

To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
 - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
 - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
 - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

M32C FoUSB/UART Debugger V.1.02

User's Manual

Renesas Microcomputer Development
Environment System

Active X, Microsoft, MS-DOS, Visual Basic, Visual C++, Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

IBM and AT are registered trademarks of International Business Machines Corporation.

Intel and Pentium are registered trademarks of Intel Corporation.

Adobe and Acrobat are registered trademarks of Adobe Systems Incorporated.

All other brand and product names are trademarks, registered trademarks or service marks of their respective holders.

Keep safety first in your circuit designs!

- Renesas Technology Corporation and Renesas Solutions Corporation put the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation, Renesas Solutions Corporation or a third party.
- Renesas Technology Corporation and Renesas Solutions Corporation assume no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation and Renesas Solutions Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation, Renesas Solutions Corporation or an authorized Renesas Technology product distributor for the latest product information before purchasing a product listed herein. The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corporation and Renesas Solutions Corporation assume no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors. Please also pay attention to information published by Renesas Technology Corporation and Renesas Solutions Corporation by various means, including the Renesas home page (<http://www.renesas.com>).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation and Renesas Solutions Corporation assume no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation, Renesas Solutions Corporation or an authorized Renesas Technology product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation and Renesas Solutions Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination. Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation or Renesas Solutions Corporation for further details on these materials or the products contained therein.

For inquiries about the contents of this document or product, fill in the text file the installer generates in the following directory and email to your local distributor.

\\SUPPORT\Product-name\\SUPPORT.TXT

Renesas Tools Homepage <http://www.renesas.com/en/tools>

Overview

The High-performance Embedded Workshop is a Graphical User Interface intended to ease the development and debugging of applications written in C/C++ programming language and assembly language for Renesas microcomputers. Its aim is to provide a powerful yet intuitive way of accessing, observing and modifying the debugging platform in which the application is running.

This help explains the function as a "debugger" of High-performance Embedded Workshop.

Target System

The Debugger operates on the FoUSB/UART or starter-kit.

Supported CPU

This help explains the debugging function corresponding to the following CPUs.

- M32C/80, M16C/80 Series

Note: In this help, the information which depends on this CPU is described as "for M32C".

- M16C/60, M16C/Tiny, M16C/20, M16C/10, R8C/Tiny Series

Note: In this help, the information which depends on this CPU is described as "for M16C/R8C".

(Blank Page)

Setup of Debugger

1

1. Features	3
1.1 Break Functions.....	3
1.1.1 Software Breaks Function.....	3
1.2 Real-Time OS Debugging Function.....	3
1.3 GUI Input/Output Function	4
2. Preparation before Use	5
2.1 Workspaces, Projects, and Files	5
2.2 Starting the High-performance Embedded Workshop	6
2.2.1 Creating a New Workspace (Toolchain Used)	7
2.2.2 Creating a New Workspace (Toolchain Not Used)	12
2.3 Starting the Debugger	17
2.3.1 Connecting the Emulator	17
2.3.2 Ending the Emulator	17
3. Setup the Debugger	18
3.1 Init Dialog.....	18
3.1.1 MCU Tab.....	19
3.1.2 Debugging Information Tab	20
3.1.3 Run Mode Tab.....	21
3.1.4 Script Tab.....	22
3.2 Setting of the Communication Interface	23
3.2.1 Setting of the USB Interface	23
3.2.2 Setting of the Serial Interface	24
3.3 Setup the Debugger for M32C	25
3.3.1 Emem Dialog.....	25

Tutorial

27

4. Tutorial	29
4.1 Introduction.....	29
4.2 Usage	30
4.2.1 Step1 : Starting the Debugger	30
4.2.2 Step2 : Checking the Operation of RAM.....	31
4.2.3 Step3 : Downloading the Tutorial Program	32
4.2.4 Step4 : Setting a Breakpoint.....	34
4.2.5 Step5 : Executing the Program	35
4.2.6 Step6 : Reviewing Breakpoints	37
4.2.7 Step7 : Viewing Register	38
4.2.8 Step8 : Viewing Memory	39
4.2.9 Step9 : Watching Variables.....	40
4.2.10 Step10 : Stepping Through a Program	42
4.2.11 Step11 : Forced Breaking of Program Executions.....	45
4.2.12 Step12 : Displaying Local Variables	46
4.2.13 Step13 : Stack Trace Function	47
4.2.14 What Next?	48

Reference

49

5. Windows/Dialogs	51
5.1 RAM Monitor Window	52
5.1.1 Extended Menus	53
5.1.2 Setting the RAM monitor area.....	54
5.2 ASM Watch Window	55
5.2.1 Extended Menus	56
5.3 C Watch Window.....	57
5.3.1 Extended Menus	58
5.4 Script Window	59
5.4.1 Extended Menus	60
5.5 S/W Break Point Setting Window	61
5.5.1 Command Button.....	62
5.5.2 Setting and Deleting a Break Points from Editor(Source) Window	63
5.6 GUI I/O Window.....	64
5.6.1 Extended Menus	65
5.7 MR Window	66
5.7.2 Display the Task Status	68
5.7.3 Display the Ready Queue Status	72
5.7.4 Display the Timeout Queue Status.....	73
5.7.5 Display the Event Flag Status.....	75
5.7.6 Display the Semaphore Status.....	77
5.7.7 Display the Mailbox Status.....	79
5.7.8 Display the Data Queue Status	81
5.7.9 Display the Cycle Handler Status	83
5.7.10 Display the Alarm Handler Status	85
5.7.11 Display the Memory Pool Status	86
5.7.12 Display the Task Context.....	88
6. Table of Script Commands	90
6.1 Table of Script Commands (classified by function)	90
6.1.1 Execution Commands.....	90
6.1.2 File Operation Commands	90
6.1.3 Register Operation Commands.....	91
6.1.4 Memory Operation Commands	91
6.1.5 Assemble/Disassemble Commands.....	91
6.1.6 Software Break Setting Commands.....	91
6.1.7 Script/Log File Commands.....	92
6.1.8 Program Display Commands	92
6.1.9 C Language Debugging Commands.....	92
6.1.10 Real-time OS Command.....	92
6.1.11 Utility Commands	93
6.2 Table of Script Commands (alphabetical order)	94
7. Writing Script Files	96
7.1 Structural Elements of a Script File	96
7.1.1 Script Command	97
7.1.2 Assign Statement.....	97
7.1.3 Conditional Statement	97
7.1.4 Loop Statement(while,endw) and Break Statement	97
7.1.5 Comment statements	98
7.2 Writing Expressions.....	98
7.2.1 Constants	98
7.2.2 Symbols and labels	99
7.2.3 Macro Variables.....	100
7.2.4 Register variables	101
7.2.5 Memory variables	101
7.2.6 Line Nos.	101

7.2.7 Character constants	102
7.2.8 Operators	102
8. C/C++ Expressions	103
8.1 Writing C/C++ Expressions	103
8.1.1 Immediate Values	103
8.1.2 Scope Resolution	104
8.1.3 Mathematical Operators	104
8.1.4 Pointers	104
8.1.5 Reference	104
8.1.6 Sign Inversion	105
8.1.7 Member Reference Using Dot Operator	105
8.1.8 Member Reference Using Arrow	105
8.1.9 Pointers to Members	106
8.1.10 Parentheses	106
8.1.11 Arrays	106
8.1.12 Casting to Basic Types	106
8.1.13 Casting to typedef Types	107
8.1.14 Variable Name	107
8.1.15 Function Name	107
8.1.16 Character Constants	107
8.1.17 Character String Literals	107
8.2 Display Format of C/C++ Expressions	108
8.2.1 Enumeration Types	108
8.2.2 Basic Types	108
8.2.3 Pointer Types	109
8.2.4 Array Types	110
8.2.5 Function Types	110
8.2.6 Reference Types	110
8.2.7 Bit Field Types	110
8.2.8 When No C Symbol is Found	111
8.2.9 Syntax Errors	111
8.2.10 Structure and Union Types	111
9. Display the Cause of the Program Stoppage	112
10. Attention	113
10.1 Common Attention	113
10.1.1 File operation on Windows	113
10.1.2 Area where software breakpoint can be set	114
10.1.3 Get or set C variables	114
10.1.4 Function name in C++	114
10.1.5 Debugging multi modules	114
10.1.6 Synchronized debugging	114
10.1.7 RAM monitor function	115
10.1.8 Line assemble function	115
10.1.9 Unavailable debugging functions	115
10.1.10 Software break function	115
10.2 Attention of the M32C Debugger	116
10.2.1 Option of C Compiler/Assembler/Linker	116
10.3 Attention of the M16C/R8C Debugger	117
10.3.1 Options for compiler, assembler, and linker	117
10.3.2 TASKING C Compiler	117
10.3.3 Cautions for R8C/Tiny series	117
10.4 Options for compiler, assembler, and linker	118
10.4.1 When Using NCxx	118
10.4.2 When Using the IAR C Compiler (EW)	118
10.4.3 When Using the IAR C Compiler (ICC)	119

10.4.4 When Using the TASKING C Compiler (EDE)	120
10.4.5 When Using the TASKING C Compiler (CM)	121
10.4.6 When Using the IAR EC++ Compiler (EW)	121

Setup of Debugger

(Blank Page)

1. Features

1.1 Break Functions

This debugger have the following functions.

1.1.1 Software Breaks Function

Software Break breaks the target program before execution of the command at the specified address. This break point is called software breakpoint.

The software breakpoint is set/reset in the Editor (Source) window or in the S/W Breakpoint Setting window. You can also disable/enable a software breakpoint temporarily.

The count of breakpoints depends on the connected MCU. When specifying two or more software breakpoints, the breakpoint combination is based on the OR logic. (Arrival to any one of breakpoints breaks the target program.)

1.1.1.1 Setting of software breakpoint

The software breakpoint can be set by the following windows.

- Editor (Source) Window
- S/W Break Point Setting Window

You can double-click the mouse to set/reset the software breakpoint in the Editor (Source) window.

You can also switch to temporarily disable/enable the software breakpoint in the S/W Breakpoint Setting window.

1.1.1.2 Area where software breakpoint can be set

The area which can be set for software breakpoint varies depending on the product.

For the areas available for software breakpoint, see the following:

"10.1.2 Area where software breakpoint can be set"

1.2 Real-Time OS Debugging Function

This function debugs the realtime OS-dependent parts of the target program that uses the realtime OS.

This function helps to show the status of the realtime OS.

1.3 GUI Input/Output Function

This function simulates the user target system's key input panel (buttons) and output panel on a window.

Buttons can be used for the input panel, and labels (strings) and LEDs can be used for the output panel.

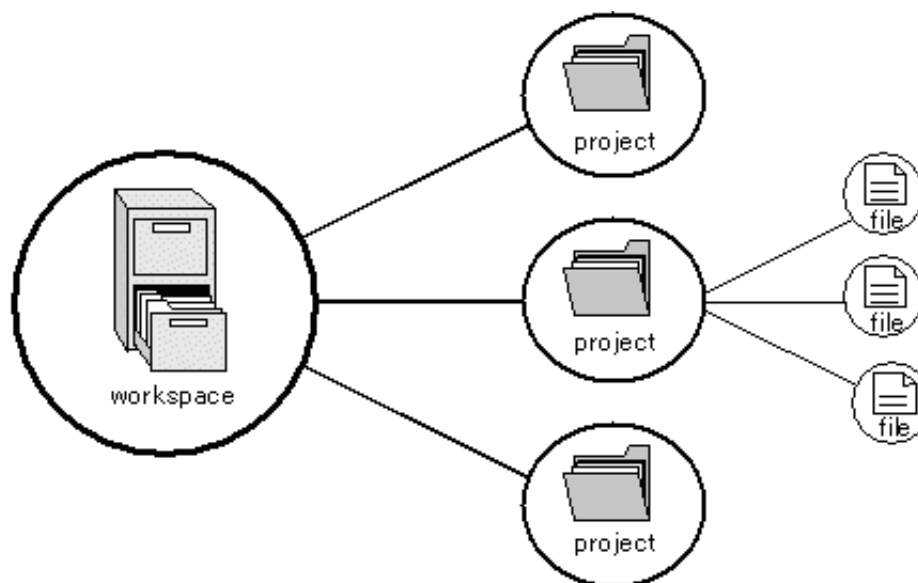
2. Preparation before Use

Please run the High-performance Embedded Workshop and connect the emulator .
In addition, in order to debug with this product, it is necessary to create a workspace.

2.1 Workspaces, Projects, and Files

Just as a word processor allows you to create and modify documents, this product allows you to create and modify workspaces.

A workspace can be thought of as a container of projects and, similarly, a project can be thought of as a container of project files. Thus, each workspace contains one or more projects and each project contains one or more files.

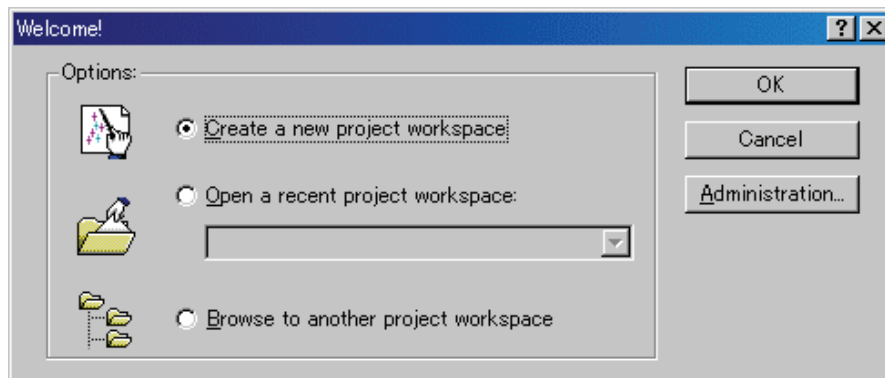


Workspaces allow you to group related projects together. For example, you may have an application that needs to be built for different processors or you may be developing an application and library at the same time. Projects can also be linked hierarchically within a workspace, which means that when one project is built all of its "child" projects are built first.

However, workspaces on their own are not very useful, we need to add a project to a workspace and then add files to that project before we can actually do anything.

2.2 Starting the High-performance Embedded Workshop

Activate the High-performance Embedded Workshop from [Programs] in the [Start] menu. The [Welcome!] dialog box is displayed.



In this dialog box, A workspace is created or displayed.

- [Create a new project workspace] radio button:
Creates a new workspace.
- [Open a recent project workspace] radio button:
Uses an existing workspace and displays the history of the opened workspace.
- [Browse to another project workspace] radio button:
Uses an existing workspace;
this radio button is used when the history of the opened workspace does not remain.

In the case of Selecting an Existing Workspace, select [Open a recent project workspace] or [Browse to another project workspace] radio button and select the workspace file (.hws).

Please refer to the following about the method to create a new workspace.

- Refer to "2.2.1 Creating a New Workspace (Toolchain Used)"
- Refer to "2.2.2 Creating a New Workspace (Toolchain Not Used)"

* When debugging the existing load module file with this product, a workspace is created by this method.

The method to create a new workspace depends on whether a toolchain is or is not in use. Note that this product does not include a toolchain. Use of a toolchain is available in an environment where the C/C++ compiler package for the CPU which you are using has been installed.

For details on this, refer to the manual attached to your C/C++ compiler package.

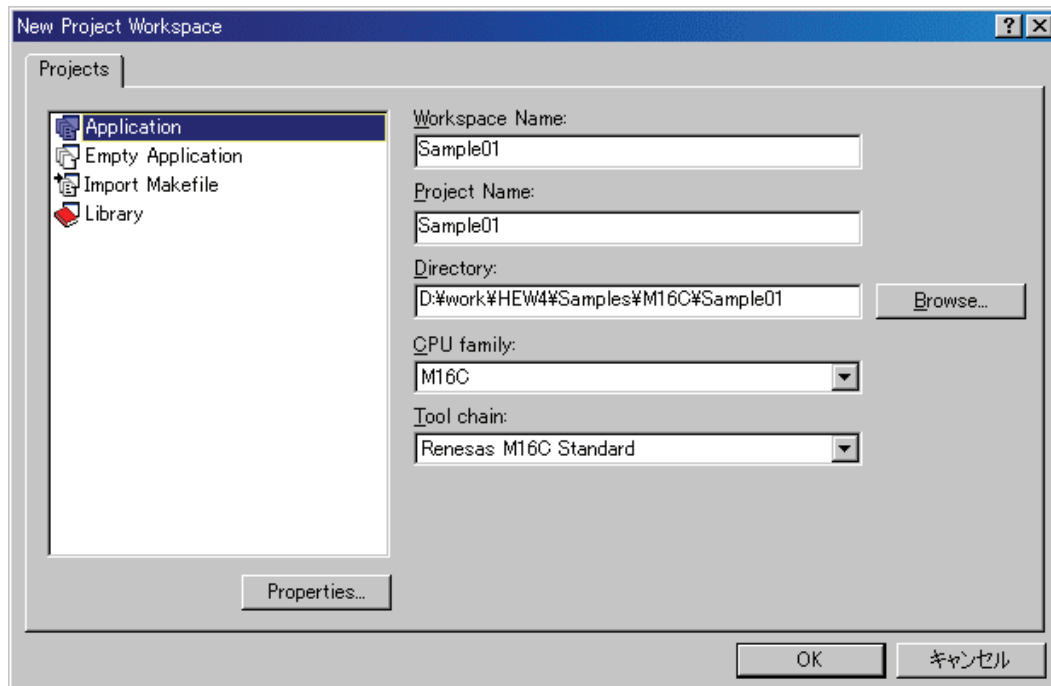
2.2.1 Creating a New Workspace (Toolchain Used)

2.2.1.1 Step1 : Creation of a new workspace

In the [Welcome!] dialog box that is displayed when the High-performance Embedded Workshop is activated, select the [Create a new project workspace] radio button and click the [OK] button.

Creation of a new workspace is started.

The following dialog box is displayed.

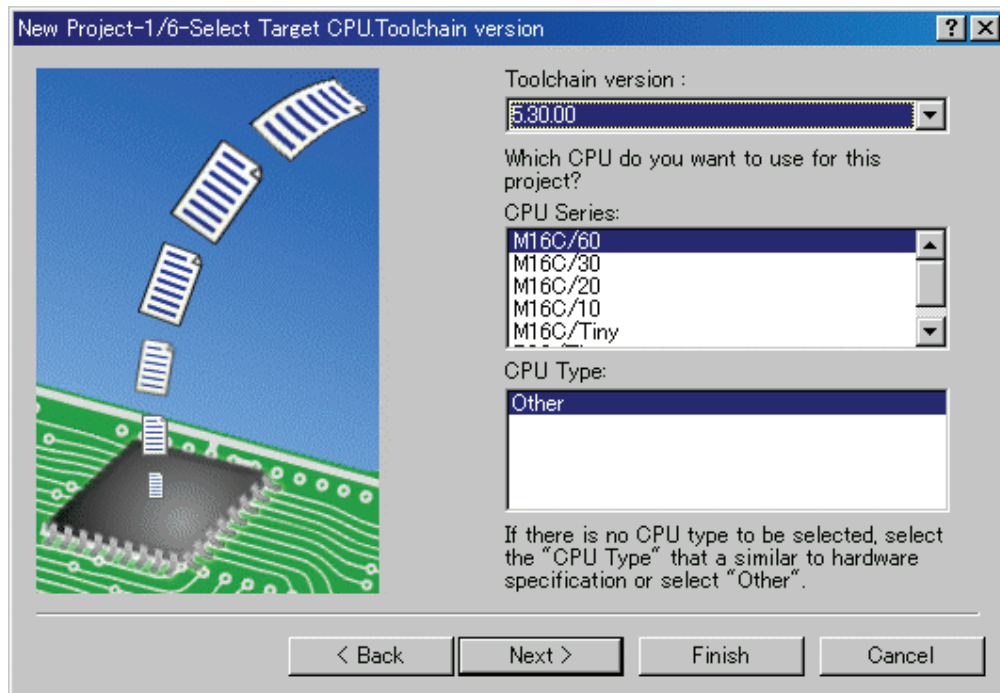


1. Select the target CPU family
In the [CPU family] combo box, select the target CPU family.
2. Select the target toolchain
In the [Tool chain] combo box, select the target toolchain name when using the toolchain.
3. Select the project type
In the [Project type] list box, select the project type to be used.
In this case, select "Application".
(Please refer to the manual attached to your C/C++ compiler package about the details of the project type which can be chosen.)
4. Specify the workspace name and project name
 - In the [Workspace Name] edit box, enter the new workspace name.
 - In the [Project Name] edit box, enter the project name. When the project name is the same as the workspace name, it needs not be entered.
 - In the [Directory] edit box, enter the directory name in which the workspace will be created.
Click the [Browse...] button to select a directory.

After a setting, click the [OK] button.

2.2.1.2 Step2 : Setting for the Toolchain

A wizard for the project creation starts.



Here, the following contents are set.

- toolchain
- the setting for the real-time OS (when using)
- the setting for the startup file, heap area, stack area, and so on

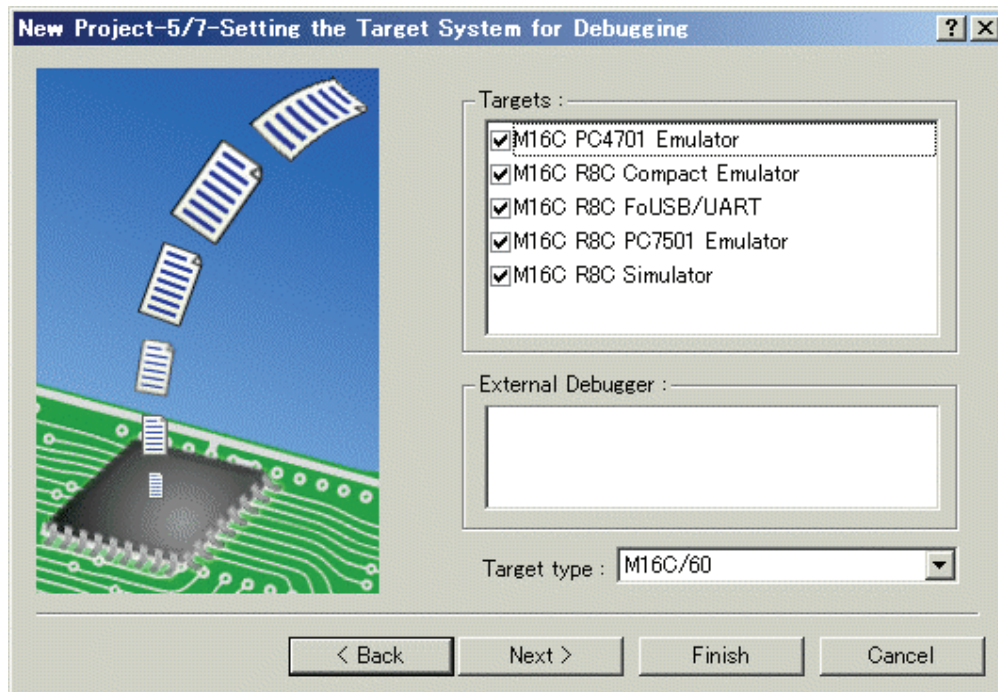
Please set required information and click the [Next] button.

The contents of a setting change with C/C++ compiler packages of use. Please refer to the manual attached to your C/C++ compiler package about the details of the contents of a setting.

2.2.1.3 Step 3: Selecting of the Target Platform

Select the target system used for your debugging (emulator, simulator).

When the setting for the toolchain has been completed, the following dialog box is displayed.



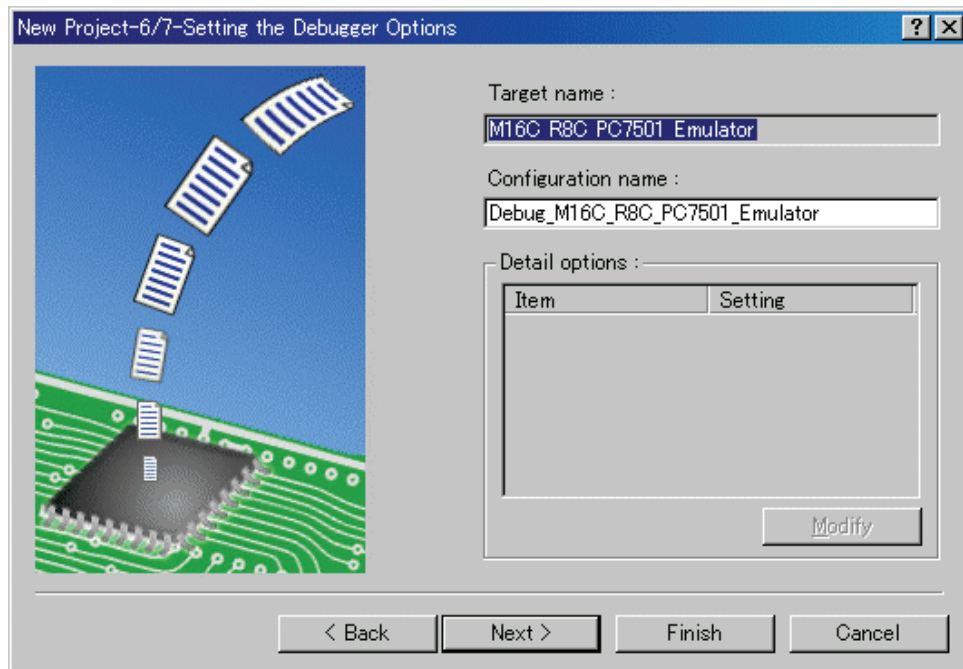
1. Selecting of the Target type
In the [Target type] list box, select the target CPU type.
2. Selecting of the Target Platform
In the [Targets] area, the target for the session file used when this debugger is activated must be selected here.
Check the box against the target platform. (And choose other target as required.)

And click the [Next] button.

2.2.1.4 Step4 : Setting the Configuration File Name

Set the configuration file name for each of the all selected target.

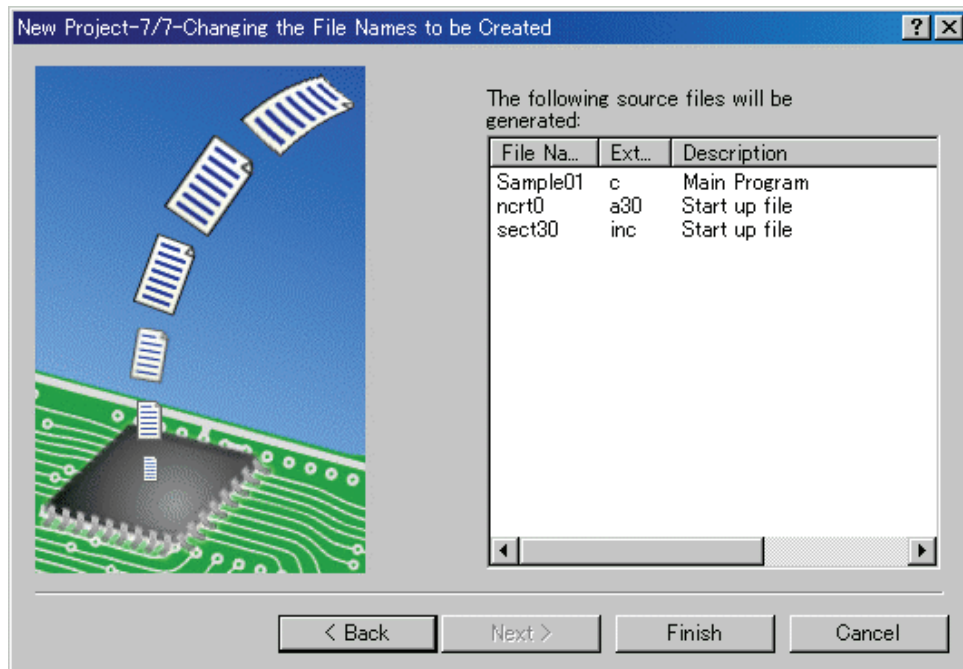
The configuration file saves the state of High-performance Embedded Workshop except for the target (emulator, simulator).



The default name is already set. If it is not necessary to change, please click the [next] button as it is.

2.2.1.5 Step5 : The check of a created file name

Finally, confirm the file name you create. The files which will be generated by the High-performance Embedded Workshop are displayed. If you want to change the file name, select and click it then enter the new name.



This is the end of the emulator settings.

Exit the Project Generator following the instructions on the screen.

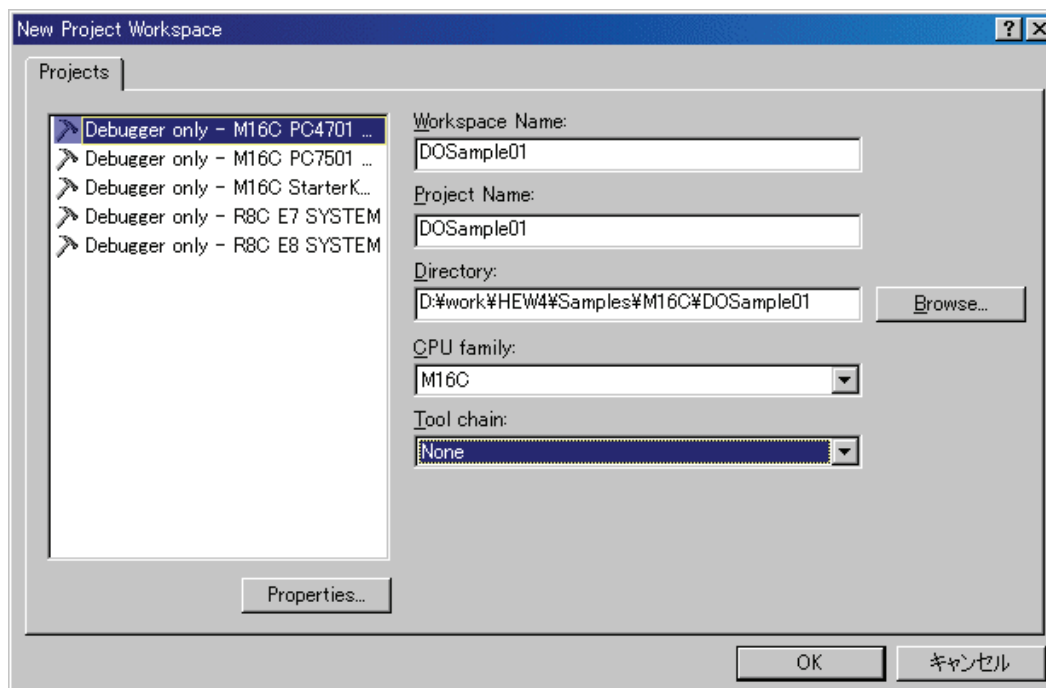
2.2.2 Creating a New Workspace (Toolchain Not Used)

When debugging the existing load module file with this product, a workspace is created by this method.(It can work even if the tool chain is not installed.)

2.2.2.1 Step1 : Creation of a new workspace

In the [Welcome!] dialog box that is displayed when the High-performance Embedded Workshop is activated, select the [Create a new project workspace] radio button and click the [OK] button.

Creation of a new workspace is started. The following dialog box is displayed.



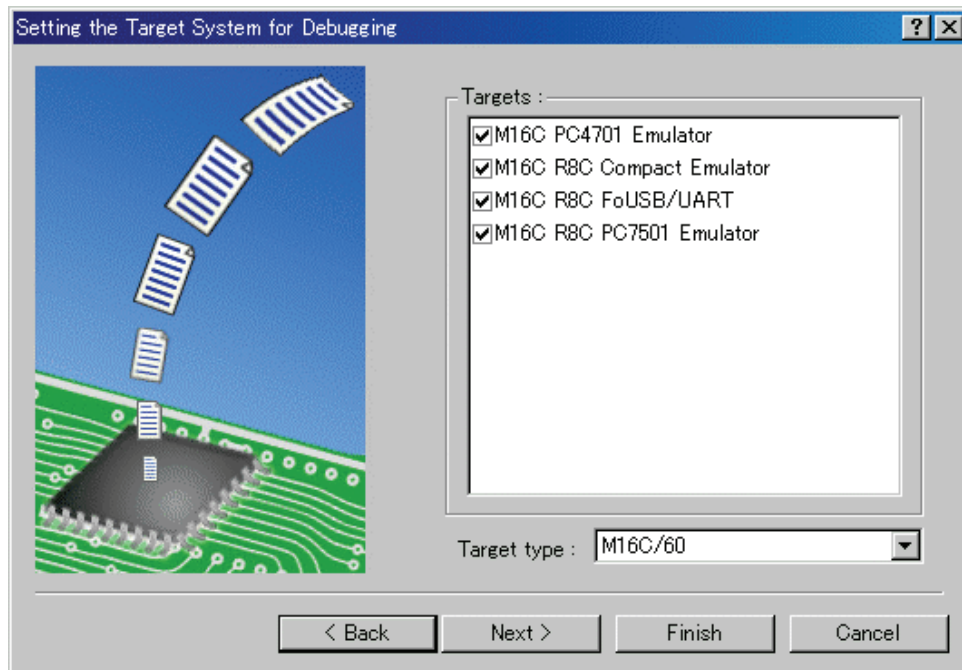
1. Select the target CPU family
In the [CPU family] combo box, select the target CPU family.
2. Select the target toolchain
In the [Tool chain] combo box, select "None". In this case, toolchain is not used.
(When the toolchain has not been installed, the fixed information is displayed in this combo box.)
3. Select the project type
(When the toolchain is not used, it is displayed on a [Project Type] list box as "Debugger only - Target Name". Select it. (When two or more project types are displayed, please select one of them.)
4. Specify the workspace name and project name
 - In the [Workspace Name] edit box, enter the new workspace name.
 - In the [Project Name] edit box, enter the project name. When the project name is the same as the workspace name, it needs not be entered.
 - In the [Directory] edit box, enter the directory name in which the workspace will be created. Click the [Browse...] button to select a directory.

After a setting, click the [OK] button.

2.2.2.2 Step 2: Selecting of the Target Platform

Select the target system used for your debugging (emulator, simulator).

A wizard starts and the following dialog box is displayed.



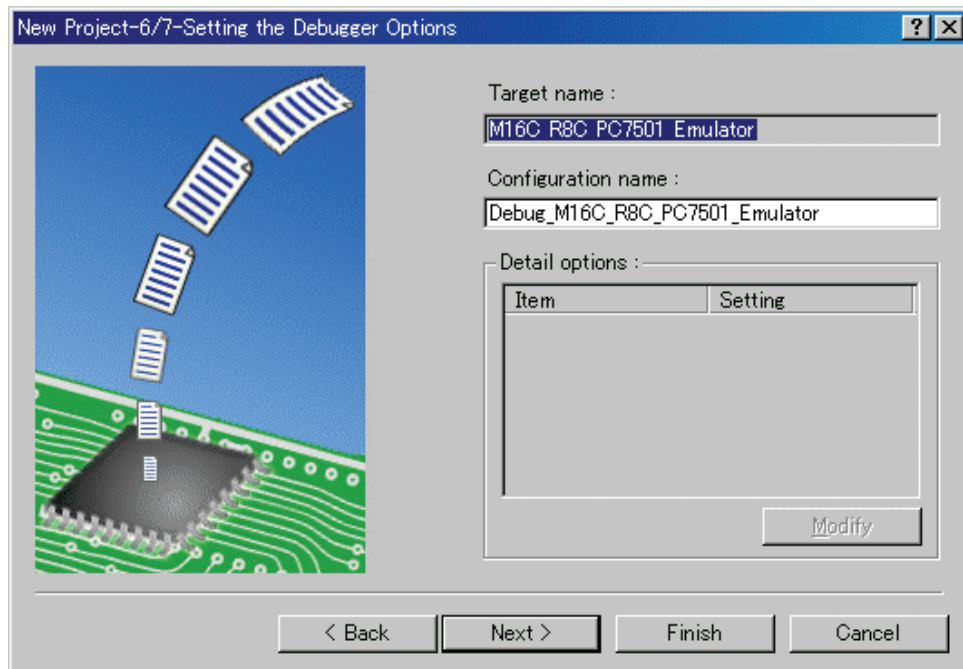
1. Selecting of the Target type
In the [Target type] list box, select the target CPU type.
2. Selecting of the Target Platform
In the [Targets] area, the target for the session file used when this debugger is activated must be selected here.
Check the box against the target platform. (And choose other target as required.)

And click the [Next] button.

2.2.2.3 Step3 : Setting the Configuration File Name

Set the configuration file name for each of the all selected target.

The configuration file saves the state of High-performance Embedded Workshop except for the target (emulator, simulator).



The default name is already set. If it is not necessary to change, please click the [next] button as it is. This is the end of the emulator settings.

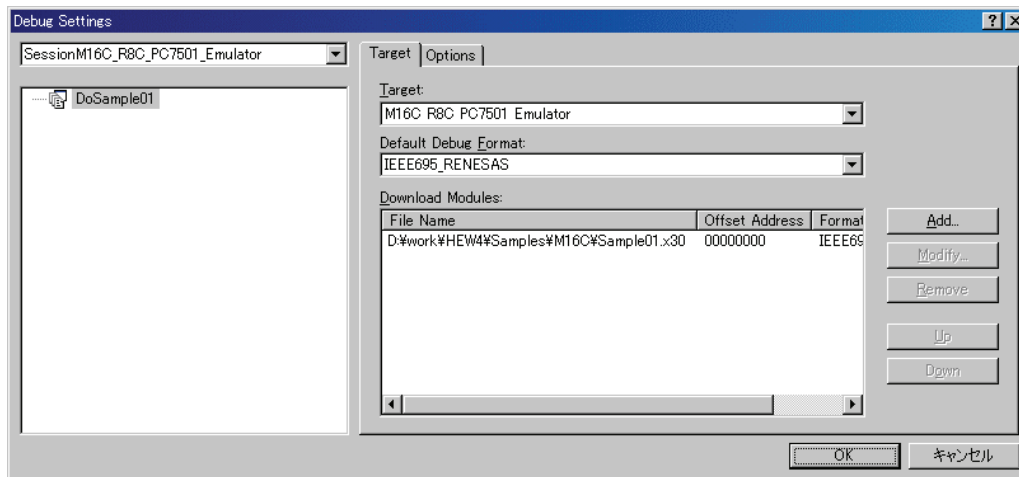
Exit the Project Generator following the instructions on the screen.

And the dialog for the setup of a debugger is also displayed at this time . If preparation of an emulator is completed, set up the debugger in this dialog box and connect with an emulator.

2.2.2.4 Step4 : Registering the Load modules to be downloaded

Finally, register the load module file to be used.

Select [Debug Settings...] from the [Debug] menu to open the [Debug Settings] dialog box.

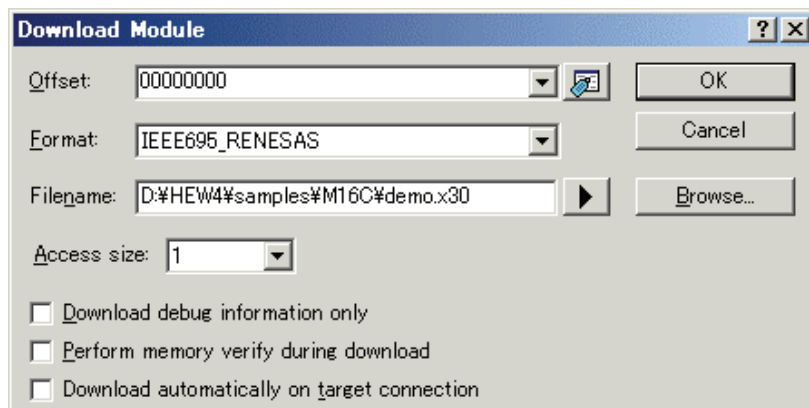


1. Select the product name to be connected in the [Target] drop-down list box.
2. Select the format of the load module to be downloaded in the [Default Debug Format] drop-down list box.

Format Name	Contents
IEEE695_RENESAS	IEEE-695 format file (When Using NCxx)
IEEE695_IAR	IEEE-695 format file (When Using IAR cross tool)
IEEE695_TASKING	IEEE-695 format file (When Using Tasking cross tool)
ELF/DWARF2_IAR	ELF/DWARF2 format file (When Using IAR cross tool)
ELF/DWARF2_TASKING	ELF/DWARF2 format file (When Using Tasking cross tool)

This debugger does not support the object formats, which are not shown in the drop down list.

-
3. Then register the corresponding download module in the [Download Modules] list box.
A download module can be specified in the dialog opened with a [Add...] button.



- Enter the offset at which to load the download module in the [Offset] edit box.
- Select the format of the download module in the [Format] edit box. Please refer to the upper table about the format name of a download module.
- Enter the full path and filename of the download module in the [Filename] edit box.
- Specifies the access size for the current download module in the [Access size] list box.

After that, click the [OK] button.

ATTENTION

"Access size" and "Perform memory verify during download" is ignored.
The access size is always set to 1 and the verification does not work.

2.3 Starting the Debugger

The debugging can be started by connecting with an emulator.

2.3.1 Connecting the Emulator

Connect the emulator by simply switching the session file to one in which the setting for the emulator use has been registered.

The session file is created by default. The session file has information about the target selected when a project was created.

In the circled list box in the following tool bars, select the session name including the character string of the target to connect.



After the session name is selected, the dialog box for setting the debugger is displayed and the emulator will be connected.

2.3.2 Ending the Emulator

The emulator can be exited by using the following methods:

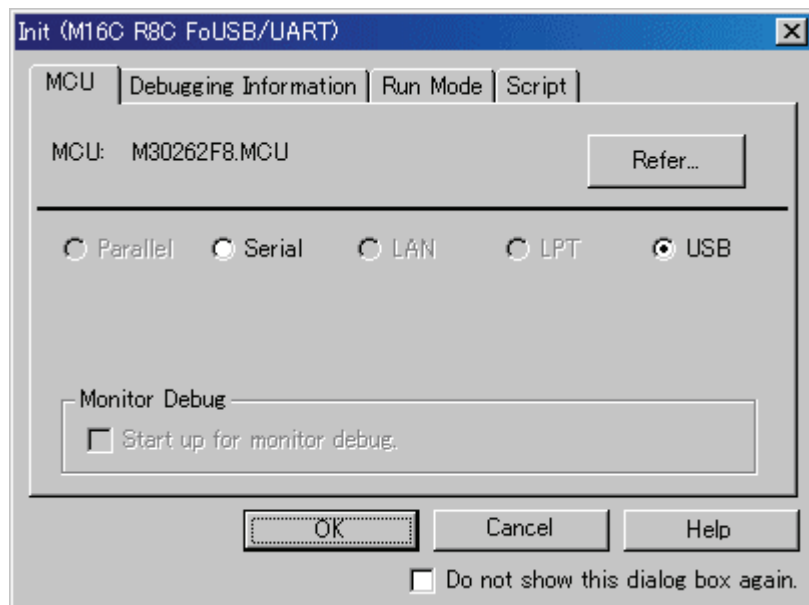
1. Selecting the "DefaultSession"
Select the "DefaultSession" in the list box that was used at the time of emulator connection.
2. Exiting the High-performance Embedded Workshop
Select [Exit] from the [File] menu. High-performance Embedded Workshop will be ended.

The message box, that asks whether to save a session, will be displayed when an emulator is exited. If necessary to save it, click the [Yes] button. If not necessary, click the [No] button.

3. Setup the Debugger

3.1 Init Dialog

The Init dialog box is provided for setting the items that need to be set when the debugger starts up. The contents set from this dialog box are also effective the next time the debugger starts. The data set in this dialog remains effective for the next start.



The tabs available on this dialog box vary with each product used. For details, click the desired tab name shown in the table below.

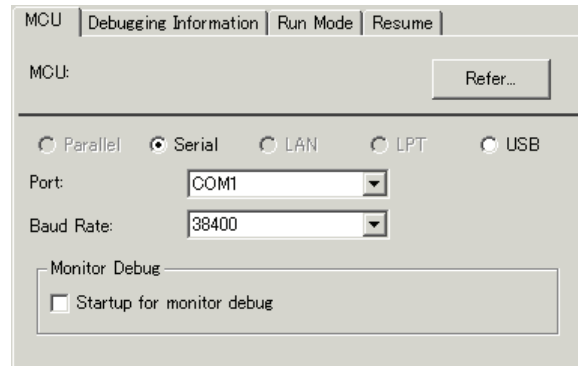
Tab Name	Product Name	
	The debugger for M32C	The debugger for M16C/R8C
MCU	exist	exist
Debugging Information	exist	exist
Run Mode	exist	exist
Script	exist	exist

You can open the Init dialog using either one of the following methods:

- After the debugger gets started, select Menu - [Setup] -> [Emulator] -> [System...].
- Start Debugger while holding down the Ctrl key.

3.1.1 MCU Tab

The specified content becomes effective when the next being start.



3.1.1.1 Specifying the MCU file



Click the "Refer" button.

The File Selection dialog is opened. Specify the corresponding MCU file.

- An MCU file contains the information specific to the target MCU.
- The specified MCU file is displayed in the MCU area of the MCU tab.

3.1.1.2 Setting of the Communication Interface

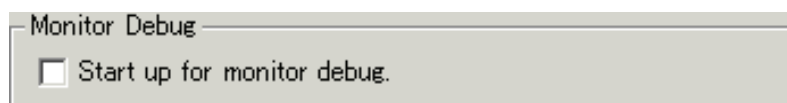
The displayed data varies depending on the specified communication interface.

The available communication interface varies depending on the products.

The following shows the setting for each communication interface.

Refer to "3.2 Setting of the Communication Interface"

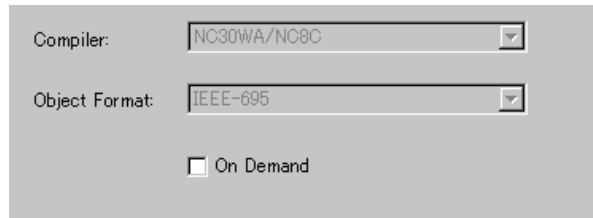
3.1.1.3 Setting of the Monitor Debug



When you start up this debugger for the monitor debug, check the checkbox "Start up for monitor debug". If this checkbox is checked, this debugger starts up without initializing the system (To use this function, you need the monitor file corresponding to the monitor debug function).

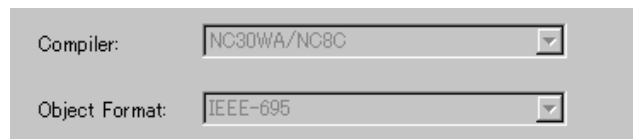
3.1.2 Debugging Information Tab

The specified content becomes effective when the next being download.



3.1.2.1 display the compiler used and its object format

Display the compiler used and its object file format.



Please specify the compiler used and its object file format in the dialog opened by menu [Debug] -> [Debug Settings...].

3.1.2.2 Specify the Storing of Debugging Information

There are two methods for storing debugging information: on-memory and on-demand.

Select one of these two methods. (The on-memory method is selected by default.)

To select the on-demand method, click the On Demand check box.

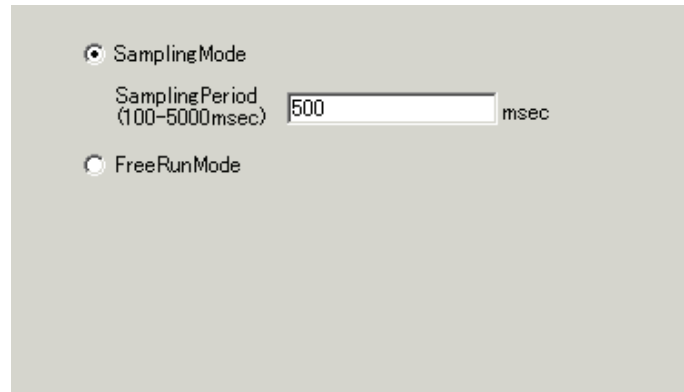
- On-memory method
Debugging information is stored in the internal memory of your computer.
This method is suitable when the load module (target program) size is small.
- On-demand method
Debugging information is stored in a reusable temporary file on the hard disk of your computer.
Because the stored debugging information is reused, the next time you download the same load module it can be downloaded at high speed.
This method is suitable when the load module (target program) size is large.

Notes

- If the load module size is large, the on-memory method may be inefficient because it requires a very large amount of time for downloading. In such a case, select the on-demand method.
- In the on-demand method, a folder in which to store a reusable temporary file is created in the folder that contains the downloaded load module. This folder is named after the load module name by the word "~INDEX_" to it. If the load module name is "sample.abs", for example, the folder name is "~INDEX_sample". This folder is not deleted even after quitting the debugger.

3.1.3 Run Mode Tab

The specified content becomes effective when the next being start.



3.1.3.1 Setting Run Mode

Set the mode in which you want the user program to be run when executing Go or Come. With this debugger, the monitor program periodically monitors the user program's execution status to check whether the user program is halted by a break, etc. This means that the monitor program interrupts the CPU when executing the user program and, hence, the user program's real-time capability is lost. To solve this problem, this debugger provides two execution modes:

- **Sampling Mode:**
the user program's execution status is periodically monitored when executing Go or Come. Therefore, it is possible to check whether the user program is halted by a break, etc. The monitoring interval time is determined by the value you set in the Sampling period field. Set an appropriate interval time that will not affect execution of the user program.
- **Free Run Mode:**
the user program's execution status is not monitored when executiong Go or Come. Therefore, although the user program's real-time execution capability is maintained, it is impossible to check whether the user program is halted by a break, etc. Consequently, this debugger shows executing Go or Come while the user program has been halted. To show the stop status, press the STOP button.

Also, if the operations (which causes memory dump like using RAM monitor window or switching the display mode of Source window to disassembly, and so on) are invoked while target program is running, the real-time execution capability will be lost.

3.1.4 Script Tab

The specified content becomes effective when the next being start.



3.1.4.1 Automatically Execute the Script Commands

To automatically execute the script command at start of Debugger, click the "Refer" button to specify the script file to be executed.



By clicking the "Refer" button, the File Selection dialog is opened.

The specified script file is displayed in the "Init File:" field.

To disable auto-execution of the script command, erase a character string displayed in the "Init File:" field.

3.2 Setting of the Communication Interface

3.2.1 Setting of the USB Interface

USB communication uses the personal computer's USB interface. It is compliant with USB 1.1.

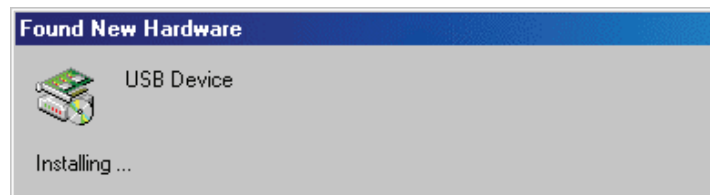
Before USB communication can be performed, the computer must have a dedicated device driver installed in it.

For connection by USB communication, click the "USB" radio button on the MCU tab.

3.2.1.1 Install of USB device driver

The USB devices connected are detected by Windows' Plug & Play function. The installation wizard for USB device drivers starts after the device had been detected. The following shows the procedure for installing the USB device drivers.

1. Connect the host computer and the emulator with USB cable.
2. Set the emulator's communication interface switch to the "USB" position. Then turn on the power to the emulator.
3. The dialog box shown below appears.



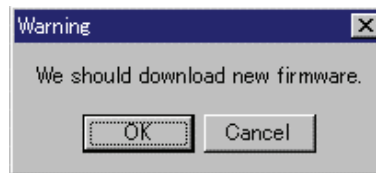
Go on following the wizard, and a dialog box for specifying the setup information file (inf file) is displayed. Specify the musbdrv.inf file stored in a location below the directory where this debugger is installed.

ATTENTION

- Before the USB device drivers can be installed, the debugger you use must already be installed. Install this debugger first.
- USB communication can be used only in Windows Me/98/2000/XP, and cannot be used in any other OSs.
- When using Windows 2000/XP, a user who install the USB device driver need administrator rights.
- During installation, a message may be output indicating that the device driver proper musbdrv.sys cannot be found. In this case, specify the musbdrv.sys which is stored in the same directory as is the musbdrv.inf file.

3.2.1.2 Download the Monitor Program

When the monitor program, that is corresponding to the mcu that is specified by the mcu file, is different from the one that is downloaded in the target, the debugger notices you to download it. If the mcu file you specified is correct, download it by pressing OK button.

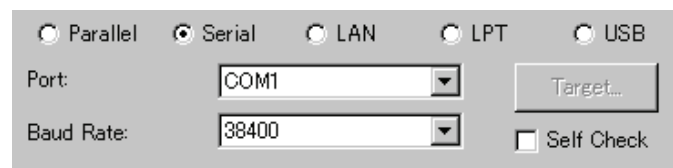


3.2.2 Setting of the Serial Interface

Serial communication uses a serial interface (RS-232C) of the personal computer. This communication method is available for all the PC4701 emulator series.

3.2.2.1 Setting of the Serial Interface

To set the Serial communication, click the "Serial" radio button of the MCU tab in the Init dialog. The setting looks like the figure below.

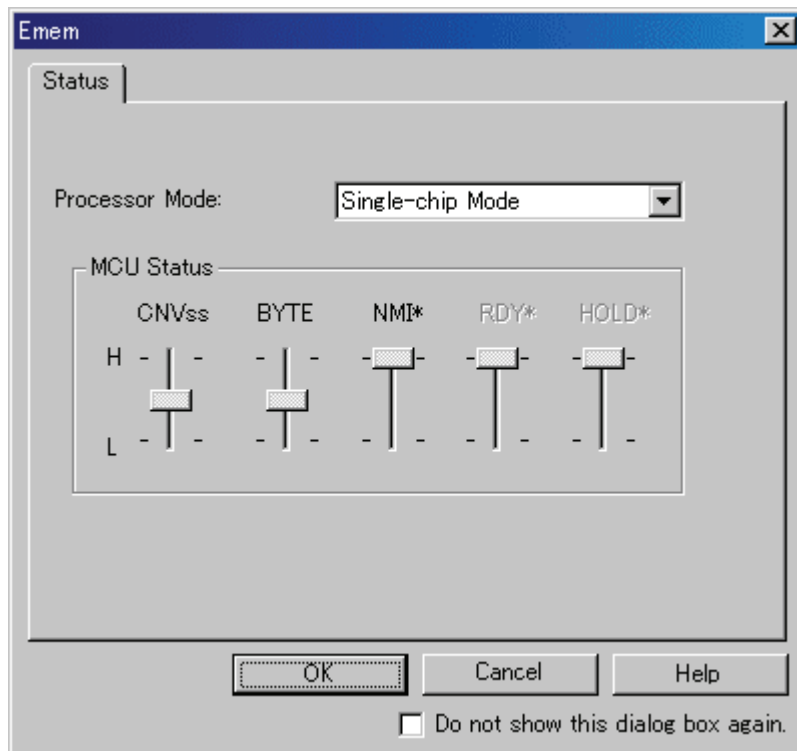


Specify the communications port in the Port field and the baud rate in the Baud Rate field.

3.3 Setup the Debugger for M32C

3.3.1 Emem Dialog

In the Emem dialog box, setting information on the user target. The Emem dialog box opens after closing the Init dialog box.



The tabs available on this dialog box vary with each product used. For details, click the desired tab name shown in the table below.

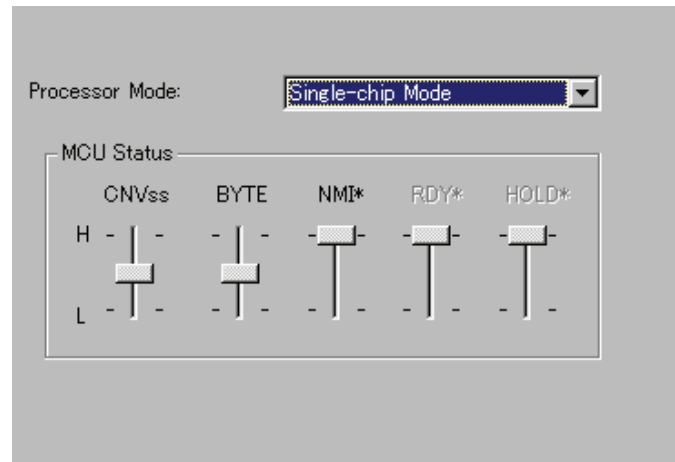
Tab Name	Contents
Status	Specify the processor mode.

To keep the Emem dialog box closed next time the debugger is started, check "Next Hide" at the bottom of the Emem dialog box. You can open the Emem dialog using either one of the following methods:

- After the debugger gets started, select Menu - [Setup] -> [Emulator] -> [Target...].

3.3.1.1 Status Tab

The specified content becomes effective when the next being start.



3.3.1.1.1 *Select the Processor Mode*

Specify the processor mode for the target system.

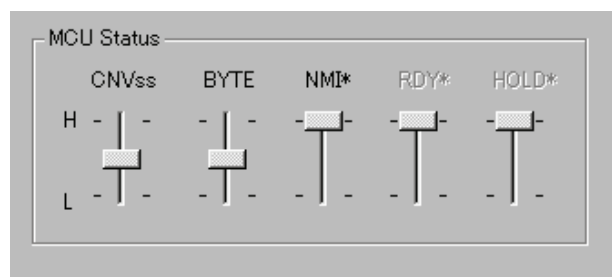


Either the following can be specified.

- Single-chip Mode
Single-chip Mode
- Memory Expansion
Memory Expansion Mode
- Microprocessor
Microprocessor Mode

3.3.1.1.2 *Inspecting the MCU status*

Clicking this tab displays the status of each MCU pin. It allows to check whether the MCU pin status matches the processor mode to be set.



If the slider is at the middle position, it means that the value is indeterminate.

Tutorial

(Blank Page)

4. Tutorial

4.1 Introduction

This section describes the main functions of this debugger by using a tutorial program. The tutorial programs are installed to the directory ¥Workspace¥Tutorial of the drive you installed High-performance Embedded Workshop. There are workspaces for each targets and each MCUs. Please select the corresponding one to your system, and open the workspace file (*.hws) from the menu [Open Workspace...].

The tutorial program is based on the C program that sorts ten random data items in ascending or descending order.

The tutorial program performs the following actions:

- The tutorial function generates random data to be sorted.
- The sort function sorts the generated random data in ascending order.
- The change function then sorts the data in descending order.

Note

After recompilation, the addresses may differ from those given in this section.

4.2 Usage

Please follow these instructions:

4.2.1 Step1 : Starting the Debugger

4.2.1.1 Preparation before Use

To run the High-performance Embedded Workshop and connect the emulator, refer to "2 Preparation before Use".

4.2.1.2 Setup the Debugger

If it connects with an emulator, the dialog box for setting up a debugger will be displayed. Please set up the debugger in this dialog box.

To setup the debugger in this dialog box, refer to "3 Setup the Debugger".

After the setup of a debugger, it will function as a debugger.

4.2.2 Step2 : Checking the Operation of RAM

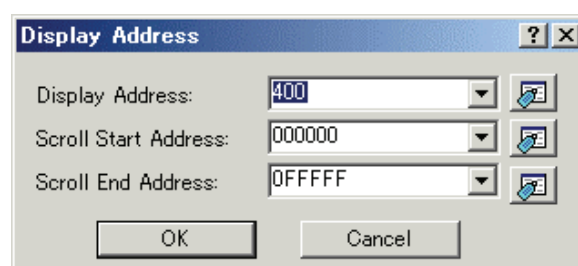
Check that RAM is operating correctly. Display and edit the contents of the memory in the [Memory] window to check that the memory is operating correctly.

Note

The memory can be installed on the board in some microcomputers. In this case, however, the above way of checking the operation of memory may be inadequate. It is recommended that a program for checking the memory be created.

4.2.2.1 Checking the Operation of RAM

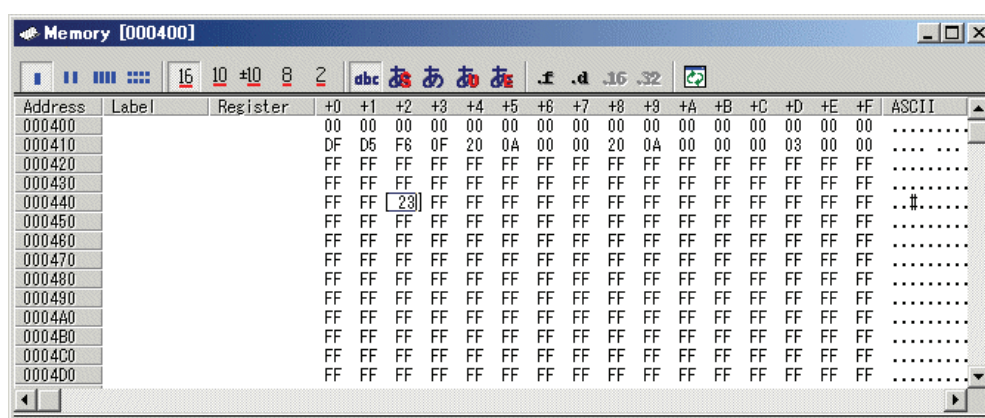
Select [Memory] from the [CPU] submenu of the [View] menu and enter the RAM address (Here, enter H'400) in the [Display Address] edit boxes. The [Scroll Start Address] and [Scroll End Address] editing box is left to a default setting. (By default, the scroll range is set to 0h to the maximum address of MCU.)



Note

The settings of the RAM area differ depending on the product. For details, refer to the hardware manual.

Click the [OK] button. The [Memory] window is displayed and shows the specified memory area.



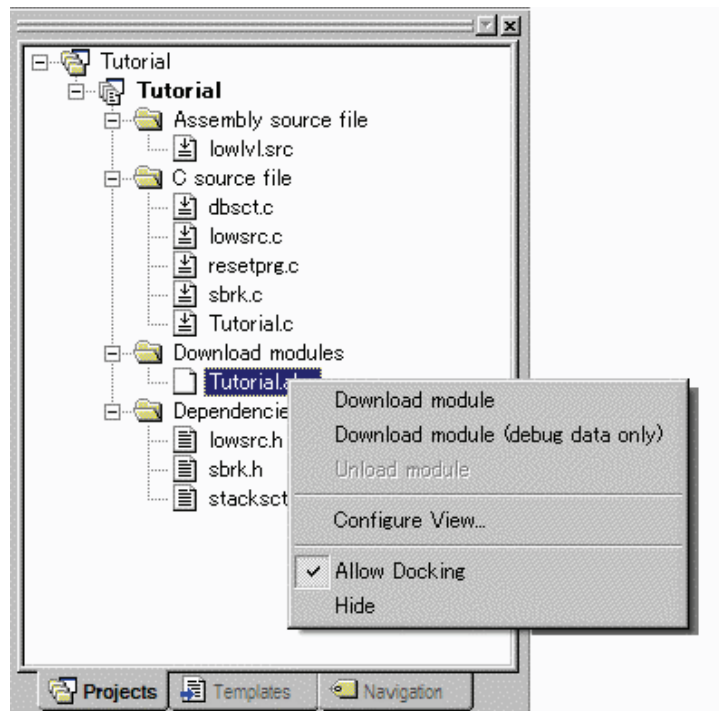
Placing the mouse cursor on a point in the display of data in the [Memory] window and double-clicking allows the values at that point to be changed.

4.2.3 Step3 : Downloading the Tutorial Program

4.2.3.1 Downloading the Tutorial Program

Download the object program to be debugged. The download file and the address to be downloaded will depend on the target mcu you use. Please replace the screen image and addresses with corresponding one to your target mcu.

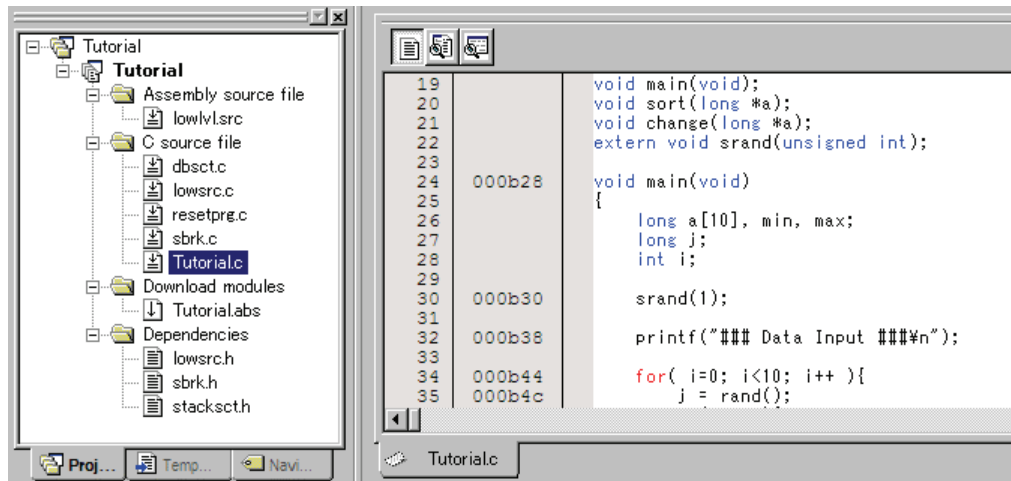
- The Debugger for M16C/R8C or M32C
Select [Download module] from [Tutorial.x30] under [Download modules].



4.2.3.2 Displaying the Source Program

This debugger allows the user to debug a user program at the source level.

Double-click [tutorial.c] under [C source file]. A [Editor(Source)] window opens and the contents of a "Tutorial.c" file are displayed.



Select the [Format Views...] option from the [Setup] menu to set a font and size that are legible, if necessary.

Initially the [Editor(Source)] window shows the start of the user program, but the user can use the scroll bar to scroll through the user program and look at the other statements.

4.2.4 Step4 : Setting a Breakpoint

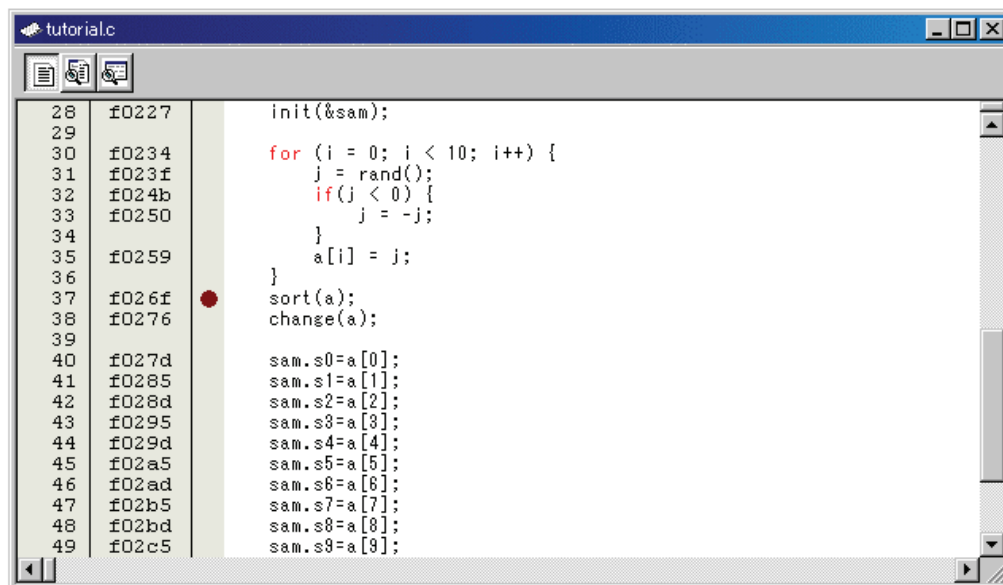
A software breakpoint is a basic debugging function.

The [Editor(Source)] window provides a very simple way of setting a software breakpoint at any point in a program.

4.2.4.1 Setting a Software Breakpoint

For example, to set a software breakpoint at the sort function call:

Double-click the [S/W breakpoints] column on the line containing the sort function call.




The red symbol will appear on the line containing the sort function call. This shows that a softwarebreak breakpoint has been set.

4.2.5 Step5 : Executing the Program


Execute the program as described in the following:

4.2.5.1 Resetting of CPU

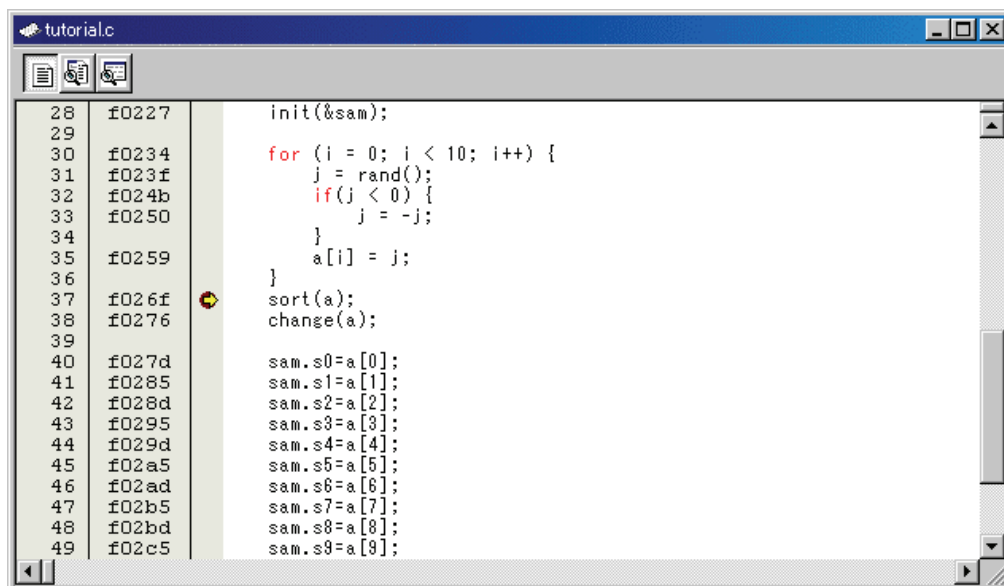
By default, CPU is not reset after downloading a program.

To reset the CPU, select [Reset CPU] from the [Debug] menu, or click the [Reset CPU] button  on the toolbar.

4.2.5.2 Executing the Program

To execute the program, select [Go] from the [Debug] menu, or click the [Go] button  on the toolbar.

The program will be executed up to the breakpoint that has been set, and an arrow will be displayed in the [S/W Breakpoints] column to show the position that the program has halted.

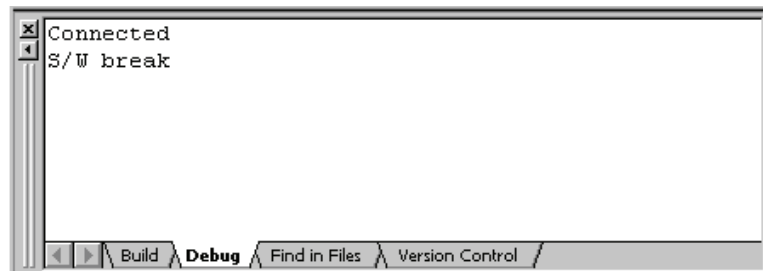


Note

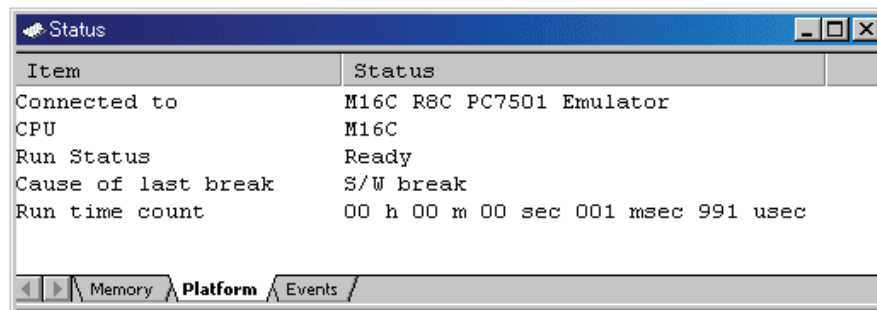
When the source file is displayed after a break, a path of the source file may be inquired. In this case, please specify the location of a source file.

4.2.5.3 Reviewing Cause of the Break

The break factor is displayed in the [Output] window.



The user can also see the cause of the break that occurred last time in the [Status] window. Select [Status] from the [CPU] submenu of the [View] menu. After the [Status] window is displayed, open the [Platform] sheet, and check the Status of Cause of last break.



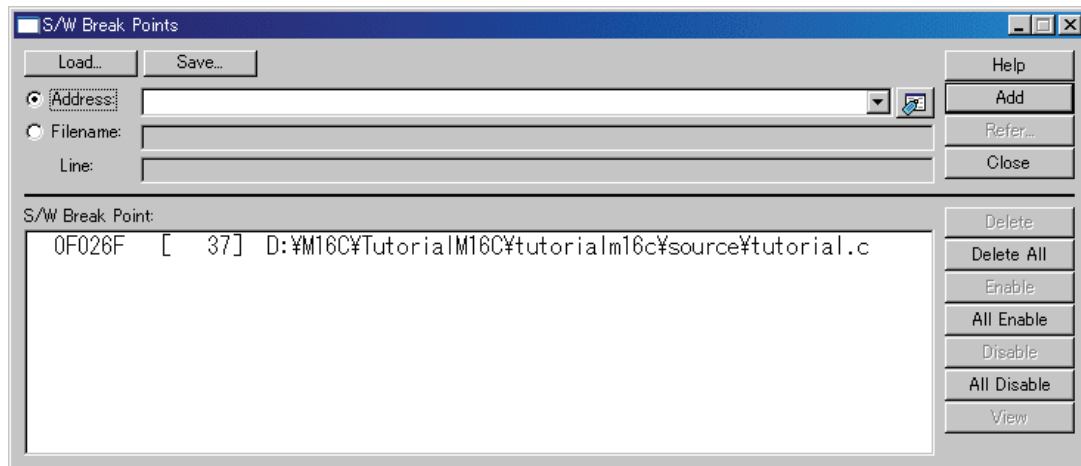
Please refer to "9 Display the Cause of the Program Stoppage" about the notation of a break factor.

4.2.6 Step6 : Reviewing Breakpoints

The user can see all the breakpoints set in the program in the [S/W Break Points] window.

4.2.6.1 Reviewing Breakpoints

Select [S/W Break Points] from the [Break] submenu of the [View] menu. The [S/W Break Points] window is displayed.



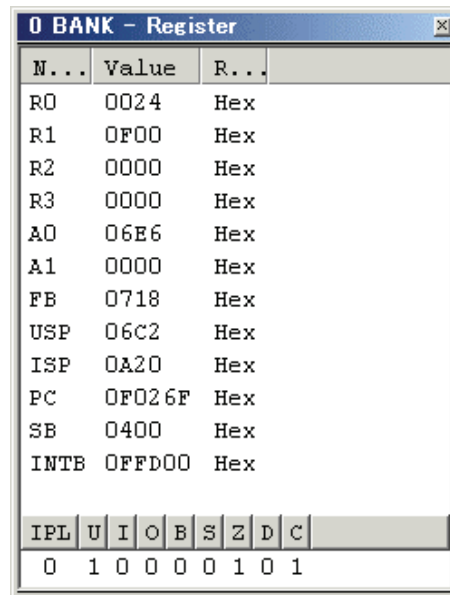
This window allows the user to set or change breakpoints, define new breakpoints, and delete, enable, or disable breakpoints.

4.2.7 Step7 : Viewing Register

The user can see all registers/flags value in the [Register] window.

4.2.7.1 Viewing Register

Select [Registers] from the [CPU] submenu of the [View] menu. The [Register] window is displayed. The figure below shows a Register window of the debugger for M16C/R8C.



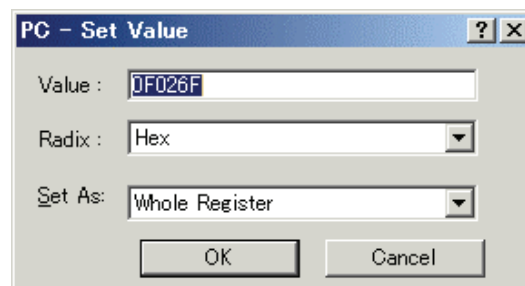
N...	Value	R...
R0	0024	Hex
R1	0F00	Hex
R2	0000	Hex
R3	0000	Hex
A0	06E6	Hex
A1	0000	Hex
FB	0718	Hex
USP	06C2	Hex
ISP	0A20	Hex
PC	0F026F	Hex
SB	0400	Hex
INTB	0FFD00	Hex

IPL	U	I	O	B	S	Z	D	C
0	1	0	0	0	0	1	0	1

4.2.7.2 Setting the Register Value

You can change a register/flag value from this window.

Double-click the register line to be changed. The dialog is opened. Enter the value to be changed.



PC - Set Value

Value : 0F026F

Radix : Hex

Set As: Whole Register

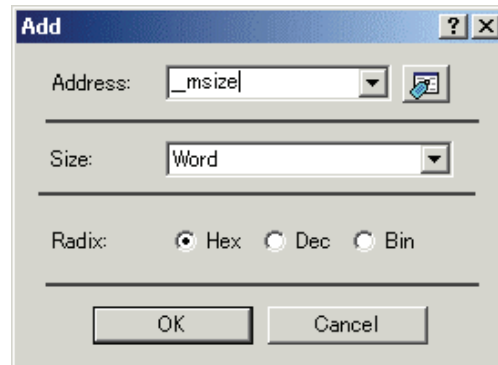
OK Cancel

4.2.8 Step8 : Viewing Memory

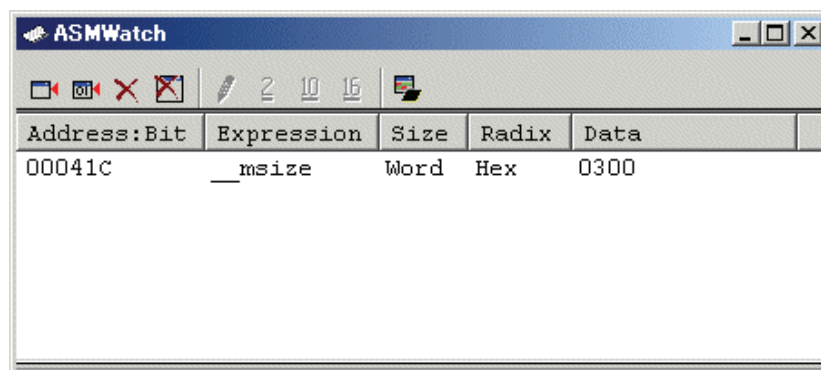
When the label name is specified, the user can view the memory contents that the label has been registered in the [ASM Watch] window.

4.2.8.1 Viewing Memory

For example, to view the memory contents corresponding to `__msize` in word size:
Select [ASM Watch] from the [Symbol] submenu of the [View] menu, open the [ASM Watch] window.
And click the [ASM Watch] window with the right-hand mouse button and select [Add...] from the popup menu, enter `__msize` in the [Address] edit box, and set Word in the [Size] combo box.



Click the [OK] button. The [ASM Watch] window showing the specified area of memory is displayed.



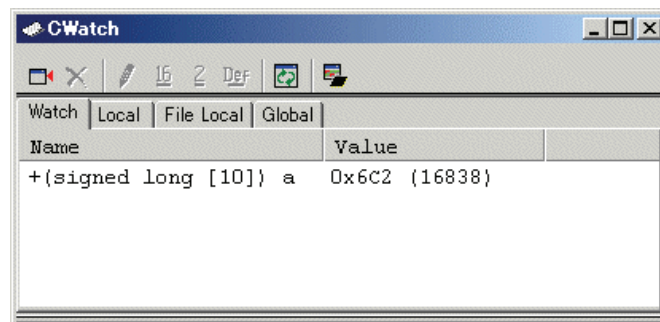
4.2.9 Step9 : Watching Variables

As the user steps through a program, it is possible to watch that the values of variables used in the user program are changed.

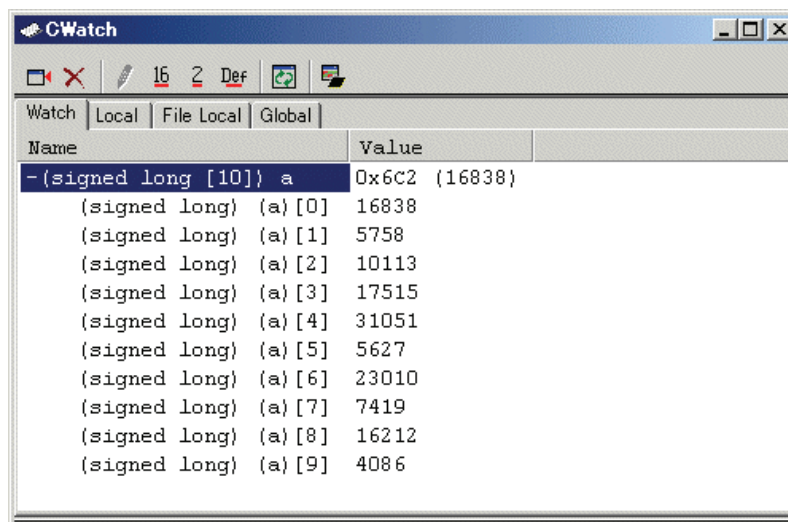
4.2.9.1 Watching Variables

For example, set a watch on the long-type array `a` declared at the beginning of the program, by using the following procedure:

Click the left of displayed array `a` in the [Editor(Source)] window to position the cursor, and select [Add C Watch...] with the right-hand mouse button. The [Watch] tab of [C watch] window in which the variable is displayed opens.



The user can click mark '+' at the left side of array `a` in the [C Watch] window to watch all the elements.



4.2.9.2 Registering Variable

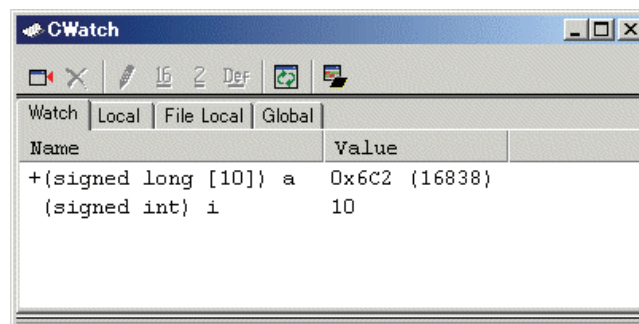
The user can also add a variable to the [C Watch] window by specifying its name.

Click the [C Watch] window with the right-hand mouse button and select [Add...] from the popup menu.

The following dialog box will be displayed. Enter variable i.



Click the [OK] button. The [C Watch] window will now also show the int-type variable i.



4.2.10 Step10 : Stepping Through a Program

This debugger provides a range of step menu commands that allow efficient program debugging.

1. Step In
Executes each statement, including statements within functions(subroutines).
2. Step Out
Steps out of a function(subroutine), and stops at the statement following the statement in the program that called the function(subroutine).
3. Step Over
Executes a function(subroutine) call in a single step.
4. Step...
Steps the specified times repeatedly at a specified rate.

4.2.10.1 Executing [Step In] Command

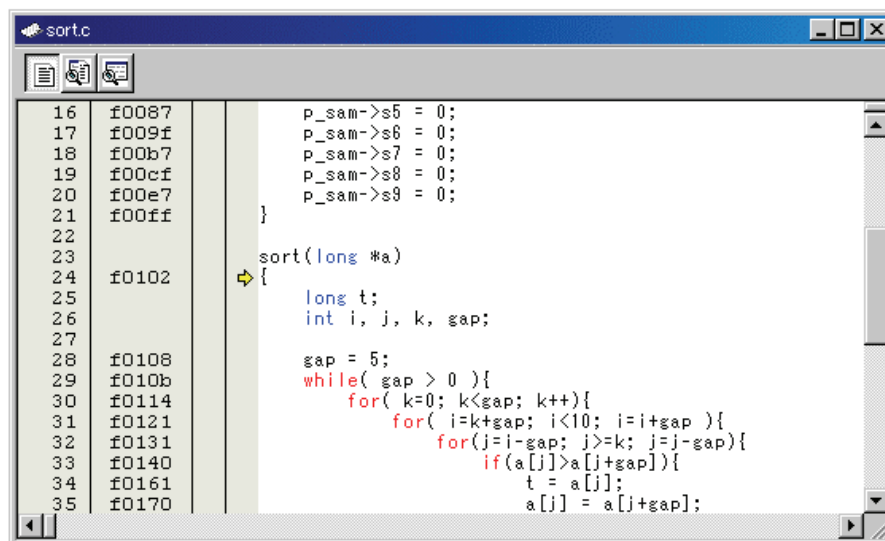
The [Step In] command steps into the called function(subroutine) and stops at the first statement of the called function(subroutine).

To step through the sort function, select [Step In] from the [Debug] menu, or click the [Step In] button



on the toolbar.

The PC cursor moves to the first statement of the sort function in the [Editor(Source)] window.




```
sort.c
16 f0087 p_sam->s5 = 0;
17 f009f p_sam->s6 = 0;
18 f00b7 p_sam->s7 = 0;
19 f00cf p_sam->s8 = 0;
20 f00e7 p_sam->s9 = 0;
21 f00ff }
22
23 sort(long #a)
24 f0102 {
25     long t;
26     int i, j, k, gap;
27
28     gap = 5;
29     while( gap > 0 ){
30         for( k=0; k<gap; k++){
31             for( i=k+gap; i<10; i=i+gap ){
32                 for( j=i-gap; j>=k; j=j-gap){
33                     if(a[j]>a[j+gap]){
34                         t = a[j];
35                         a[j] = a[j+gap];
```

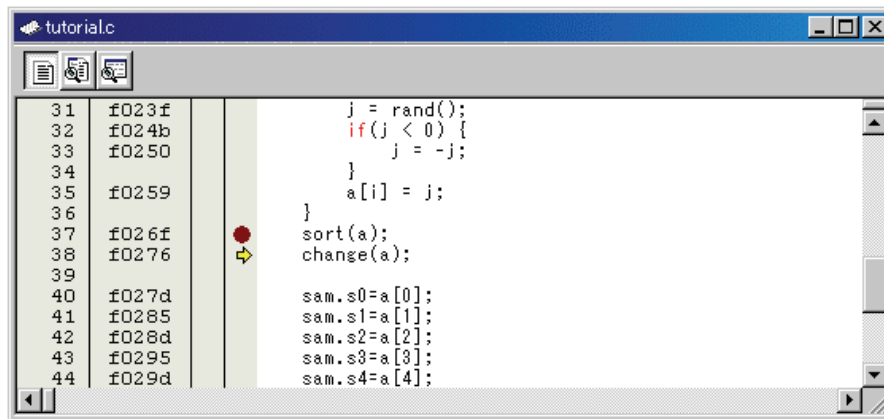
4.2.10.2 Executing [Step Out] Command

The [Step Out] command steps out of the called function(subroutine) and stops at the next statement of the calling statement in the main function.

To step out of the sort function, select [Step Out] from the [Debug] menu, or click the [Step Out]

button  on the toolbar.

The PC cursor slips out of a sort function, and moves to the position before a change function.



Note

It takes time to execute this function. When the calling source is clarified, use [Go To Cursor].

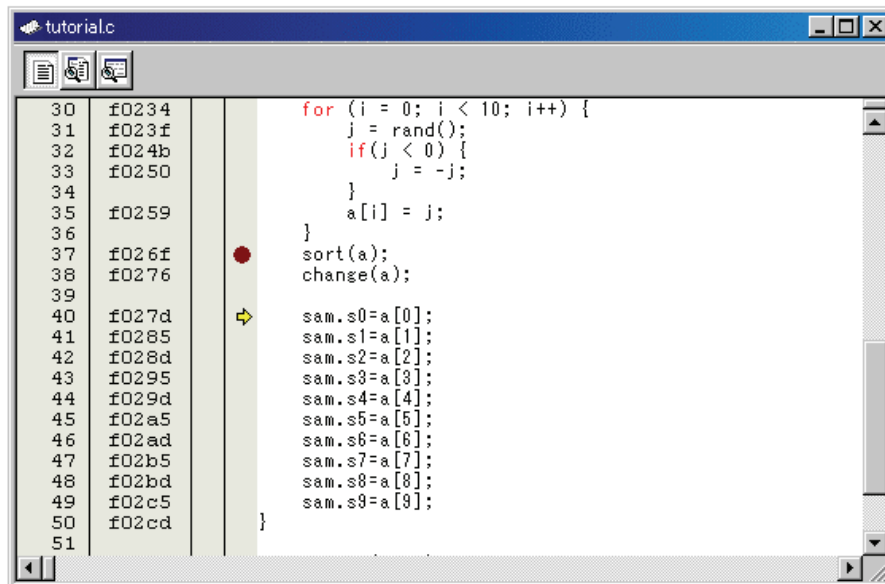
4.2.10.3 Executing [Step Over] Command

The [Step Over] command executes a function(subroutine) call as a single step and stops at the next statement of the main program.

To step through all statements in the change function at a single step, select [Step Over] from the

[Debug] menu, or click the [Step Over] button  on the toolbar.

The PC cursor moves to the next position of a change function.



4.2.11 Step11 : Forced Breaking of Program Executions

This debugger can force a break in the execution of a program.

4.2.11.1 Forced Breaking of Program Executions

Cancel all breaks.

To execute the remaining sections of the main function, select [Go] from the [Debug] menu or the [Go]



button on the toolbar.

The program goes into an endless loop. To force a break in execution, select [Halt Program] from the



[Debug] menu or the [Halt] button on the toolbar.

4.2.12 Step12 : Displaying Local Variables

The user can display local variables in a function using the [C Watch] window.

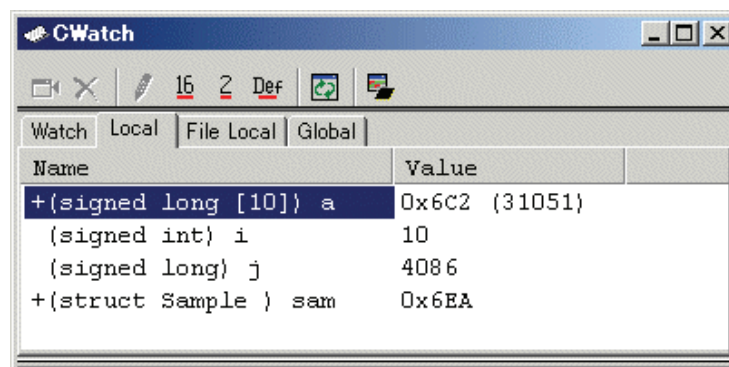
4.2.12.1 Displaying Local Variables

For example, we will examine the local variables in the tutorial function, which declares four local variables: a, j, i, and sam.

Select [C Watch] from the [Symbol] submenu of the [View] menu. The [C Watch] window is displayed. By default, [C watch] window has four tabs as following:

- [Watch] tab
Only the variable which the user registered is displayed.
- [Local] tab
All the local variables that can be referred to by the scope in which the the PC exists are displayed. If a scope is changed by program execution, the contents of the [Local] tab will also change.
- [File Local] tab
All the file local variables of the file scope in which the PC exists are displayed. If a file scope is changed by program execution, the contents of the [File Local] tab will also change.
- [Global] tab
All the global variables currently used by the downloaded program are displayed.

Please choose the [Local] tab, when you display a local variable.



Click mark '+' at the left side of array a in the [Locals] window to display the elements.

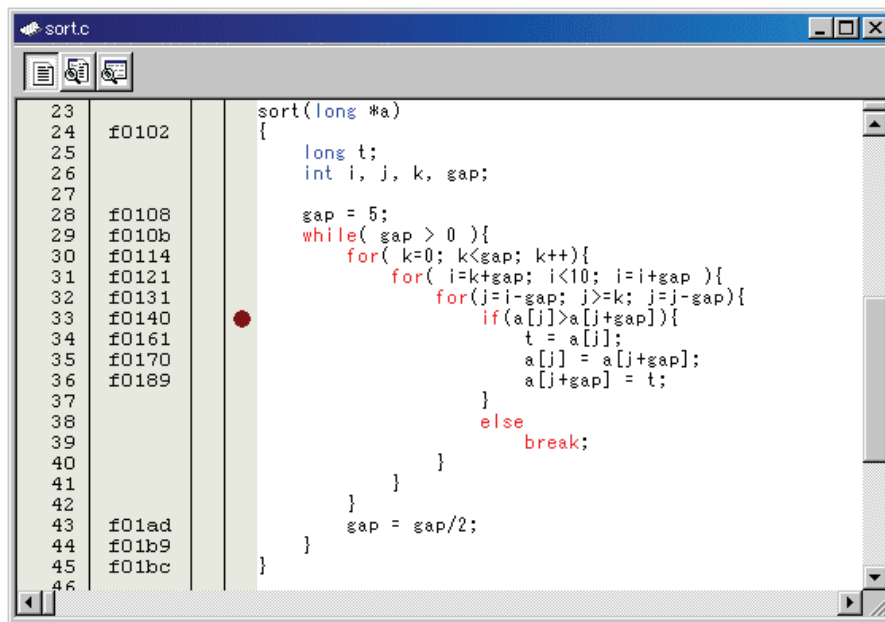
When the user refers to the elements of array a before and after the execution of the sort function, it is clarified that random data is sorted in descending order.

4.2.13 Step13 : Stack Trace Function


The debugger uses the information on the stack to display the names of functions in the sequence of calls that led to the function to which the program counter is currently pointing.

4.2.13.1 Reference the function call status

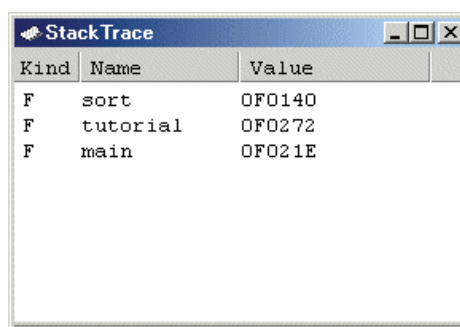
Double-click the [S/W Breakpoints] column in the sort function and set a software breakpoint.



To executes the user program from the reset vector address, select [Reset Go] from the [Debug] menu,

or click the [Reset Go] button  on the toolbar.

After the break in program execution, select [Stack Trace] from the [Code] submenu of the [View] menu to open the [Stack Trace] window.



The upper figure shows that the position of the program counter is currently at the selected line of the sort() function, and that the sort() function is called from the tutorial() function.

4.2.14 What Next?

This tutorial has described the usage of this debugger.

Sophisticated debugging can be carried out by using the emulation functions that the emulator offers.

This provides for effective investigation of hardware and software problems by accurately isolating and identifying the conditions under which such problems arise.

Reference

(Blank Page)

5. Windows/Dialogs

The window of this debugger is shown below.

When the window name is clicked, the reference is displayed.

Window Name	View Menu
RAM Monitor Window	[View]->[CPU]->[RamMonitor]
ASM Watch Window	[View]->[Symbol]->[ASMWatch]
C Watch Window	[View]->[Symbol]->[CWatch]
Script Window	[View]->[Script]
S/W Break Point Setting Window	[View]->[Break]->[S/W Break Points]
GUI I/O Window	[View]->[Graphic]->[GUI I/O]
MR Window	[View]->[RTOS]->[MR]

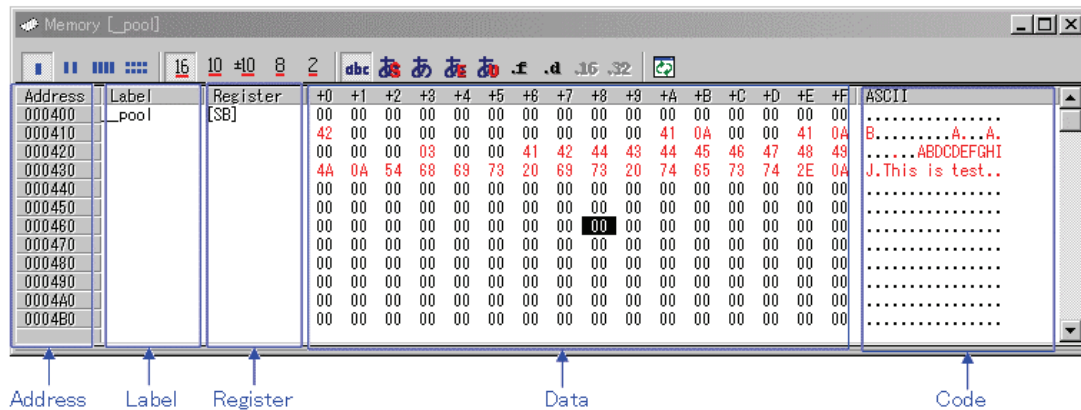
For the reference of the following windows, refer to the help attached to a High-performance Embedded Workshop main part.

- Differences Window
- Map Window
- Command Line Window
- Workspace Window
- Output Window
- Disassembly Window
- Memory Window
- IO Window
- Status Window
- Register Window
- Image Window
- Waveform Window
- Stack Trace Window

5.1 RAM Monitor Window

The RAM monitor window is a window in which changes of memory contents are displayed while running the target program.

The displayed contents are updated at given intervals while running the target program.



- This system provides a 1Kbytes of RAM monitor area, which can be placed at any continuous addresses.
- The RAM monitor area can be changed to any desired address range.
Refer to "5.1.2 Setting the RAM monitor area" for details on how to change the RAM monitor area. The default RAM monitor area is mapped into a 1-Kbyte area beginning with the start address of the internal RAM.
- The display content updating interval can be set for each window individually.
The actual updating interval at which the display contents are actually updated while running the target program is shown in the title field of the Address display area.

ATTENTION

- The interval time at which intervals the display is updated may be longer than the specified interval depending on the operating condition (shown below).
 - Host machine performance/load condition
 - Communication interface
 - Window size (memory display range) or the number of windows displayed

5.1.1 Extended Menus

This window has the following popup menus that can be brought up by right-clicking in the window.

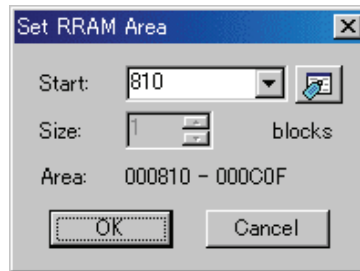
Menu		Function
RAM Monitor Area...		Set RAM monitor base address.
Sampling Period...		Set RAM monitor sampling period.
Clear		Clear access attribute.
Up		Moves display position to the immediately preceding RAM monitor area (smaller address)
Down		Moves display position to the immediately following RAM monitor area (larger address)
Address...		Display from specified address.
Scroll Area...		Specify scroll range.
Data Length	1byte	Display in 1Byte unit.
	2bytes	Display in 2Byte unit.
	4bytes	Display in 4Byte unit.
	8bytes	Display in 8Byte unit.
Radix	Hex	Display in Hexadecimal.
	Dec	Display in Decimal.
	Single Dec	Display in Signed Decimal.
	Oct	Display in Octdecimal.
	Bin	Display in Binary.
Code	ASCII	Display as ASCII character.
	SJIS	Display as SJIS character.
	JIS	Display as JIS character.
	UNICODE	Display as UNICODE character.
	EUC	Display as EUC character.
	Float	Display as Floating-point.
	Double	Display as Double Floating-point.
Layout	Label	Switch display or non-display of Label area.
	Register	Switch display or non-display of Register area.
	Code	Switch display or non-display of Code area.
Column...		Set the number of columns displayed on one line.
Split		Split window.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

5.1.2 Setting the RAM monitor area

Choose the popup menu [RAM Monitor Area...] in the RAM monitor window.

The Set RRAM Area dialog box shown below will appear.

The start address of the currently set RAM monitor area and the range of the RAM monitor area are displayed in the Start and the Area fields of this dialog box. (No values can be entered in the Size field.)



Use this dialog box to change the position of the RAM monitor area.

- Specify the RAM monitor area by its start address. The size cannot be changed (fixed to 1 Kbyte).
- The start address can be specified in 0x10 byte units.
If you specify a non-aligned address value, it is rounded off to the nearest address value in 0x10 byte units before being set.

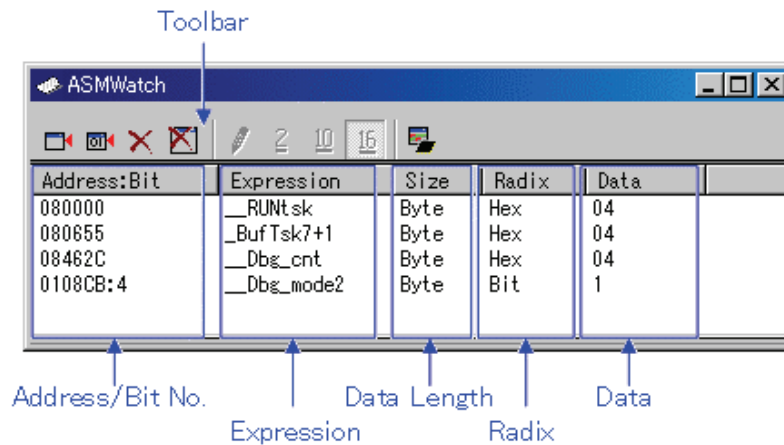
5.1.2.1 Changing the RAM Monitor Area

The start address of the RAM monitor area can be changed.

Specify the start address of the RAM monitor area in the Start field of the Set RRAM Area dialog box. (No values can be entered in the Size field.)

5.2 ASM Watch Window

The ASM watch window is a window in which you can register specific addresses as watchpoints and inspect memory contents at those addresses.



- The addresses to be registered are called the "watchpoints." One of the following can be registered:
 - Address (can be specified using a symbol)
 - Address + Bit number
 - Bit symbol
- The registered watchpoints are saved in the debugger when the ASM watch window is closed and are automatically registered when the window is reopened.
- If symbols or bit symbols are specified for the watchpoints, the watchpoint addresses are recalculated when downloading the target program.
- The invalid watchpoints are marked by "-<not active>-" when displayed on the screen.
- The order in which the watchpoints are listed can be changed by a drag-and-drop operation.
- The watchpoint expressions, sizes, radices and datas can be changed by in-place editing.

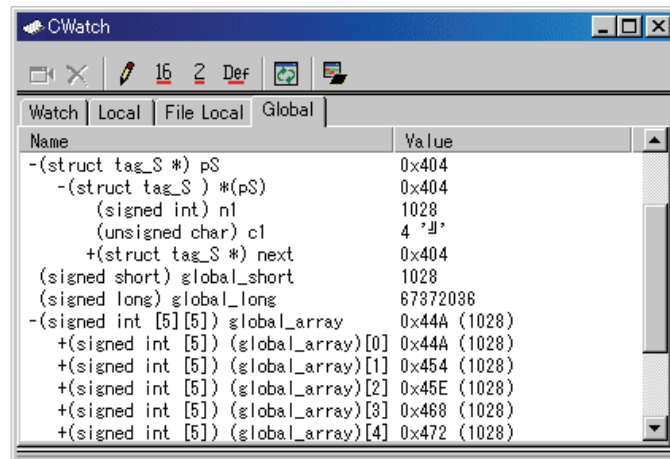
5.2.1 Extended Menus

This window has the following popup menus that can be brought up by right-clicking in the window.

Menu		Function
Add...		Add watchpoint.
Add Bit...		Add bit-level watchpoint.
Remove		Remove the selected watchpoint.
Remove All		Remove all watchpoints.
Set...		Set new data to selected watchpoint.
Radix	Bin	Display in Binary.
	Dec	Display in Decimal.
	Hex	Display in Hexadecimal.
Refresh		Refresh memory data.
Layout	Address Area	Switch display or non-display of Address area.
	Size Area	Switch display or non-display of Size area.
RAM Monitor	Enable RAM Monitor	Switch enable or disable RAM monitor function.
	Sampling Period...	Set RAM monitor sampling period.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

5.3 C Watch Window

The C Watch Window displays C/C++ expressions and their values (results of calculations). The C/C++ expressions displayed in the C Watch Window are known as C watchpoints. The displays of the results of calculating the C watchpoints are updated each time a command is executed.



- Variables can be inspected by scope (local, file local or global).
- The display is automatically updated at the same time the PC value changes.
- Variable values can be changed.
- The display radix can be changed for each variable individually.
- Any variable can be registered to the Watch tab, so that it will be displayed at all times:
 - The registered content is saved for each project separately.
 - If two or more of the C watch window are opened at the same time, the registered
- The C watchpoints can be registered to separate destinations by adding Watch tabs.
- Variables can be registered from another window or editor by a drag-and-drop operation.
- The C watchpoints can be sorted by name or by address.

ATTENTION

- You cannot change the values of the C watch points listed below:
 - Bit field variables
 - Register variables
 - C watch point which does not indicate an address (invalid C watch point)
- If a C/C++ language expression cannot be calculated correctly (for example, when a C/C++ symbol has not been defined), it is registered as invalid C watch point. It is displayed as "--<not active>--". If that C/C++ language expression can be calculated correctly at the second time, it becomes an effective C watch point.
- The display settings of the Local, File Local and Global tabs are not saved. The contents of the Watch tab and those of newly added tabs are saved.

5.3.1 Extended Menus

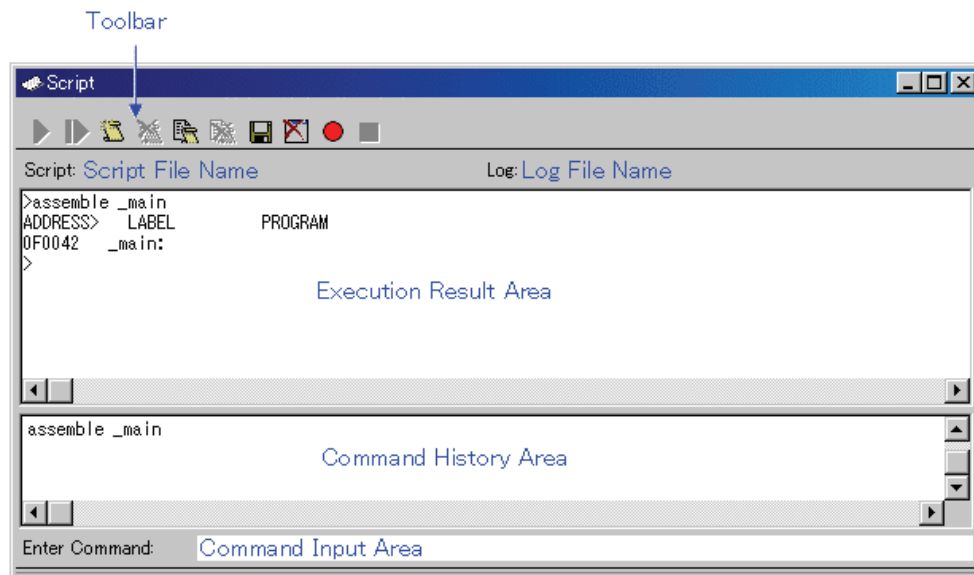
This window has the following popup menus that can be brought up by right-clicking in the window.

Menu		Function
Add...		Add C watchpoint.
Remove		Remove the selected C watchpoint.
Remove All		Remove all C watchpoints.
Initialize		Reevaluates the selected C watchpoint.
Set New Value...		Set new data to selected C watchpoint.
Radix	Hex	Display in Hexadecimal.
	Bin	Display in Binary.
	Default	Display in Default Radix.
	Toggle(All Variables)	Change radix (toggle).
Refresh		Refresh memory data.
Hide type name		Hide type names from variables.
Show char* as string		Selects whether to display char* type as a string.
Sort	Sort by Name	Sort variables by its name.
	Sort by Address	Sort variables by its address.
RAM Monitor	Enable RAM Monitor	Switch enable or disable RAM monitor function.
	Sampling Period...	Set RAM monitor sampling period.
Add New Tab...		Add new tab.
Remove Tab		Remove the selected tab.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

5.4 Script Window

The Script Window displays the execution of text -format script commands and the results of that execution.

Script commands can be executed using a script file or interactively. You can also write script commands in the script file so that they are automatically executed. The results of script command execution can also be stored in a previously specified log file.



- The Script Window has a view buffer that stores the results of executing the last 1000 lines. The results of execution can therefore be stored in a file (view file) without specifying a log file.
- When a script file is opened, the command history area changes to become the script file display area and displays the contents of the script file. When script files are nested, the contents of the last opened script file are displayed. The script file display area shows the line currently being executed in inverse vide.
- When a script file is open, you can invoke script commands from the command input area provided the script file is not being executed.
- The Script Window can record the history of the executed commands to a file. This function is not the same as the log function. This function records not the result but only the executed commands, so the saved files can be used as the script files.

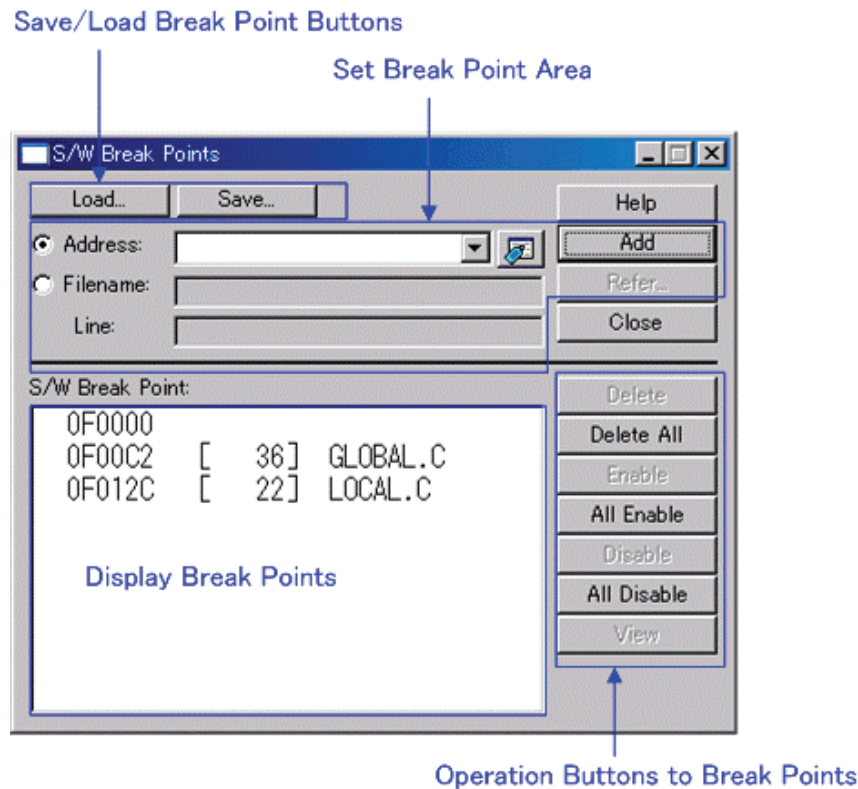
5.4.1 Extended Menus

This window has the following popup menus that can be brought up by right-clicking in the window.

Menu		Function
Script	Open...	Open script file.
	Run	Run script file.
	Step	One step execution of script file.
	Close	Close script file.
View	Save...	Save view buffer to file.
	Clear	Clear view buffer.
Log	On...	Open log file and start recording (start output to file).
	Off	Close log file and end recording (stop output to file).
Record	On...	Record the executed commands to a file.
	Off	Stop recording the executed commands.
Copy		Copy the selection and put it on the Clipboard.
Paste		Insert Clipboard contents.
Cut		Cut the selection and put it on the Clipboard.
Delete		Erase the selection.
Undo		Undo the last action.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

5.5 S/W Break Point Setting Window

The S/W Break Point Setting window allows you to set software break points. Software breaks stop the execution of instructions immediately before the specified break point.



- If you have set multiple software breakpoints, program execution stops when any one software break address is encountered (OR conditions).
- You can continue to set software breakpoints until you click the "Close" button to close the S/W Break Point Setting Window.
- You can clear, enable or disable software breakpoints selected by clicking in the software breakpoint display area. You can also enable and disable software breakpoints by double-clicking on them.
- Click on the "Save" button to save the software break points in the file. To reload software break point settings from the saved file, click the "Load" button. If you load software break points from a file, they are added to any existing break points.

5.5.1 Command Button

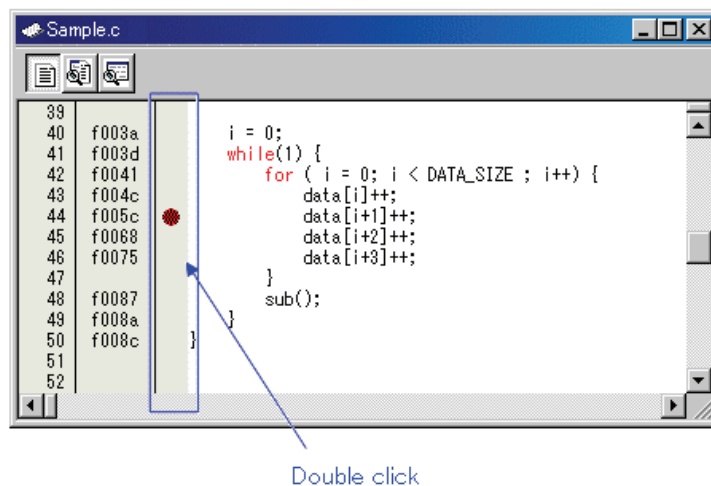
The buttons on this window has the following meanings.

Button	Function
Load...	Load setting information from a file in which it was saved.
Save...	Save the contents set in the window to a file.
Help	Display the help of this window.
Add	Add the break point.
Refer...	Open file selection dialog box.
Close	Close the window.
Delete	Remove the selected break point.
Delete All	Remove all break points.
Enable	Enable the selected break points.
All Enable	Enable all break points.
Disable	Disable the selected break point.
All Disable	Disable all break points.
View	Shows the selected breakpoint positions in the Editor(Source) window.

5.5.2 Setting and Deleting a Break Points from Editor(Source) Window

The area which can be set in the software breakpoint is different according to the product. Please refer to "10.1.2 Area where software breakpoint can be set" for details.

You can set break points in the Editor(Source) Window. To do so, double-click the break point setting area ("S/W breakpoints" column) for the line in which you want to set the break. (A red marker is displayed on the line to which the break point was set.)

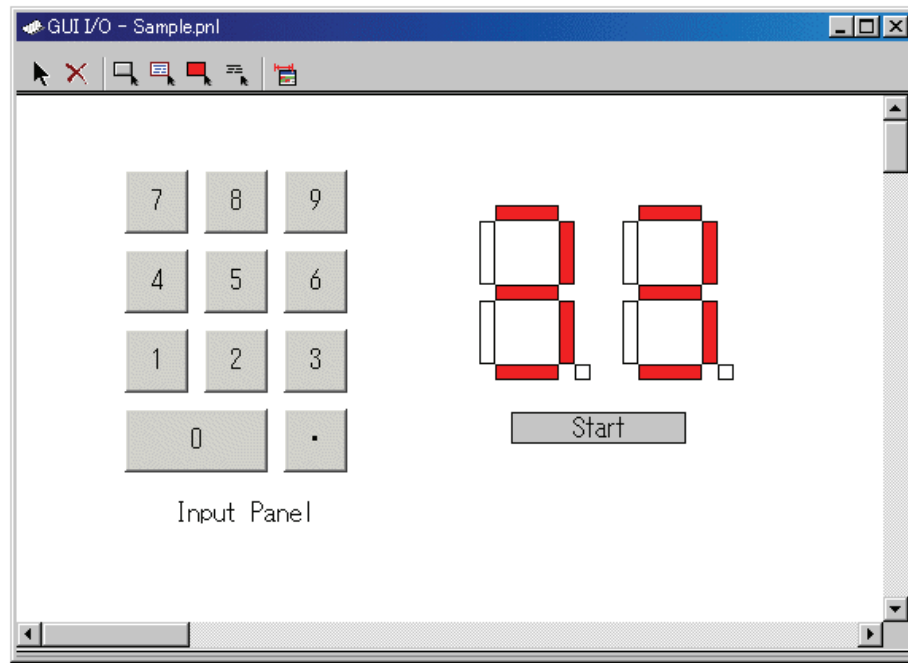


You can delete the break point by double-clicking again in the break point setting area ("S/W breakpoints" column).

In the Editor(Source) window, a display of "S/W breakpoints" column is set to "Enable" by default. To erase this column, deselect the [S/W breakpoints] check box in the dialog box opened by choosing the main menu - [Edit] -> [Define Column Format]. The "S/W breakpoints" column is erased from all Editor (Source) windows. And select popup menu - [Columns] -> [S/W breakpoints] in the Editor (Source) window, A column can be set up for each Editor (Source) windows.

5.6 GUI I/O Window

The GUI I/O window allows you for port input by creating a user target system key input panel (button) in the window and clicking the created button. And this window also allows you to implement the user target system output panel in the window.



- You can arrange the following parts on the window.
 - Label (character string)
Displays/erases a character string specified by the user when any value is written to the specified address (bit).
 - LED
Changes the display color of any area when any value is written to the specified address (bit). (Substitution for LED ON)
 - Button
A virtual port input can be executed at the time the button is pressed.
 - Text
Display the text string.
- You can also save the created panel in a file and reload it.
- You can set up to 200 address points to the created part. If different addresses are set to the individual parts, you can arrange up to 200 parts.

5.6.1 Extended Menus

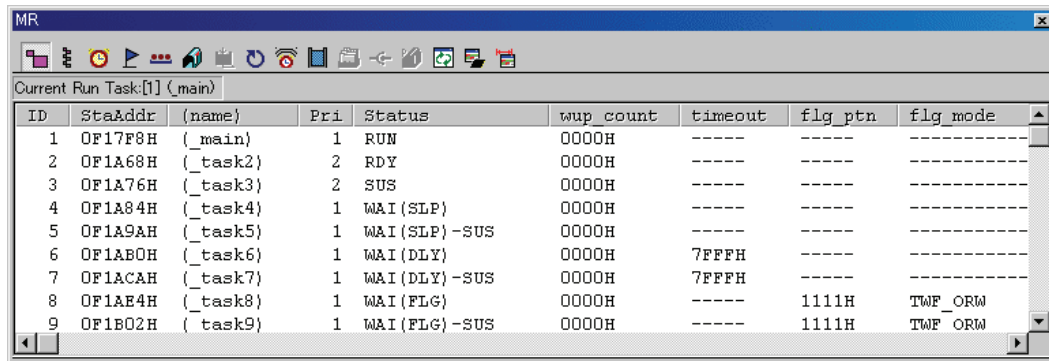
This window has the following popup menus that can be brought up by right-clicking in the window.

Menu	Function
Select Item	Select an I/O item.
Delete	Delete the selected I/O item.
Copy	Copy the selected I/O item.
Paste	Paste the copied I/O item.
Create Button	Create a new button item.
Create Label	Create a new label item.
Create LED	Create a new LED item.
Create Text	Create a new text item.
Display grid	Display the grid line.
Save...	Save I/O panel file.
Load...	Load I/O panel file.
Sampling Period...	Set RAM monitor sampling period.
Toolbar display	Display toolbar.
Customize toolbar...	Open toolbar customize dialog box.
Allow Docking	Allow window docking.
Hide	Hide window.

5.7 MR Window

Use the MR Window to display the status of the realtime OS.

You can only use the MR Window when you have downloaded a program that uses the realtime OS (if the downloaded program does not use the MR, nothing is displayed in the MR Window when it is opened.)



The screenshot shows the MR window with a toolbar at the top and a table of task status below. The table has columns for ID, StaAddr, (name), Pri, Status, wup_count, timeout, flg_ptn, and flg_mode. The current run task is (_main).

ID	StaAddr	(name)	Pri	Status	wup_count	timeout	flg_ptn	flg_mode
1	0F17F8H	(_main)	1	RUN	0000H	----	----	-----
2	0F1A68H	(_task2)	2	RDY	0000H	----	----	-----
3	0F1A76H	(_task3)	2	SUS	0000H	----	----	-----
4	0F1A84H	(_task4)	1	WAI (SLP)	0000H	----	----	-----
5	0F1A9AH	(_task5)	1	WAI (SLP) -SUS	0000H	----	----	-----
6	0F1AB0H	(_task6)	1	WAI (DLY)	0000H	7FFFH	----	-----
7	0F1ACAH	(_task7)	1	WAI (DLY) -SUS	0000H	7FFFH	----	-----
8	0F1AE4H	(_task8)	1	WAI (FLG)	0000H	----	1111H	TWF_ORW
9	0F1B02H	(_task9)	1	WAI (FLG) -SUS	0000H	----	1111H	TWF_ORW

- You can open the MR window as many as the number of display modes .
- By clicking the desired button, the MR window display mode changes and the display data also changes.
- By double-clicking the desired task line, you can display the context data of the task.
- You can drag the cursor to change the width of the display area in each mode.
- If the downloaded program does not use MR, you cannot select all the menu which will select the display mode.
- The supported display mode is as follows.

If a target program created by MR30 or MR308 conformed to uITRON4 is downloaded, this window supports the displays listed below.

- Task status
- Ready queue status
- Timeout queue status
- Event flag status
- Semaphore status
- Mailbox status
- Data queue status
- Cyclic handler status
- Alarm handler status
- Memory pool status

If a target program created by MR30 or MR308 conformed to uITRON3 is downloaded, this window supports the displays listed below.

If a target program created on MR30 V.1.00 is downloaded, the MPL mode cannot be used on MR30. (You cannot select the menu which changes the current mode to the MPL mode.)

- Task status
- Ready queue status
- Timeout queue status
- Event flag status
- Semaphore status
- Mailbox status
- Cyclic handler status
- Alarm handler status
- Memory pool status

ATTENTION

Please use the startup file (crt0mr.axx/start.axx) whose contents matches with the version of MRxx, when you make downloaded program. The MR Window and MR command will not run properly if the startup file you uses don't match with the version of MRxx.

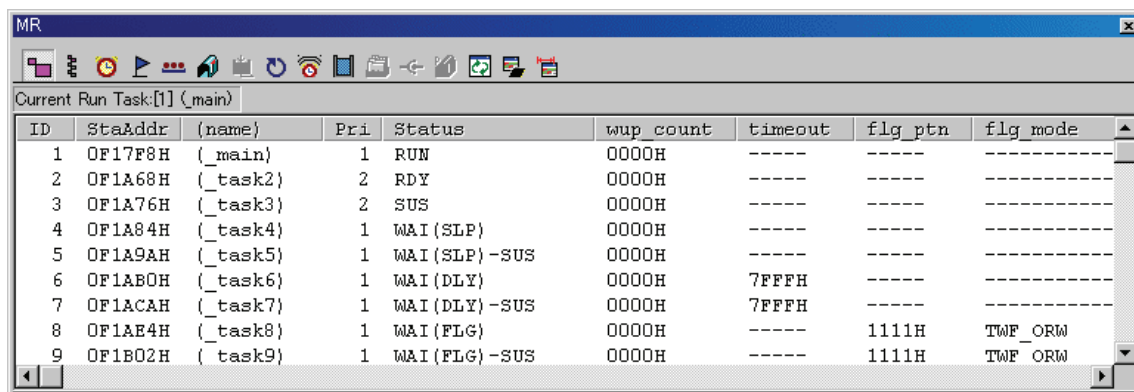
5.7.1.1 Extended Menus

This window has the following popup menus that can be brought up by right-clicking in the window.

Menu		Function
Mode	Task	Displays Task status.
	Ready Queue	Displays Ready status.
	Timeout Queue	Displays Timeout status.
	Event Flag	Displays Event Flag status.
	Semaphore	Displays Semaphore status.
	Mailbox	Displays Mailbox status.
	Data Queue	Displays Data Queue status.
	Cyclic Handler	Displays Cyclic Handler status.
	Alarm Handler	Displays Alarm Handler status.
	Memory Pool	Displays Memory Pool status.
	Message Buffer	Displays Message Buffer status.
	Port	Displays Port status.
	Mailbox(with Priority)	Displays Mailbox(with Priority) status.
Context...		Displays Context.
Layout	Status Bar	Switch display or non-display of status bar.
Refresh		Refresh memory data.
RAM Monitor	Enable RAM Monitor	Switch enable or disable RAM Monitor function.
	Sampling Period...	Set RAM Monitor sampling period.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

5.7.2 Display the Task Status

In the MR window, select Popup Menu - [Mode] -> [Task].



ID	StaAddr	(name)	Pri	Status	wup_count	timeout	flg_ptn	flg_mode
1	0F17F8H	(_main)	1	RUN	0000H	----	----	----
2	0F1A68H	(_task2)	2	RDY	0000H	----	----	----
3	0F1A76H	(_task3)	2	SUS	0000H	----	----	----
4	0F1A84H	(_task4)	1	WAI(SLP)	0000H	----	----	----
5	0F1A9AH	(_task5)	1	WAI(SLP)-SUS	0000H	----	----	----
6	0F1AB0H	(_task6)	1	WAI(DLY)	0000H	7FFFH	----	----
7	0F1ACAH	(_task7)	1	WAI(DLY)-SUS	0000H	7FFFH	----	----
8	0F1AE4H	(_task8)	1	WAI(FLG)	0000H	----	1111H	TWF_ORW
9	0F1B02H	(_task9)	1	WAI(FLG)-SUS	0000H	----	1111H	TWF_ORW

By double-clicking any line, the information on the task context is displayed in the Context dialog. For details on the Context dialog, see "5.7.12 Display the Task Context"

The following data is displayed in the status bar.

Current Run Task:[1] (_main)

5.7.2.1 Display the Task Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

All the tasks defined in the configuration are listed in the order of ID number. The function of each item is as described below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Items	Contents
ID	Task ID
StaAddr	Starting address of task
(name)	Task name
Pri	Priority
Status*1	Task status
wup_count	Wake-up count
timeout	Timeout value
flg_ptn	Wait bit pattern of event flag
flg_mode*2	Wait cancellation condition of event flag

- *1Task Status

Display	Status
RUN	Run status
RDY	Ready Status
SUS	Suspend status
DMT	Dormant status
WAI(SLP)	Sleep state
WAI(SLP)-SUS	Sleep state(double wait)
WAI(DLY)	Time wait state duw to dly_tsk
WAI(DLY)-SUS	Time wait state duw to dly_tsk(double wait)
WAI(FLG)	Event flagwait status
WAI(FLG)-SUS	Event flagwait status(double wait)
WAI(SEM)	Semaphore wait status
WAI(SEM)-SUS	Semaphore wait status(double wait)
WAI(MBX)	Message wait status
WAI(MBX)-SUS	Message wait status(double wait)
WAI(SLP-TMO)	Sleep state with time-out
WAI(SLP-TMO)-SUS	Sleep state with time-out(double wait)
WAI(FLG-TMO)	Event flag wait state with time-out
WAI(FLG-TMO)-SUS	Event flag wait state with time-out(double wait)
WAI(SEM-TMO)	Semaphore wait state with time-out
WAI(SEM-TMO)-SUS	Semaphore wait state with time-out(double wait)
WAI(MBX-TMO)	Message wait state with time-out
WAI(MBX-TMO)-SUS	Message wait state with time-out(double wait)

- *2Display the Wait Cancellation Condition of Event Flag

flg_mode	Status
TWF_ANDW	Waits for all bits set in the wait bit pattern to be set (AND wait)
TWF_ANDW+TWF_CLR	Clears the event flag to 0 when an AND wait has occurred and the task wait status has been cancelled
TWF_ORW	Waits for any one bit set in the wait bit pattern to be set (OR wait)
TWF_ORW+TWF_CLR	Clears the event flag to 0 when an OR wait has occurred and the task wait status has been cancelled

5.7.2.2 Display the Task Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

All the tasks defined in the configuration are listed in the order of ID number. The function of each item is as described below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Items	Contents
ID	Task ID
Name	Task name
Pri	Priority
Status*1	Task status
Wupcnt	Wake-up count
Actcnt	Activated count
Tmout	Timeout value
Flgptn	Wait bit pattern of event flag
Wfmode*2	Wait cancellation condition of event flag

- *1Task Status

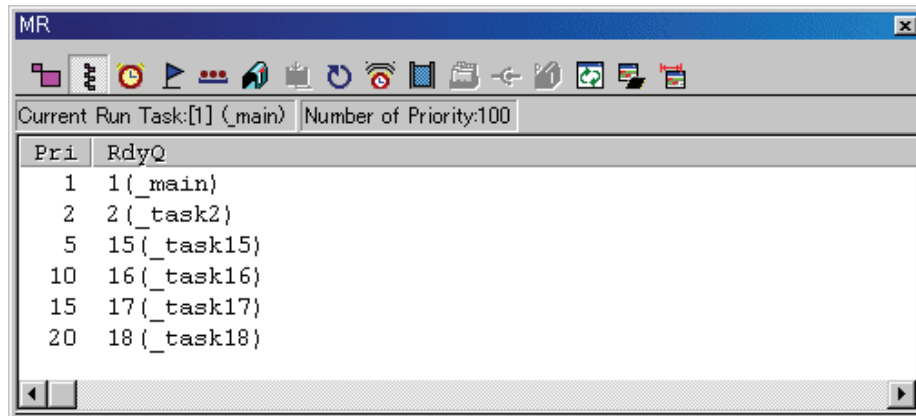
Display	Status
RUN	Run status
RDY	Ready Status
SUS	Suspend status
DMT	Dormant status
WAI(SLP)	Sleep state
WAI(SLP)-SUS	Sleep state(double wait)
WAI(DLY)	Time wait state due to dly_tsk
WAI(DLY)-SUS	Time wait state due to dly_tsk(double wait)
WAI(FLG)	Event flagwait status
WAI(FLG)-SUS	Event flagwait status(double wait)
WAI(SEM)	Semaphore wait status
WAI(SEM)-SUS	Semaphore wait status(double wait)
WAI(MBX)	Message wait status
WAI(MBX)-SUS	Message wait status(double wait)
WAI(SDTQ)	Transmission data wait status
WAI(SDTQ)-SUS	Transmission data wait status(double wait)
WAI(RDTQ)	Reception data wait status
WAI(RDTQ)-SUS	Reception data wait status(double wait)
WAI(VSDTQ)	Transmission extended data wait status
WAI(VSDTQ)-SUS	Transmission extended data wait status(double wait)
WAI(VRDTQ)	Reception extended data wait status
WAI(VRDTQ)-SUS	Reception extended data wait status(double wait)
WAI(MPF)	Fixed length memory pool wait
WAI(MPF)-SUS	Fixed length memory pool wait(double wait)
WAI(SLP-TMO)	Sleep state with timeout
WAI(SLP-TMO)-SUS	Sleep state with timeout(double wait)
WAI(FLG-TMO)	Event flag wait state with timeout
WAI(FLG-TMO)-SUS	Event flag wait state with timeout(double wait)
WAI(SEM-TMO)	Semaphore wait state with timeout
WAI(SEM-TMO)-SUS	Semaphore wait state with timeout(double wait)
WAI(MBX-TMO)	Message wait state with time-out
WAI(MBX-TMO)-SUS	Message wait state with timeout(double wait)
WAI(SDTQ-TMO)	Transmission data with timeout wait status
WAI(SDTQ-TMO)-SUS	Transmission data with timeout wait status(double wait)
WAI(RDTQ-TMO)	Reception data with timeout wait status
WAI(RDTQ-TMO)-SUS	Reception data with timeout wait status(double wait)
WAI(VSDTQ-TMO)	Transmission extended data with timeout wait status
WAI(VSDTQ-TMO)-SUS	Transmission extended data with timeout wait status(double wait)
WAI(VRDTQ-TMO)	Reception extended data with timeout wait status
WAI(VRDTQ-TMO)-SUS	Reception extended data with timeout wait status(double wait)
WAI(MPF-TMO)	Fixed length memory pool with timeout wait
WAI(MPF-TMO)-SUS	Fixed length memory pool with timeout wait(double wait)

- *2Display the Wait Cancellation Condition of Event Flag

Wfmode	Status
TWF_ANDW	Waits for all bits set in the wait bit pattern to be set (AND wait)
TWF_ORW	Waits for any one bit set in the wait bit pattern to be set (OR wait)

5.7.3 Display the Ready Queue Status

In the MR window, select Popup Menu - [Mode] -> [Ready Queue].



The following data is displayed in the status bar.

Current Run Task:[1] (_main)	Number of Priority:100
------------------------------	------------------------

5.7.3.1 Display the Ready Queue Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

The function of each item is as described below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
Pri	Displays priority
RdyQ	Shows the ID Nos. and task names of tasks in the ready queue

- Up to 8 characters of the task name is displayed in the RdyQ field. When the task name exceeds 8 characters, the extra characters are omitted.

5.7.3.2 Display the Ready Queue Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

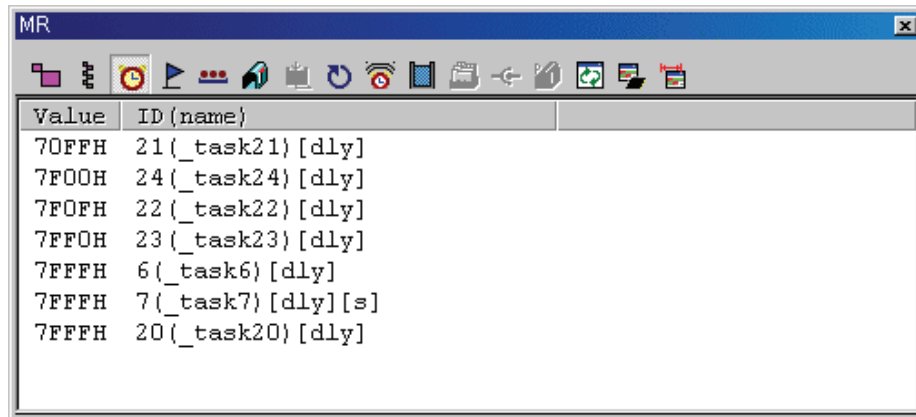
The function of each item is as described below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Item	Contents
Pri	Displays priority
Ready Queue	Shows the ID Nos. and task names of tasks in the ready queue

- Up to 8 characters of the task name is displayed in the Ready Queue field. When the task name exceeds 8 characters, the extra characters are omitted.

5.7.4 Display the Timeout Queue Status

In the MR window, select Popup Menu - [Mode] -> [Timeout Queue].



5.7.4.1 Display the Timeout Queue Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

The function of each item is as described below.

Tasks waiting at present are displayed in the descending order of timeout value. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
Value	Shows the timeout value of each task
ID(name)	Shows the ID No. and task name of the tasks in the timeout queue

- Following character strings are used to indicate the type of wait state.

Character string	Wait state
[slp]	Wait due to tslp_tsk
[dly]	Wait due to dly_tsk
[flg]	Wait due to twai_flg
[sem]	Wait due to twai_sem
[mbx]	Wait due to trcv_msg

- When a task connected to the timeout queue is in the state of forced waiting (double waiting), a string "[s]", which indicates double waiting, is appended to a string displayed in the ID (name) field.

Normal display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]

5.7.4.2 Display the Timeout Queue Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

The function of each item is as described below.

Tasks waiting at present are displayed in the descending order of timeout value. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Item	Contents
Tmout	Shows the timeout value (ms) of each task
ID(Name)	Shows the ID No. and task name of the tasks in the timeout queue

- Following character strings are used to indicate the type of wait state.

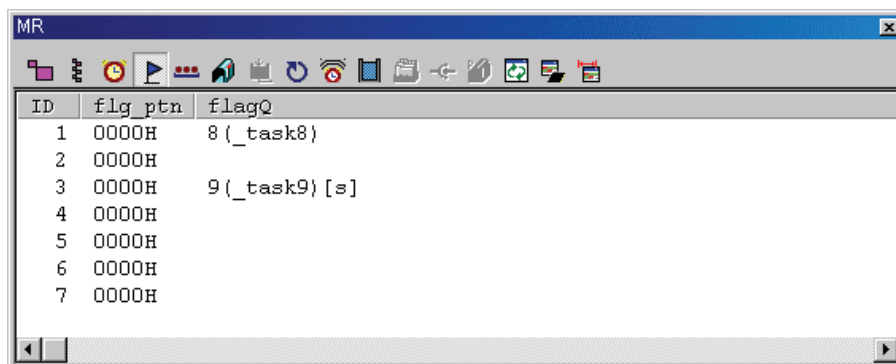
Character string	Wait state
[slp]	Wait due to tslp_tsk
[dly]	Wait due to dly_tsk
[flg]	Wait due to twai_flg
[sem]	Wait due to twai_sem
[mbx]	Wait due to trcv_mbx
[mpf]	Wait due to tget_mpf
[sdtq]	Wait due to tsnd_dtq
[rdtq]	Wait due to trcv_dtq
[vsdtq]	Wait due to vtsnd_dtq
[vrdtq]	Wait due to vtrcv_dtq

- When a task connected to the timeout queue is in the state of forced waiting (double waiting), a string "[s]", which indicates double waiting, is appended to a string displayed in the ID(Name) field.

Normal display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]

5.7.5 Display the Event Flag Status

In the MR window, select Popup Menu - [Mode] -> [Event Flag].



5.7.5.1 Display the Event Flag Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

All the event flags defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
ID	ID No. of event flag
flg_ptn	Bit pattern of each event flag
flagQ	Task ID Nos. and task names in the event flag queue

- When a task connected to the event flag queue is in the state of waiting with timeout enabled (waiting in twai_flg), a string "[tmo]", which indicates a state of waiting with timeout enabled, is appended to a string displayed in the flag Q field.
When a task connected to the event flag queue is in the state of forced waiting (double waiting), a string "[s]", which indicates double waiting, is appended to a string displayed in the flag Q field.

Normal Display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]
Display when in WAIT-SUSPEND with time out	26(_task26)[tmo][s]

- Up to 8 characters can be displayed in the task name in the flag Q field. If a task name exceeds 8 characters, the extra characters are omitted.

5.7.5.2 Display the Event Flag Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

All the event flags defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Item	Contents
ID	ID No. of event flag
Flgatr	Attribute of each event flag
Flgpbn	Bit pattern of each event flag
Flag Queue	Task ID Nos. and task names in the event flag queue

- The following are displayed in the Flgatr area:

TA_TFIFO	Task wait queue is in FIFO order
TA_TPRI	Task wait queue is in task priority order
TA_WSGI	Only one task is allowed to be in the waiting state for the eventflag
TA_WMUL	Multiple tasks are allowed to be in the waiting state for the eventflag
TA_CLR	Eventflag's bit pattern is cleared when a task is released from the waiting state for that eventflag

- When a task connected to the event flag queue is in the state of waiting with timeout enabled (waiting in twai_flg), a string "[tmo]", which indicates a state of waiting with timeout enabled, is appended to a string displayed in the Flag Queue field.

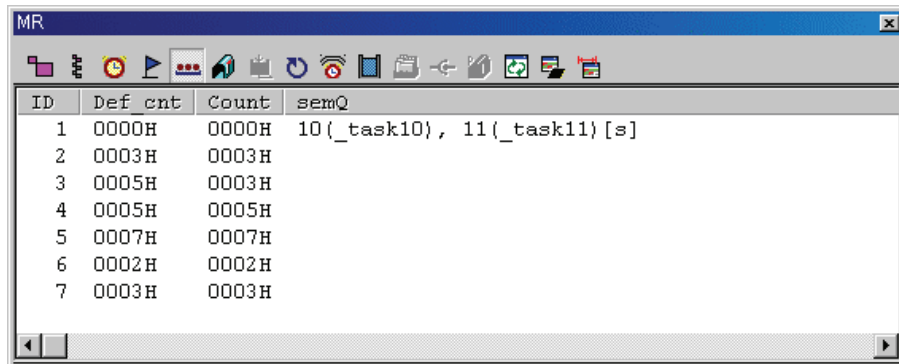
When a task connected to the event flag queue is in the state of forced waiting (double waiting), a string "[s]", which indicates double waiting, is appended to a string displayed in the Flag Queue field.

Normal Display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]
Display when in WAIT-SUSPEND with time out	26(_task26)[tmo][s]

- Up to 8 characters can be displayed in the task name in the Flag Queue field.
If a task name exceeds 8 characters, the extra characters are omitted.

5.7.6 Display the Semaphore Status

In the MR window, select Popup Menu - [Mode] -> [Semaphore].



5.7.6.1 Display the Semaphore Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

All the SEMs defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
ID	ID No. of semaphore
Def_cnt	Default value of semaphore counter
Count	Semaphore count
semQ	Task ID Nos. and task names in the semaphore queue

- When a task connected to the SEM queue is in the state of waiting with timeout enabled (waiting in twai_sem), a string "[tmo]", which indicates a state of waiting with timeout enabled, is appended to a string displayed in the semQ field.
When a task connected to the SEM queue is in the state of forced waiting (double waiting), a string "[s]", which indicates double waiting, is appended to a string displayed in the semQ field.

Normal Display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]
Display when in WAIT-SUSPEND with time out	26(_task26)[tmo][s]

- Up to 8 characters can be displayed in the task name in the semQ field.
If a task name exceeds 8 characters, the extra characters are omitted.

5.7.6.2 Display the Semaphore Status (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

All the SEMs defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Item	Contents
ID	ID No. of semaphore
Sematr	Attribute of each semaphore
Semcnt	Semaphore count
Semaphore Queue	Task ID Nos. and task names in the semaphore queue

- The following are displayed in the Sematr area:

TA_TFIFO	Task wait queue is in FIFO order
TA_TPRI	Task wait queue is in task priority order

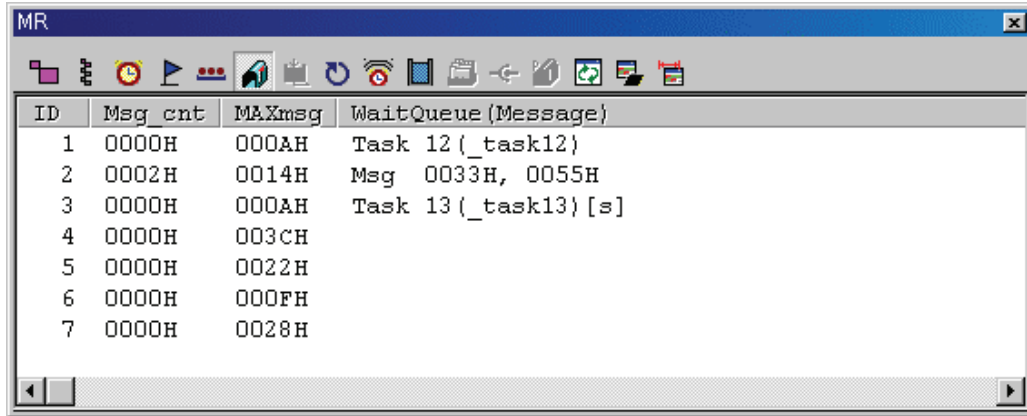
- When a task connected to the SEM queue is in the state of waiting with timeout enabled (waiting in twai_sem), a string "[tmo]", which indicates a state of waiting with timeout enabled, is appended to a string displayed in the Semaphore Queue field.
When a task connected to the SEM queue is in the state of forced waiting (double waiting), a string "[s]", which indicates double waiting, is appended to a string displayed in the Semaphore Queue field.

Normal Display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]
Display when in WAIT-SUSPEND with time out	26(_task26)[tmo][s]

- Up to 8 characters can be displayed in the task name in the Semaphore Queue field. If a task name exceeds 8 characters, the extra characters are omitted.

5.7.7 Display the Mailbox Status

In the MR window, select Popup Menu - [Mode] -> [Mailbox].



5.7.7.1 Display the Mailbox Status (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

All the mail boxes defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
ID	ID No. of mailbox
Msg_cnt	Number of messages in each mailbox
MAXmsg	Maximum number of messages that can be contained in each mailbox
Wait Queue(Message)	The messages stored in the mailbox or ID No. and task name of tasks waiting for messages

- The WaitQueue (Message) field shows a string "Msg" when a message is stored (when Msg_cnt as described above is non-zero), and then displays the stored message.
When no message is stored (when Msg_cnt is zero), the WaitQueue field displays a string "Task" if a task waiting for a message exists, and then displays the ID number and name of the task waiting for a message.
- When a task connected to the mail box queue is in the state of waiting with timeout enabled (waiting in trcv_msg), a string "[tmo]", which indicates the state of timeout enabled, is appended to a string displayed in the WaitQueue (Message) field.

When a task connected to the mail box queue is in the state of forced waiting (Double waiting), a string "[s]", which indicates the state of double waiting, is appended to a string displayed in the WaitQueue (Message) field.

Normal Display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]
Display when in WAIT-SUSPEND with time out	26(_task26)[tmo][s]

- Up to 8 characters can be displayed in the task name in the WaitQueue (Message) field.
If a task name exceeds 8 characters, the extra characters are omitted.

5.7.7.2 Display the Mailbox Status (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

All the mail boxes defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Item	Contents
ID	ID No. of mailbox
Mbxatr	Attribute of each mailbox
Mailbox Queue (Wait)	ID No. and task name of tasks waiting for messages
Mailbox Queue (Message)	The messages stored in the mailbox

- The following are displayed in the Mbxatr area:

TA_TFIFO	Task wait queue is in FIFO order
TA_TPRI	Task wait queue is in task priority order
TA_MFIFO	Message queue is in FIFO order
TA_MPRI	Message queue is in message priority order

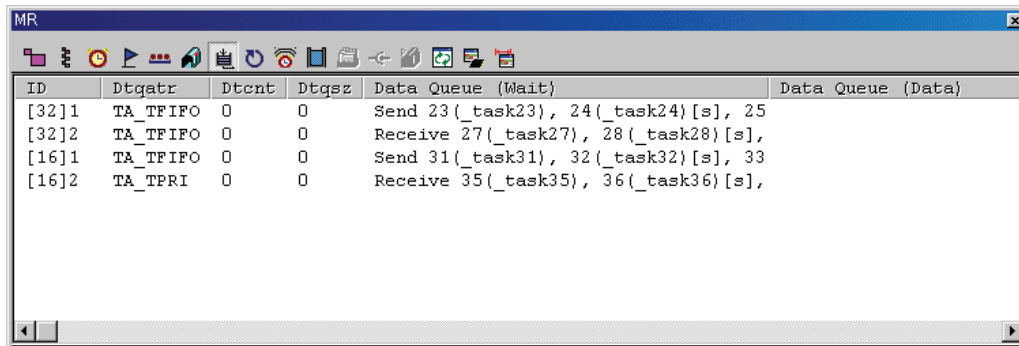
- When a task connected to the mail box queue is in the state of waiting with timeout enabled (waiting in `trcv_mbx`), a string "[tmo]", which indicates the state of timeout enabled, is appended to a string displayed in the Mailbox Queue (Wait) field.
When a task connected to the mail box queue is in the state of forced waiting (Double waiting), a string "[s]", which indicates the state of double waiting, is appended to a string displayed in the Mailbox Queue (Wait) field.

Normal Display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]
Display when in WAIT-SUSPEND with time out	26(_task26)[tmo][s]

- Up to 8 characters can be displayed in the task name in the Mailbox Queue (Wait) field.
If a task name exceeds 8 characters, the extra characters are omitted.

5.7.8 Display the Data Queue Status

In the MR window, select Popup Menu - [Mode] -> [Data Queue].



5.7.8.1 Display the Data Queue Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

All the data queues defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Item	Contents
ID	ID No. of data queue
Dtqatr	Attribute of each data queue
Dtcnt	Number of messages in each data queue
Dtqsz	Maximum number of messages that can be contained in each data queue
Data Queue (Wait)	ID No. and task name of tasks waiting for message transmission waiting or message reception waiting
Data Queue (Data)	The messages stored in the data queue

- The display of the ID field varies depending on which one is specified, the standard data(32 bits) or the extended data(16 bits).

MR308/4

- If the standard data(32 bits), the ID field displays a string "[32]" and data queue ID number.
- If the extended data(16 bits), the ID field displays a string "[16]" and data queue ID number.

MR30/4

- If the standard data(16 bits), the ID field displays a string "[16]" and data queue ID number.
- If the extended data(32 bits), the ID field displays a string "[32]" and data queue ID number.

-
- The following are displayed in the Dtqatr area:

TA_TFIFO	Task wait queue is in FIFO order
TA_TPRI	Task wait queue is in task priority order

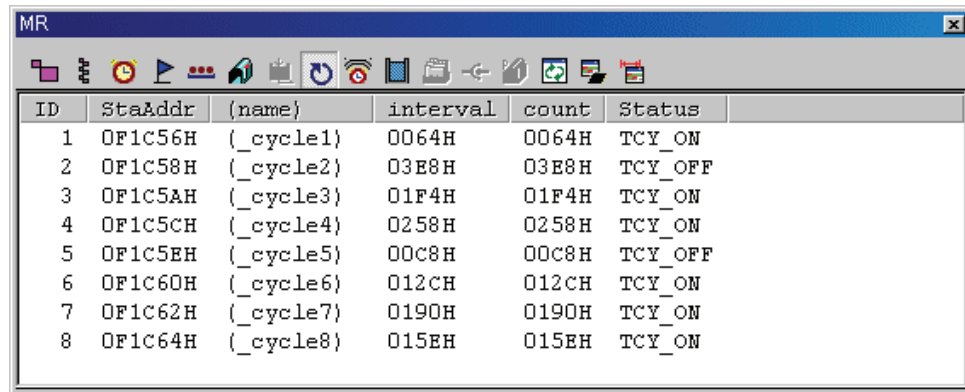
- The Data Queue (Wait) field displays a string "Send" if a task waiting for a message sending, and then displays the ID number and name of the task waiting for a message sending. Also, if a task waiting for a message receiving, displays a string "Receive" and then displays the ID number and name of the task waiting for a message receiving.
- When a task connected to the data queue is in the state of waiting with timeout enabled , a string "[tmo]", which indicates the state of timeout enabled, is appended to a string displayed in the Data Queue (Wait) field.
When a task connected to the data queue is in the state of forced waiting (Double waiting), a string "[s]", which indicates the state of double waiting, is appended to a string displayed in the Data Queue (Wait) field.

Normal Display	26(_task26)
Display when in WAIT-SUSPEND	26(_task26)[s]
Display when in WAIT-SUSPEND with time out	26(_task26)[tmo][s]

- Up to 8 characters can be displayed in the task name in the Data Queue (Wait) field.
If a task name exceeds 8 characters, the extra characters are omitted.

5.7.9 Display the Cycle Handler Status

In the MR window, select Popup Menu - [Mode] -> [Cyclic Handler].



ID	StaAddr	(name)	interval	count	Status
1	0F1C56H	(_cycle1)	0064H	0064H	TCY_ON
2	0F1C58H	(_cycle2)	03E8H	03E8H	TCY_OFF
3	0F1C5AH	(_cycle3)	01F4H	01F4H	TCY_ON
4	0F1C5CH	(_cycle4)	0258H	0258H	TCY_ON
5	0F1C5EH	(_cycle5)	00C8H	00C8H	TCY_OFF
6	0F1C60H	(_cycle6)	012CH	012CH	TCY_ON
7	0F1C62H	(_cycle7)	0190H	0190H	TCY_ON
8	0F1C64H	(_cycle8)	015EH	015EH	TCY_ON

5.7.9.1 Display the Cycle Handler Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

All the cycle handlers defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
ID	ID No. of cycle handler
StaAddr	Starting address of cycle handler
(name)	Name of cycle handler
interval	Interrupt interval
count	Interrupt count
Status	Activity status of cycle start handler

- The following are displayed in the Status area:

TCY_ON	Cycle handler enabled
TCY_OFF	Cycle handler disabled

5.7.9.2 Display the Cycle Handler Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

All the cycle handlers defined in the configuration are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

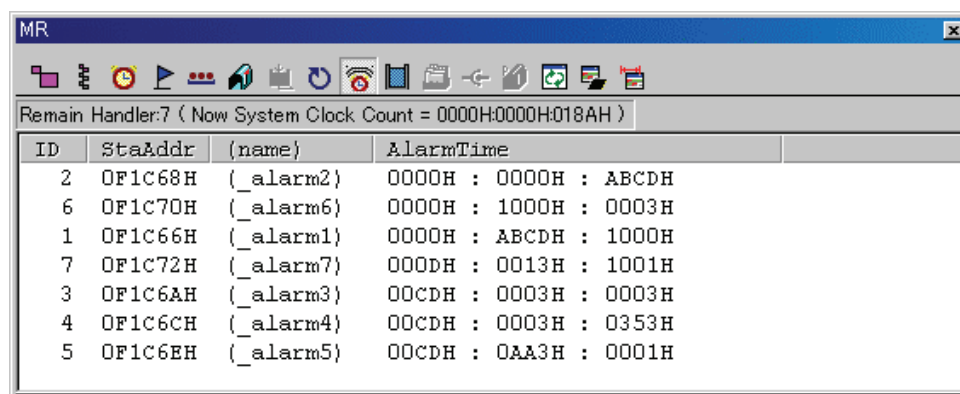
Item	Contents
ID	ID No. of cycle handler
Name	Name of cycle handler
Cycphs	The activation phase (by the millisecond)
Cyctim	The activation cycle time (by the millisecond)
Tmout	The amount of time by the millisecond remaining before the cyclic handler's next activation time
Status	Activity status of cycle start handler

- The following are displayed in the Status area:

TCYC_STA	Cycle handler is in an operational state
TCYC_STP	Cycle handler is in a non-operational state

5.7.10 Display the Alarm Handler Status

In the MR window, select Popup Menu - [Mode] -> [Alarm Handler].



When the realtime OS is MRxx conformed to uITRON specifications V.3.0, the following data is displayed in the status bar.

Remain Handler:7 (Now System Clock Count = 0000H:0000H:018AH)

5.7.10.1 Display the Alarm Handler Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Of all the cycle start handlers defined in the configuration, only those which are not started yet at present are listed in the ascending order of start time. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
ID	ID No. of alarm handler
StaAddr	Starting address of alarm handler
(name)	Name of alarm handler
AlarmTime	Starting time of alarm handler

5.7.10.2 Display the Alarm Handler Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Of all the cycle start handlers defined in the configuration, only those which are not started yet at present are listed in the ascending order of start time. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

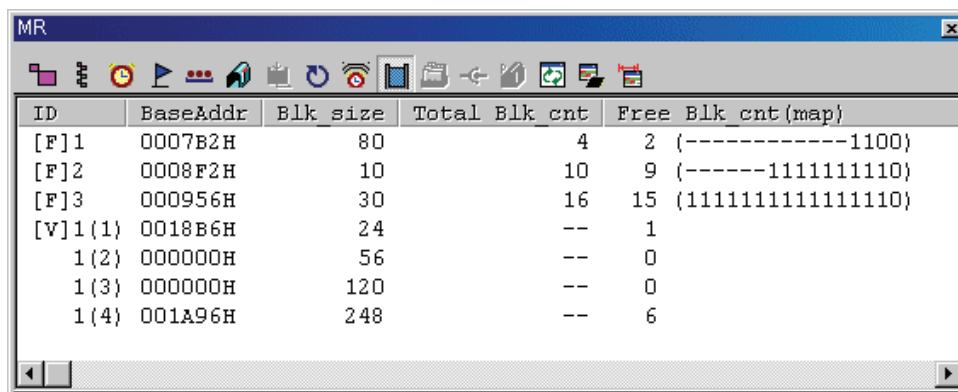
Item	Contents
ID	ID No. of alarm handler
Name	Name of alarm handler
Almtim	The amount of time by the millisecond remaining before the alarm handler's activation time
Status	Activity status of alarm handler

- The following are displayed in the Status area:

TALM_STA	Alarm handler is in an operational state
TALM_STP	Alarm handler is in a non-operational state

5.7.11 Display the Memory Pool Status

In the MR window, select Popup Menu - [Mode] -> [Memory Pool].



ID	BaseAddr	Blk size	Total Blk cnt	Free Blk cnt (map)
[F]1	0007B2H	80	4	2 (-----1100)
[F]2	0008F2H	10	10	9 (-----111111110)
[F]3	000956H	30	16	15 (111111111111110)
[V]1(1)	0018B6H	24	--	1
1(2)	000000H	56	--	0
1(3)	000000H	120	--	0
1(4)	001A96H	248	--	6

5.7.11.1 Display the Memory Pool Status(When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

All the memory pools defined in the configuration are listed in the order of ID number. (The fixed length data comes first, and the optional length data comes after the fixed length data.) The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

Item	Contents
ID	ID No. of memory pool
BaseAddr	Base address of memory pool
Blk_Size	Block size of memory pool
Total Blk_cnt	Tot a l block count of memory pool
Free Blk_cnt(map)	Number of unused blocks and information on unused memory blocks (bit information)

- The display of the ID field varies depending on which one is specified, fixed length or optional length.
 - If the data is of fixed length, the ID field displays a string "[F]" and memory pool ID number.
 - For an arbitrary length, the contents displayed on the first line are the character string "[V]," a memory pool ID number, and a block ID number. Displayed on the second to fourth lines are the memory pool ID and block ID numbers. The block ID numbers are enclosed in parentheses.
- When specifying the optional length memory pool, "--" is displayed in the Total Blk_cnt field. No bit information is displayed in the Free Blk_cnt (map) field.
- When specifying the fixed-length memory pool, the display format of each bit in the memory block information in Free Blk_cnt (map) is as shown below:

item	Contents
'0'	Memory block in use (busy)
'1'	Memory block not in use (ready)
','	No memory block

5.7.11.2 Display the Memory Pool Status(When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

All the memory pools are listed in the order of ID number. The function of each item is listed below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

Item	Contents
ID	ID No. of memory pool
Mplatr	Attribute of each memory pool
Mpladr	Base address of memory pool
Mplsz	Size of memory pool
Blkcnt	Total block count of fixed length memory pool
Fblkcnt	Number of unused blocks and information on unused memory blocks
Memory Pool Queue	Displays the ID number and name of tasks waiting in the memory pool.

- The following are displayed in the Mplatr area:

TA_TFIFO	Task wait queue is in FIFO order
TA_TPRI	Task wait queue is in task priority order

- The display of the ID field varies depending on which one is specified, fixed length or optional length.
 - If the data is of fixed length, the ID field displays a string "[F]" and memory pool ID number.
 - For an arbitrary length, the contents displayed on the first line are the character string "[V]," a memory pool ID number, and a block ID number. Displayed on the second to fourth lines are the memory pool ID and block ID numbers. The block ID numbers are enclosed in parentheses.

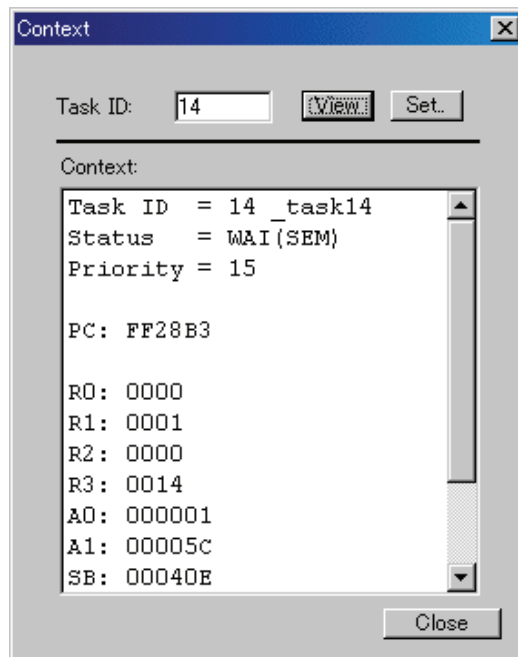
5.7.12 Display the Task Context

5.7.12.1 Display the Task Context

In the MR window, select Popup Menu - [Context...].

The Context dialog box is opened. The Context dialog box is used to reference/specify the context information of the specified task.

You can also open the Context dialog box by double-clicking the data display area in the task state display mode .



Enter the task ID number in the Task ID field and click the View button (or press the Enter key).

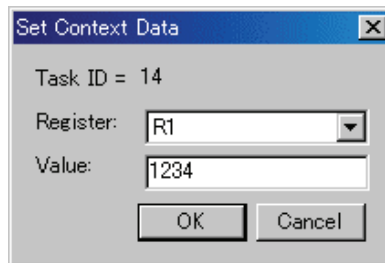
The context of the specified task appears in the Context field.

- If the task entered in the Task ID field is "RUN" or "DMT" when clicking the View button, the context is not displayed. (In the Context field, only the task ID and task state are displayed.)
- If a task ID number which does not exist is entered in the Task ID field when clicking the View button, an error occurs.

5.7.12.2 Change the task context

Enter the task ID number in the Task ID field in the Context dialog and click the Set button. The Set Context dialog is opened.

The Set Context dialog is used to set the specified context register value of the specified task.



Specify the register to be changed in the Register field list box and enter the value to be set in the "Value:" field.

If an expression description set in the "Value:" field is wrong, or if the specified value is outside the allowable range set for the specified register, an error occurs.

6. Table of Script Commands

The following script commands are prepared.

The commands with yellow color displaying can be executed at run time.

The command to which "*" adheres behind is not supported according to the product.

6.1 Table of Script Commands (classified by function)

6.1.1 Execution Commands

Command Name	Short Name	Contents
Go	G	Program execution with breakpoints
GoFree	GF	Free run program execution
GoProgramBreak*	GPB	Run target program with software break point
GoBreakAt*	GBA	Run target program with software break point
Stop	-	Stops program execution
Status	-	Checks the operating status of the MCU
Step	S	Halts for user input until the specified time has elapsed
StepInstruction	SI	Step execution of instructions
OverStep	O	Overstep execution of source lines
OverStepInstruction	OI	Overstep execution of instructions
Return	RET	Executes a source line return
ReturnInstruction	RETI	Executes an instruction return
Reset	-	Resets the target MCU

6.1.2 File Operation Commands

Command Name	Short Name	Contents
Load	L	Downloads the target program
LoadHex	LH	Downloads an Intel HEX-format file
LoadMot*	LM	Downloads a Motorola S-format file
LoadSymbol	LS	Loads source line/ASM symbol information
Reload	-	Re-downloads the target program
UploadHex	UH	Outputs data to an Intel HEX-format file
UploadMot*	UM	Outputs data to a Motorola S-format file

6.1.3 Register Operation Commands

Command Name	Short Name	Contents
Register	R	Checks and sets a register value

6.1.4 Memory Operation Commands

Command Name	Short Name	Contents
DumpByte	DB	Displays the contents of memory (in 1-byte units)
DumpWord*	DW	Displays the contents of memory (in 2-byte units)
DumpLword*	DL	Displays the contents of memory (in 4-byte units)
SetMemoryByte	MB	Checks and changes memory contents (in 1-byte units)
SetMemoryWord*	MW	Checks and changes memory contents (in 2-byte units)
SetMemoryLword*	ML	Checks and changes memory contents (in 4-byte units)
FillByte	FB	Fills a memory block with the specified data (in 1-byte units)
FillWord*	FW	Fills a memory block with the specified data (in 2-byte units)
FillLword*	FL	Fills a memory block with the specified data (in 4-byte units)
Move	-	Moves memory blocks
MoveWord*	MOVEW	Moves memory blocks(in 2-byte units)

6.1.5 Assemble/Disassemble Commands

Command Name	Short Name	Contents
Assemble	A	Line-by-line assembly
DisAssemble	DA	Disassembles memory contents line by line
Module	MOD	Displays modules names
Scope	-	Sets and checks the effective local symbol scope
Section	SEC	Checks section information
Bit*	-	Checks and sets bit symbols
Symbol	SYM	Checks assembler symbols
Label	-	Checks assembler labels
Express	EXP	Displays an assembler expression

6.1.6 Software Break Setting Commands

Command Name	Short Name	Contents
SoftwareBreak	SB	Sets and checks software breaks
SoftwareBreakClear	SBC	Clears software breaks
SoftwareBreakClearAll	SBCA	Clears all software breaks
SoftwareBreakDisable	SBD	Disables software breakpoints
SoftwareBreakDisableAll	SBDA	Disables all software breaks
SoftwareBreakEnable	SBE	Enables software breakpoints
SoftwareBreakEnableAll	SBEA	Enables all software breaks
BreakAt	-	Sets a software breakpoint by specifying a line No.
BreakIn	-	Sets a software breakpoint by specifying a function

6.1.7 Script/Log File Commands

Command Name	Short Name	Contents
Script	-	Opens and executes a script file
Exit	-	Exits the script file
Wait	-	Waits for an event to occur before command input
Pause	-	Waits for user input
Sleep	-	Halts for user input until the specified time has elapsed
Logon	-	Outputs the screen display to a log file
Logoff	-	Stops the output of the screen display to a log file
Exec	-	Executes external application

6.1.8 Program Display Commands

Command Name	Short Name	Contents
Func	-	Checks function names and displays the contents of functions
Up*	-	Displays the calling function
Down*	-	Displays a called function
Where*	-	Displays a function call status
Path	-	Sets and checks the search path
AddPath	-	Adds the search path
File	-	Checks a filename and displays the contents of that file

6.1.9 C Language Debugging Commands

Command Name	Short Name	Contents
Print	-	Check value of specified C variable expression
Set	-	Set specified data in specified C variable expression

6.1.10 Real-time OS Command

Command Name	Short Name	Contents
MR*	-	Displays status of realtime OS (MRxx)

6.1.11 Utility Commands

Command Name	Short Name	Contents
Radix	-	Sets and checks the radix for numerical input
Alias	-	Specifies and checks command alias definitions
UnAlias	-	Cancels the alias defined for a command
UnAliasAll	-	Cancels all aliases defined for commands
Version	VER	Displays the version No.
Date	-	Displays the date
Echo	-	Displays messages
CD	-	Window open

6.2 Table of Script Commands (alphabetical order)

Command Name	Short Name	Contents
AddPath	-	Adds the search path
Alias	-	Specifies and checks command alias definitions
Assemble	A	Line-by-line assembly
Bit*	-	Checks and sets bit symbols
BreakAt	-	Sets a software breakpoint by specifying a line No.
BreakIn	-	Sets a software breakpoint by specifying a function
CD	-	Specifies and checks the current directory
Date	-	Displays the date
DisAssemble	DA	Disassembles memory contents line by line
Down*	-	Displays a called function
DumpByte	DB	Displays the contents of memory (in 1-byte units)
DumpLword*	DL	Displays the contents of memory (in 4-byte units)
DumpWord*	DW	Displays the contents of memory (in 2-byte units)
Echo	-	Displays messages
Exec	-	Executes external application
Exit	-	Exits the script file
Express	EXP	Displays an assembler expression
File	-	Checks a filename and displays the contents of that file
FillByte	FB	Fills a memory block with the specified data (in 1-byte units)
FillLword*	FL	Fills a memory block with the specified data (in 4-byte units)
FillWord*	FW	Fills a memory block with the specified data (in 2-byte units)
Func	-	Checks function names and displays the contents of functions
Go	G	Program execution with breakpoints
GoBreakAt*	GBA	Run target program with software break point
GoFree	GF	Free run program execution
GoProgramBreak*	GPB	Run target program with software break point
Label	-	Checks assembler labels
Load	L	Downloads the target program
LoadHex	LH	Downloads an Intel HEX-format file
LoadMot*	LM	Downloads a Motorola S-format file
LoadSymbol	LS	Loads source line/ASM symbol information
Logoff	-	Stops the output of the screen display to a log file
Logon	-	Outputs the screen display to a log file
Module	MOD	Displays modules names
Move	-	Moves memory blocks
MoveWord*	MOVEW	Moves memory blocks(in 2-byte units)
MR*	-	Displays status of realtime OS (MRxx)
OverStep	O	Overstep execution of source lines
OverStepInstruaction	OI	Overstep execution of instructions
Path	-	Sets and checks the search path
Pause	-	Waits for user input
Print	-	Check value of specified C variable expression.
Radix	-	Sets and checks the radix for numerical input
Register	R	Checks and sets a register value
Reload	-	Re-downloads the target program

Reset	-	Resets the target MCU
Return	RET	Executes a source line return
ReturnInstruction	RETI	Executes an instruction return
Scope	-	Sets and checks the effective local symbol scope
Script	-	Opens and executes a script file
Section	SEC	Checks section information
Set	-	Set specified data in specified C variable expression
SetMemoryByte	MB	Checks and changes memory contents (in 1-byte units)
SetMemoryLword*	ML	Checks and changes memory contents (in 4-byte units)
SetMemoryWord*	MW	Checks and changes memory contents (in 2-byte units)
Sleep	-	Halts for user input until the specified time has elapsed
SoftwareBreak	SB	Sets and checks software breaks
SoftwareBreakClear	SBC	Clears software breaks
SoftwareBreakClearAll	SBCA	Clears software breaks
SoftwareBreakDisable	SBD	Disables software breakpoints
SoftwareBreakDisableAll	SBDA	Disables all software breaks
SoftwareBreakEnable	SBE	Enables software breakpoints
SoftwareBreakEnableAll	SBEA	Enables all software breaks
Status	-	Checks the operating status of the MCU
Step	S	Step execution of source line
StepInstruction	SI	Step execution of instructions
Stop	-	Stops program execution
Symbol	SYM	Checks assembler symbols
UnAlias	-	Cancels the alias defined for a command
UnAliasAll	-	Cancels all aliases defined for commands
Up*	-	Displays the calling function
UploadHex	UH	Outputs data to an Intel HEX-format file
UploadMot*	UM	Outputs data to a Motorola S-format file
Version	VER	Displays the version No.
Wait	-	Waits for an event to occur before command input
Where*	-	Displays a function call status

7. Writing Script Files

This debugger allows you to run script files in a Script Window. The script file contains the controls necessary for automatically executing the script commands.

7.1 Structural Elements of a Script File

You can include the following in script files:

- Script commands
- Assign statements
- Conditional statements (if, else, endi)
Program execution branches to the statement(s) to be executed according to the result of the conditional expression.
- Loop statements (while, endw)
A block of one or more statements is repeatedly executed according to the expression.
- break statement
Exits from the innermost loop.
- Comment statements
You can include comments in a script file. The comment statements are ignored when the script commands are executed.

Specify only one statement on each line of the script file. You cannot specify more than one statement on a line, or write statements that span two or more lines.

Notes

- You cannot include comments on the same lines as script commands.
- You can nest script files up to five levels.
- You can nest if statements and while statements up to 32 levels.
- If statements must be paired with endi statements, and while statements with endw statements in each script file.
- Expressions included in script files are evaluated as unsigned types. Therefore, operation cannot be guaranteed if you use negative values for comparison in if or while statements.
- You can specify up to 4096 characters per line. An error occurs if a line exceeds this number of characters.
- When a script file containing illegal commands is automatically executed (when you select [Option] -> [Script]-> [Run] from the Script Window menu after opening a script file, or click the button in the Script Window), execution of the script file continues even after the error is detected, except when the script line itself cannot be read. If an error is detected and the script file continues to be executed, operation after detection of the error cannot be guaranteed. Reliability cannot therefore be placed on the results of execution after an error has been detected.

7.1.1 Script Command

You can use the same script commands that you enter in the Script Window. You can also call script files from within other script files (nesting up to 10 levels).

7.1.2 Assign Statement

Assign statement s define and initialize macro variables and assign values. The following shows the format to be used.

<code>%macro-variable = expression</code>

- You can use alphanumerics and the underscore (`_`) in macro variable names. However , you cannot use a numeric to start a macro variable name.
- You can specify any expression of which the value is an integer between 0h and FFFFFFFFh to be assigned in a macro variable. If you specify a negative number, it is processed as twos complement.
- You can use macro variables within the expression.
- Always precede macro variables with the "%" sign.

7.1.3 Conditional Statement

In a conditional statement, different statements are executed according to whether the condition is true or false. The following shows the format to be used.

<pre>if (expression) statement 1 else statement 2 endi</pre>
--

- If the expression is t rue (other than 0), statement 1 is executed. If false, (0), statement 2 is executed.
- You can omit the else statement. If omitted and the expression is false, execution jumps to the line after the endi statement.
- if statements can be nested (up to 32 levels).

7.1.4 Loop Statement(while,endw) and Break Statement

In loop statements, execution of a group of statements is repeated while the expression is true. The following shows the format to be used.

<pre>while (expression) statement endw</pre>
--

- If the expression is t rue, the group of statements is repeated. If false, the loop is exited (and the statement following the endw statement is executed).
- You can nest while statements up to 32 levels.
- Use the break statement to forcibly exit a while loop. If while statements are nested, break exits from the inner most loop.

7.1.5 Comment statements

You can include comments in a script file. Use the following format.

<code>;character string</code>

- Write the statement after a semicolon (;). You can include only spaces and tabs in front of the semicolon
- Lines with comment statements are ignored when the script file is executed.

7.2 Writing Expressions

This debugger allows you to use expressions for specifying addresses, data, and number of passes, etc. The following shows example commands using expressions.

```
>DumpByte TABLE1
>DumpByte TABLE1+20
```

You can use the following elements in expressions:

- Constants
- Symbols and labels
- Macro variables
- Register variables
- Memory variables
- Line Nos.
- Character constants
- Operators

7.2.1 Constants

You can use binary, octal, decimal, or hexadecimal. The prefix or suffix symbol attached to the numerical value indicates which radix is used.

The debugger for M32C and M16C/R8C and 740

	Hexadecimal	Decimal	Octal	Binary *
Prefix	0x,0X	@	None	%
Suffix	h,H	None	o,O	b,B
Examples	0xAB24 AB24h	@1234	1234o	%10010 10010b

*You can only specify % when the predetermined radix is hexadecimal.

- If you are inputting a radix that matches the predetermined radix, you can omit the symbol that indicates the radix (excluding binary).
- Use the RADIX command to set the predetermined value of a radix. However, in the cases shown below, the radix is fixed regardless of what you specify in a RADIX command.

Type	Radix
Address	Hex
Line No. No. of executions No. of passes	Dec

7.2.2 Symbols and labels

You can include symbols and labels defined in your target program, or symbols and labels defined using the `Assemble` command.

- You can include alphanumerics, the underscore (`_`), period (`.`), and question mark (`?`) in symbols and labels. However, do not start with a numeric.
- Symbols and labels can consist of up to 255 characters.
- Uppercase and lowercase letters are unique.

Product Name	Notes
The debugger for M32R, The debugger for M32C, The debugger for M16C/R8C,	<ul style="list-style-type: none">• You cannot include the assembler structured instructions, pseudo instructions, macro instructions, operation code, or reserved words (<code>.SECTION</code>, <code>.BYTE</code>, <code>switch</code>, <code>if</code>, etc.).• You cannot use strings that start with two periods (<code>..</code>) for symbols or labels.

7.2.2.1 Local label symbol and scope

This debugger supports both global label symbols, which can be referenced from the whole program area, and local label symbols, which can only be referenced within the file in which they are declared. The effective range of local label symbols is known as the scope, which is measured in units of object files. The scope is switched in this debugger in the following circumstances:

- When a command is entered
The object file that includes the address indicated by the program counter becomes the current scope. When the `SCOPE` command is used to set the scope, the specified scope is the active scope.
- During command execution
The current scope automatically switches depending on the program address being handled by the command.

7.2.2.2 Priority levels of labels and symbols

The conversion of values to labels or symbols, and vice versa, is subject to the following levels of priority:

- Conversion of address values
 1. Local labels
 2. Global labels
 3. Local symbols
 4. Global symbols
 5. Local labels outside scope
 6. Local symbols outside scope
- Conversion of data values
 1. Local symbols
 2. Global symbols
 3. Local labels
 4. Global labels
 5. Local labels outside scope
 6. Local symbols outside scope
- Conversion of bit values
 1. Local bit symbols
 2. Global bit symbols
 3. Local bit symbols outside scope

7.2.3 Macro Variables

Macro variables are defined by assign statements in the script file. See Section "7.1.2 Assign Statement" in the Reference part for details. Precede variables with '%' for use as macro variables.

- You can specify alphanumerics and/or the underbar (_) in the variable name following the percent sign (%). However, do not start the names with a numeric.
- You cannot use the names of registers as variable names.
- Uppercase and lowercase letters are differentiated in variable names.
- You can define a maximum of 32 macro variables. Once defined, a macro variable remains valid until you quit the debugger.

Macro variables are useful for specifying the number of iterations of the while statement.

7.2.4 Register variables

Register variables are used for using the values of registers in an expression. Precede the name of the register with '%' to use it as a register variable. Use the following format. (The debugger for 740 can use '_' instead of '%'.)

Product Name	Register name
The debugger for M32C	PC, USP, ISP, INTB, FLB, SVF, SVP, VCT, DMD0,DMD1, DCT0, DCT1, DRC0, DRC1, DMA0,DMA1, DCA0, DCA1, DRA0, DRA1, 0R0, 0R1, 0R2, 0R3, 0A0, 0A1, 0FB, 0SB <- Bank 0 Register 1R0, 1R1, 1R2, 1R3, 1A0, 1A1, 1FB, 1SB <- Bank 1 Register
The debugger for M16C/R8C	PC, USP, ISP, SB, INTB, FLG 0R0, 0R1, 0R2, 0R3, 0A0, 0A1, 0FB <- Bank 0 Register 1R0, 1R1, 1R2, 1R3, 1A0, 1A1, 1FB <- Bank 1 Register

Uppercase and lowercase letters are not unique in register names. You can specify either.

7.2.5 Memory variables

Use memory variables to use memory values in expressions. The format is as follows:

[Address].data-size

- You can specify expressions in addresses (you can also specify memory variables).
- The data size is specified as shown in the following table. (The debugger for 740 doesn't support four byte length.)

data Length	Debugger	Specification
1 Byte	All	B or b
2 Bytes	The debugger for M32R	H or h
	Other	W or w
4 bytes	The debugger for M32R	W or w
	The debugger for M32R, M16C/R8C	L or l

Example: Referencing the contents of memory at address 8000h in 2 bytes
`[0x8000].w`

- The default data size is word, if not specified.

7.2.6 Line Nos.

These are source file line Nos. The format for line Nos. is as follows:

```
#line_no
#line_no."source file name"
```

- Specify line Nos. in decimal.
- You can only specify line Nos. in which software breaks can be set. You cannot specify lines in which no assembler instructions have been generated, including comment lines and blank lines.
- If you omit the name of the source file, the line Nos. apply to the source file displayed in active Editor(Source) Window.
- Include the file attribute in the name of the source file.
- Do not include any spaces between the line No. and name of the source file.

7.2.7 Character constants

The specified character or character string is converted into ASCII code and processed as a constant.

- Enclose characters in single quote marks.
- Enclose character strings in double quote marks.
- The character string must consist of one or two characters (16 bits max.). If more than two characters are specified, the last two characters of the string are processed. For example, "ABCD" would be processed as "CD", or value 4344h.

7.2.8 Operators

The table below lists the operators that you can use in expressions.

- The priority of operators is indicated by the level, level 1 being the highest and level 8 the lowest. If two or more operators have the same level of priority, they are evaluated in order from the left of the expression.

Operator	Function	Priority level
()	Brackets	level 1
+, -, ~	Monadic positive, monadic negative, monadic logical NOT	level 2
*, /	Dyadic multiply, dyadic divide	level 3
+, -	Dyadic add, dyadic subtract	level 4
>>, <<	Right shift, left shift	level 5
&	Dyadic logical AND	level 6
~, ^	Dyadic logical OR, dyadic exclusive OR	level 7
<, <=, >, >=, ==, !=	Dyadic comparison	level 8

8. C/C++ Expressions

8.1 Writing C/C++ Expressions

You can use C/C++ expressions consisting of the tokens shown below for registering C watchpoints and for specifying the values to be assigned to C watchpoints.

Token	Example
Immediate values	10, 0x0a, 012, 1.12, 1.0E+3
Scope	::name, classname::member
Mathematical operators	+, -, *, /
Pointers	*, **, ...
Reference	&
Sign inversion	-
Member reference using dot operator	Object.Member
Member reference using arrow	Pointer->Member, this->Member
Pointers to Members	Object.*var, Pointer->*var
Parentheses	(,)
Arrays	Array[2], DArray[2] [3] , ...
Casting to basic types	(int), (char*), (unsigned long *), ...
Casting to typedef types	(DWORD), (ENUM), ...
Variable names and function names	var, i, j, func, ...
Character constants	'A', 'b', ...
Character string literals	"abcdef", "I am a boy.", ...

8.1.1 Immediate Values

You can use hexadecimal, decimals, octals as immediate values. Values starting with 0x are processed as hexadecimal, those with 0 as octals, and those without either prefix as decimals.

Floating-point numbers can also be used to assign values to variables.

Notes

- You cannot register only immediate values as C watchpoints.
- The immediate value is effective only when it is used in C/C++ language expressions that specify C/C++ watchpoints or when it is used to specify the value to be assigned to those expressions. When using floating-point numbers, operation cannot be performed on an expression like 1.0+2.0.

8.1.2 Scope Resolution

The scope resolution operator `::` is available as following.

Global scope: `::variable name`

`::x, ::val`

Class scope: `class name::member name`, `class name::class name::member name`, e.g.

`T::member, A::B::member`

8.1.3 Mathematical Operators

You can use the addition (+), subtraction (-), multiplication (*), and division (/) mathematical operators. The following shows the order of priority in which they are evaluated.

`(*)`, `(/)`, `(+)`, `(-)`

Notes

- There is no support currently for mathematical operators for floating point numbers.

8.1.4 Pointers

Pointers are indicated by the asterisk (*). You can use pointer to pointers **, and pointer to pointer to pointers ***, etc.

Examples: `"*variable_name"`, `***variable_name"`, etc.

Notes

- Immediate values cannot be processed as pointers. That is, you cannot specify `*0xE000`, for example.

8.1.5 Reference

References are indicated by the ampersand (&). You can only specify `"&variable_name"`.

8.1.6 Sign Inversion

Sign inversion is indicated by the minus sign (-). You can only specify "-immediate_value" or "-variable_name". No sign inversion is performed if you specify 2 (or any even number of) minus signs.

Notes

- There is no support currently for sign inversion of floating point numbers.

8.1.7 Member Reference Using Dot Operator

You can only use "variable_name.member_name" for checking the members of structures and unions using the dot operator.

Example:

```
class T {
public:
int member1;
char member2;
};
class T t_cls;
class T *pt_cls = &t_cls;
```

In this case, t_cls.member1, (*pt_cls).member2 correctly checks the members.

8.1.8 Member Reference Using Arrow

You can only use "variable_name->member_name" for checking the members of structures and unions using the arrow.

Example:

```
class T {
public:
int member1;
char member2;
};
class T t_cls;
class T *pt_cls = &t_cls;
```

In this case, (&t_cls)->member1, pt_cls->member2 correctly checks the members.

8.1.9 Pointers to Members

Pointers to members using the ".*" or "->*" operator can be referred only in the forms of variable name .* member name or variable name ->* member name.

Example:

```
class T {
public:
    int member;
};
class T t_cls;
class T *pt_cls = &t_cls;

int T::*mp = &T::member;
```

In this case, t_cls.*mp and tp_cls->*mp can correctly reference the variable of pointer-to-member type.

Note

- Note that the expression *mp cannot be considered as the variable of pointer-to-member type.

8.1.10 Parentheses

Use the '(' and ')' to specify priority of calculation within an expression.

8.1.11 Arrays

You can use the ' [' and '] ' to specify the elements of an array. You can code arrays as follows: "variable_name [(element_No or variable)] ", "variable_name [(element_No or variable)] [(element_No or variable)] ", etc.

8.1.12 Casting to Basic Types

You can cast to C basic types char, short, int, and long, and cast to the pointer types to these basic types. When casting to a pointer type, you can also use pointers to pointers and pointers to pointers to pointers, etc.

Note that if signed or unsigned is not specified, the default values are as follows:

Basic type	Default
char	unsigned
short	signed
int	signed
long	signed

Notes

- Of the basic types of C++, casts to bool type, wchar_t type, and floating-point type (float or double) cannot be used.
- Casts to register variables cannot be used.

8.1.13 Casting to typedef Types

You can use casting to typedef types (types other than the C basic types) and the pointer types to them. When casting to a pointer type, you can also use pointers to pointers and pointers to pointers to pointers, etc.

Notes

- You cannot cast to struct or union types or the pointers to those types.

8.1.14 Variable Name

Variable names that begin with English alphabets as required under C/C++ conventions can be used.

The maximum number of characters for variable name is 255.

And 'this' pointer is available.

8.1.15 Function Name

Function names that begin with English alphabets as required under C conventions can be used.

Notes

- In the case of C++, no function names can be used.

8.1.16 Character Constants

You can use characters enclosed in single quote marks (') as character constants. For example, 'A', 'b', etc. These character constants are converted to ASCII code and used as 1-byte immediate values.

Notes

- You cannot register character constants only as C watchpoints.
- Character constants are valid only when used in a C/C++ expression that specifies a C watchpoint, and when specifying a value to be assigned (character constants are processed in the same manner as immediate values).

8.1.17 Character String Literals

You can use character strings enclosed in double quote marks (") as character string literals. Examples are "abcde", "I am a boy.", etc.

Notes

- Character string literals can only be placed on the right side of an assignment operator in an expression. They can only be used when the left side of the assignment operator is a char array or a char pointer type. In all other cases, a syntax error results.

8.2 Display Format of C/C++ Expressions

C/C++ expressions in the data display areas of the C Watch Windows are displayed as their type name, C/C++ expression (variable name), and result of calculation (value), as shown below.

The following describes the display formats of the respective types.

8.2.1 Enumeration Types

- When the result (value) of calculation has been defined, its name is displayed.
`(DATE) date = Sunday(all Radices)_`
- If the result (value) of calculation has not been defined, it is displayed as follows:
`(DATE) date = 16 (when Radix is in initial state)`
`(DATE) date = 0x10(when Radix is hex)`
`(DATE) date = 0000000000010000B(when Radix is binary)`

8.2.2 Basic Types

- When the result of calculation is a basic type other than a char type or floating point type, it is displayed as follows:
`(unsigned int) i = 65280(when Radix is in initial state)`
`(unsigned int) i = 0xFF00(when Radix is hex)`
`(unsigned int) i = 1111111100000000B(when Radix is binary)`
- When the result of calculation is a char type, it is displayed as follows:
`(unsigned char) c = 'J'(when Radix is in initial state)`
`(unsigned char) c = 0x4A(when Radix is hex)`
`(unsigned char) c = 10100100B(when Radix is binary)`
- When the result of calculation is a floating point, it is displayed as follows:
`(double) d = 8.207880399131839E-304(when Radix is in initial state)`
`(double) d = 0x10203045060708(when Radix is hex)`
`(double) d = 0000000010.....1000B(when Radix is binary)`
`(..... indicates abbreviation)`

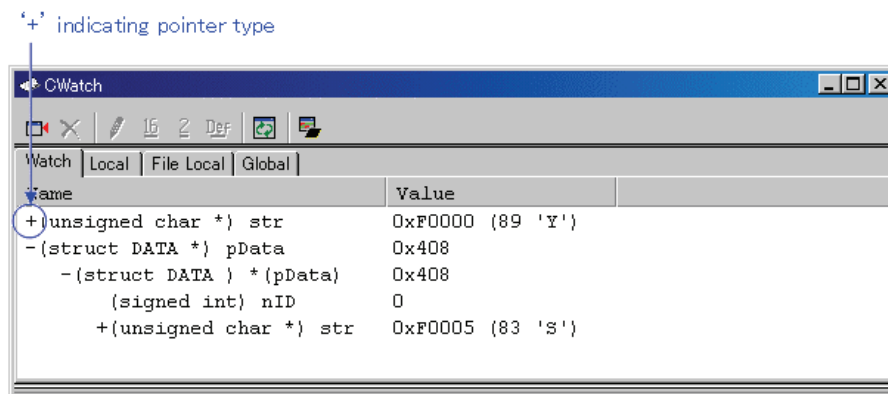
8.2.3 Pointer Types

- When the result of calculation is a pointer type to other than a `char*` type, it is displayed in hexadecimal as follows:
`(unsigned int *) p = 0x1234(all Radices)`
- When the result of calculation is a `char*` type, you can select the display format of the string or a character in the C Watch window's menu [Display String].
 - string types
`(unsigned char *) str = 0x1234 "Japan"(all Radices)`
 - character types
`(unsigned char *) str = 0x1234 (74 'J')(all Radices)`

1 When the result of calculation is a `char*` type, it is displayed as follows:

`(unsigned char *) str = 0x1234 "Jap(all Radices)`

If the string contains a non-printing code prior to the code to show the end of the string (0), it is displayed up to the non-printing character and the closing quote mark is not displayed.



You can double-click on lines indicated by a '+' to see the members of that structure or union. The '+' changes to a '-' while the members are displayed. To return to the original display, double click the line, now indicated by the '-'.

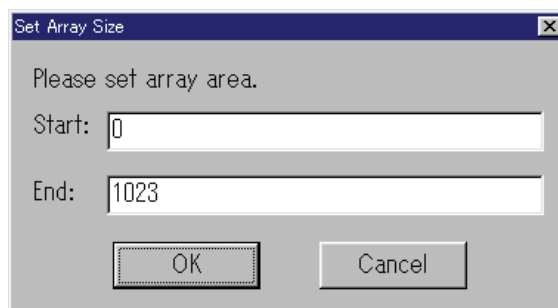
8.2.4 Array Types

- When the result of calculation is an array type other than a char [] type, the starting address is displayed in hex as follows:
`(signed int [10]) z = 0x1234(all Radices)`
- When the result of calculation is a char [] type, it is displayed as follows:
`(unsigned char [10]) str = 0x1234 "Japan"(all Radices)`

If the string contains a non-printing code prior to the code to show the end of the string (0), it is displayed up to the non-printing character and the closing quote mark is not displayed.

`(unsigned char [10]) str = 0x1234 "Jap(all Radices)`

Also if the string contains more than 80 characters, the closing quote mark is not displayed. When the C/C++ expression is an array type as same as pointer type, a '+' is display to the left of the type name. You can see the elements of the array by using this indicating. (for the details, refer to "8.2.3 Pointer Types") When the number of the array elements is more than 100, the following dialog box open. Specify the number of the elements in the dialog box.



The elements from the index specified in "Start" to the index specified in "End" are displayed. If you specify the value more than the max index of the array, the value is regarded as max index of the array. When you click the "Cancel" button, the elements are not displayed.

8.2.5 Function Types

- When the result of calculation is a function type, the starting address is displayed in hex as follows:
`(void()) main = 0xF000(all Radices)`

8.2.6 Reference Types

- When the result of calculation is a reference type, the reference address is displayed in hex as follows:
`(signed int &) ref = 0xD038(all Radices)`

8.2.7 Bit Field Types

- When the result of calculation is a bit field type, it is displayed as follows:
`(unsigned int :13) s.f = 8191(when Radix is in initial state)`
`(unsigned int :13) s.f = 0x1FFF(when Radix is hex)`
`(unsigned int :13) s.f = 1111111111111B(when Radix is binary)`

8.2.8 When No C Symbol is Found

If the calculated expression contained a C symbol that could not be found, it is displayed as follows:

```
() x = <not active>(all Radices)
```

8.2.9 Syntax Errors

- When the calculated expression contains a syntax error, it is displayed as follows:

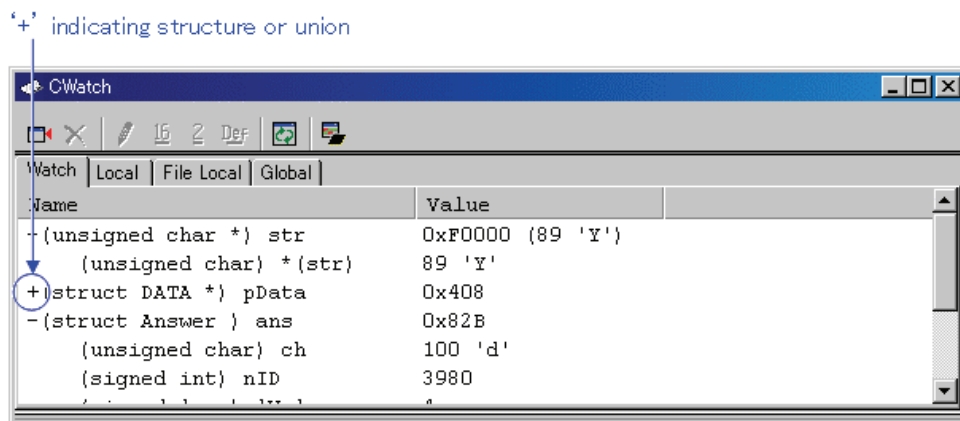
```
() str*(p = <syntax error>(all Radices)
(where str*(p is the syntax error)
```

8.2.10 Structure and Union Types

- When the result of calculation is a structure or union type, the address is displayed in hex as follows:

```
(Data) v = 0x1234 (all Radices)
```

If, as in structures and unions, the C/C++ expression consists of members, a '+' is displayed to the left of the type name (tag name).



You can double-click on lines indicated by a '+' to see the members of that structure or union. The '+' changes to a '-' while the members are displayed. To return to the original display, double click the line, now indicated by the '-'. This function allows you to check the members of structures and unions.

Attention

If a variable is declared with the same name as the type definition name declared by typedef, you cannot reference that variable.

- Register Variables

When the result of calculation is a register variable, "register" is displayed to the left of the type name as follows:

```
(register signed int) j = 100
```

9. Display the Cause of the Program Stoppage

If the program is stopped by the debug function, the cause of the stoppage is displayed in the Output window or Status window ([Platform] sheet).

The contents of a display and the meaning of "the cause of the stoppage" are as follows.

Display	The cause of the stoppage
Halt	The stop by the [Halt Program] button/menu
S/W break	Software break
Address match interrupt break	Address interrupt break
H/W event, Combination	Hardware break, logical combination AND or AND(same time)condition was met
H/W event, Combination, Ax	Hardware break, logical combination OR condition was met (Ax: The event number of which condition was met.)
H/W event, State transition, from xx	Hardware break, State Transition condition was met (from xx: previous state (start, state1, state2))
H/W event, State transition, Timeout	Hardware break, State Transition, Time Out condition was met
H/W event, Access protect error	Protect break

Note

To be able to show the cause of break or not depends on the connected target. Some targets may always show "Halt" or show "---".

10. Attention

10.1 Common Attention

10.1.1 File operation on Windows

The following points should be noted:

1. File Name and Directory Name
 - Do not use directory names or filenames that include blanks.
 - Operation is not guaranteed if your directory names and filenames include kanji.
 - Use only one period in a filename.
2. Specify the File and Directory
 - You cannot use ".." to specify two levels upper directories.
 - You cannot use a network pathname. You must allocate a drive.

10.1.2 Area where software breakpoint can be set

The Internal RAM area and Internal ROM area can be designated for software breakpoint.

10.1.3 Get or set C variables

- If a variable is declared with the same name as the type definition name declared by typedef, you cannot reference that variable.
- Values cannot be changed for register variables and bit fields.
- Values cannot be changed for 64 bit width variables (long long, double, and so on).
- Values cannot be changed for C/C++ expressions that do not indicate the memory address and size.
- For the sake of optimization, the C compiler may place different variables at the same address. In this case, values of the C variable may not be displayed correctly.
- Literal character strings can only be substituted for char array and char pointer type variables.
- No arithmetic operations can be performed on floating point types.
- No sign inversion can be performed on floating point types.
- Casting cannot be performed on floating point types.
- Casting cannot be performed on register variables.
- Casting cannot be performed on structure types, union types, or pointer types to structure or union types.
- Character constants and literal character strings cannot contain escape sequences.

10.1.4 Function name in C++

- When you input the address using the function name in setting display address, setting break points, and so on, you can not specify the member function, operator function, and overloaded function, of a class.
- You can not use function names for C/C++ expression
- No script commands (e.g., breakin and func) can be used in which function names are specified for arguments.
- In address value specifying columns of dialog boxes, no addresses can be specified using function names.

10.1.5 Debugging multi modules

If you register two or more absolute module file in one session, you can download only one file in same time.

If you register one absolute module file and one or more machine language file in one session, you can download all file in same time.

10.1.6 Synchronized debugging

Synchronized debugging function is not available.

10.1.7 RAM monitor function

The RAM monitor function of this debugger is realized with memory dump. Real-time capability is impaired when this function is used. Moreover, a setup of the display color by the access attribute cannot be performed.

10.1.8 Line assemble function

You can specify the internal RAM area included in from 00000H to 0FFFFH for the line assemble function.

10.1.9 Unavailable debugging functions

This debugger does not support the Hardware break, Protect, Coverage, Trace, Time measurement, and other functions which depend on the specification of emulator system. These functions are not available.

10.1.10 Software break function

- The software break function uses the address match interrupt of an mcu. Therefore, the count of break points depends of the connected mcu.
- If the PASS column of a Source Window is double-clicked, the pass counts (number of times passed) can be specified.

10.2 Attention of the M32C Debugger

10.2.1 Option of C Compiler/Assembler/Linker

The information may not be downloaded/debugged normally depending on the option designation of the compiler, assembler, and linker.

Please refer to the following for the option specification.

Refer to "10.4 Options for compiler, assembler, and linker"

The compiler that can be used by M32C debugger:

- NCxx
- the IAR EC++ Compiler
- the IAR C Compiler

10.3 Attention of the M16C/R8C Debugger

10.3.1 Options for compiler, assembler, and linker

The information may not be downloaded/debugged normally depending on the option designation of the compiler, assembler, and linker.

Please refer to the following for the option specification.

Refer to "10.4 Options for compiler, assembler, and linker"

The compiler that can be used by M16C/R8C debugger:

- NCxx
- the IAR EC++ Compiler
- the IAR C Compiler
- the TASKING C Compiler

10.3.2 TASKING C Compiler

When you debug programs compiled by the TASKING C Compiler "CCM16", the type of bit field is fixed on "unsigned short int". Because CCM16 outputs the debug information for the type of bit field as "unsigned short int."

10.3.3 Cautions for R8C/Tiny series

- You can not use the R8C/12-17 group mcu, if '0' is written to the bit0 of FFFFH. If '0' is written, please erase the flash ROM by a flash writer.
- You can not debug the program which is using WDT, if the target mcu is R8C/10-17 group.

10.4 Options for compiler, assembler, and linker

We do not evaluate other settings, so we can not recommend to append other options.

10.4.1 When Using NCxx

When -O, -OR or -OS option is specified at compilation, the source line information may not be generated normally due to optimization, causing step execution to be operated abnormally.

To avoid this problem, specify -ONBSD (or -Ono_Break_source_debug) option together with -O, -OR or -OS option.

In the options other than the above-mentioned, the operation check is not done. Please acknowledge that the options other than the above-mentioned cannot be recommended.

10.4.2 When Using the IAR C Compiler (EW)

Please specify the project setting by following process.

1. The Setting in the IAR Embedded Workbench
When you select the menu [Project] -> [Options...], the dialog for "Options For Target " target"" will open. In this dialog, please select the "XLINK" as category, and set the project setting.
 - Output Tab
In the "Format" area, check the "Other" option, and select the "ieee-695" as "Output Format".
 - Include Tab
In the "XCL File Name" area, specify your XCL file (ex: lnkm16c.xcl).
2. Edit the XCL file
Add the command line option "-y" to your XCL file. The designation of "-y" option varies depending on the product.

Product Name	-y Option
The debugger for M32C	-ylmb
The debugger for M16C/R8C	-ylmb

3. Build your program after the setting above.

In the options other than the above-mentioned, the operation check is not done. Please acknowledge that the options other than the above-mentioned cannot be recommended.

10.4.3 When Using the IAR C Compiler (ICC)

10.4.3.1 Specify the Option

Please compile according to the following procedures and link.

- At compilation
Specify the "-r" option.
- Before linking
Open the linker's option definition file (extension .xcl) to be read when linking and add "-FIEEE695" and "-y" options. The designation of "-y" option varies depending on the product.

Product Name	-y Option
The debugger for M32C	-ylmb
The debugger for M16C/R8C	-ylmb

- At link
Specify the linker's option definition file name using "-f" option.

In the options other than the above-mentioned, the operation check is not done. Please acknowledge that the options other than the above-mentioned cannot be recommended.

10.4.3.2 Command Execution Examples

The following shows examples of entering commands depending on the product

- The debugger for M32C


```
>ICCMC80 -r file1.c<Enter>
>ICCMC80 -r file2.c<Enter>
>XLINK -o filename.695 -f lnkm80.xcl file1 file2<Enter>
```
- The debugger for M16C/R8C


```
>ICCM16C -r file1.c<Enter>
>ICCM16C -r file2.c<Enter>
>XLINK -o filename.695 -f lnkm16c.xcl file1 file2<Enter>
```

The XCL file name varies depending on the product and memory model. For details, see the ICCxxxx manual.

10.4.4 When Using the TASKING C Compiler (EDE)

Please specify the project setting by following process.

1. Select menu - [EDE]->[C Compiler Option]->[Project Options...]. The "M16C C Compiler Options [Project Name]" dialog opens.
Please set as follows by this dialog.
 - Optimize Tab
Please specify "No optimization" by Optimization level.
 - Debug Tab
Please check only ""Enable generation of any debug information(including type checking)"" and "Generate symbolic debug information".
2. Select menu - [EDE]->[Linker/Locator Options...]. The "M16C Linker/Locator Options [Project Name]" dialog opens.
Please set as follows by this dialog.
 - Format Tab
Please specify "IEEE 695 for debuggers(abs)" by Output Format.
3. Build your program after the setting above.

In the options other than the above-mentioned, the operation check is not done. Please acknowledge that the options other than the above-mentioned cannot be recommended.

10.4.5 When Using the TASKING C Compiler (CM)

10.4.5.1 Specify the Option

Please specify "-g" and "-O0" options when compiling.

In the options other than the above-mentioned, the operation check is not done. Please acknowledge that the options other than the above-mentioned cannot be recommended.

10.4.5.2 Command Execution Examples

The following shows examples of entering commands.

```
>CM16 -g -O0 file1.c<Enter>
```

10.4.6 When Using the IAR EC++ Compiler (EW)

Please specify the project setting by following process.

1. The Setting in the IAR Embedded Workbench
When you select the menu [Project] -> [Options...], the dialog for "Options For Target " target"" will open. In this dialog, please select the "XLINK" as category, and set the project setting.
 - Output Tab
In the "Format" area, check the "Other" option, and select the "elf/dwarf" as "Output Format".
 - Include Tab
In the "XCL File Name" area, specify your XCL file (ex: lnkm32cf.xcl).
2. Edit the XCL file
Add the command line option "-y" to your XCL file. The designation of "-y" option varies depending on the product.

Product Name	-y Option
The debugger for M32C	-yspc
The debugger for M16C/R8C	-yspc

3. Build your program after the setting above.

In the options other than the above-mentioned, the operation check is not done. Please acknowledge that the options other than the above-mentioned cannot be recommended.

[MEMO]

M32C FoUSB/UART Debugger V.1.02
User's Manual

Publication Date: Aug. 01, 2006 Rev.1.00

Published by: Sales Strategic Planning Div.
Renesas Technology Corp.

Edited by: Microcomputer Tool Development Department
Renesas Solutions Corp.

© 2006. Renesas Technology Corp. and Renesas Solutions Corp., All rights reserved. Printed in Japan.

M32C FoUSB/UART Debugger V.1.02 User's Manual



Renesas Electronics Corporation

1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REJ10J1377-0100