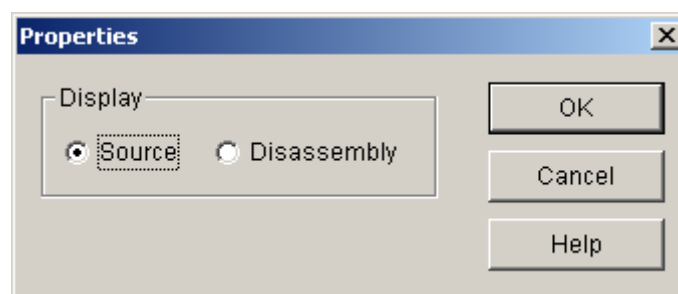
Under this label is shown the source line or execution instruction that corresponds to the traced PC.

Switching of the field display between Source and Disassembly is enabled on the Properties dialog box accessible from the Properties command on the Trace window context menu.

## 9.2 Mode of Display

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

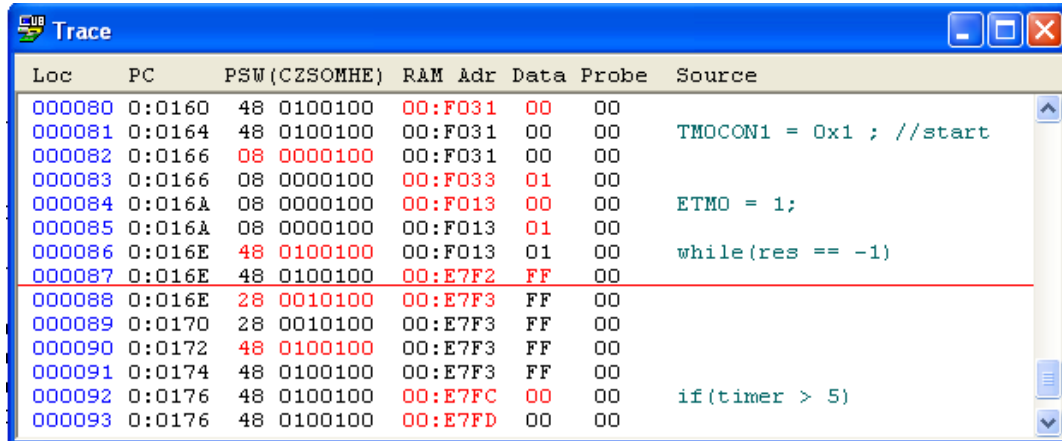
Switching of the field display between Source and Disassembly is enabled on the Properties dialog box accessible from the Properties command on the Trace window context menu.



## 9. Tracing Function

### ■ Source display mode

If the traced PC is in an area in which source debug information exists, the corresponding source line is displayed. For an area containing no source debug information, the result obtained by disassembling the contents of the execution code is displayed.



| Loc    | PC     | PSW(CZSOMHE) | RAM Adr | Data | Probe | Source                  |
|--------|--------|--------------|---------|------|-------|-------------------------|
| 000080 | 0:0160 | 48 0100100   | 00:F031 | 00   | 00    |                         |
| 000081 | 0:0164 | 48 0100100   | 00:F031 | 00   | 00    | TMOCON1 = 0x1 ; //start |
| 000082 | 0:0166 | 08 0000100   | 00:F031 | 00   | 00    |                         |
| 000083 | 0:0166 | 08 0000100   | 00:F033 | 01   | 00    |                         |
| 000084 | 0:016A | 08 0000100   | 00:F013 | 00   | 00    | ETMO = 1;               |
| 000085 | 0:016A | 08 0000100   | 00:F013 | 01   | 00    |                         |
| 000086 | 0:016E | 48 0100100   | 00:F013 | 01   | 00    | while(res == -1)        |
| 000087 | 0:016E | 48 0100100   | 00:E7F2 | FF   | 00    |                         |
| 000088 | 0:016E | 28 0010100   | 00:E7F3 | FF   | 00    |                         |
| 000089 | 0:0170 | 28 0010100   | 00:E7F3 | FF   | 00    |                         |
| 000090 | 0:0172 | 48 0100100   | 00:E7F3 | FF   | 00    |                         |
| 000091 | 0:0174 | 48 0100100   | 00:E7F3 | FF   | 00    |                         |
| 000092 | 0:0176 | 48 0100100   | 00:E7FC | 00   | 00    | if(timer > 5)           |
| 000093 | 0:0176 | 48 0100100   | 00:E7FD | 00   | 00    |                         |

A C source code consists of multiple instruction codes. If the traced PC is at the starting address of a C source, the source code is displayed in the Trace window.

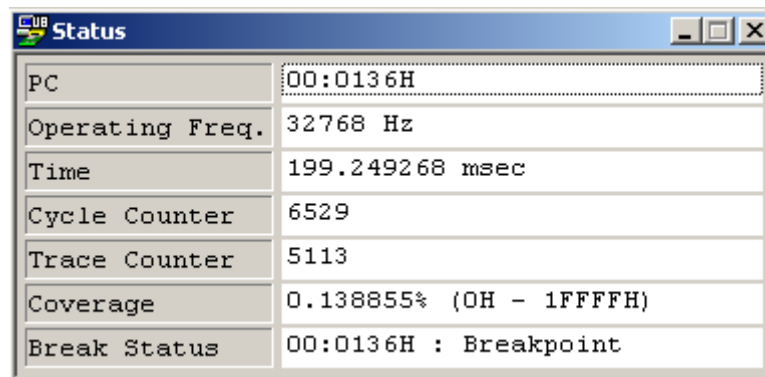
### ■ Disassembly display mode

The disassembled execution code of the traced PC is displayed.

## 9.3 Trace Counter

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

The trace counter shows the step count currently stored in the trace memory. The trace counter value is indicated in the Trace Counter field in the Status window.<sup>8</sup>



| Status          |                         |
|-----------------|-------------------------|
| PC              | 00:0136H                |
| Operating Freq. | 32768 Hz                |
| Time            | 199.249268 msec         |
| Cycle Counter   | 6529                    |
| Trace Counter   | 5113                    |
| Coverage        | 0.138855% (0H - 1FFFFH) |
| Break Status    | 00:0136H : Breakpoint   |

If the contents of the trace memory exceed 256K steps, “(Overflow)” is indicated in the Trace Counter field.

<sup>8</sup> The contents to be displayed in the Status window vary depending on the emulation system connected. The Status window shown above is an example in simulation mode.

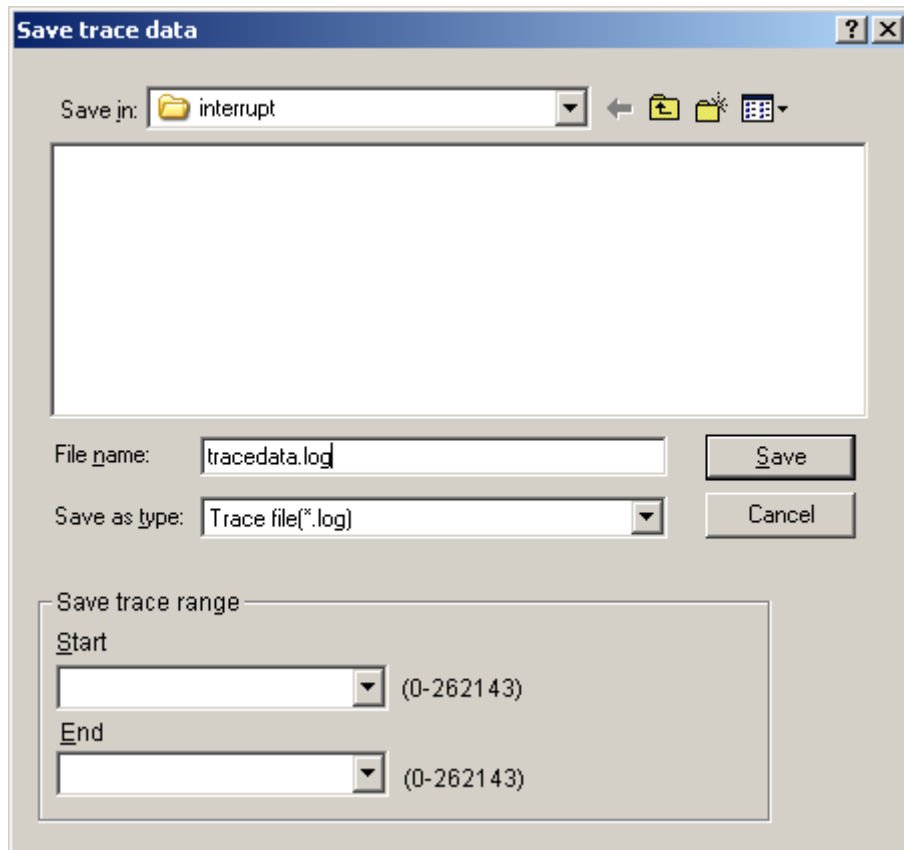


## 9.4 Storing Trace Data

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

Contents of the trace memory can be stored in a disk file.

Trace memory contents can be saved on the Save trace data dialog box accessible from the *Save trace data* command on the context menu.



## 9. Tracing Function

If nothing is specified in the Save trace range field, all the areas in the trace memory will be stored in a file.

Data will be stored as such text data as is shown below.

```
.....
002909 0:00B2 40 0100000 00:0739 FF FF BL 0H:6H
002910 0:0006 40 0100000 00:0739 FF FF RT
002911 0:00B6 40 0100000 00:073A 7E FF L R0,0H:[EA]
-----
Loc      PC      PSW(CZSOMHE) RAM Adr Data Probe Source
-----
002912 0:00B6 40 0100000 00:073A 7E FF
002913 0:00BA 00 0000000 00:073A 7E FF B _main ;600H
002914 0:0600 00 0000000 00:073A 7E FF io_init(); /* Initialize I/O */
002915 0:037A 00 0000000 00:87CF 00 FF {
002916 0:037A 00 0000000 00:87CE 00 FF
002917 0:037A 00 0000000 00:87CD 06 FF
002918 0:037A 00 0000000 00:87CC 04 FF
002919 0:037C 00 0000000 00:87CC 04 FF asm_init_proc();
002920 0:0436 00 0000000 00:87CB 00 FF PUSH LR ; required if subroutine is called.
002921 0:0436 00 0000000 00:87CA 00 FF
002922 0:0436 00 0000000 00:87C9 03 FF
002923 0:0436 00 0000000 00:87C8 80 FF
.....
```

The title is inserted every 32 steps.

As for the Source / Disassembly field of the stored trace data, it is stored in the display mode specified in the Properties dialog box.

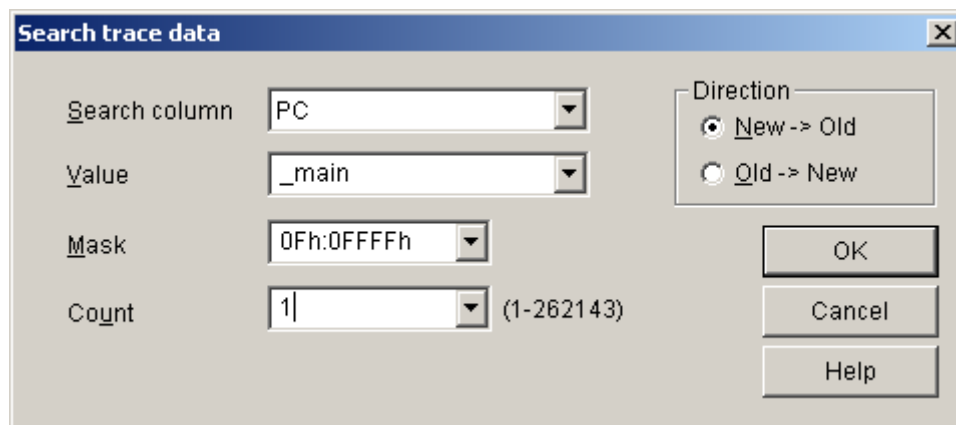
Once trace data is stored, the name of the stored file is indicated on the title bar of the Trace window.

## 9.5 Searching for Trace Data

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

256K steps worth of trace data is a huge amount of data and storing the whole of it takes considerable amount time. For this reason, an emulator is equipped with a function that searches the contents of the trace memory that the emulator itself has.

Selecting the *Search trace data* command on the Trace window context menu opens the following dialog box:



### ■ Search column field

Select a search target.

- PC : Program counter
- PSW : Program status word
- C : Carry flag
- Z : Zero flag
- S : Sign flag
- OV : Overflow flag
- MIE : Master interrupt enable flag
- HC : Half-carry flag
- ELEVEL : Interrupt level
- RAM Address : RAM address to which read/write operations were performed
- RAM Data : RAM data that was read/written
- Probe Data : Probe data

### ■ Direction radio buttons

- New → Old : Searches in the direction from the latest trace data to older data

## 9. Tracing Function

·Old → New : Searches in the direction from the oldest trace data to newer data

### ■ **Value field**

Specify the value of the search target as a search condition. When the search target is the PC, the bit 0 of the PC is masked with 0.

### ■ **Mask field**

Specify in this field a mask value with the value of the search target. When a mask value is specified and then the mask value ANDed with the value of the search target bit by bit matches the search value, a match occurs.

If no mask value is specified, masking is not performed.

### ■ **Count field**

Specify the number of times a match with the search value occurs. A value of 1 to the current trace counter value can be specified.

A search value is compared with the value of the search target only when the value of the search target is changed in Dr.ICE mode. Therefore, when the same data continues, only the first data is compared.

## 9.6 Clearing Trace Memory

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

Selecting the *Clear trace memory* command on the Trace window context menu clears the trace memory contents.

Selecting the Trace memory check box for reset target selection on the Reset tab in the Tool menu *Operation settings* menu command dialog box clears the trace memory contents at reset.

## 9.7 Trace Trigger

Dr.U8 ICE  
Dr.U16 ICE

The trace trigger function enables to control events to start or stop the trace. Select [Run] menu → [Trace trigger set] to get this dialog box.

### ■ Start trigger ON/OFF

Free run:

Trace is started when an emulation starts. When the “Free run” is checked, “PC match” or “RAM match” is not available.

PC match:

Trace will be started if the specified address set in the [Address] field is executed with specified number of time set in the [Pass count value] field.

When you set up a count value, please click [set >>] button after typing numbers in [Pass count value]. The [Count value] indicates a count value set in the emulator. So, it may be updated when you get any emulation breaks before starting the trace. In that case, you can count up it again by clicking the [set >>] button.

## 9. Tracing Function

RAM match:

Trace is started by conditions of RAM address during the program execution.

Specify a RAM address in the [RAM address] field, which can be bitwise ANDed to data specified in [RAM address mask] field.

Also, select one of “Read”, “Write”, or “Read/Write” in the [Access method] field. If [Specify RAM data] is checked, trace will be started when the access data while executing the code matches to data specified in [RAM data] field. You can also specify access methods, mask data which is bitwise ANDed for the RAM data, and “Equal” or “Not Equal” in the [Compare] field..

### ■ Trace is stopped when the trace counter overflows

If this check box is checked, trace will be stopped when a trace memory exceeds 256K step, allows the trace memory not to be overwritten that help check the 1<sup>st</sup> 256K step of execution history.

## 9.8 Others

### 9.8.1 Jumping to Source Window /Disassembly Window

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

Selecting the *Jump to Source window* command on the Trace window context menu jumps to the Source window that indicates the line containing the trace PC at the cursor position.

Selecting the *Jump to Disassembly window* command on the Trace window context menu jumps to the Disassembly window that indicates the line containing the trace PC at the cursor position.

Double-clicking on the “PC” field of the Trace window jumps to either a Source window or a Disassembly window:

- In source display mode, if the trace PC is in the source code area, control jumps to a Source window, and if the trace PC is outside the trace code area, control jumps to a Disassembly window.
- In disassembly display mode, control jumps to a Disassembly window.

### 9.8.2 Jumping to Specified Line

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

Selecting the *Jump* command on the Trace window context menu opens a dialog box, where the cursor can be jumped by specifying a line number of the trace memory (LOC field).

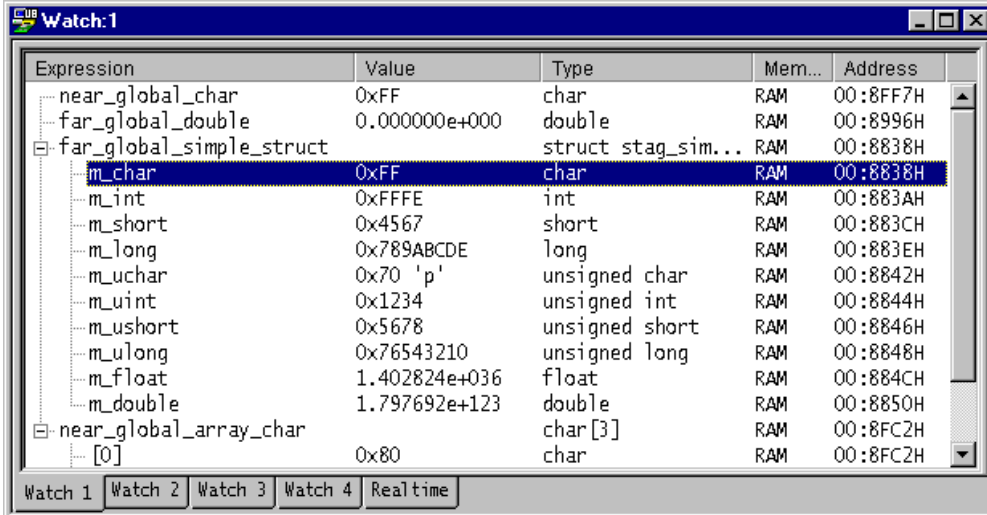
The dialog box for specifying a jump destination line number can also be displayed by double-clicking on the “LOC” field in the Trace window.

# 10. Watch Function

## 10.1 Overview

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The *Watch window* command on the View menu opens the Watch window for monitoring C variables, data addresses, registers, SFRs, and other items. Item definitions can include register indirect references and other complicated addressing modes. The window even supports memory references using labels.



| Expression               | Value         | Type               | Mem... | Address  |
|--------------------------|---------------|--------------------|--------|----------|
| near_global_char         | 0xFF          | char               | RAM    | 00:8FF7H |
| far_global_double        | 0.000000e+000 | double             | RAM    | 00:8996H |
| far_global_simple_struct |               | struct stag_sim... | RAM    | 00:8838H |
| m_char                   | 0xFF          | char               | RAM    | 00:8838H |
| m_int                    | 0xFFFE        | int                | RAM    | 00:883AH |
| m_short                  | 0x4567        | short              | RAM    | 00:883CH |
| m_long                   | 0x789ABCDE    | long               | RAM    | 00:883EH |
| m_uchar                  | 0x70 'p'      | unsigned char      | RAM    | 00:8842H |
| m_uint                   | 0x1234        | unsigned int       | RAM    | 00:8844H |
| m_ushort                 | 0x5678        | unsigned short     | RAM    | 00:8846H |
| m_ulong                  | 0x76543210    | unsigned long      | RAM    | 00:8848H |
| m_float                  | 1.402824e+036 | float              | RAM    | 00:884CH |
| m_double                 | 1.797692e+123 | double             | RAM    | 00:8850H |
| near_global_array_char   |               | char[3]            | RAM    | 00:8FC2H |
| [0]                      | 0x80          | char               | RAM    | 00:8FC2H |

Using the *Watch window* command on the View menu for the first time opens an empty Watch window with no items. Use the *Add item...* command on the Watch window context menu and similar commands to add watch items.

Although a total of five tabs, Watch 1 to Watch 4 and Realtime, are displayed in the above example, the display of the tabs varies depending on the type of emulator used.

Appendix 13.2 "Watch Window Candidates" lists the types of items that can be added to the watch list.

### 10.2 Adding Items

#### ■ From Watch Window Context Menu



The *Add item...* command on the Watch window context menu displays the following dialog box for adding a watch item.

The 'Watch item' dialog box is shown with the following settings:

- Type:** ☐ C, ☒ Assembly
- Address:** 0E000h
- Address Attribute:** ☐ Code, ☒ Data, ☐ Bit
- Data size:** ☒ Byte, ☐ Word, ☐ Double word
- Byte Number of Display:** [1-128]
- Radix:** ☒ Hexadecimal, ☐ Decimal, ☐ Binary
- Reverse order:** ☒

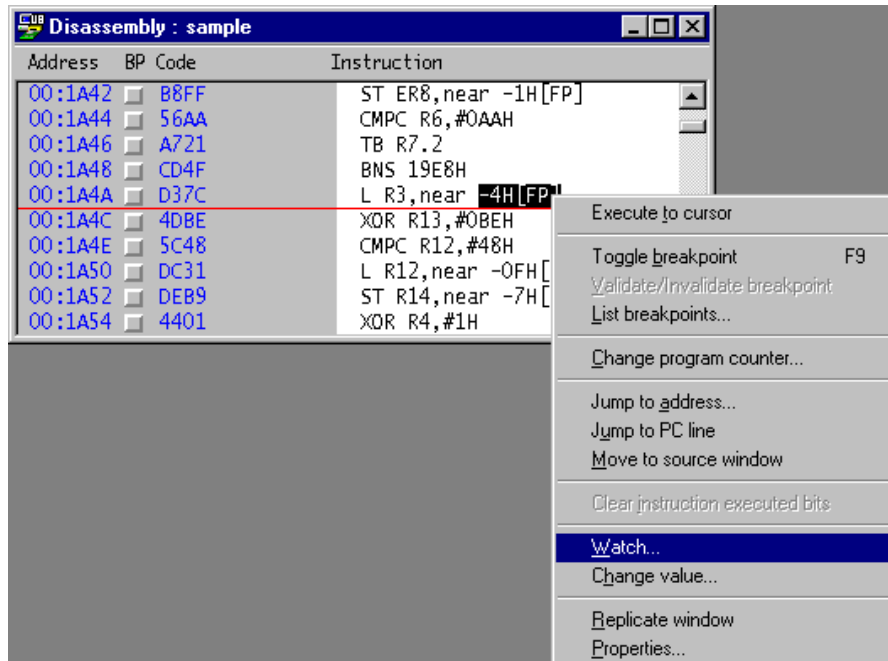
- [Type] Specify the type of watch item to add.
- [Address] Enter an expression for the watch item.
- [Address Attribute] If the watch item is an address expression, specify the size of the watch item at that address. C variables do not have this field. C variables do not need this setting, so gray out this portion.
- [Data size] If the watch item is an address expression, specify the size of the object at that address. C variables do not have this field.
- [Byte Number of Display] If the watch item is a data address expression, specify the number (1 to 128) of objects to display starting at that address. C variables do not have this field.
- [Radix] Specify the radix for displaying the data.
- [Reverse order] Selecting this check box arranges the data in descending order from the value left. C variables do not have this field.



### ■ From Source or Disassembly window

Right-click on the string to add and choose the *Add watch item...* command on the context menu to display the dialog box for adding the watch item to the Watch window

If the expression consists of more than a single word-- -4H[FP], for example--select the entire expression with the mouse before right-clicking.



### ■ From SFR or Register Window

Choosing the *Add watch item...* command on the window's context menu displays the dialog box for adding the SFR or register at the cursor position to the Watch window.

### ■ From Symbol List

Choosing the *Symbol list...* command on the Tools menu displays the symbol list dialog box.

Pushing the Add to watch button displays the dialog box for adding the currently selected symbol to the Watch window.

### ■ Watch Item List Limitations

The Watch window holds 254 items per tab on four tabs for a maximum of 1024 items. The only way to use more is to save a tab to a file, delete those watch items from the debugger, and read in new ones.

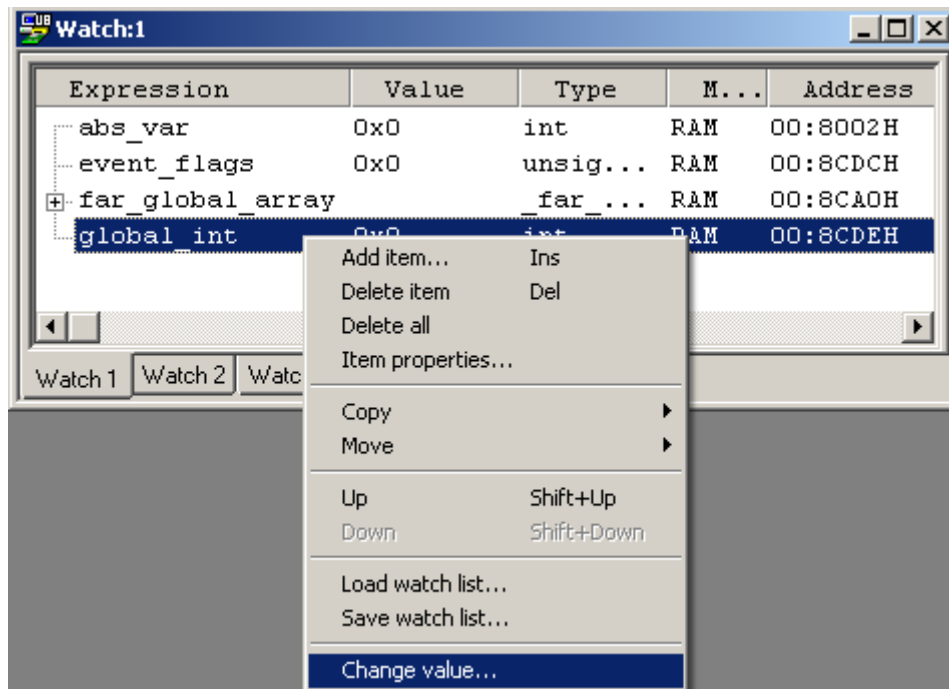
## 10. Watch Function

### 10.3 Changing Data on Watch Window

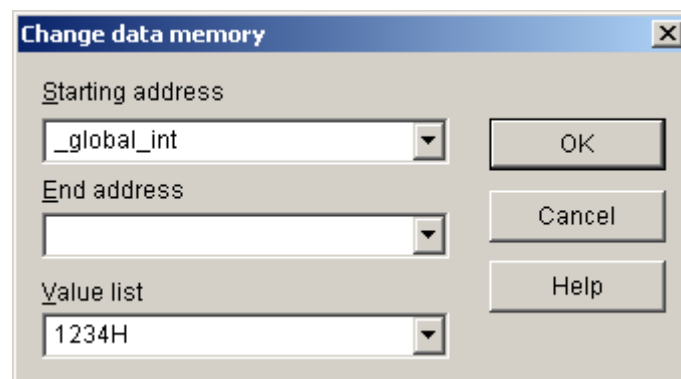
Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The DTU8 debugger has a function to change the value of an item held in the Watch window.

Double-clicking on an item in the Watch window or choosing an item and then choosing the *Change value...* command on the window's context menu displays a dialog box for changing the value of the item.



Choosing the *Change value...* command on the window's context menu displays the following dialog box:



In this example, the contents (contents of RAM of 8CDEH) of variable “global\_int” are changed to 1234H. Symbols can also be specified in the Value list field

When an end address is specified, the data specified in the Value list field is written to the range from the start address to the end address.

In the above example, the RAM value is to be changed. The dialog box to be displayed varies depending on the type of item whose value is to be changed, as shown below.

| Target item   | Dialog box to be displayed                              |
|---|---|
| RAM   | Change data memory dialog box                           |
| ROM   | Change code memory dialog box                           |
| Address in memory in physical segments 1 and higher | Change physical segments 1 and higher memory dialog box |
| SFR   | Change SFR dialog box                                   |
| Register  | Change register dialog box                              |

## 10.4 Saving Watch Item List

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The *Save watch item list* command on the Watch window context menu saves the current watch item list to a disk file.

Saving such lists makes it unnecessary to set up the same list anew for repeated debugging sessions.

This menu command saves only the watch item list for the currently displayed tab.

Debugger project files save all four watch item lists.

Note that this command saves the watch items, but not their contents. Loading a list displays the current values for the items on the list, not the values that they had when the list was saved.

## 10.5 Loading Watch Item List

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The *Load watch item list...* command on the Watch window context menu is for adding a list of watch items to the Watch window.

Note that specifying a directory in the Watch item files field on the Directory tab of the Tools menu *Environment settings* menu command dialog box eliminates the need to specify the path when reading in watch item files.

The menu commands for saving and reading in watch item lists together allow the developer to maintain separate watch item lists for individual source code files and functions.

### ■ Important Note on Reading in Watch Item Lists

A Watch window tab holds a maximum of 254 watch items. Make sure that the operation does not cause the list to exceed this limit.

## 10. Watch Function

### 10.6 Editing Watch Item List

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

#### ■ Reordering List

To move a watch item up or down the list, select it and then choose the appropriate command on the Watch window context menu: *Up* one or *Down* one.

Alternatively, use the keyboard shortcut Shift plus Up or Down arrow.

#### ■ Changing Radix

To change the display radix for a watch item, select the watch item and choose the *Properties...* command on the Watch window context menu.

#### ■ Deleting Watch Items

To delete a watch item, select it and choose the *Delete item* command on the Watch window context menu.

To delete all watch items, use the Delete all command on the Watch window context menu.

#### ■ Copying Watch Items

The context menu *Copy to tab* menu command copies the currently selected watch item to the specified tab.

#### ■ Moving Watch Items

The context menu *Move to tab* menu command copies the currently selected watch item to the specified tab and then deletes that item from the current tab.

### 10.7 Real-Time Watch

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

The Watch window has a [Realtime] tab that enables monitoring variations in values during program execution. By registering RAM addresses to the [Realtime] tab, it is possible to monitor variations in data with respect to the registered addresses while executing the program.

This function may not be supported depending on emulator. For details refer to the manual of the emulator used.

The following can be registered to the [Realtime] tab:

- Addresses that have been assigned to memory that can be used as RAM
- SFRs
- Symbol that has the DATA attribute of the assembly language source level
- C variable assigned to the available RAM area.

But in the case of a variable with child items, such as array and a structure, a child item is not displayed.

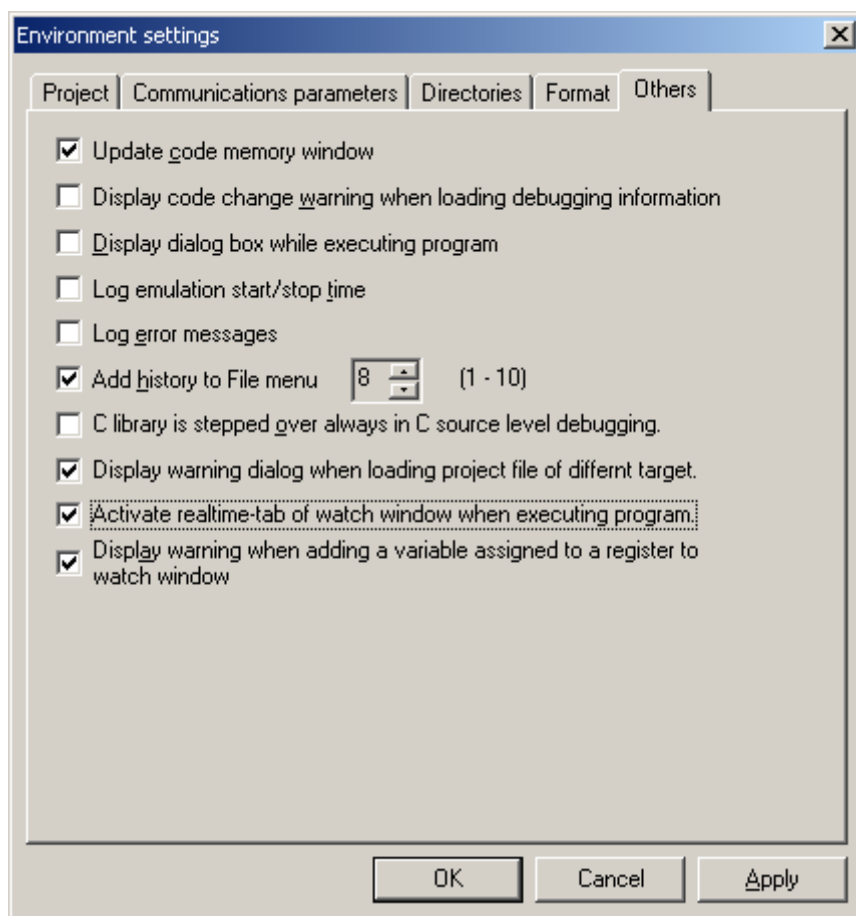
The number of addresses that can be registered to the [Realtime] tab is as follows:

- In Dr.U8 ICE mode: A maximum of 8 addresses (bytes); specification of 8/16/32 bit size is available about each address.
- In simulation mode: A maximum of 16 addresses (bytes); specification of 8/16/32 bit size is available about each address.

### 10.7.1 Real-Time Watch Display Options

Whether to displays the Watch window's Realtime tab by bringing it to the front when program execution is started can be specified.

Choosing the Others tab in the Environment settings dialog box accessible from the Tool menu *Environment settings* menu command displays the following dialog box:



#### ■ **Activate realtime-tab of watch window when executing program**

Selecting this check box displays the Watch window's Realtime tab by bringing it to the front when program execution is started. Note, however, that this foreground display is disabled if the Watch window is closed or no watch item has been registered in the Realtime tab.

This check box is put into a deselected state when you run the DTU8 debugger for the first time.

## **10. Watch Function**

The settings information of this check box is stored in the appropriate project file of the DTU8 debugger.

# 11. Other Functions

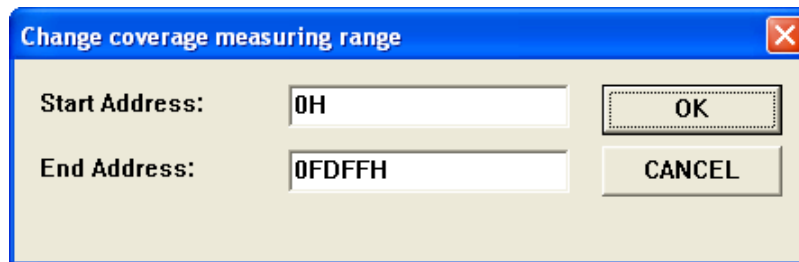
---

## 11.1 Coverage Function

Dr.U16 ICE  
Simulate

The debugger in Dr.U16ICE mode and simulation mode has a coverage function that calculates a program execution coverage rate based on the information about code addresses that have already been executed. Using the Coverage Function

To Change coverage measuring range, use the corresponding “Set coverage area” command on the Status window context menu or double-click the Coverage in the Status window.

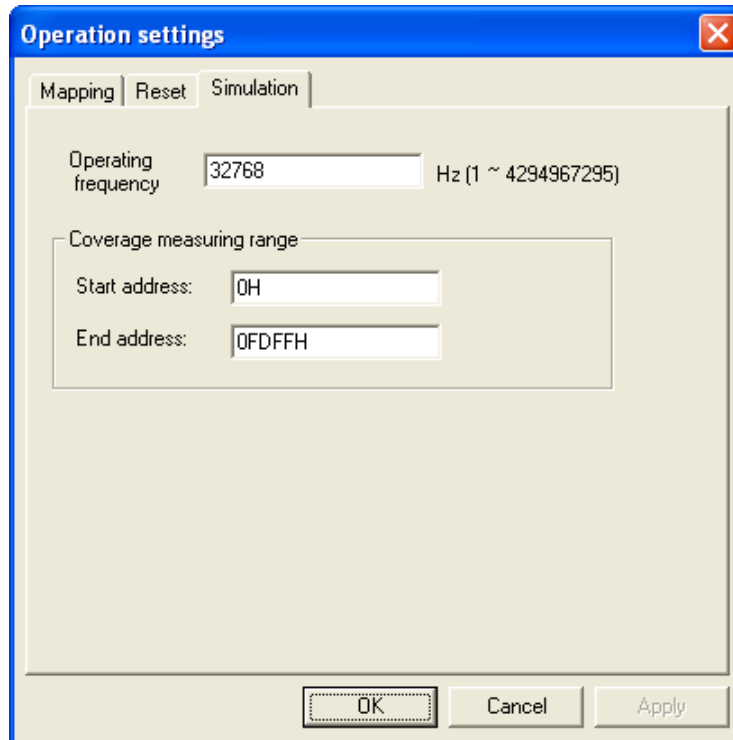


[Start address] Specify the address at which coverage measurement is to be started.

[End address] Specify the address at which coverage measurement is to be terminated.

In simulation mode, on the [Simulation] tab of the dialog box opened by the [Operation settings] menu command of the Tool menu are the input fields for specifying the coverage measurement range, as shown below.

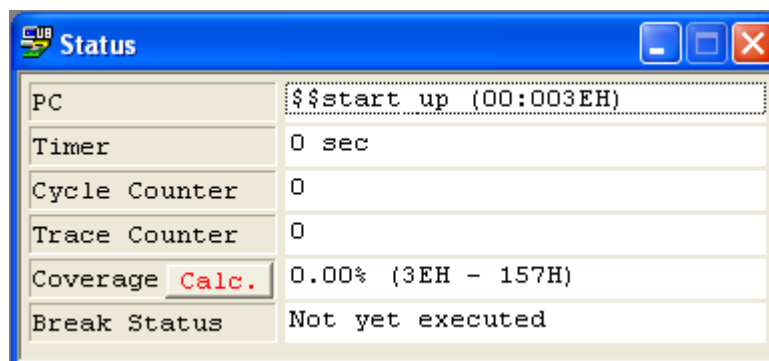
## 11. Other Functions



- [Start address] Specify the address at which coverage measurement is to be started.
- [End address] Specify the address at which coverage measurement is to be terminated.

When an address range (normally, the range from the leading address to the end address of the loaded program) for measuring a coverage rate is specified in the [Coverage measuring range] field and the program is executed, the execution coverage rate at the time a break occurred during the Simulation Execution is displayed in the [Coverage] field in the Status window.

### 11.1.1 Displaying the Coverage Rate



(Status window in Dr.U16ICE mode)

The execution coverage rate is displayed in the [Coverage] field in the Status window in the following format:

**Coverage**                      *rate% (startaddress - endaddress)*



For *startaddress* and *endaddress*, specify the program address range for performing coverage measurement. *rate* shows a value in percent figures where the ratio of the executed instructions to all the instructions in *startaddress* and *endaddress* has been converted to percentage.

In Dr.U16ICE mode, the "Calc." button is displayed at the Coverage field. If the string of the "Calc." button is red, it means that the display field is not updated. To know the execution coverage rate, please push the "Calc." button and update a display field.

The string of the "Calc." button changes red when DTU8 is started, or when the IE bits is cleared, or when the emulation is started.

Execution coverage rate is calculated based on the IE (Instruction Executed) bits in the Source or Disassembly window. Therefore, to initialize the execution coverage rate, select the [Clear instruction executed bits] menu command of the pop-up window of the Source or Disassembly window.

## 11.2 Macro Function

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

In addition to keyboard input, the debugger supports macros written in a script language. These text files can be created with any text editor.

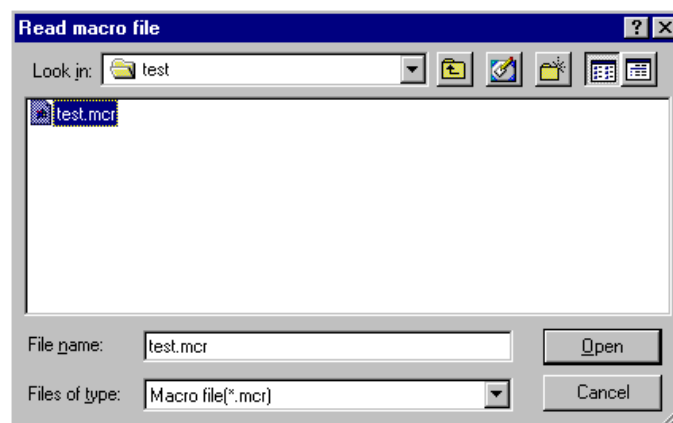
The execution results go to the Log window.

For further details on the script language, see Chapter 13 "Appendices."

### 11.2.1 Overview

The *Run macro...* menu command on the Tools menu is for executing a text file as a macro.

Double-clicking a text file in the resulting dialog box starts execution.



The execution results go to the Log window. If necessary, the Log window contents can be saved to a disk file.

Note that specifying a directory in the Macro files field on the Directory tab of the Tools menu *Environment settings* menu command dialog box eliminates the need to specify the path when reading in macro files.

## 11. Other Functions

### ■ Example

This script file generates a log file (TEST1.LOG) with the following output.

```
CLRLOG                ; Clear Log
ECHO "TEST1 MACRO("$DATE ")" ; Output title to header
LOD TEST.ABS          ; Reading object file
SBP _sub1              ; Setting breakpoint to sub1
ECHO "START AT " $TIME ; Output starting time
RST                   ; Reset
G                     ; Start execution
ECHO "STOP AT " $TIME  ; Output ending time
DBS                   ; Display the break status
DDM _AREA1            ; Display the Data memory
DSYM _sub1             ; Display symbol information of sub1
IF PC == 7CH GOTO OK   ; If break address is 7CH, OK
ECHO "NG"              ;
GOTO FIN              ;
:OK
ECHO "GOOD!"           ;
:FIN
SAVELOG TEST1.LOG      ; Saving Log file to disk
```

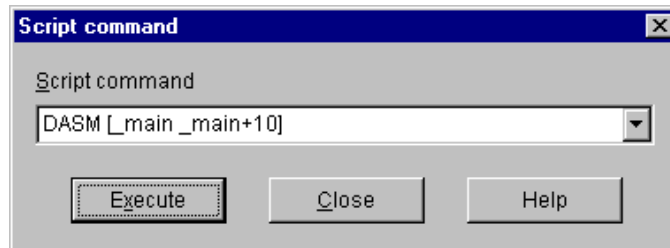
This script file generates a log file (TEST1.LOG) with the following output.

```
> TEST1 MACRO(2001/07/02)
> LOD TEST.ABS                ; Reading object file
> SBP _sub1                    ; Setting breakpoint to sub1
> START AT 15:38:05
> RST                          ; Reset
> G                            ; Start execution
> STOP AT 15:38:06
> DBS                          ; Display the break status
Break status
Breakpoint
> DDM _AREA1                  ; Display the Data memory
8BFE : 77H
> DSYM _sub1                  ; Display symbol information of sub1
Symbol Name      Value      Type
-----
_sub1            7CH (124)    CODE
> IF PC == 7CH GOTO OK        ; If break address is 7CH, OK
>
> GOOD!
> SAVELOG TEST1.LOG           ; Saving Log file to disk
```

### 11.2.2 Executing a Single Script Command

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The Tools menu *Execute script command* menu command allows the developer to enter and execute DTU8 debugger script language commands one command at a time.



Specifying a script command in the entry field and pressing the Execute button executes the command and sends its results to the Log window. Typical uses include saving the contents of a Disassembly window or memory dump window to a file. To save disassemble results from a Disassembly window to a file, for example, choose the Tools menu *Execute script command* menu command, type DASM followed by the option address range pair, and press the Execute button to send the results to the Log window. The Log window context menu *Save log* menu command then saves the disassemble results to a file.

## 11.3 Log Function

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The Log window maintains a record of debugger activities.

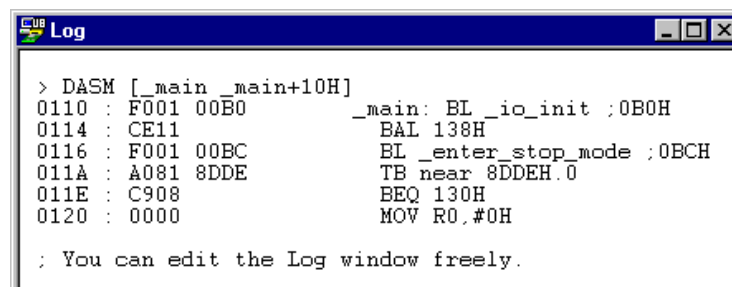
The Log window incorporates rudimentary text editor facilities for editing text, searching for strings, and replacing strings. You can, for example, add your own comments to macro output.

### 11.3.1 Using Log Window

The Log window command on the View menu opens the Log window

This window displays the following.

- Macro execution results
- Program file comparison results
- Emulation start and end times
- Error and warning messages
- Text



Note: Sending error and warning messages to the Log window requires selecting the Log messages check box on the Other tab in the Tools menu *Environment settings* menu command dialog box.

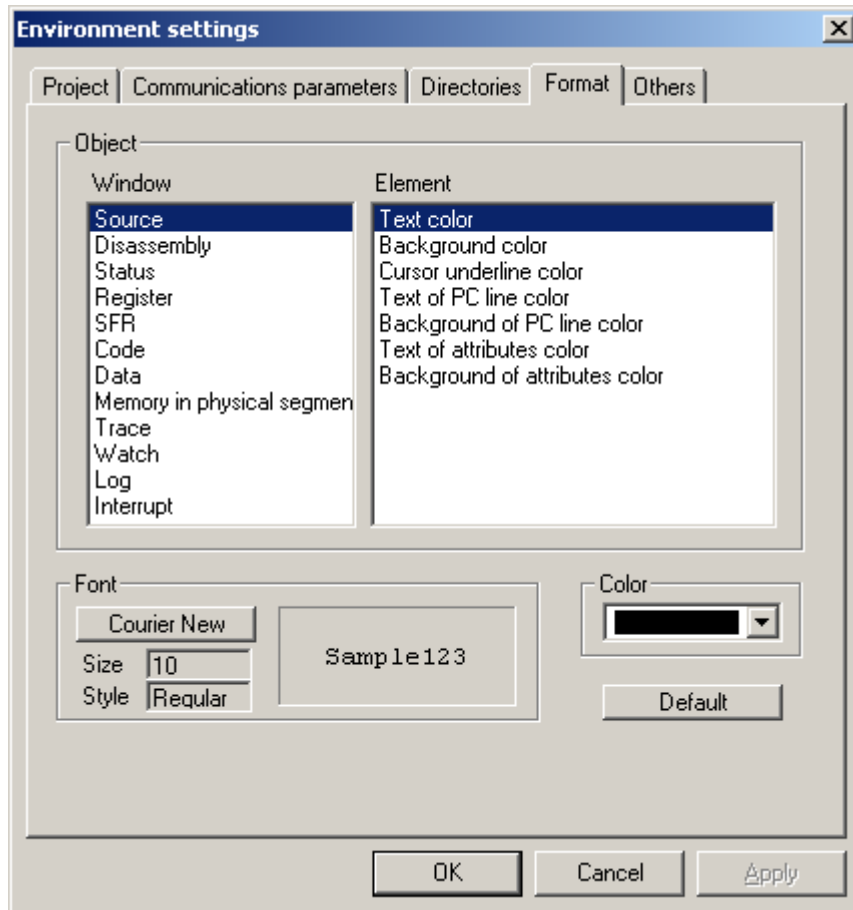
## 11. Other Functions

### 11.4 Customizing Various Windows

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Display font and colors for display can be changed for each window.

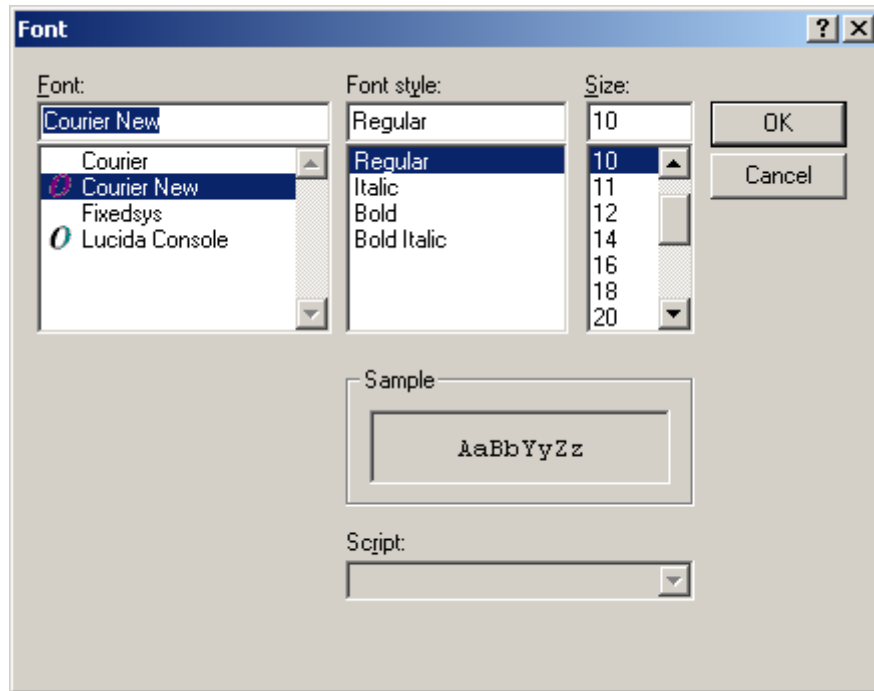
Choosing the Format tab in the Environment settings dialog box accessible from the Tool menu *Environment settings* menu command displays the following dialog box:



The Font field on the Format tab displays the font name and size for the window currently selected in the Object field. Clicking on the button showing the font name (called font-name button) opens a dialog box for changing font settings.

#### ■ Changing font

Clicking on the font-name button opens the following dialog box for changing font settings:



Select the font you want to change, font style, and size from their respective list boxes.

Only monospaced fonts can be used in the DTU8 debugger. The Font list does not display any proportional font.

#### ■ **Windows whose font can be changed**

The following windows allow their font to be changed:

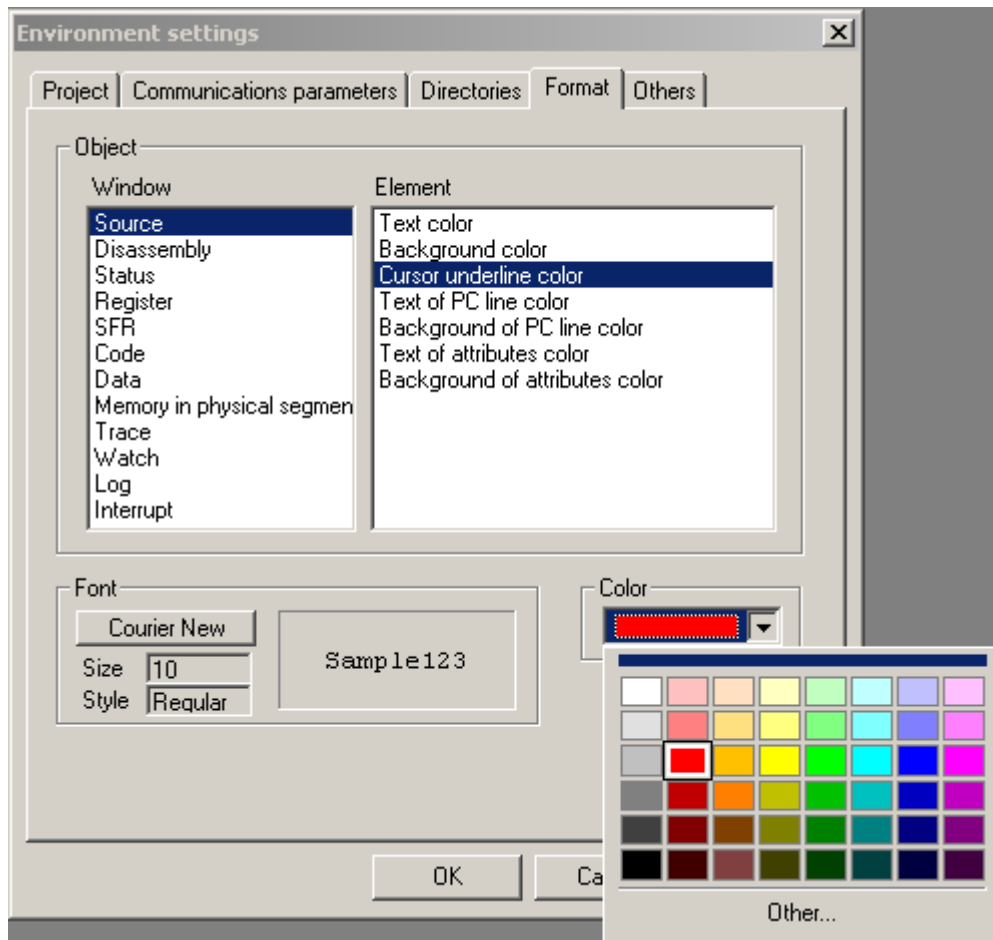
- Source
- Disassembly
- SFR
- Code memory
- Data memory
- Memory in physical segments 1 and higher
- Trace
- Watch
- Log

The Status, Register, and Interrupt windows only allow changing colors; it does not allow changing fonts.

#### ■ **Changing color**

To change color, choose a desired one from the Color field pull-down list.

## 11. Other Functions



### ■ Using the default

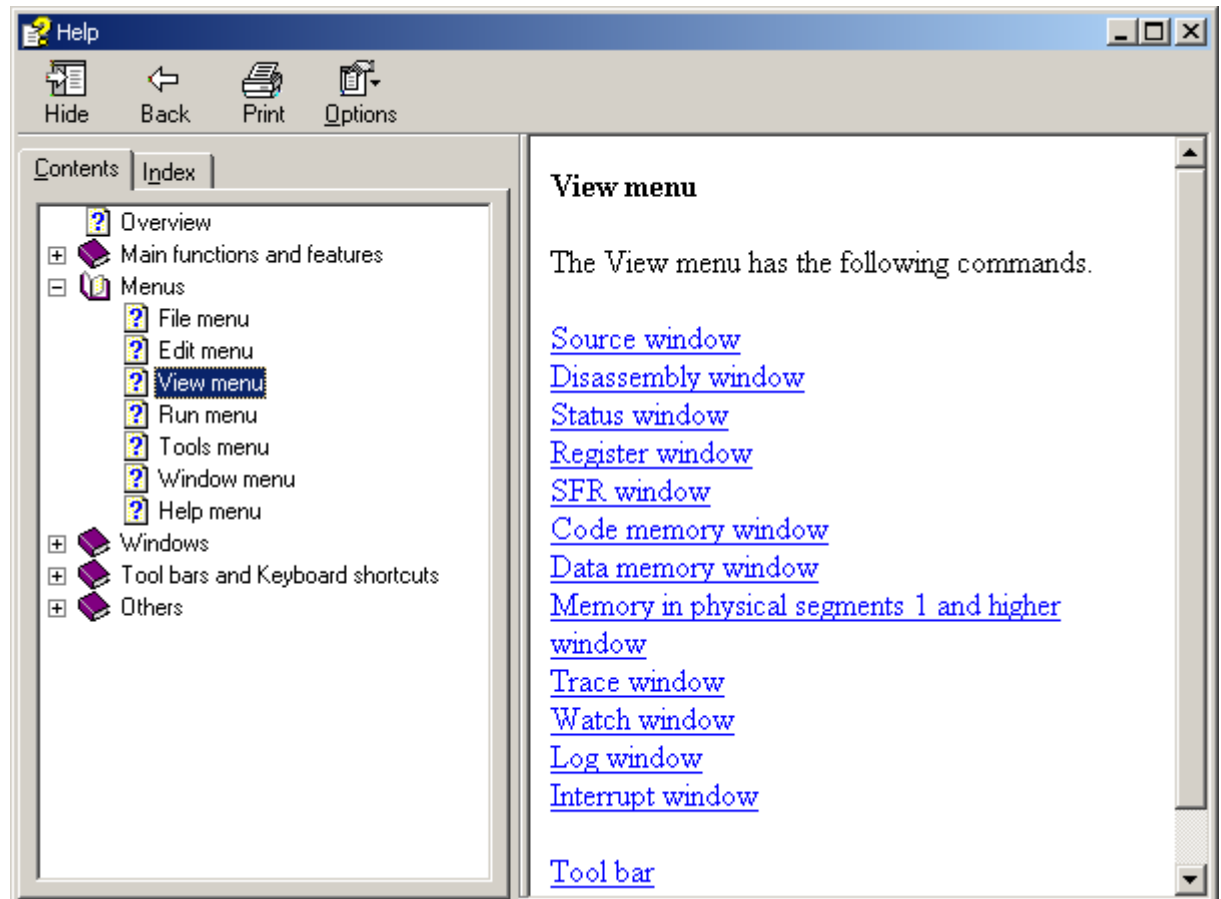
Clicking on the Default button brings the font and color settings made for the currently selected window back to the default settings.

The font information for each window is stored in the appropriate project file of the DTU8 debugger.

## 11.5 On-Line Help

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The debugger provides on-line help in HTML format for all menus, the context menus for each window, and all dialog boxes.

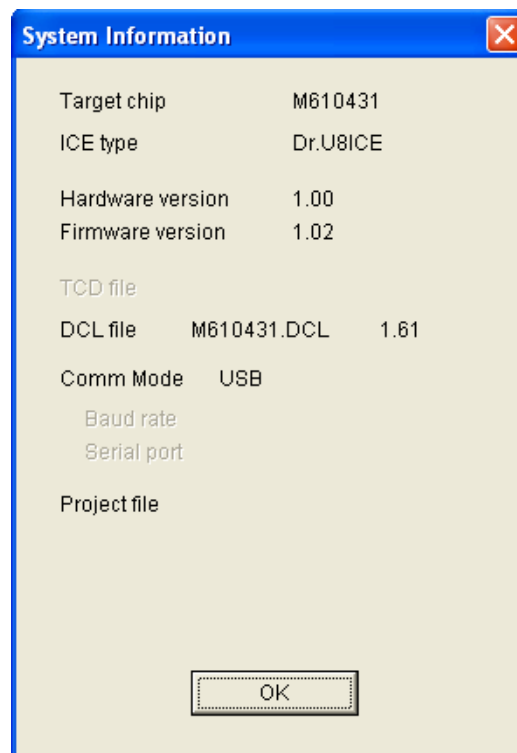


The User's Manual provides a tutorial on using various debugging functions. Refer to the on-line help for a complete reference to menus and windows.

### 11.6 Displaying Version Data

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

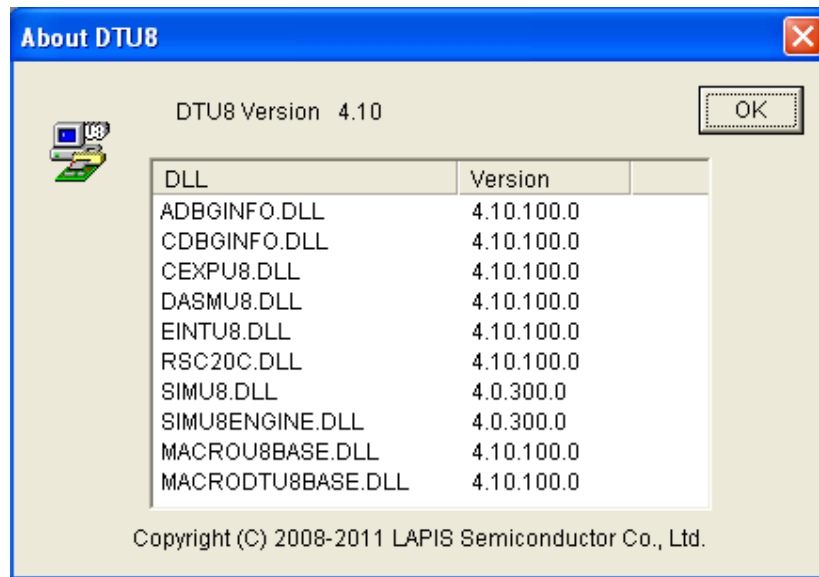
The *System information...* command on the Help menu displays information such as the target chip model number, the type of in-circuit emulator (ICE) (= the mode of debugging), emulator hardware and firmware version numbers, the name and version number of the emulator's target chip data (TCD) file, the name and version number of the debugger configuration list (DCL) file, the communication mode, the baud rate, project file name, and the serial port. The content to be displayed varies depending on the mode of debugging. Shown below is an example of display in Dr.U8 ICE mode.





Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

The About DTU8... command on the Help menu displays version information for the debugger and the DLLs that it uses.

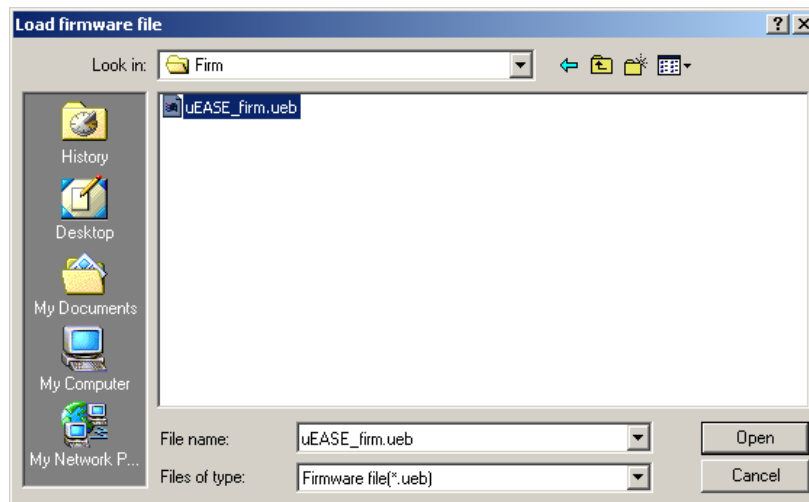


## 11.7 Firmware Update Function

Dr.U8 ICE  
Dr.U16 ICE  
uEASE  
nanoEASE

This function allows the developer to update the firmware of Dr. U8 ICE, Dr.U16 ICE, uEASE or nanoEASE.

The *Update Firmware...* commands on the Help menu displays the following dialog box:



### ■ Important Note on Firmware Update

When Firmware Updating completed, terminates DTU8 debugger.

In the case of Dr.U8 ICE or Dr.U16 ICE, please once turn off ICE, re-switch on a power supply again and reboot.

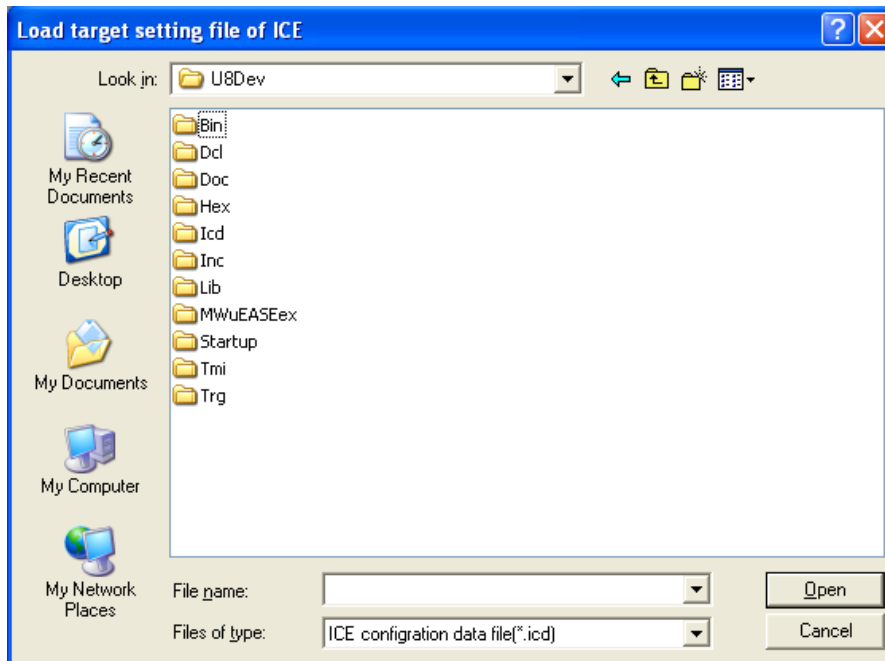
In the case of uEASE or nanoEASE, the USB cable of uEASE, nanoEASE is once pulled out and please re-put it.

## 11. Other Functions

## 11.8 Target Device change Function

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE

The target device of Dr.U8 ICE or Dr.U16 ICE can be changed on the DTU8 debugger. Select [Help] menu → [Change target device of ICE] to get this dialog box.



Select a file for the model conversions and click [Open] button to complete the target change. Please refer to a “Dr.U8 ICE user’s manual” or “Dr.U16 ICE user’s manual” for the further information of a Target Device change function.

## 11.9 Realtime LCD monitor Function

Dr.U8 ICE  
Dr.U16 ICE

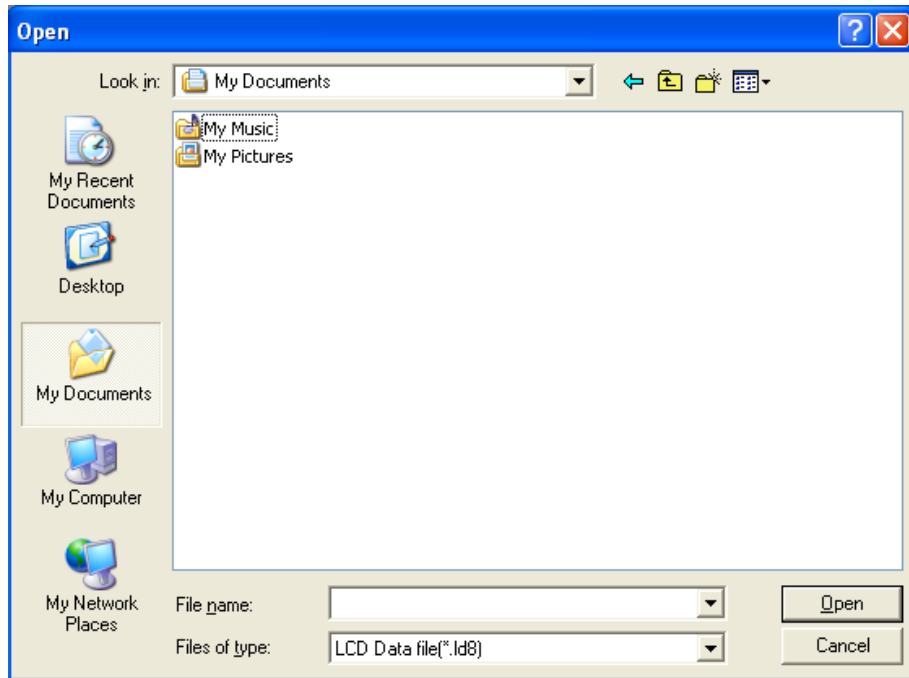
The realtime LCD monitor function can update the LCD display image in realtime, by reading LCD terminal information (COM/SEG) from Dr.U8 ICE and sending them to the LCD image check tool.

### 11.9.1 How to use the Realtime LCD monitor

Before using this function, please define the LCD panel using the LCD image assignment tool, and save the contents of definition into a LCD data file. Please refer to "5. LCD Image assignment tool" of a "LCD Image Tool User's Manual" for more detail about the definition method of the LCD panel.

Select [Tool] menu → [Realtime LCD monitor] on DTU8 to get this dialog.

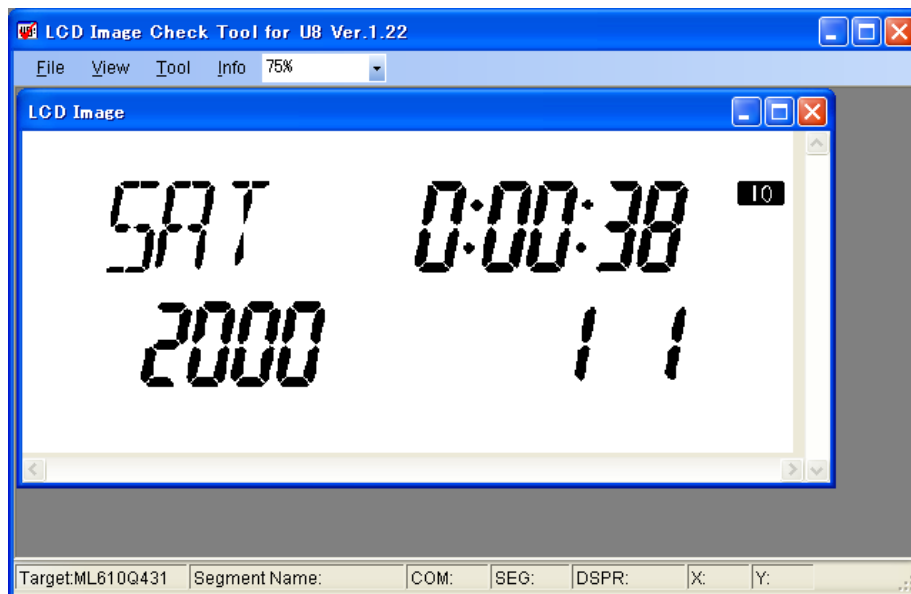
## 11. Other Functions



After selecting a LCD data file and click the “Open” button will start to run the LCD image check tool.

Select “Tool” menu → “Start Realtime LCD Monitor Mode” on the LCD image check tool to start the realtime LCD monitor mode.

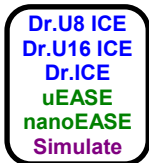
Then, if program codes that control the LCD panel are executed on DTU8, display data on the LCD image check tool will be updated in realtime according to the program code.



# 12. Appendix

---

## 12.1 Symbolic Debugging and Input Formats



### 12.1.1 Symbols

Symbols are names for specific addresses, numeric values, and the like. They are easier to remember than the constants that they represent.

The debugger treats the following strings within expressions as symbols.

The first character must be a letter (a to z or A to Z) or special character (underscore, dollar, or question mark).

Additional characters must be a letter (a to z or A to Z), special character (underscore, dollar, or question mark), or a digit (0 to 9).

The following characters indicate the end of the string: tab (09H), space (20H), carriage return (0DH), operator (+, -, \*, /, %, &, |, ^, !, ., <, >, =, (, or )), or other special character (backslash, left bracket, right bracket, colon, semicolon, at-sign, hash, double quote, or single quote).

#### ■ *Preparing Symbol Table*

Symbolic debugging requires that the object file be created with the RASU8 assembler's /D command line option for including debugging information in the output.

#### ■ *Symbol Value*

This is the value that the symbol has.

#### ■ *Symbol Attribute*

This classifies the value that the symbol has. The debugger recognizes the following attributes.

|        |                                 |
|--------|---------------------------------|
| CODE   | Code address symbol             |
| DATA   | Data address symbol             |
| NVDATA | Nonvolatile data address symbol |
| TABLE  | Table address symbol            |
| BIT    | Bit address symbol              |
| NVBIT  | Nonvolatile bit address symbol  |
| TBIT   | Table bit address symbol        |
| NUMBER | Numeric value symbol            |

## 12. Appendix

### ■ **Symbol Types**

This indicates the type that the symbol had when it was defined: reserved word or user defined symbol. This type plays a role in symbol search order.

Reserved word symbols are defined by the target microcontroller. Their primary use is to assign labels to registers in the SFR address space. The debugger reads their definitions from the corresponding debugger configuration list (DCL) file. These symbols cannot be redefined.

User defined symbols are defined by the programmer in source code files. All symbols defined with such assembler directives as EQU, CODE, DATA, and XDATA and all labels are thus user-defined symbols.

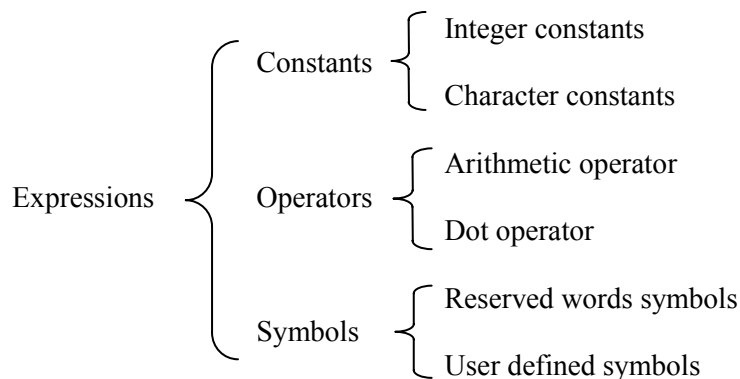
### ■ **Displaying Symbol List**

The *Symbol list...* command on the Tools menu displays a list of all reserved word symbols and user-defined symbols currently defined.

This dialog box includes facilities for adding, deleting, and redefining user-defined symbols.

### 12.1.2 Expressions

Debugger expressions are made up of three element types: constants, operators, and symbols. The following Figure provides complete details.



### ■ **Constants**

Constants directly specify an address or data value. They are divided into integer constants and character constants

### ■ **- Integer Constants**

The debugger treats as integer constants strings starting with a decimal digit (0 to 9) and satisfying the conditions in the following table. There are four bases available: binary, octal, decimal, and hexadecimal. The debugger distinguishes the base from the suffixes (and 0x prefix) listed in the table.

If a hexadecimal constant starts with a nondigit (A to F, a to f, or \_), prefix it with a 0 to distinguish it from a label.

The debugger skips all underscores ( \_ ) in integer constants. It does not distinguish case.

## 12.1 Symbolic Debugging and Input Formats

| Radix       | Permitted characters      | Radix specifier | Example             |
|-------------|---------------------------|-----------------|---------------------|
| Binary      | 0, 1, _                   | B               | 1010B               |
| Octal       | 0 to 7, _                 | O, Q            | 271O, 514Q          |
| Decimal     | 0 to 9, _                 | None            | 1234                |
| Hexadecimal | 0 to 9, A to F, a to f, _ | H, 0x           | 753H, 0C6E7H, 0x332 |

### ■ - Character Constants

A character constant has a value between 00H to 0FFH and consists of a single character or escape sequence between single quotes ('). If the character following the opening single quote is other than a backslash (\), the character constant value is the ASCII code for that character. If the character following the opening single quote is a backslash (\), however, the value is that specified for the corresponding C escape sequence.

### ■ Operators

The debugger uses 32-bit unsigned arithmetic for all operators in expressions. If a result is negative, the debugger replaces it with its twos complement. The debugger ignores any overflow.

Operators are divided into arithmetic operators and the dot operator.

### ■ - Arithmetic Operators

- + binary addition
- binary subtraction
- & binary bitwise AND

### ■ - Dot Operator

. ((left\_operand \* bit\_count) + right\_operand)

This operator constructs a bit address from a data address and a bit position.

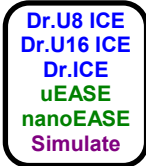
### 12.1.3 Value Lists

The *Change xxx memory* commands on the context menus for each Dump window accept these as input for changing multiple values at once. Each value in the list can be either an immediate value or a symbol. The values in the list can be separated by blanks, commas, or semicolons.

The default radix is 10. Use the H suffix (e.g., 10H) to indicate hexadecimal notation. Precede any hexadecimal values starting with A to F with the prefix 0 (e.g., 0ABH).

Note that value lists cannot use minus signs to specify negative values.

### 12.2 Watch Item Candidates



The following is a list of items eligible for addition to the Watch window.

- C variables
- Assembly symbols
- Program counter (PC)
- Program status word (PSW, EPSW, EPSW1 to EPSW3, C, Z, S, OV, MIE, HC, and ELVL)
- Register direct addressing (R0 to R15, ER0 to ER14, XR0 to XR12, QR0 to QR8, LR, ELR, ELR1 to ELR2, EA, CSR, LCSR, ECR1 to ECR2, DSR, BP, FP, and SP)
- Register bit direct addressing (Rn.bit)
- Data memory direct addressing
- SFR names
- General-purpose register indirect addressing ([ERm])
- FP indirect addressing with 6-bit base (Disp6[FP])
- BP indirect addressing with 6-bit base (Disp6[BP])
- General-purpose register indirect addressing with 16-bit base (Disp16[ERm])
- EA register indirect addressing ([EA])

#### 12.2.1 Adding Local Variables

Adding a local variable to a Watch window requires first moving the program counter (PC) inside the scope for the function defining the local variable.

If multiple blocks use the same name for local variables, the Watch window displays the value for the local variable for the block currently containing the program counter (PC).

Note that the Watch window does not display correct values for such local variables when the program counter (PC) is in the prologue or epilogue portions of the corresponding function.

When an attempt is made to add a local variable assigned to a register to the Watch window, a confirmation dialog box is displayed notifying that the variable has been assigned to a register. This dialog box can be made not to appear by an optional setting in the Environment settings dialog box. See Section 13.2.6, “Variable Assigned to Register,” for the setting method.

#### 12.2.2 Adding Global Variables

Global variables can be added to the Watch window at any time, regardless of the current program counter (PC) position. If multiple global variables of different types share the same name, the largest one is added to the Watch window.

#### 12.2.3 Adding Function Parameters

If an enum type function argument is added to the Watch window, no enumeration symbol is displayed in the Value field.



When an attempt is made to add a function argument assigned to a register to the Watch window, a confirmation dialog box is displayed notifying that the function argument has been assigned to a register. This dialog box can be made not to appear by an optional setting in the Environment settings dialog box. See Section 13.2.6, “Variable Assigned to Register,” for the setting method.

### 12.2.4 Displaying C Variable Types

The Type field in the Watch window sometimes fails to properly display complex type declarations.

Consider, for example, the following declarations.

```
extern void f1(void) ;
extern void f2(void) ;
extern void f3(void) ;
const void (*const func_ptr[3])(void) = {f1, f2, f3} ;
```

The Watch window Type field gives the following as the type for the pointer to a function func\_ptr.

```
void([3]*)()
```

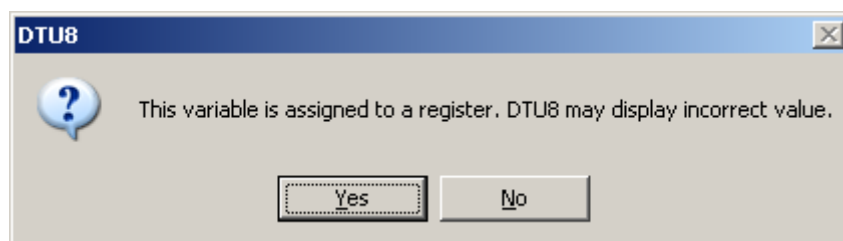
### 12.2.5 Mixing C and ASM

The Watch window does not support arrays allocated in assembler language source code and then declared extern in the C source code without array specifications. The extern declaration must explicitly supply the array dimensions.

### 12.2.6 Variable Assigned to Register

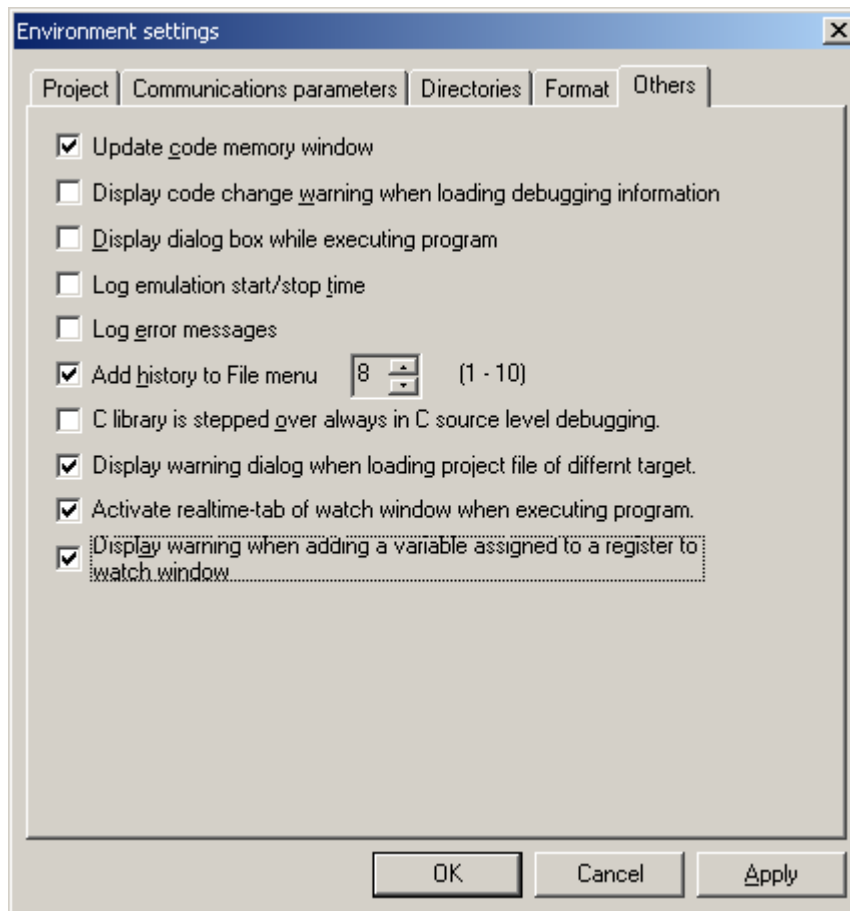
When a local variable or a function argument is assigned to a register, it is possible that the assigned register will temporarily be used for a different purpose and a value different from an expected one will be displayed on the Watch window.

Therefore, if an attempt is made to add a local variable or function argument that has been assigned to the Watch window, the DTU8 debugger displays the following confirmation dialog box:



This dialog box can be made not to appear by an optional setting in the Environment settings dialog box.

## 12. Appendix



Deselect the Display warning when adding a variable assigned to a register to watch window check box on the Others tab in the Environment settings dialog box, and the above mentioned confirmation dialog box will not appear.

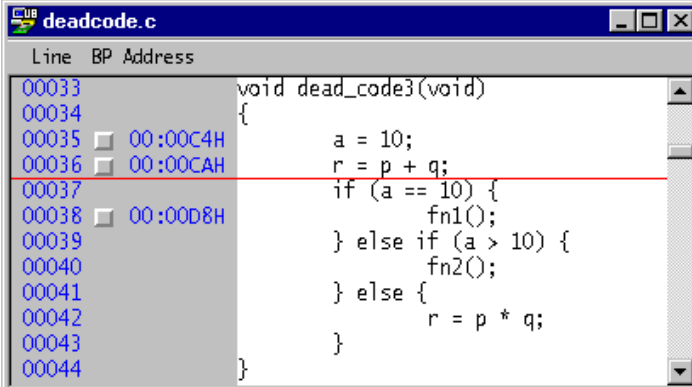
## 12.3 Debugging Optimized C Source Code

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Attempting to debug C source code that has been optimized by the compiler sometimes produces a Source window execution order with no apparent connection to the original C source code. Executable source lines have no addresses, making it impossible to set breakpoints, because the compiler optimization phase has folded code, eliminated redundant code, moved code, etc.

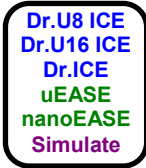
### ■ Example

The following is an example where optimization deletes code.



| Line  | BP                       | Address  | Code                  |
|-------|--------------------------|----------|-----------------------|
| 00033 |                          |          | void dead_code3(void) |
| 00034 |                          |          | {                     |
| 00035 | <input type="checkbox"/> | 00:00C4H | a = 10;               |
| 00036 | <input type="checkbox"/> | 00:00CAH | r = p + q;            |
| 00037 |                          |          | if (a == 10) {        |
| 00038 | <input type="checkbox"/> | 00:00D8H | fn1();                |
| 00039 |                          |          | } else if (a > 10) {  |
| 00040 |                          |          | fn2();                |
| 00041 |                          |          | } else {              |
| 00042 |                          |          | r = p * q;            |
| 00043 |                          |          | }                     |
| 00044 |                          |          | }                     |

Lines 39 to 43 have no BP buttons or addresses because the optimization phase determines that the code that should appear is dead code that is never executed.



## 12.4 Macro Script References

### 12.4.1 Script Commands

#### ■ *Script commands for projects*

|       |                   |
|-------|-------------------|
| PJLOD | Load project file |
| PJSAV | Save project file |

#### ■ *Script commands for referencing/changing CPU contents*

|     |  |
|-----|--|
| D   | Display contents of SFR or register    |
| C   | Change contents of SFR or register     |
| DPC | Display contents of PC                 |
| CPC | Change contents of PC                  |
| OSC | Display/change OSC clock supply source |
| XT  | Display/change XT clock supply source  |

#### ■ *Script commands for referencing/changing C variables*

|       |                                |
|-------|--------------------------------|
| WATCH | Display contents of C variable |
|-------|--------------------------------|

#### ■ *Script commands for code memory*

|          |   |
|----------|---|
| DCM      | Display contents of code memory                                 |
| CCM      | Change contents of code memory                                  |
| MCM      | MCM Copy contents of code memory range to the specified address |
| LOD      | Load program file   |
| SAV      | Save program file   |
| VER      | Compare program file and program code in code memory            |
| ASM      | Assemble specified string                                       |
| STARTASM | Start continuous assembly block                                 |
| ENDASM   | End continuous assembly block                                   |
| DASM     | Disassemble specified address range in code memory              |
| MMAP     | Display/change code memory mapping                              |

#### ■ *Script commands for data memory*

|      |                                    |
|------|------------------------------------|
| DDM  | Display contents of data memory    |
| CDM  | Change contents of data memory     |
| MDM  | Copy contents of data memory range |
| DLOD | Load data memory file              |

|      |                                    |
|------|------------------------------------|
| DSAV | Save data memory file              |
| PMAP | Display/change data memory mapping |

■ **Script commands for memory in physical segments 1 and higher**

|      |   |
|------|---|
| DGM  | Display contents of memory in physical segments 1 and higher        |
| CGM  | Change contents of memory in physical segments 1 and higher         |
| MGM  | Copy contents of memory range in physical segments 1 and higher     |
| GMAP | Display/change mapping for memory in physical segments 1 and higher |

■ **Script commands for emulation**

|       |                 |
|-------|-----------------|
| G     | Execute program |
| STP   | Step in         |
| STPOV | Step over       |

■ **Script commands for resetting**

|      |   |
|------|---|
| RST  | Reset entire emulator                     |
| SRC  | Display/set/clear reset items             |
| URST | Enable/disable user cable RESET pin input |

■ **Script commands for breaks**

|     |                                    |
|-----|------------------------------------|
| SBC | Display/set/clear break conditions |
| DBP | Display breakpoints                |
| SBP | Set software breakpoint            |
| RBP | Remove breakpoint                  |
| DBS | Display break status (source)      |

■ **Script commands for tracing**

|       |   |
|-------|---|
| DTM   | Display contents of trace memory        |
| DTP   | Display contents of trace pointer       |
| RTP   | Clear trace memory                      |
| TRSAV | Save trace memory contents to disk file |
| S     | Search trace memory                     |

■ **Script commands for performance and coverage**

|      |                                  |
|------|----------------------------------|
| TIME | Display timer value              |
| CCC  | Change contents of cycle counter |
| DCC  | Display cycle counter            |
| DCV  | Display coverage                 |
| SCV  | Set coverage address range       |

## 12. Appendix

### ■ **Script commands for macros**

|       |   |
|-------|---|
| PAUSE | Suspend macro execution                                       |
| ECHO  | Send specified string to log                                  |
| GOTO  | Jump to macro line with specified label                       |
| IF    | Perform conditional branch to macro line with specified label |

### ■ **Script commands for symbols**

|      |                         |
|------|-------------------------|
| DSYM | Display symbol value    |
| CSYM | Change symbol value     |
| RSYM | Remove symbol           |
| SSYM | Add user defined symbol |

### ■ **Script commands for self-diagnostics**

|          |                      |
|----------|----------------------|
| SELFTEST | Run self-diagnostics |
|----------|----------------------|

### ■ **Miscellaneous script commands**

|         |   |
|---------|---|
| VERSION | Display debugger version information    |
| EXIT    | Terminate debugger execution            |
| SAVELOG | Save results of macro execution to file |
| CLRLOG  | Clear Log window                        |

## 12.4.2 Script Commands for Projects

Load debugger settings from disk file

**Syntax:** Load debugger settings from disk file  
PJLOD *filename*  
*filename:* Name of project file

Save debugger settings to disk file

**Syntax:** PJSAV filename  
PJSAV *filename*  
*filename:* Name of project file

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

## 12.4.3 Script Commands for Referencing/Changing CPU Contents

Display contents of SFR or register

**Syntax:**

Display contents of SFR or register

*D parameter**parameter:*

reg\_mnemonic or SFR

*Reg\_mnemonic:*

PSW

Processor status word

EPSW

PSW PUSH register

EPSW1 to EPSW3

PSW PUSH registers 1 to 3

LR

Link register

ELR

Link register

ELR1 to ELR3

Link registers 1 to 3

CSR

Code segment register

LCSR

LCSR register

ECSR1 to 3

CSR PUSH registers 1 to 3

DSR

Data segment register

EA

EA register

R0 to R15

Byte-sized registers 0 to 15

ER0 to ER14

Word-sized registers 0 to 14

XR0 to XR12

Long word-sized registers 0 to 12

QR0 to QR8

Quad word-sized registers 0 to 8

CR0 to CR15

Byte-sized coprocessor registers 0 to 15

CER0 to CER14

Word-sized coprocessor registers 0 to 14

CXR0 to CXR12

Long word-sized coprocessor registers 0 to 12

CQR0 and CQR8

Quad word-sized coprocessor registers 0 and 8

SP

Stack pointer

Change contents of SFR or register

**Syntax 1:**

Change contents of register

*C Reg\_mnemonic = data**data:*

New value

*Reg\_mnemonic:*

PSW

Processor status word

EPSW

PSW PUSH register

EPSW1 to EPSW3

PSW PUSH registers 1 to 3

LR

Link register

ELR

Link register

ELR1 to ELR3

Link registers 1 to 3

CSR

Code segment register

LCSR

LCSR register

ECSR1 to 3

CSR PUSH registers 1 to 3

DSR

Data segment register

EA

EA register

R0 to R15

Byte-sized registers 0 to 15

ER0 to ER14

Word-sized registers 0 to 14

XR0 to XR12

Long word-sized registers 0 to 12

QR0 to QR8

Quad word-sized registers 0 to 8

CR0 to CR15

Byte-sized coprocessor registers 0 to 15

CER0 to CER14

Word-sized coprocessor registers 0 to 14

CXR0 to CXR12

Long word-sized coprocessor registers 0 to 12

CQR0 and CQR8

Quad word-sized coprocessor registers 0 and 8

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

## 12. Appendix

### Syntax 2: Change SFR contents

*C Sfr\_mnemonic = data*  
*Sfr\_mnemonic:*  
*data:*

Register assigned to SFR address space  
New value

#### Display contents of PC

##### Syntax: Display contents of PC

DPC

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

#### Change contents of PC

##### Syntax: Change contents of PC

CPC *address*

*address:*

New code address

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

## 12.4.4 Script Commands for Referencing/Changing C Variables

#### Display contents of C variable

##### Syntax: Display contents of C variable

WATCH *valueable\_name*

*valueable\_name:*

Name of C variable

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

## 12.4.5 Script Commands for Code Memory

#### Display contents of code memory

##### Syntax 1: Display contents of single address in code memory

DCM *address*

*address:*

Address in code memory

##### Syntax 2: Display contents of address range in code memory

DCM [*start-address end-address*]

*start-address:*

Starting address in code memory

*end-address:*

End address in code memory

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

#### Change contents of code memory

##### Syntax 1: Change contents of single address in code memory

CCM *address = data*

*address:*

Address in code memory

*data:*

New value

##### Syntax 2: Change contents of address range in code memory

CCM [*start-address end-address*] = *data*

*start-address:*

Starting address in code memory

*end-address:*

End address in code memory

*data:*

New value

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate



Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Copy contents of code memory range to the specified address

**Syntax:** Copy code memory

MCM [*start-address end-address*] *trans-address*

|                       |   |
|-----------------------|---|
| <i>start-address:</i> | Source starting address in code memory      |
| <i>end-address:</i>   | Source end address in code memory           |
| <i>trans-address:</i> | Destination starting address in code memory |

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Load program from disk file into code memory

**Syntax:** Load program from disk file into code memory

LOD *filename* [*option ... option*]

|                  |                      |                                     |
|------------------|----------------------|-------------------------------------|
| <i>filename:</i> | Name of file to load |                                     |
| <i>option:</i>   | /NS                  | Do not load debugging information   |
|                  | /N                   | Do not load code data               |
|                  | /B                   | Clear all breakpoints after loading |
|                  | /R                   | Reset after loading                 |

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Save code memory contents to disk file

**Syntax:** Save code memory contents to disk file

SAV *filename format* [ [*start-address end-address*] ] [*option ... option*]

|                  |                       |                           |
|------------------|-----------------------|---------------------------|
| <i>filename:</i> | Name for saved file   |                           |
| <i>format:</i>   | HEX                   | Save in Intel HEX format  |
|                  | S                     | Save in Motorola S format |
|                  | <i>start-address:</i> | Starting address          |
|                  | <i>end-address:</i>   | End address               |
| <i>option:</i>   | /NS                   | Do not save symbol table  |

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Compare program file and program code in code memory

**Syntax 1:** Compare program code for all addresses in program file

VER *filename*  
*filename:* Name of file to compare

**Syntax 2:** Compare program code for specified address range in program file

VER *filename* [*start-address end-address*]  
*filename:* Name of file to compare

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Assemble specified string

**Syntax 1:** Assemble single instruction at single address

ASM *address* "*assemble-string*"  
*address:* Address in code memory

**Syntax 2:** Assemble specified string at next address within continuous assembly block

ASM "*assemble-string*"

Note: This version is for use between a STARTASM command specifying the starting address and an ENDASM command ending the block.

## 12. Appendix

### Start continuous assembly block

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

**Syntax:** Start continuous assembly block

STARTASM *address*

*address:* Address in code memory

### End continuous assembly block

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

**Syntax:** End continuous assembly block

ENDASM

### Disassemble specified address range in code memory

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

**Syntax 1:** Disassemble program code at single address

DASM *address* [*option ... option*]

*address:* Address in code memory

*option:* /NC Do not display instruction code

/NL Do not display address

**Syntax 2:** Disassemble program code for address range

DASM [*start-address end-address*] [*option ... option*]

*start-address:* Starting address

*end-address:* End address

*option:* /NC Do not display instruction code

/NL Do not display address

### Display/change code memory mapping

Dr.ICE  
Simulate

**Syntax 1:** Display current code memory mapping

MMAP

**Syntax 2:** Change code memory mapping

MMAP [*start-address end-address*] = *parm*

*start-address:* Starting address

*end-address:* End address

*parameter:* SC System memory

UC User memory

N Not used

## 12.4.6 Script Commands for Data Memory

### Display contents of data memory

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

**Syntax 1:** Display contents of single address in data memory

DDM *address*

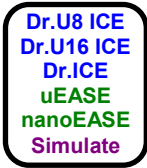
*address:* Address in data memory

**Syntax 2:** Display contents of address range in data memory

DDM [*start-address end-address*]

*start-address:* Starting address in data memory

*end-address:* End address in data memory



## Change contents of data memory

**Syntax 1:** Change contents of single address in data memory

CDM *address* = *data*

*address:*

Address in data memory

*data:*

New value

**Syntax 2:** Change contents of address range in data memory

CDM [*start-address end-address*] = *data*

*start-address:*

Starting address in data memory

*end-address:*

End address in data memory

*data:*

New value

Note: This command does not support addresses in the SFR region.



## Copy contents of data memory range to the specified address

**Syntax:** Copy data memory range

MDM [*start-address end-address*] *trans-address*

*start-address:*

Source starting address in data memory

*end-address:*

Source end address in data memory

*trans-address:*

Destination starting address in data memory

Note: This command does not support addresses in the SFR region.



## Load data from disk file into data memory

**Syntax:** Load data from disk file into data memory

DLOD *filename*

*filename:*

Name of file to load



## Save data memory contents to disk file

**Syntax:** Save data memory contents to disk file

DSAV *filename* [ [*start-address end-address*] ]

*filename:*

Name for saved file

*start-address:*

Starting address

*end-address:*

End address

Note: This command does not support addresses in the SFR region.



## Display/change data memory mapping

**Syntax 1:** Display current data memory mapping

PMAP

**Syntax 2:** Change data memory mapping

PMAP [*start-address end-address*] = *parm*

*start-address:*

Starting address

*end-address:*

End address

*parameter:*

S

System memory

U

User memory

N

Not used

## 12. Appendix



### Save SFR contents to disk file

**Syntax:** Save SFR contents to disk file

SFRSAV *filename* [ [*start-address end-address*] ]

*filename:* Name for saved file  
*start-address:* Starting address  
*end-address:* End address

Note: This command supports specified addresses in the TRG file.

## 12.4.7 Script Commands for Memory in Physical Segments 1 and Higher



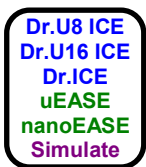
### Display contents of memory in physical segments 1 and higher

**Syntax 1:** Display contents of single address in memory in physical segments 1 and higher

DGM *address*  
*address:* Address in memory in physical segments 1 and higher

**Syntax 2:** Display contents of address range in memory in physical segments 1 and higher

DGM [*start-address end-address*]  
*start-address:* Starting address in memory in physical segments 1 and higher  
*end-address:* End address in memory in physical segments 1 and higher



### Change contents of memory in physical segments 1 and higher

**Syntax 1:** Change contents of single address in memory in physical segments 1 and higher

CGM *address* = *data*  
*address:* Address in memory in physical segments 1 and higher  
*data:* New value

**x**and higher

CGM [*start-address end-address*] = *data*  
*start-address:* Starting address in memory in physical segments 1 and higher  
*end-address:* End address in memory in physical segments 1 and higher  
*data:* New value

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Copy contents of memory range in physical segments 1 and higher range to the specified address

**Syntax:** Copy memory range in physical segments 1 and higher

MGM [*start-address end-address*] *trans-address*

*start-address:* Source starting address in memory in physical segments 1 and higher  
*end-address:* Source end address in memory in physical segments 1 and higher  
*trans-address:* Destination starting address in memory in physical segments 1 and higher

Dr.ICE  
Simulate

Display/change memory in physical segments 1 and higher mapping

**Syntax 1:** Display current memory mapping for physical segments 1 and higher

GMAP

**Syntax 2:** Change memory mapping for physical segments 1 and higher

GMAP [*start-address end-address*] = *parm*

*start-address:* Starting address  
*end-address:* End address  
*parameter:* SC System code  
SD System data  
SCD System code/data  
UC User code  
UD User data  
UCD User code/data  
N Not used region

### 12.4.8 Script Commands for Emulation

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Execute program

**Syntax 1:** Execute program

G

G *address*

**Syntax 2:** Execute program up to break at single address

G [*address*] , *break-address*

*address:* Starting address for user program  
*break-address:* Break address

**Syntax 3:** Execute program with address pass count break

G [*address*] , *break-address* [*count*]

*address:* Starting address for user program  
*break-address:* Break address  
*count:* Break count

**Syntax 4:** Execute program with RAM address access break

G [*address*] , RAM *ram-address* [ [*count*] ]

*ram-address:* RAM address  
*count:* Break count

## 12. Appendix

**Syntax 5:** Execute program with RAM data match break

G [address] , RDAT [AND mask\_data] condition data[[count]]  
mask\_data: Mask data  
condition: = Count matches  
              ! Count mismatches  
data: Data for comparison  
count: Break count

**Syntax 6:** Execute program with RAM address data match break

G [address] , RADR ram-address[AND mask\_address] condition data [ [count] ]  
address: Starting address for user program  
ram-address: RAM address  
mask\_address: Mask address  
condition: = Count matches  
              ! Count mismatches  
data: Data for comparison  
count: Break count

### Step over

**Syntax:** Step over

STPOV [address][ , count ]  
address: Starting address for step execution  
count: Number of steps to execute

### Step in

**Syntax:** Step in

STP [address][ , count ]  
address: Starting address for step execution  
count: Number of steps to execute

## 12.4.9 Script Commands for Resetting

### Reset entire emulator

**Syntax:** Reset entire emulator

RST

## 12.4.10 Script Commands for Breaks

### Display/set/clear break conditions

**Syntax 1:** Display current settings

SBC

**Syntax 2:** Enable trace full break

SBC TF *overflow*  
overflow: Trace overflow point

**Syntax 3:** Enable external break condition

SBC XP *condition*  
*condition:* DOWN Break at falling edge  
 UP Break at rising edge

**Syntax 4:** Enable power down break condition

SBC PD

**Syntax 5:** Disable break condition

SBC ~*mnemonic*  
*mnemonic:* TF Trace full break  
 XP External break  
 PD Power down break  
 \* All breaks

## Display breakpoints

**Syntax:** Display breakpoints

DBP

Dr.U8 ICE  
 Dr.U16 ICE  
 Dr.ICE  
 uEASE  
 nanoEASE  
 Simulate

## Set breakpoint

**Syntax:** Set software breakpoint

SBP *address*  
*address:* Address for breakpoint

Dr.U8 ICE  
 Dr.U16 ICE  
 Dr.ICE  
 uEASE  
 nanoEASE  
 Simulate

## Remove breakpoint

**Syntax:** Remove breakpoint

RBP *bp-address*  
*bp-address:* *address* Address with breakpoint  
 \* All breakpoints

Dr.U8 ICE  
 Dr.U16 ICE  
 Dr.ICE  
 uEASE  
 nanoEASE  
 Simulate

## Display break status (source)

**Syntax:** Display break status (source)

DBS

Dr.U8 ICE  
 Dr.U16 ICE  
 Dr.ICE  
 uEASE  
 nanoEASE  
 Simulate

## 12.4.11 Script Commands for Tracing

## Display contents of trace memory

**Syntax 1:** Display contents of entire trace memory

DTM

**Syntax 2:** Display specified number of oldest entries

DTM *step\_number*  
*step\_number:* Number of entries to display

Dr.U8 ICE  
 Dr.U16 ICE  
 Dr.ICE  
 Simulate

## 12. Appendix

**Syntax 3:** Display specified number of newest entries

DTM BACK *step\_number*  
*step\_number:*

Number of entries to display

Display contents of trace pointer

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

**Syntax:** Display contents of trace pointer

DTP

Clear trace memory

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

**Syntax:** Clear trace memory

RTP

Save trace memory contents to disk file

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

**Syntax 1:** Save entire trace memory contents to disk file

TRSAV *filename*  
*filename:*

Name for saved file

**Syntax 2:** Save specified trace memory range to disk file

TRSAV *filename* [*start* *end*]  
*filename:*  
*start:*  
*end:*

Name for saved file  
Starting position  
End position

### 12.4.12 Script Commands for Performance and Coverage

Display timer value

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE

**Syntax:** Display timer value

TIME

Display cycle counter

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

**Syntax:** Display cycle counter

DCC

Change contents of cycle counter

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
Simulate

**Syntax:** Change contents of cycle counter

CCC *count*  
*count:*

Cycle count

Display coverage

Dr.U16 ICE  
Simulate

**Syntax:** Display coverage

DCV

Set coverage address range

Dr.U16 ICE  
Simulate

**Syntax1:** Display coverage address range

SCV



**Syntax2:** Set coverage address range

SCV [*start\_address end\_address*]  
     *start\_address*: coverage start address  
     *end\_address*: coverage end address

### 12.4.13 Script Commands for Macros

Suspend macro execution until a key other than space is pressed

**Syntax:** Suspend macro

PAUSE

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Send specified string to log

**Syntax:** Send specified string to log

ECHO                    *statement*  
     *statement*:  
     *parameter*:  
                           \$TIME                    Current time  
                           \$DATE                    Current date

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Jump to macro line with specified label

**Syntax:** Jump to macro line with specified label

GOTO                    *label*  
     :*label*

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

Perform conditional branch to macro line with specified label

**Syntax:** Perform conditional branch to macro line with specified label

IF                    *compare statement*  
     *compare*:  
                           *value-comp*  
                           *lastline-comp*  
     *value-comp*:  
     *obj*:  
                           PSW, EPSW, EPSW1, EPSW2, EPSW3, LR, ELR, ELR1, ELR2,  
                           ELR3, CSR, LCSR, ECSR1, ECSR2, ECSR3, DSR, EA, R0, R1, R2,  
                           R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, ER0, ER2,  
                           ER4, ER6, ER8, ER10, ER12, ER14, XR0, XR4, XR8, XR12, QR0,  
                           QR8, CR0, CR1, CR2, CR3, CR4, CR5, CR6, CR7, CR8, CR9, CR10,  
                           CR11, CR12, CR13, CR14, CR15, CER0, CER2, CER4, CER6, CER8,  
                           CER10, CER12, CER14, CXR0, CXR4, CXR8, CXR12, CQR0, CQR8,  
                           SP, PC, SFR sfr name, data-address, CVAR C variable name  
     *cond*:  
                           ==, !=  
     *value*:  
                           expression  
     *data-address*:  
                           Data address expression  
     *lastline-comp*:  
                           LASTLINE cond "string"  
                                   String comparison with entire preceding line  
                           LASTLINE(n) cond "string"  
                                   String comparison with nth word on preceding line  
     *statement*:  
                           THEN command  
                           GOTO label

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE  
uEASE  
nanoEASE  
Simulate

## 12. Appendix

*command:* any script command

### 12.4.14 Script Commands for Symbols

Display symbol value

**Syntax:** Display symbol value

DSYM *symbol*  
*symbol:* Symbol name (may include wild cards)

Change symbol value

**Syntax:** Change symbol value

CSYM *symbol* = *data*  
*symbol:* Symbol name  
*data:* New value

Remove symbol

**Syntax:** Remove symbol

RSYM *symbol*  
*symbol:* Symbol name (may include wild cards)

Add user defined symbol

**Syntax:** Add user defined symbol

SSYM *symbol* = *data* [*attribute*]  
*symbol:* Symbol name  
*data:* Symbol value  
*attribute:* Symbol attribute

|     |           |
|-----|-----------|
| /N  | NUMBER    |
| /C  | CODE      |
| /D  | DATA      |
| /T  | TABLE     |
| /ND | NVDATA    |
| /B  | BIT DATA  |
| /TB | TABLE BIT |
| /NB | NVBIT     |

### 12.4.15 Other Script Commands

Display debugger version information

**Syntax:** Display debugger version information

VERSION

Terminate debugger execution

**Syntax:** Terminate debugger execution

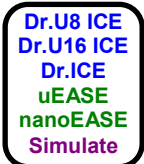
EXIT



Save results of macro execution to file

**Syntax:** Save results of macro execution to file

SAVELOG *logname*  
logname: Name for saved file

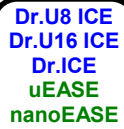


Clear Log window

**Syntax:** Clear Log window

CLRLOG

## 12.5 Errors in Emulator Connection



If an error code is received while the DTU8 debugger is operating with the emulator (Dr.U8 ICE, Dr.U16 ICE, Dr.ICE in-circuit emulator or uEASE, nanoEASE on-chip debug emulator) connected, the DTU8 debugger displays the error message corresponding to that error code.

The following subsections describe the error messages to be displayed when Dr.U8 ICE/uEASE is connected and how to handle the corresponding errors.

### 12.5.1 Errors Common to When Dr.U8 ICE / Dr.U16 ICE / Dr.ICE Connected and When uEASE / nanoEASE Connected

The table below lists the error messages concerning errors for which the error contents or action on error is the same irrespective of the type of emulator out of the errors that can occur when the emulator Dr.U8 ICE, Dr.U16 ICE, Dr.ICE or uEASE, nanoEASE is connected.

| Error message                          | Description / Action on error   |
|--|---|
| USB : Failure to open the device       | One of the errors on the left is displayed if USB communication fails.<br><br>Check the USB cable and check the connection.   |
| USB : Cannot close device              |   |
| Cannot open EP $n$                     |   |
| USB : Cannot close pipe                |   |
| USB : Failure to bulk out              |   |
| USB : Failure to bulk in               |   |
| Illegal data received                  | DTU8 or the emulator being connected may be malfunctioning.<br><br>Check the connection with the emulator and reactivate DTU8 or the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs. |
| ICE error: 6000H<br>Illegal data type. | DTU8 or the emulator may be malfunctioning.<br><br>Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.              |

## 12.5 Errors in Emulator Connection

| Error message  | Description / Action on error  |
|--|--|
| ICE error: 6001H<br>Invalid address range specified. | DTU8 or the emulator may be malfunctioning.<br>Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |
| ICE error: 6002H<br>Resource number not found.       | DTU8 or the emulator may be malfunctioning.<br>Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |
| ICE error: 6003H<br>Too many data.                   | DTU8 or the emulator may be malfunctioning. This error may be displayed even when a USB cable is connected or disconnected while DTU8 is operating.<br>Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs. |
| ICE error: 6005H<br>Invalid compare data.            | DTU8 or the emulator may be malfunctioning.<br>Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |
| ICE error: 6006H<br>Break number not found.          | DTU8 or the emulator may be malfunctioning.<br>Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |

## 12. Appendix

| Error message   | Description / Action on error   |
|---|---|
| ICE error: 6011H<br>Number of the hardware breakpoint exceeds the limit..   | It is displayed when it is going to set up a hardware break point exceeding the number which can be set up. DTU8 or an emulator may be malfunctioning. Please check connection of an emulator and reboot DTU8 and an emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs. |
| ICE error: 6200H<br>Emulation busy.   | The specified command cannot be executed during emulation. Specify the command with the program execution stopped with a break.   |
| ICE error: 6201H<br>Cannot execute this command except that in emulation/step mode.   | DTU8 or the emulator may be malfunctioning. Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |
| ICE error: 6300H<br>Invalid direct instruction.   | DTU8 or the emulator may be malfunctioning. Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |
| ICE error: 630AH<br>No setting ICE information.   |   |
| ICE error: 630CH<br>Error reading information of target LSI.  |   |
| ICE error: 630DH<br>Setting of the firmware update mode was a failure. Check the connection, and then restart DTU8 and uEASE. |   |
| ICE error: 6FFAH<br>Unexpected error was received from ICE.   | DTU8 or the emulator may be malfunctioning. Check the connection with the emulator, then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |

## 12.5.2 Errors in uEASE / nanoEASE Connection

uEASE  
nanoEASE

The table below lists the error messages that are displayed when connected to uEASE or nanoEASE. Even in cases where the same error message as when connected to Dr.U8 ICE is displayed, the error messages in such cases are listed here if the action on error is different.

| Error message   | Description / Action on error   |
|---|---|
| ICE error: 0001H<br>The target system was disconnected.<br>Please restart DTU8 and the target system. | The target system was disconnected.<br>Check the connection with uEASE and target system, and then restart DTU8, uEASE and the target system.   |
| ICE error: 6100H<br>Mismatch in compare result at memory writing.                                     | The flash memory on the target microcontroller being connected to uEASE may be damaged. This error may be displayed even when a USB cable is connected or disconnected while DTU8 is operating.<br>Check the connection with uEASE, reactivate DTU8 and uEASE, and then write data again. If this error still occurs, replace the target microcontroller. |
| ICE error: 6302H<br>Error resetting.  | Check the connection between uEASE and the user target board, then reactivate DTU8, uEASE, and the user target board. If this error recurs even after reactivation, replace the target microcontroller.   |
| ICE error: 6303H<br>Error accessing to FLASH memory.  | Check the connection between uEASE and the user target board, then reactivate DTU8, uEASE, and the user target board. If this error recurs even after reactivation, replace the target microcontroller.   |
| ICE error: 6304H<br>Timeout error.  | It is possible that there is a problem in the connection between uEASE and the user target board. Check the connection between uEASE and the user target board, then reactivate them.   |
| ICE error: 6305H<br>VTref of target LSI is too low to control Flash memory.                           | The power supply voltage of the target microcontroller (VTref) falls below the voltage required to control the Flash memory. Check the power supply.  |
| ICE error: 6306H<br>Specified block address out of valid range.                                       | DTU8 or uEASE may be malfunctioning.<br>Check the connection with uEASE, then reactivate DTU8 and uEASE. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.   |

## 12. Appendix

| Error message   | Description / Action on error   |
|---|---|
| ICE error: 6307H<br>Failed to start On-Chip ICE.  | Check the connection between uEASE and the user target board, then reactivate DTU8, uEASE, and the user target board. If this error recurs even after reactivation, replace the target microcontroller.                                       |
| ICE error: 6308H<br>VDDL voltage supplied from uEASE is out of valid range.   | uEASE may be malfunctioning.<br>Check the connection with uEASE, then reactivate DTU8 and uEASE. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.         |
| ICE error: 6309H<br>VTref voltage of target is abnormal.<br>Please restart DTU8 and the target system, after checking the power supply of the target. | The power supply voltage of the target (VTref) is outside the normal range. Check the power supply of the target, then reactivate DTU8 and the target board.  |
| ICE error: 630BH<br>Only during emulation.  | DTU8 or uEASE may be malfunctioning.<br>Check the connection with uEASE, then reactivate DTU8 and uEASE. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs. |



## 12.5.3 Errors in Dr.U8 ICE / Dr.U16 ICE / Dr.ICE Connection

Dr.U8 ICE  
Dr.U16 ICE  
Dr.ICE

The table below lists the error messages that are displayed when connected to Dr.U8 ICE, Dr.U16 ICE or Dr.ICE. Even in cases where the same error message as when connected to uEASE is displayed, the error messages in such cases are listed here if the action on error is different.

| Error message   | Description / Action on error   |
|---|---|
| ICE error: 6100H<br>Mismatch in compare result at memory writing. | DTU8 or the Dr.U8 ICE in-circuit emulator being connected may be malfunctioning.<br>Check the connection, and then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs.                     |
| ICE error: 6302H<br>Error resetting.                              |   |
| ICE error: 6304H<br>Timeout error.                                |   |
| ICE error: 6FFFH<br>Undefined function.                           | DTU8 or the Dr.U8 ICE, Dr.U16 ICE, Dr.ICE in-circuit emulator being connected may be malfunctioning.<br>Check the connection, and then reactivate DTU8 and the emulator. If this error recurs even after reactivation, contact your nearest LAPIS Semiconductor sales office and report how the error occurs. |