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.

User's Manual

740 Compact Emulator Debugger V.1.02

User's Manual

Renesas Microcomputer Development Environment System

Renesas Electronics
www.renesas.com

Rev.1.00 2006.08

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 compact emulator system.

Supported CPU

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

- 740 Family
  Note: In this help, the information which depends on this CPU is described as "for 740".

(Blank Page)

Setup of Debugger

(Blank Page)

1. Features

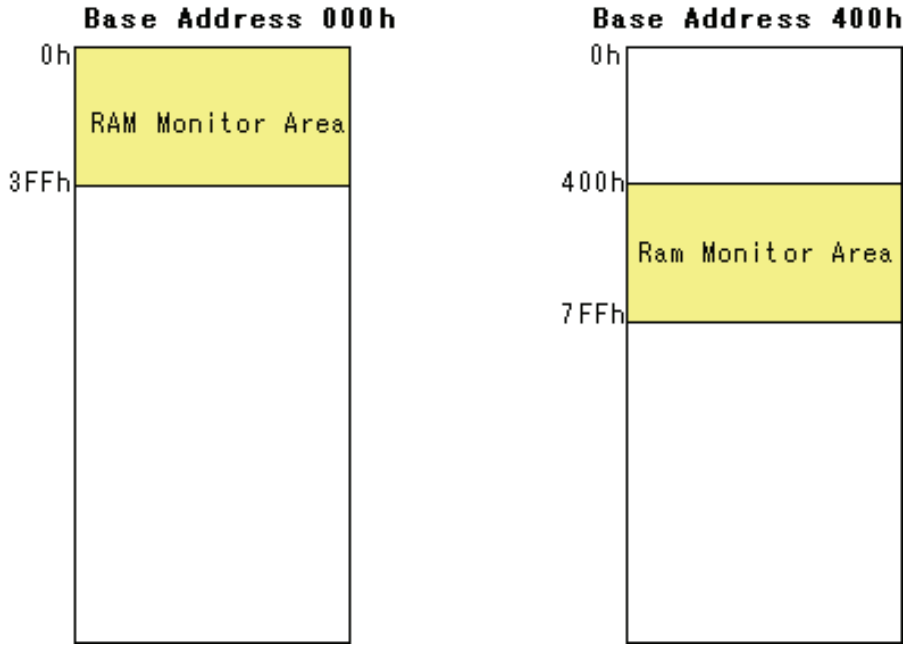
This debugger have the following functions.

1.1 Real-Time RAM Monitor Function

This function allows you to inspect changes of memory contents without impairing the realtime capability of target program execution.
The compact emulator system contains a 1-Kbyte RAM monitor area (which cannot be divided into smaller areas).

1.1.1 RAM Monitor Area

This debugger provides a 1KB of RAM monitor area, which can be placed at any continuous addresses.



1.1.2 Sampling Period

Sampling cycle means the display update interval.
You can specify this function in any window which supports the RAM monitor. (The interval of 100 ms is set by default.)
The actual sampling cycle may take longer time than the specified cycle depending on the operating environment. (Sampling cycle depends on the following environments.)

- Communication interface
- Number of the RAM Monitor windows displayed
- Size of the RAM Monitor window displayed
- Number of ASM watch points within the RAM monitor area of the ASM Watch window
- Number of C watch points within the RAM monitor area of the C Watch window

1.1.3 Related Windows

The window where the function of the real time RAM monitor function can be used is shown below.

- RAM Monitor Window
- ASM Watch Window
- C Watch Window

1.2 Break Functions

1.2.1 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.
You can specify up to 64 software breakpoints. 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.)

1.2.1.1 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.

1.2.1.2 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:
"11.1.2 Area where software breakpoint can be set "

1.2.2 Hardware Break

This function causes the target program to stop upon detecting a data read/write to memory or instruction execution.
You can set one address breakpoints with pass counts.

1.3 Real-Time Trace Function

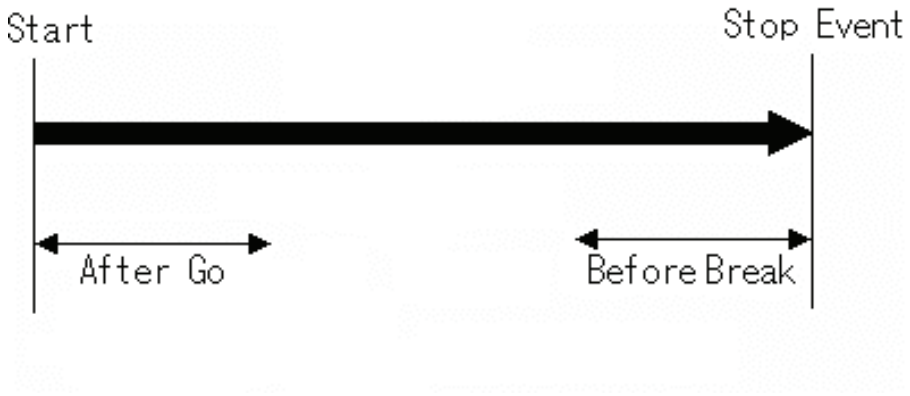
This function records a target program execution history.
Up to 32K cycles of execution history can be recorded. This record allows inspecting the bus information, executed instructions, and source program execution path for each cycle.
The real-time trace function records the execution history of the target program.
The execution history is referred to in the tracing window.
The execution history can be referred to in the following mode.

- BUS mode
  This mode allows you to inspect cycle-by-cycle bus information. The display content depends on the MCU and emulator system used. In addition to bus information, this mode allows disassemble, source line or data access information to be displayed in combination.
- Disassemble mode
  This mode allows you to inspect the executed instructions. In addition to disassemble information, this mode allows source line or data access information to be displayed in combination.
- Data access mode
  This mode allows you to inspect the data read/write cycles. In addition to data access information, this mode allows source line information to be displayed in combination.
- Source mode
  This mode allows you to inspect the program execution path in the source program.

1.3.1 Trace Area

The 32K cycles execution history can be referred to with this debugger.
The trace area of the following 5 mode is being supported.

- Beore Break
  32K cycles before target program stops
- After Go
  Until 32K cycles are written in the trace memory



"Before Break" is set by default. To refer the execution history before stopping the target program, use "Before Break" (designation of trace event is not required).

1.4 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. Before starting the debugger

:

2.1 Communication method by emulator

The supported communication methods are as follows.

- The M38000T2-CPE is supporting USB as a communication interface.

2.1.1 USB Interface

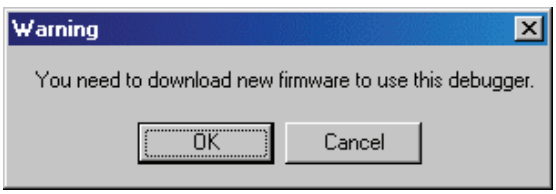
- The supported host computer OS is Windows Me/98/2000/XP. USB communication cannot be used in any other OS.
- Compliant with USB Standard 1.1.
- Connections via USB hub are not supported.
- By connecting the host computer and the emulator with USB cable, it is possible to install the supported device drivers using a wizard.
- The necessary cable is included with the emulator.

2.2 Download of Firmware

It is necessary to down-load the firmware which corresponds to connected Compact Emulator when the debugger is started to the emulator.

- The firmware downloaded to the emulator is unknown one.
- You have setup the debugger for the first time.
- You have upgraded emulator debugger.

Press the system reset switch within two seconds after powering up the Compact Emulator to establish the maintenance mode.
This debugger searches the version of the firmware downloaded to the emulator at start. Also when the firmware downloaded to the emulator is of old version, a mode which drives this debugger to download firmware is set.
When this debugger gets started while the emulator is set in the mode which drives the debugger to download firmware forcedly, the following dialog is opened at start.
Click the OK button to download the firmware.



2.3 Setting before emulator starts

2.3.1 USB communication with the Emulator

Connection of USB devices is detected by Windows' Plug & Play function. The device driver needed for the connected USB device is automatically installed. For details, see "Install of USB Device Driver".

2.3.1.1 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

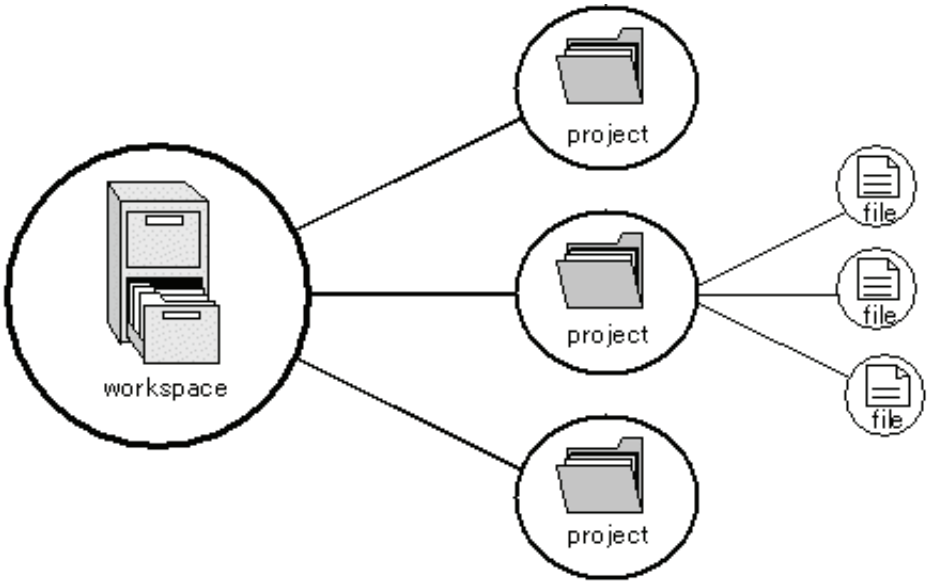
- Before the USB device drivers can be installed, the debugger you use must already be installed. Install this debugger first.
- USB communication can be used only in Windows Me/98/2000/XP, and cannot be used in any other OSs.
- When using Windows 2000/XP, a user who install the USB device driver need 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. 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.

3.1 Workspaces, Projects, and Files

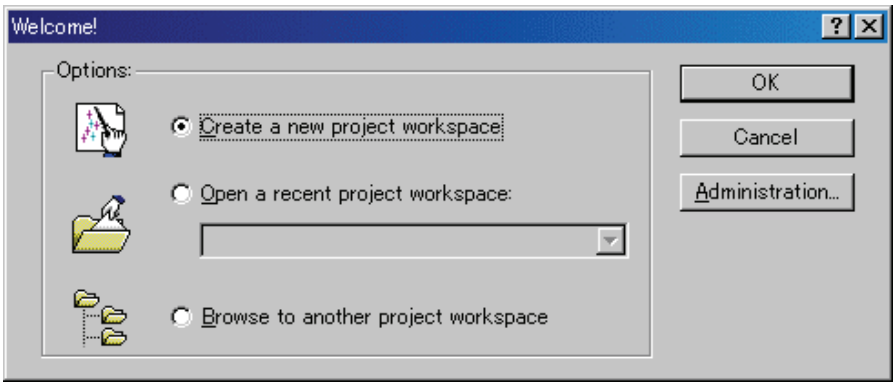
Just as a word processor allows you to create and modify documents, this product allows you to create and modify workspaces.
A workspace can be thought of as a container of projects and, similarly, a project can be thought of as a container of project files. Thus, each workspace contains one or more 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.

3.2 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).

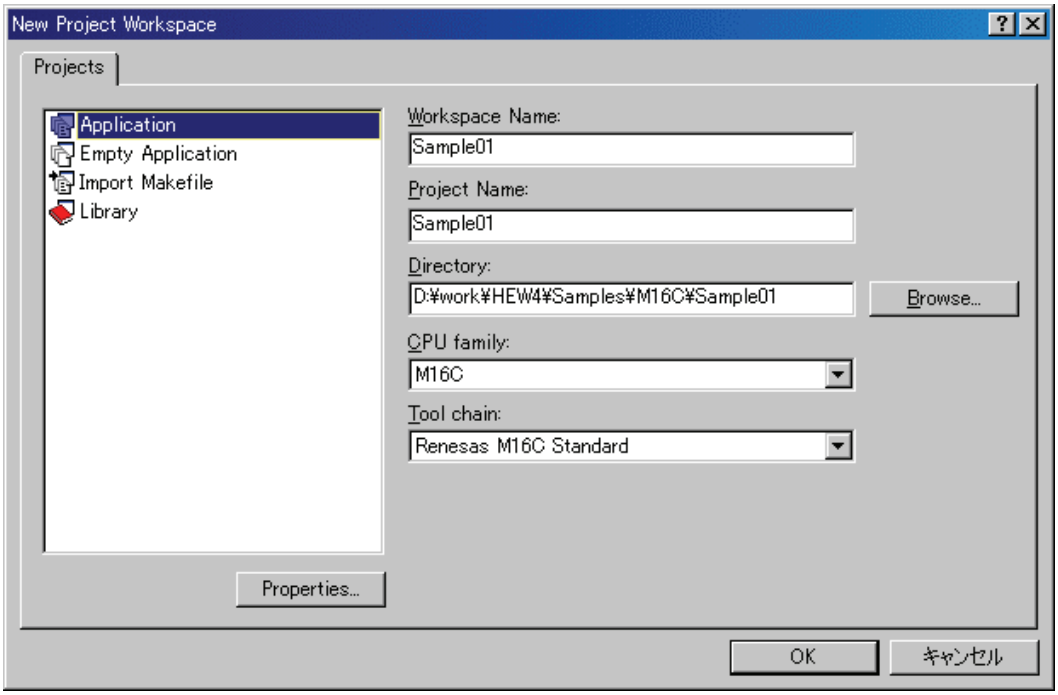
Please refer to the following about the method to create a new workspace.
Refer to "3.2.1 Creating a New Workspace (Toolchain Used)"
Refer to "3.2.2 Creating a New Workspace (Toolchain Not Used)"
* When debugging the existing load module file with this product, a workspace is created by this method.

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.

3.2.1 Creating a New Workspace (Toolchain Used)

3.2.1.1 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.

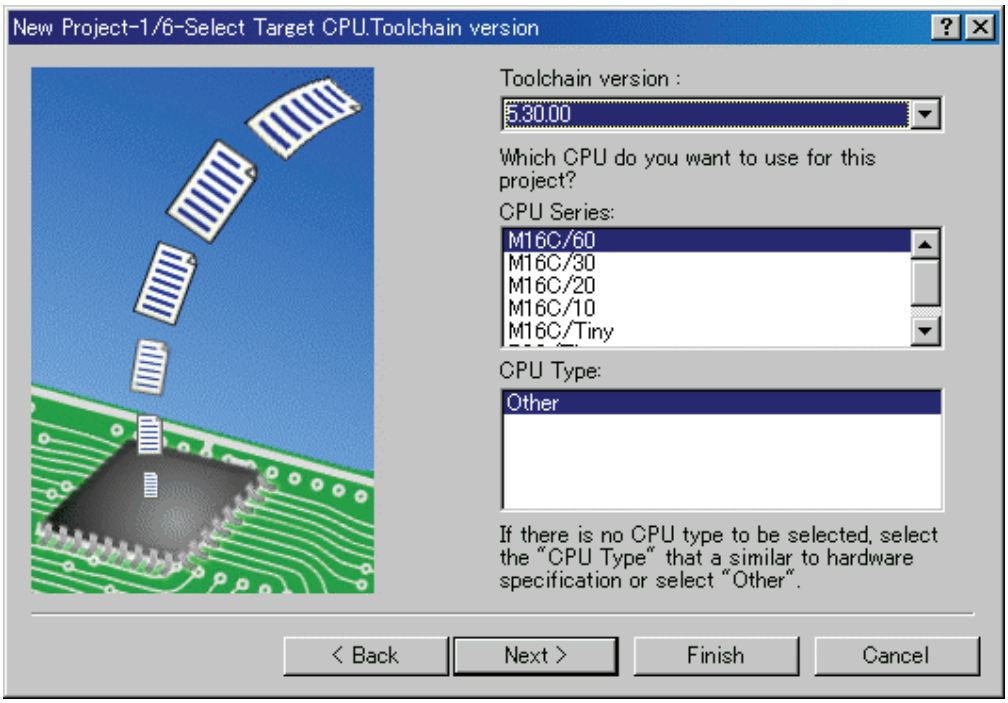


1. Select the target CPU family
   In the [CPU family] combo box, select the target CPU family.
2. Select the target toolchain
   In the [Tool chain] combo box, select the target toolchain name when using the toolchain.
3. Select the project type
   In the [Project type] list box, select the project 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.

3.2.1.2 Step2 : Setting for the Toolchain

A wizard for the project creation starts.



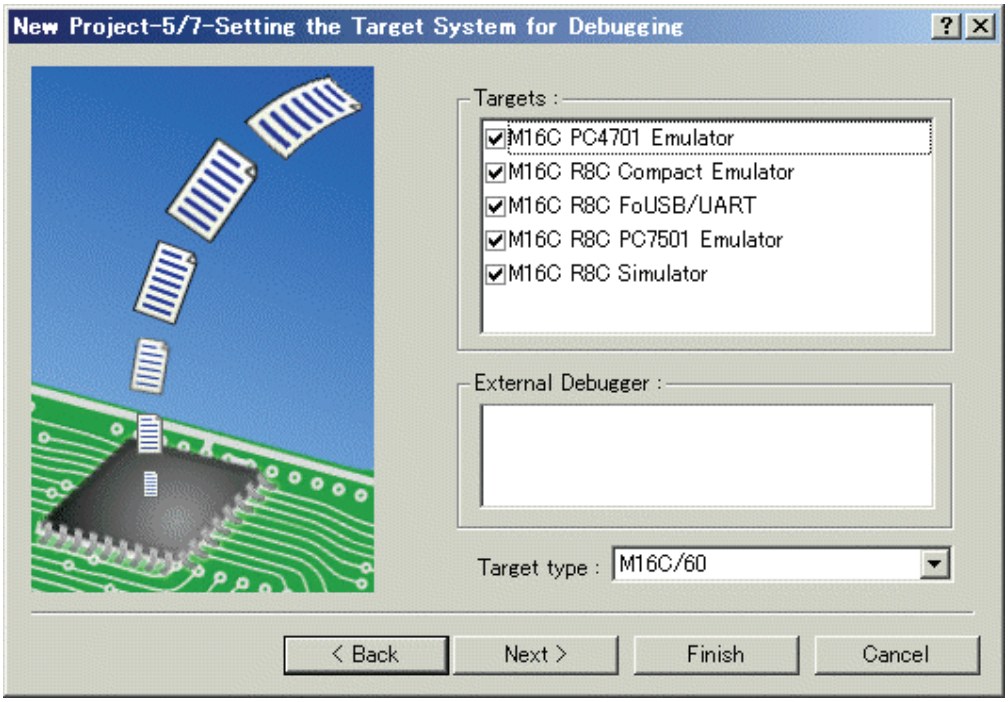
Here, the following contents are set.

- 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.

3.2.1.3 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.

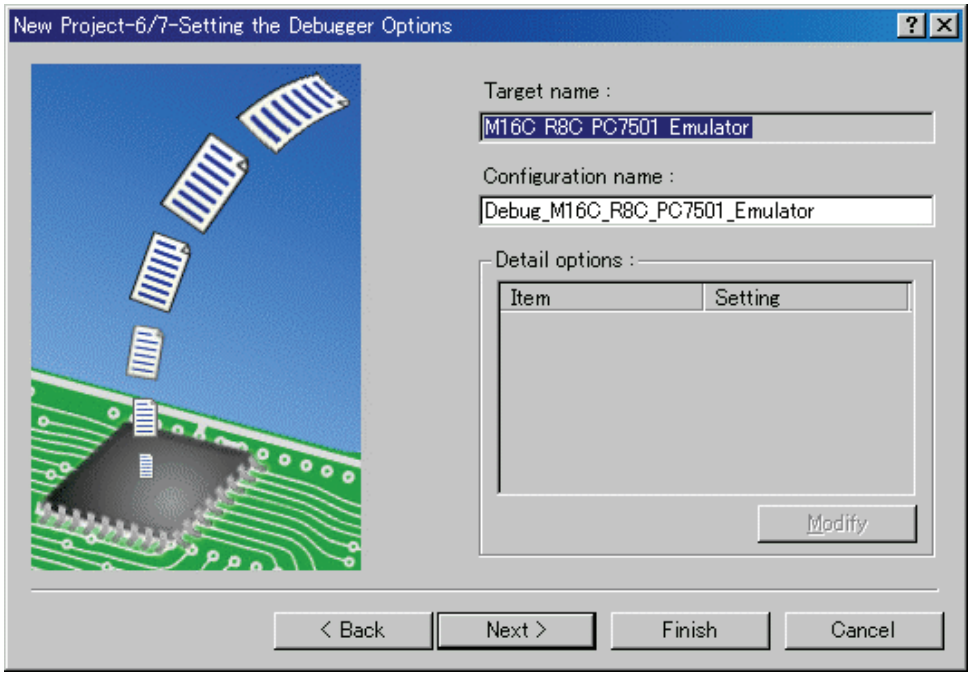


1. Selecting of the Target type
   In the [Target type] list box, select the target CPU type.
2. Selecting of the Target Platform
   In the [Targets] area, the target for the session file used when this debugger is activated must be selected here.
   Check the box against the target platform. (And choose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

And click the [Next] button.

3.2.1.4 Step4 : Setting the Configuration File Name

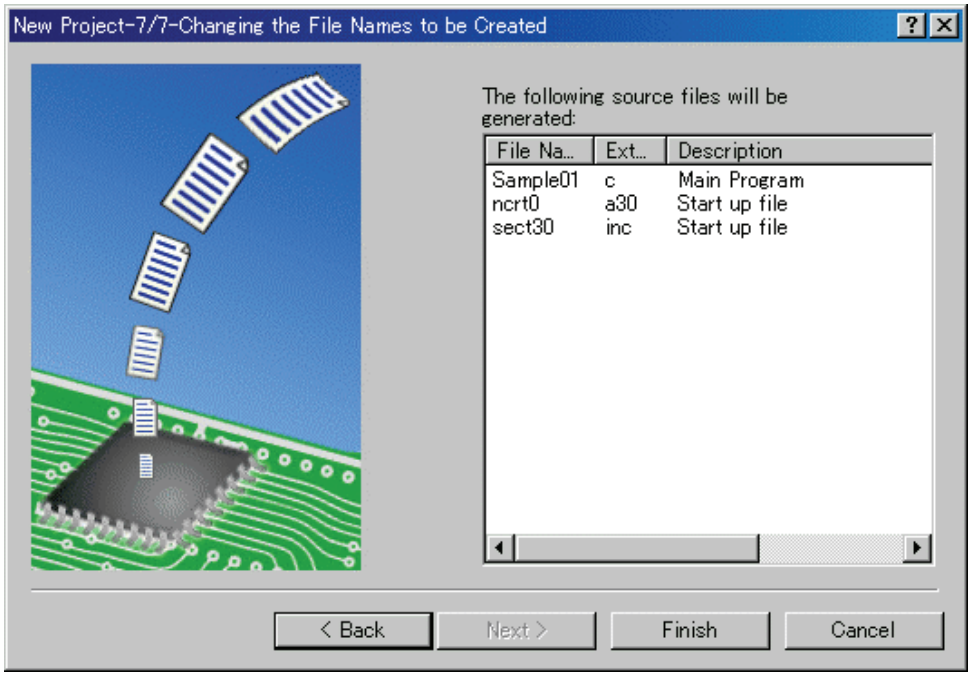
Set the configuration file name for each of the all selected target.
The configuration file saves the state of High-performance Embedded Workshop except for the target (emulator, simulator).



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

3.2.1.5 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.



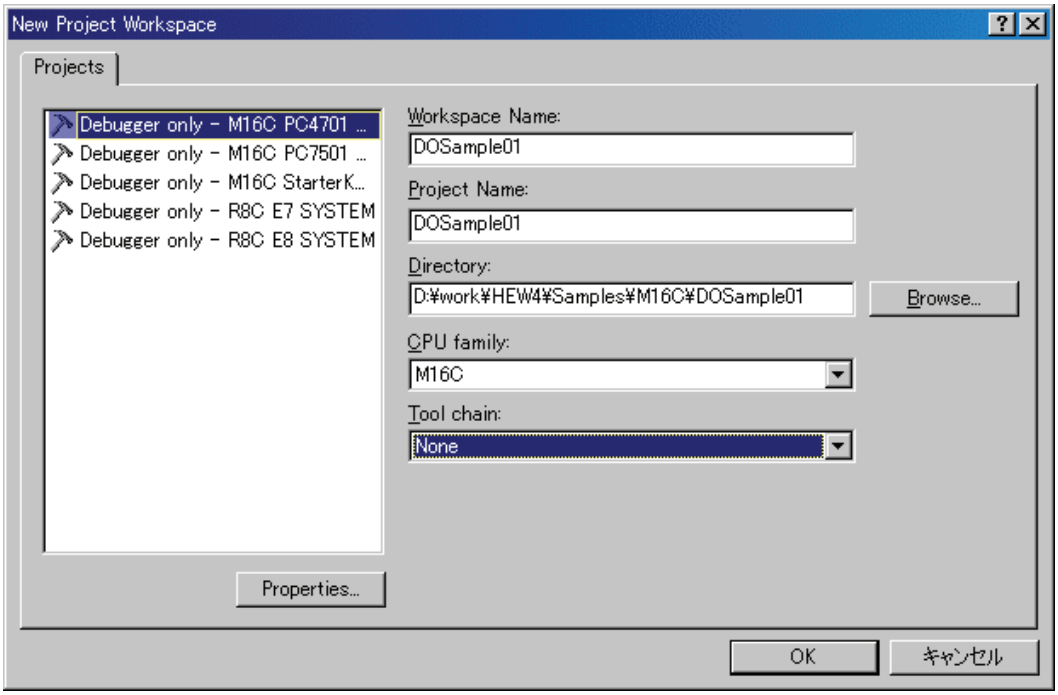
This is the end of the emulator settings.
Exit the Project Generator following the instructions on the screen.

3.2.2 Creating a New Workspace (Toolchain Not Used)

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

3.2.2.1 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.

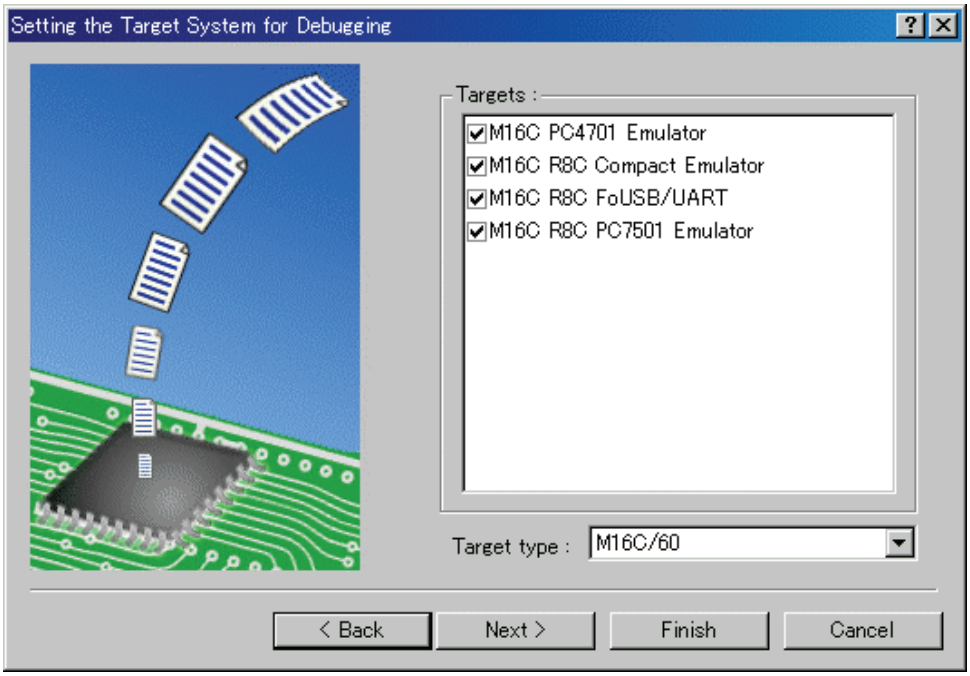


1. Select the target CPU family
   In the [CPU family] combo box, select the target CPU family.
2. Select the target toolchain
   In the [Tool chain] combo box, select "None". In this case, toolchain is not used.
   (When the toolchain has not been installed, the fixed information is displayed in this 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.

3.2.2.2 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.

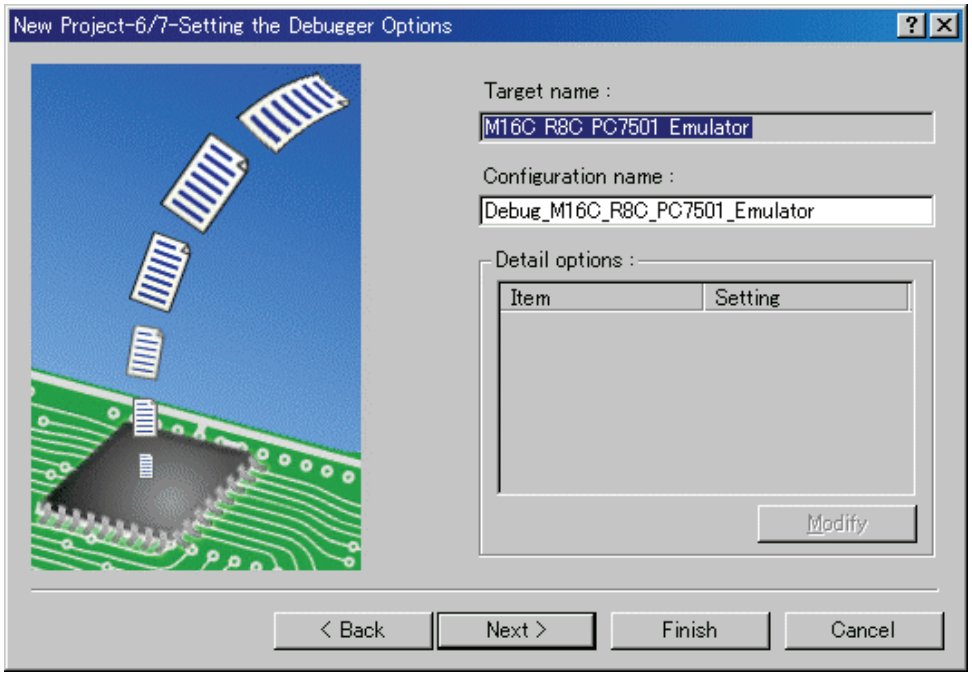


1. Selecting of the Target type
   In the [Target type] list box, select the target CPU type.
2. Selecting of the Target Platform
   In the [Targets] area, the target for the session file used when this debugger is activated must be selected here.
   Check the box against the target platform. (And choose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

And click the [Next] button.

3.2.2.3 Step3 : Setting the Configuration File Name

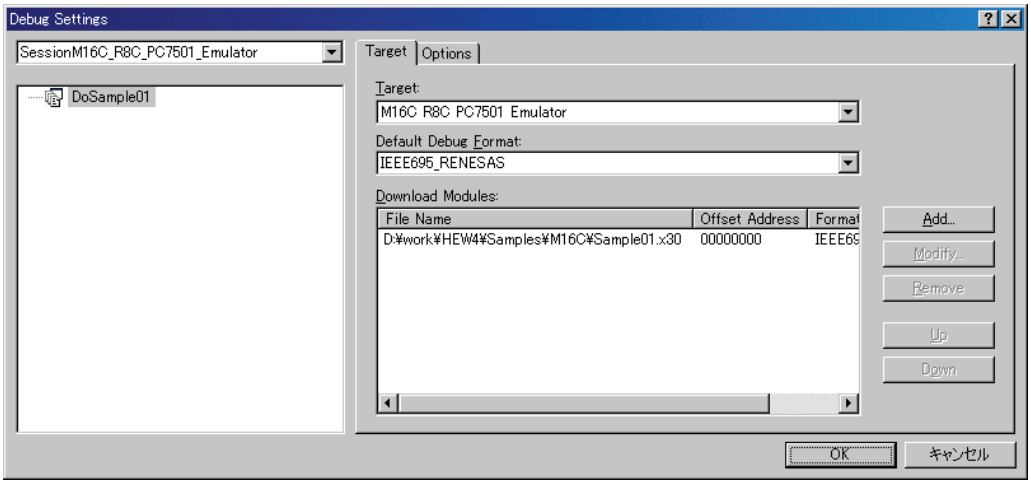
Set the configuration file name for each of the all selected target.
The configuration file saves the state of High-performance Embedded Workshop except for the target (emulator, simulator).



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.

3.2.2.4 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.

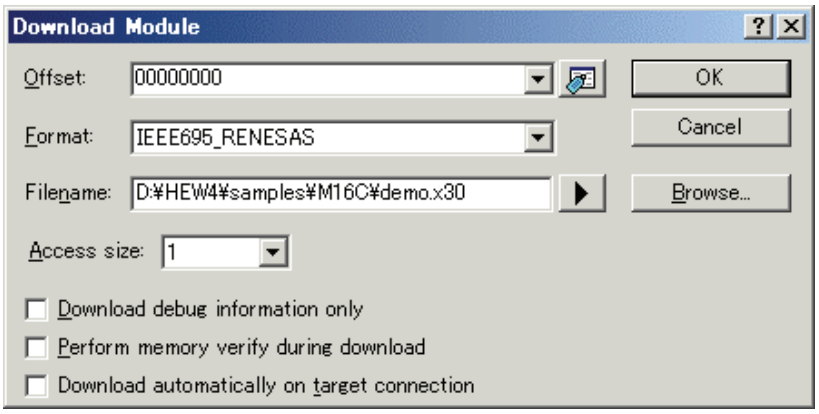


1. Select the product name to be connected in the [Target] drop-down list box.
2. Select the format of the load module to be downloaded in the [Default Debug Format] drop-down list box.

Format Name	Contents
Intel-Hex+Sym	Intel Hex format file with Symbol format file (When Using SRA74)
IEEE695_ICC740	IEEE-695 format file (When Using ICC740)

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

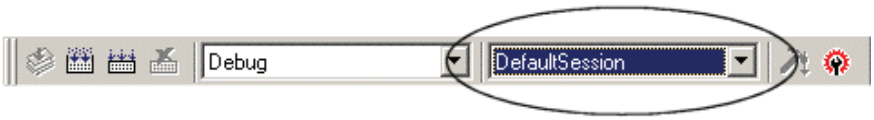
"Access size" and "Perform memory verify during download" is ignored.
The access size is always set to 1 and the verification does not work.

3.3 Starting the Debugger

The debugging can be started by connecting with an emulator.

3.3.1 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.

3.3.2 Ending the Emulator

The emulator can be exited by using the following methods:

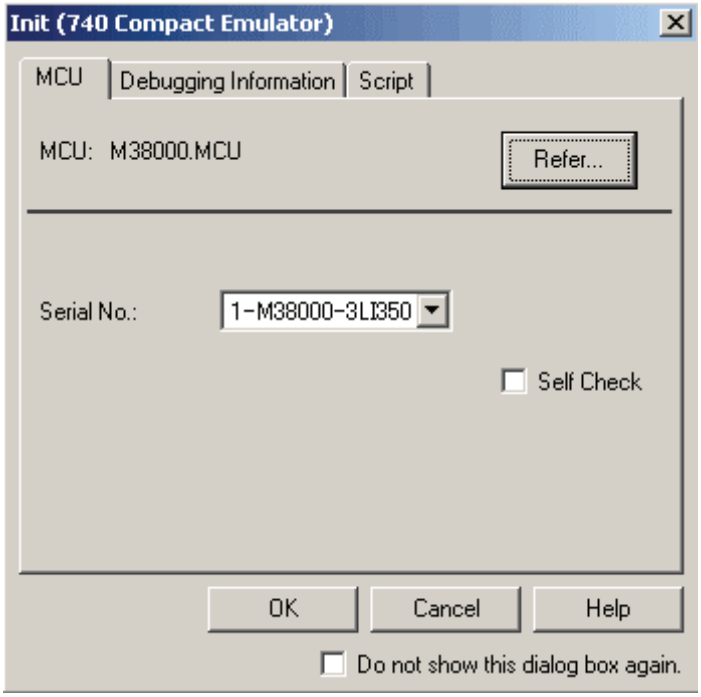
1. Selecting the "DefaultSession"
   Select the "DefaultSession" in the list box that was used at the time of emulator connection.
2. Exiting the High-performance Embedded Workshop
   Select [Exit] from the [File] menu. High-performance Embedded Workshop will be ended.

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.

4. Setup the Debugger

4.1 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.

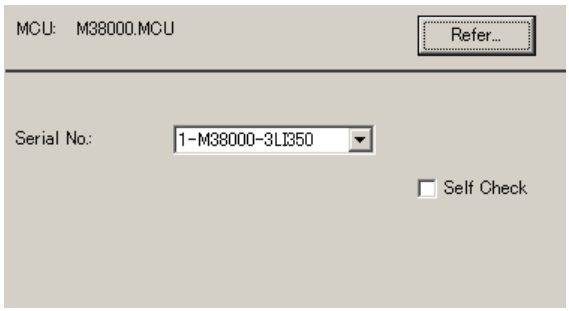


You can open the Init dialog using either one of the following methods:

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

4.1.1 MCU Tab

The specified content becomes effective when the next being start.



4.1.1.1 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.

If the corresponding MCU file is not contained in the debugger/emulation pod, you must create a new MCU file. To do this, see the following:
"4.4.1 Method of making MCU file(the 740 Debugger)"

4.1.1.1.1. 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.
For details, refer to "4.2 Setting of the Communication Interface"

4.1.1.1.2. Executing Self-Check

Specify this option to execute self-check* on the emulator when the debugger starts up.

Be sure to select the above check box only when you want to perform self-check at startup. Specify this option in the following cases:

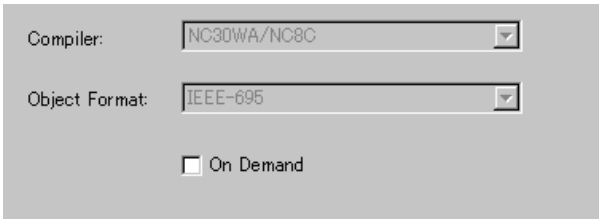
- When the firmware cannot be downloaded
- When although the firmware is successfully downloaded, the debugger does not start
- When the MCU goes wild or something is wrong with the trace results and you want to check whether the emulator is operating normally.

Select the check box to close the Init dialog box. After connecting to the emulator and confirming the firmware, the debugger will immediately start self-check on the emulator. (Self-check takes about 30 seconds to 1 minute.)
If an error is found in this self-check, the debugger displays the content of the error and is finished.
When the self-check terminated normally, the dialog box shown below is displayed. When you click OK, the debugger starts up directly in that state.

This specification is effective only when the debugger starts up.
* Self-check refers to the function to check the emulator's internal circuit boards for memory condition, etc. Refer to the user's manual of your emulator for details about the self-check function.

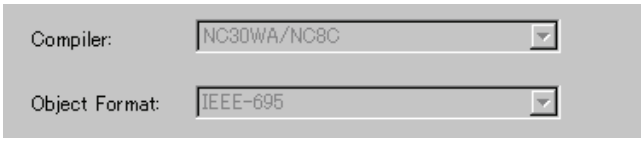
4.1.2 Debugging Information Tab

The specified content becomes effective when the next being download.



4.1.2.1 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...].

4.1.2.2 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.

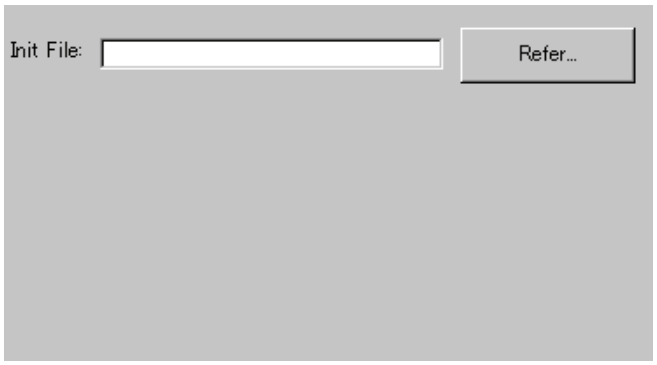
- On-memory method
  Debugging information is stored in the internal memory of your computer.
  This method is suitable when the load module (target program) size is small.
- On-demand method
  Debugging information is stored in a reusable temporary file on the hard disk of your computer.
  Because the 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.

4.1.3 Script Tab

The specified content becomes effective when the next being start.



4.1.3.1.1. Automatically Execute the Script Commands

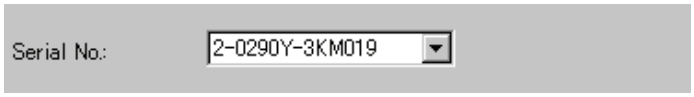
To automatically execute the script command at start of Debugger, click the "Refer" button to specify the script file to be executed.

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.

4.2 Setting of the Communication Interface

4.2.1 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 details on how to install USB device drivers, see
"2.3.1.1 Install of USB device driver"
The currently USB-connected emulators are listed in the Serial No. area. Select the serial No. of the emulator you want to connect.



4.3 Setup the Debugger for 740

4.3.1 Map Command

The memory map information must be altered to suit the target microcomputer's memory space by Map command.

Area	Mapping	Note
SFR	External	
RAM	External	
Internal ROM	Internal	
External ROM	External	Memory Expansion Mode, Microprocessor Mode

- Internal
  Enables the emulator's internal resources. The internal ROM area must be set for Internal because it is always emulated with the emulator's internal resources. If an external area is not allocated memory, you can use the emulator's internal memory by setting that area for Internal.
- External:
  Enables resources external to the emulator (including the internal SFR and RAM areas). The internal SFR and internal RAM areas must always be set for External. To enable the memory allocated for an external area, set that area for External.

The memory map attributes immediately after the emulator has started up are External for 0h-3FFFh and Internal for 4000h-FFFFh. Use the MAP command to look up or alter the memory map information. Execute the MAP command from the script window.

ATTENTION

[the case that the internal ROM area is located to the address before 4000h]
If the internal ROM area of the target mcu is located to the address before 4000h, please change the mapping of this area to INTERNAL.
Example)
when the internal ROM area is located from 1080h:

```
1080 to 3FFF -> Internal
```

[About special settings]
Always set the internal SFR and internal RAM areas for External. However, if the target MCU's RAM area is larger than the RAM included in the emulator MCU, set that area for Internal.
Example)
When the RAM area included in the emulator MCU is 40-1FF and the target MCU's internal RAM area is 40-2FF

```
40 to 1FF -> External
200 to 2FF -> Internal
```

4.4 Method of making MCU file

4.4.1 Method of making MCU file(the 740 Debugger)

The following content is sequentially described in the MCU file.
Please describe information on 1-4 referring to the data book on MCU used.

1. Number of stack page selection bit
2. Address of CPU mode register
3. End address of stack*1
4. Address of reset vector
5. POD number*2
6. Firmware name*3
7. MCU Information No.*4

*1End address of stack
Specify the last address of the area to be used as the stack. Consider the initial value of the stack page selection bit in the CPU mode register. (The initial value of the stack page selection bit depends on the microcomputer.) For a microcomputer which sets the stack page selection bit initial value to "0", the allowable designation range is a 0 page address range (0h to FFh). For a microcomputer which sets the stack page selection bit initial value to "1", the allowable designation range is a 1 page address range (100h to 1FFh).

*2POD number
In the compact emulator, please set up "0"

*3Firmware name
In the compact emulator, please set up "M38000".

*4MCU Information No.
In the compact emulator, please set up "00".

4.4.1.1 Example

```
2
3B
FF
FFFC
0
M38000
00
```

Tutorial

(Blank Page)

5. Tutorial

5.1 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.

When using the assembler package for 740 family

The tutorial program for the assembler package for 740 family is prepared. If you use the assembler package for 740 family, please use it

- Please read this tutorial with replacing function names with subroutine name. (e.g. replace "function sort()" with "subroutine sort")
- About the source file name, also please replace it with the corresponding one.
- The diagrams in this tutorial are for C program. The displayed diagram for the assembler program may different from them.
- Step9 and Step12 are descriptions of C program.

5.2 Usage

Please follow these instructions:

5.2.1 Step1 : Starting the Debugger

5.2.1.1 Preparation before Use

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

5.2.1.2 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 "4 Setup the Debugger".
After the setup of a debugger, it will function as a debugger.

5.2.2 Step2 : Checking the Operation of RAM

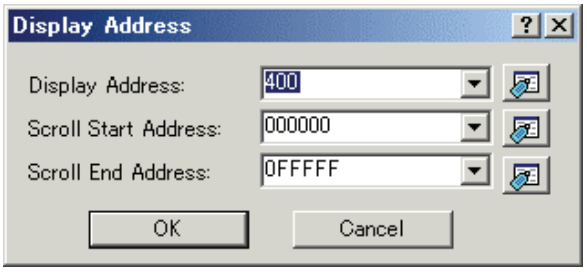
Check that RAM is operating correctly. Display and edit the contents of the memory in the [Memory] window to check that the memory is operating correctly.

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.

5.2.2.1 Checking the Operation of RAM

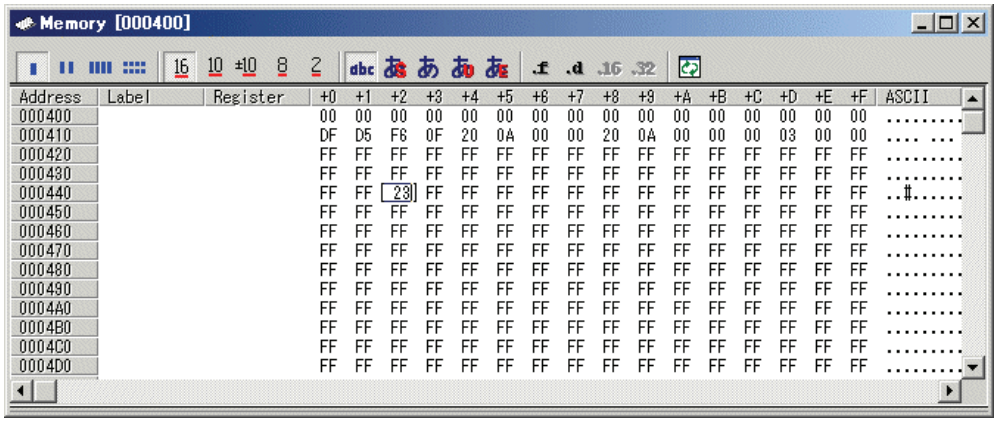
Select [Memory] from the [CPU] submenu of the [View] menu and enter the RAM address (Here, enter H'400) in the [Display Address] edit boxes. The [Scroll Start Address] and [Scroll End Address] editing box is left to a default setting. (By default, the scroll range is set to 0h to the maximum address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



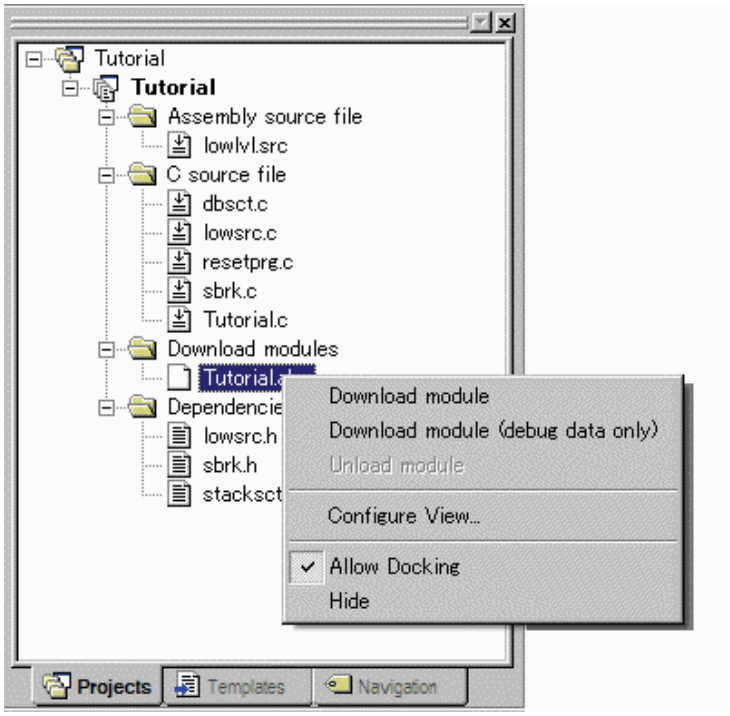
Placing the mouse cursor on a point in the display of data in the [Memory] window and double-clicking allows the values at that point to be changed.

5.2.3 Step3 : Downloading the Tutorial Program

5.2.3.1 Downloading the Tutorial Program

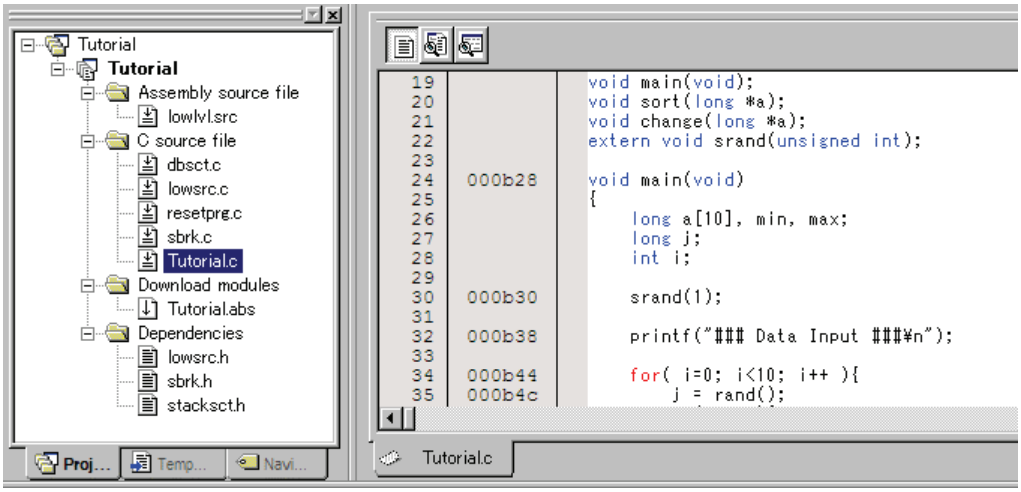
Download the object program to be debugged. The download file and the address to be downloaded will depends on the target mcu you uses. Please replace the screen image and addresses with corresponding one to your target mcu.

- The Debugger for 740
  If you use the C Compiler Package for 740 Family, select [Download module] from [Tutorial.695] under [Download modules].
  If you use the Assembler Package for 740 Family, select [Download module] from [Tutorial.hex] under [Download modules].



5.2.3.2 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.



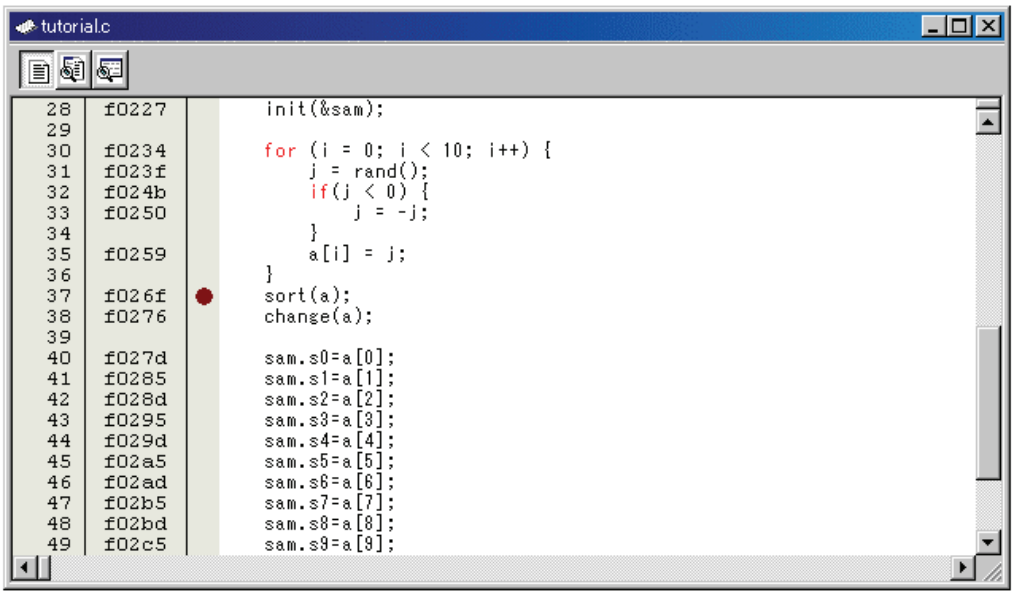
Select the [Format Views...] option from the [Setup] menu to set a font and size that are legible, if necessary.
Initially the [Editor(Source)] window shows the start of the user program, but the user can use the scroll bar to scroll through the user program and look at the other statements.

5.2.4 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.

5.2.4.1 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.

5.2.5 Step5 : Executing the Program

Execute the program as described in the following:

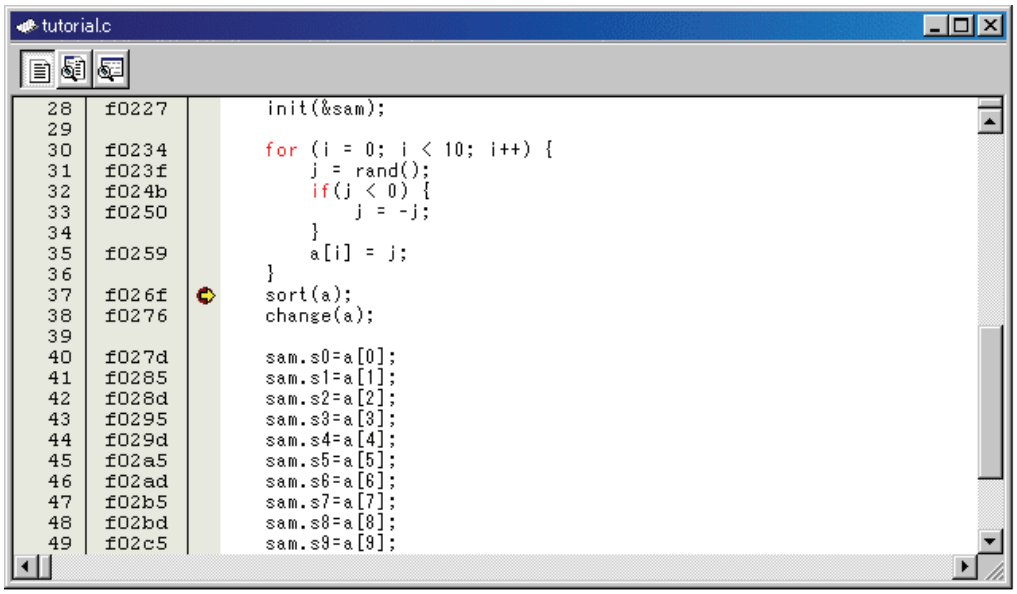
5.2.5.1 Resetting of CPU

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.

5.2.5.2 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.

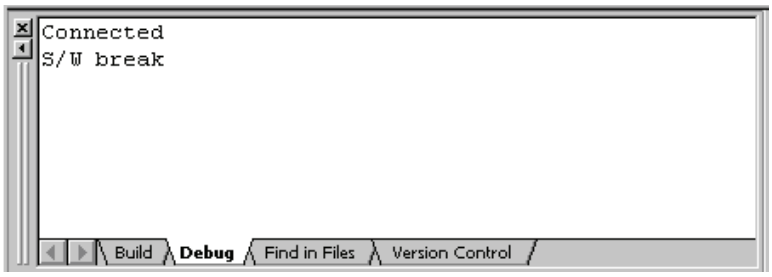


Note

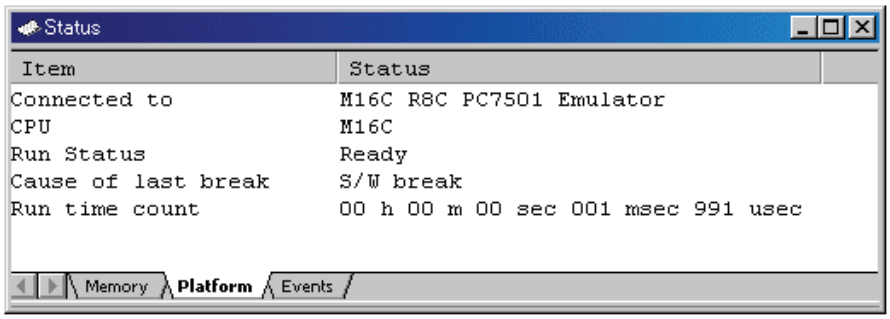
When the source file is displayed after a break, a path of the source file may be inquired. In this case, please specify the location of a source file.

5.2.5.3 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.



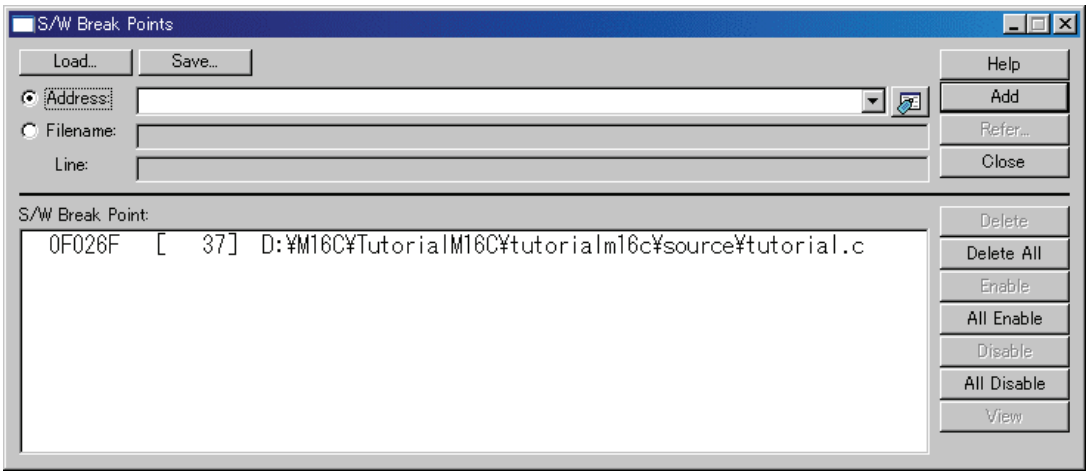
Please refer to "10 Display the Cause of the Program Stoppage " about the notation of a break factor.

5.2.6 Step6 : Reviewing Breakpoints

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

5.2.6.1 Reviewing Breakpoints

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



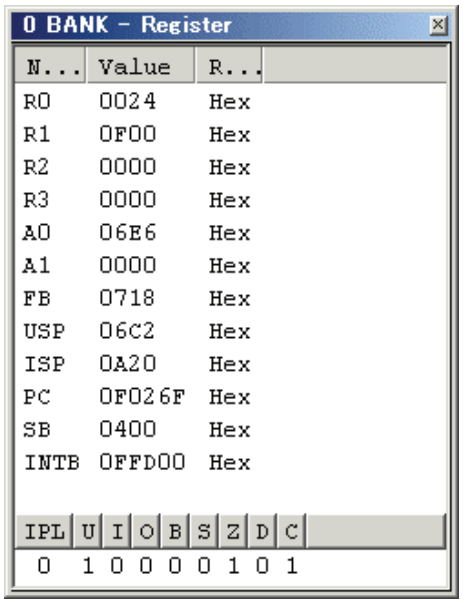
This window allows the user to set or change breakpoints, define new breakpoints, and delete, enable, or disable breakpoints.

5.2.7 Step7 : Viewing Register

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

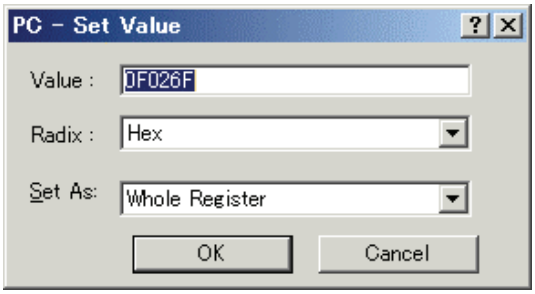
5.2.7.1 Viewing Register

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.



5.2.7.2 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.

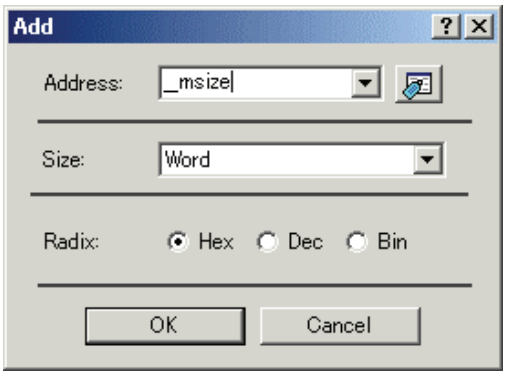


5.2.8 Step8 : Viewing Memory

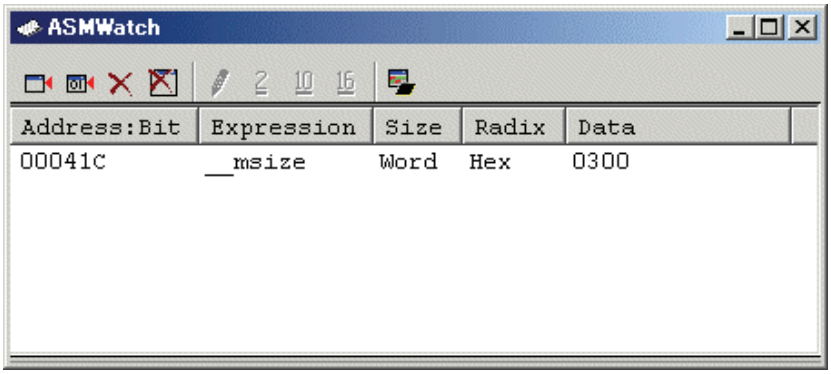
When the label name is specified, the user can view the memory contents that the label has been registered in the [ASM Watch] window.

5.2.8.1 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.

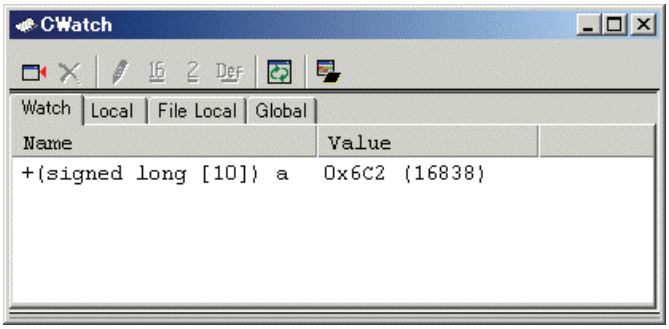


5.2.9 Step9 : Watching Variables

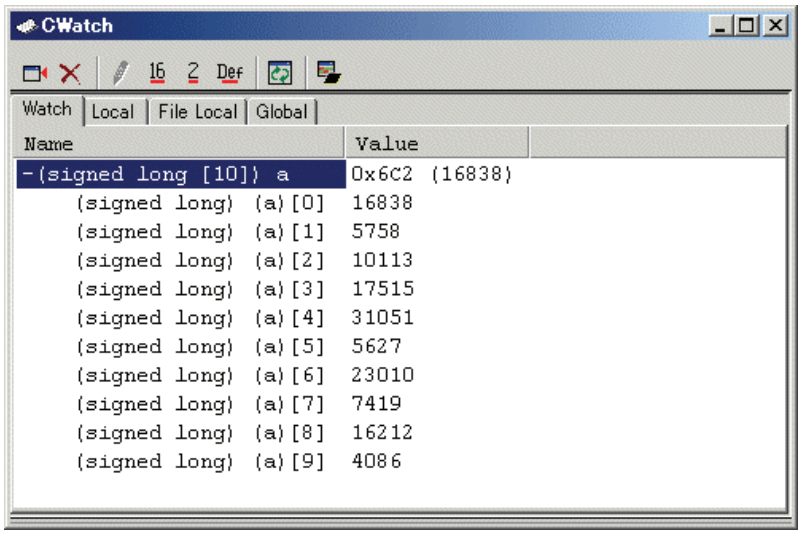
As the user steps through a program, it is possible to watch that the values of variables used in the user program are changed.
If the downloaded program is the program generated by the assembler package for 740 family, you can not watch variables in C watch window.

5.2.9.1 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.

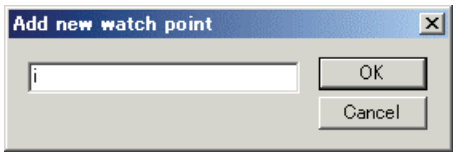


The user can click mark '+' at the left side of array a in the [C Watch] window to watch all the elements.

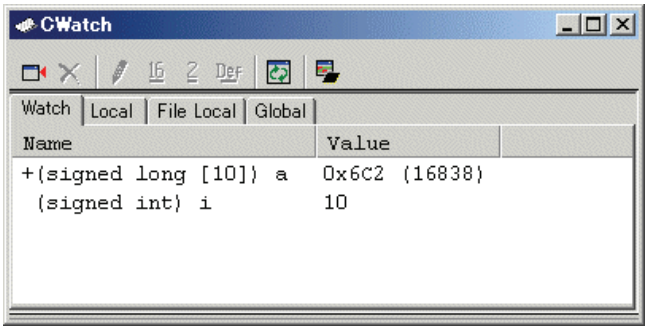


5.2.9.2 Registering Variable

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.



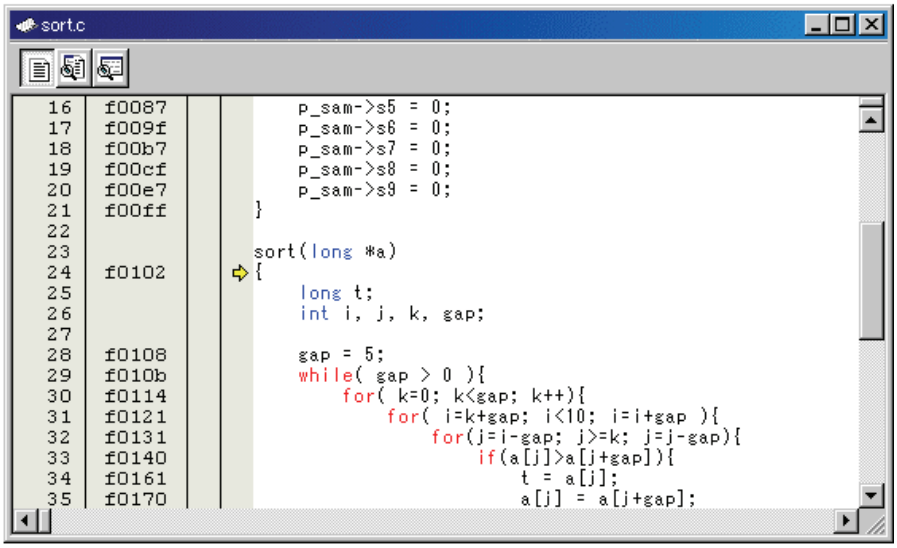
5.2.10 Step10 : Stepping Through a Program

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.

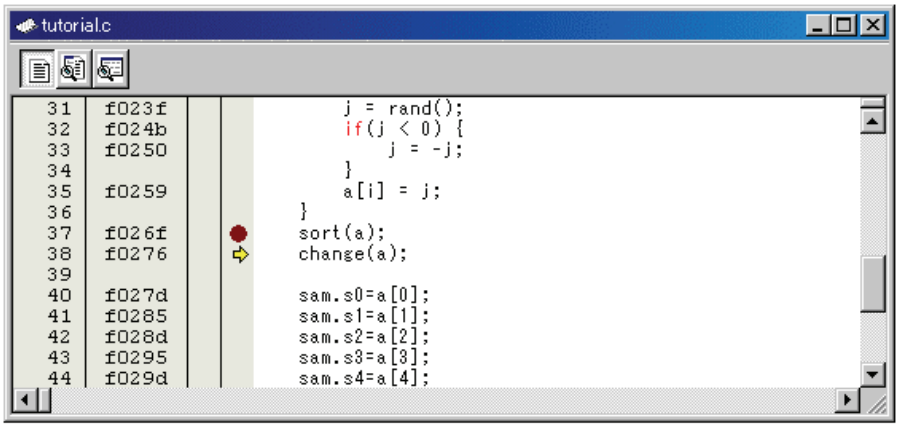
5.2.10.1 Executing [Step In] Command

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.



5.2.10.2 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.

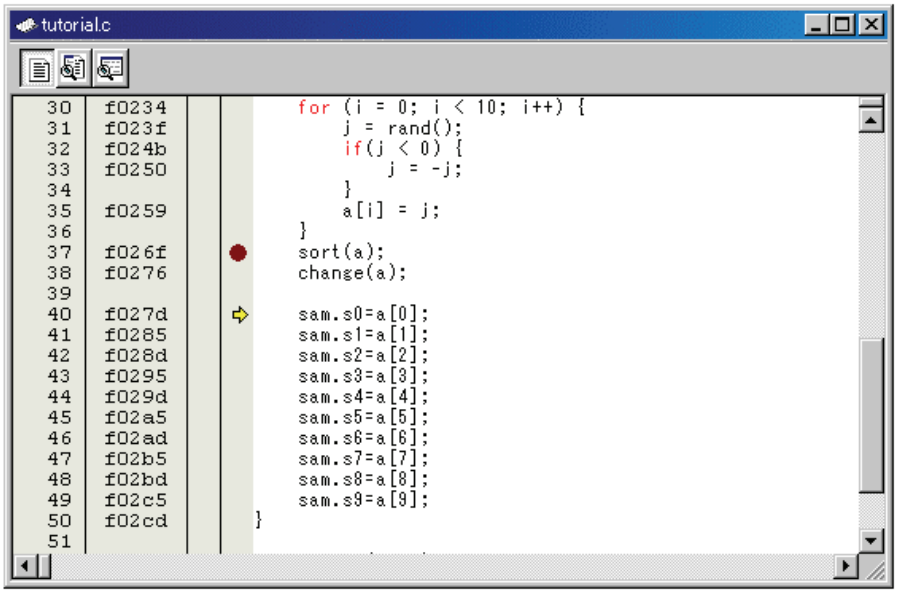


Note

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

5.2.10.3 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.



5.2.11 Step11 : Forced Breaking of Program Executions

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

5.2.11.1 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.

5.2.12 Step12 : Displaying Local Variables

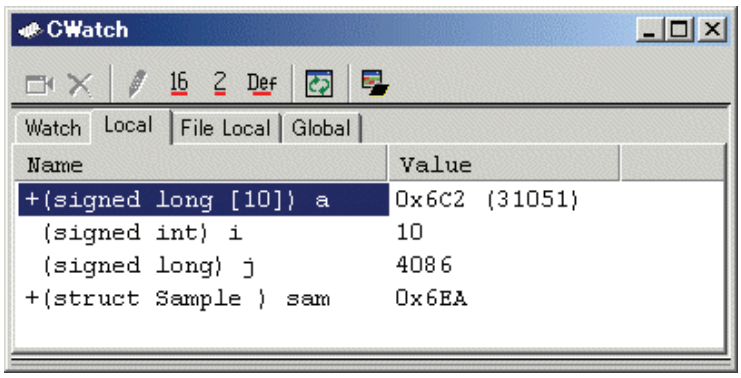
The user can display local variables in a function using the [C Watch] window.
If the downloaded program is the program generated by the assembler package for 740 family, you can not watch variables in C watch window.

5.2.12.1 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.



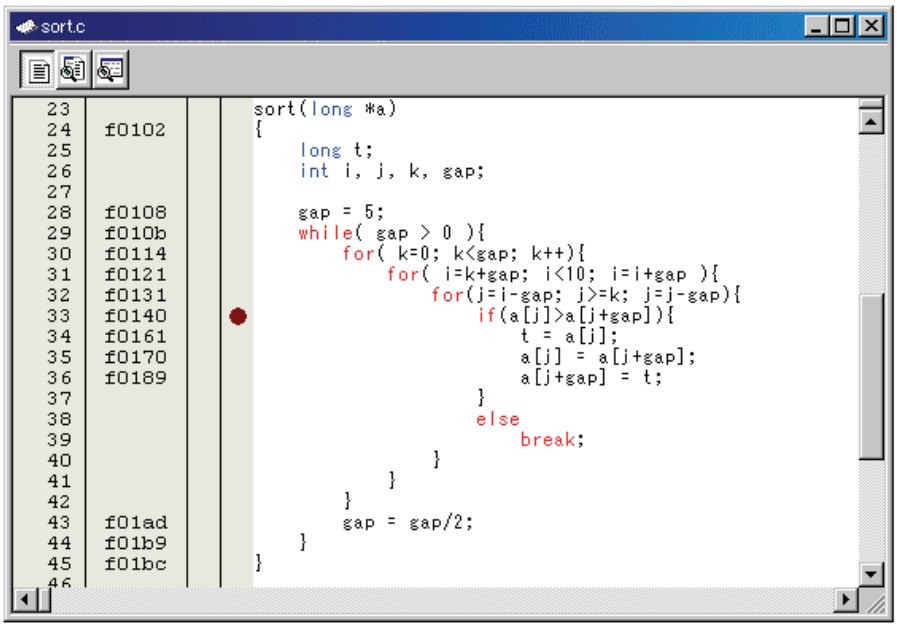
Click mark '+' at the left side of array a in the [Locals] window to display the elements.
When the user refers to the elements of array a before and after the execution of the sort function, it is clarified that random data is sorted in descending order.

5.2.13 Step13 : Stack Trace Function

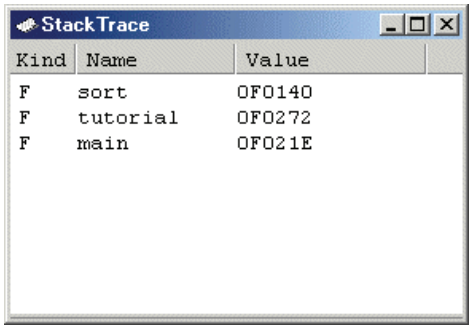
The debugger uses the information on the stack to display the names of functions in the sequence of calls that led to the function to which the program counter is currently pointing.
The debugger for 740 doesn't support the stack trace function.

5.2.13.1 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.

5.2.14 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)

6. Windows/Dialogs

The window of this debugger is shown below.
When the window name is clicked, the reference is displayed.

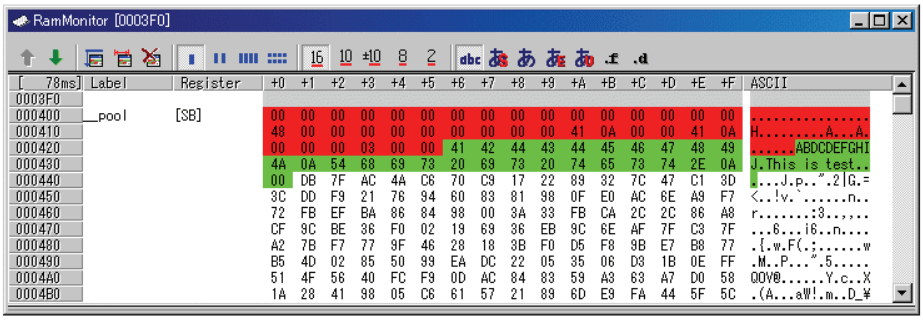
Window Name	View Menu
RAM Monitor Window	[View]->[CPU]->[RamMonitor]
ASM Watch Window	[View]->[Symbol]->[ASMWatch]
C Watch Window	[View]->[Symbol]->[CWatch]
Script Window	[View]->[Script]
S/W Break Point Setting Window	[View]->[Break]->[S/W Break Points]
H/W Break Point Setting Dialog Box	[View]->[Break]->[H/W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

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

6.1 RAM Monitor Window

The RAM monitor window is a window in which changes of memory contents are displayed while running the target program.
The relevant memory contents are displayed in dump form in the RAM monitor area by using the realtime RAM monitor function. The displayed contents are updated at given intervals (by default, every 100 ms) 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 "6.1.2 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.
- The background colors of the data display and code display areas are predetermined by access attribute, as shown below.

Access attribute	Background color
Read accessed address	Green
Write accessed address	Red
Non-accessed address	White

The background colors can be changed.

ATTENTION

- The RAM monitor window shows the data that have been accessed through the bus. Therefore, changes are not reflected in the displayed data unless they have been accessed via the target program as in the case where memory is rewritten directly from an external I/O.
- If the data in the RAM monitor area are displayed in lengths other than the byte, it is possible that the data will have different memory access attributes in byte units. If bytes in one data have a different access attribute as in this case, those data are enclosed in parentheses when displayed in the window. In that case, the background color shows the access attribute of the first byte of the data.

- The displayed access attributes are initialized by downloading the target program.
- 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

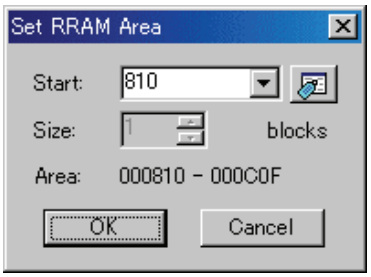
6.1.1 Extended Menus

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.

6.1.2 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.

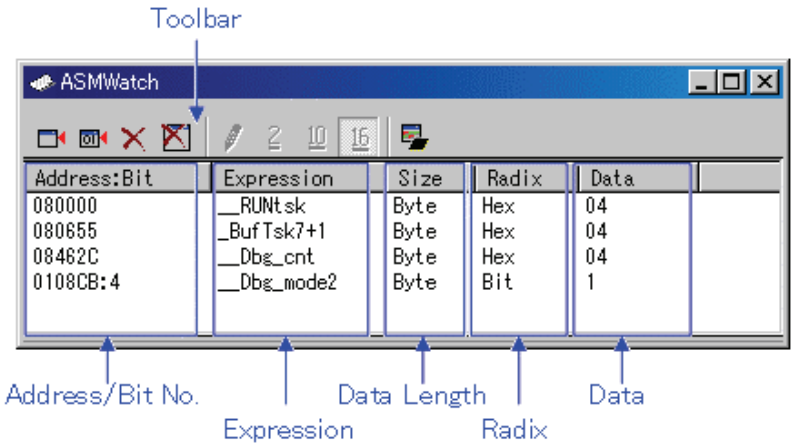
- Specify the RAM monitor area by its start address. The size cannot be changed (fixed to 1 Kbyte).
- The start address can be specified in 0x10 byte units.
  If you specify a non-aligned address value, it is rounded off to the nearest address value in 0x10 byte units before being set.

6.1.2.1 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.)

6.2 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.
If a registered address resides within the RAM monitor area, the memory content at that address is updated at given intervals (by default, every 100 ms) during program execution.



- 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, radixes and datas can be changed by in-place editing.

ATTENTION

- The RAM monitor obtains the data accessed through the bus. Any change other than the access from the target program will not be reflected.
- If the display data length of the RAM monitor area is not 1 byte, the data's access attribute to the memory may varies in units of 1 byte. In such a case that the access attribute is not unified within a set of data, the data's access attribute cannot be displayed correctly. In this case, the background colors the access attribute color of the first byte of the data.

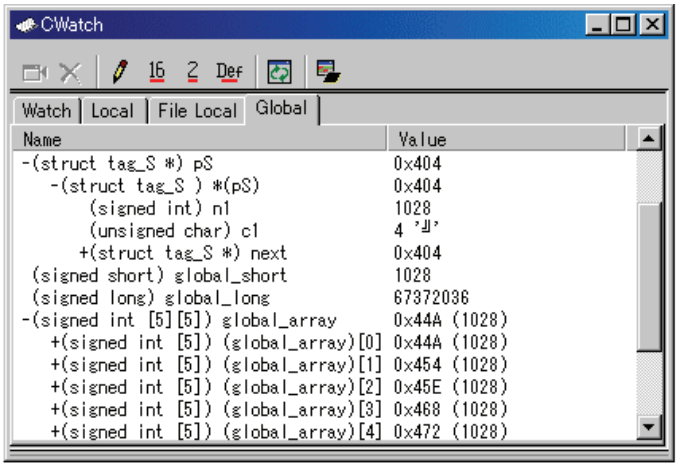
6.2.1 Extended Menus

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-lebel 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 moniter 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.

6.3 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.
When RAM monitor function is effective and the C watch points are within the RAM monitor area, the displayed values are updated during execution of the target program.



- 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.
- Values can be inspected in real time during program execution by using the RAM monitor function.

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.
- The RAM monitor obtains the data accessed through the bus. Any change other than the access from the target program will not be reflected.
- The variables, which are changed in real-time, are global variables and file local variables only.
- If the display data length of the RAM monitor area is not 1 byte, the data's access attribute to the memory may varies in units of 1 byte. In such a case that the access attribute is not unified within a set of data, the data's access attribute cannot be displayed correctly. In this case, the background colors the access attribute color of the first byte of the data.

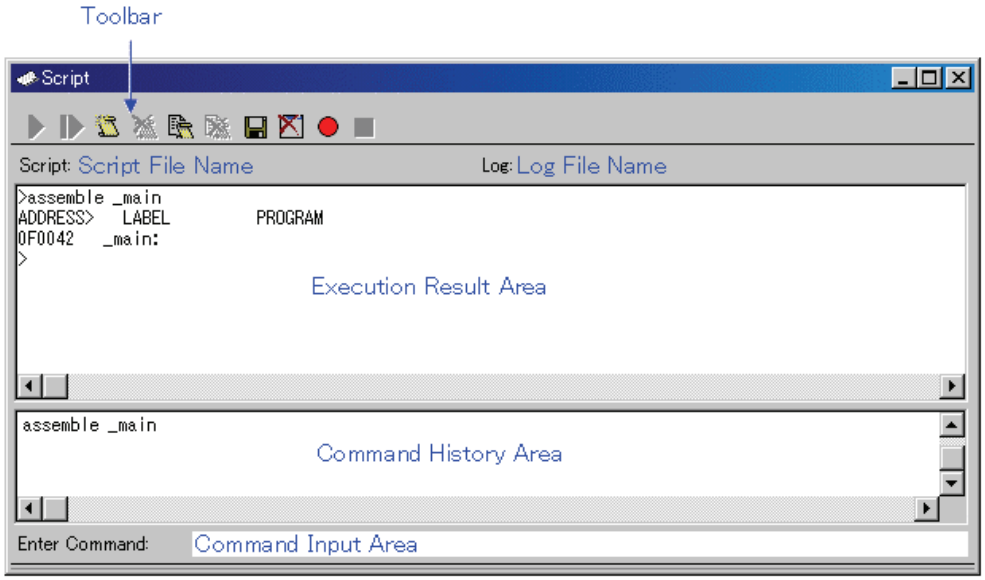
6.3.1 Extended Menus

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 moniter 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.

6.4 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.

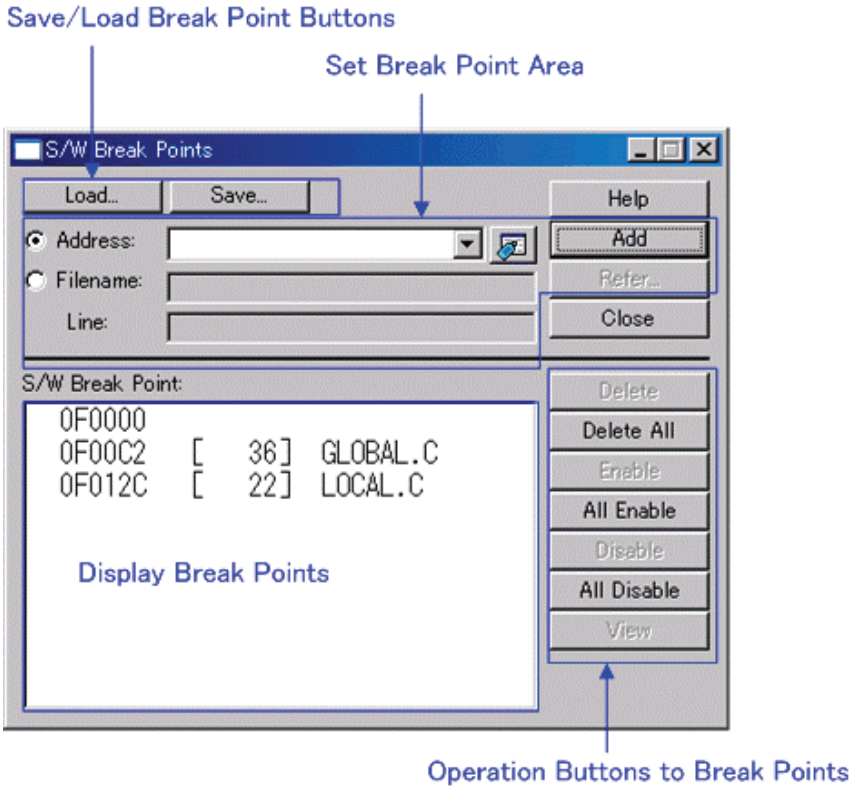
6.4.1 Extended Menus

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.

6.5 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.

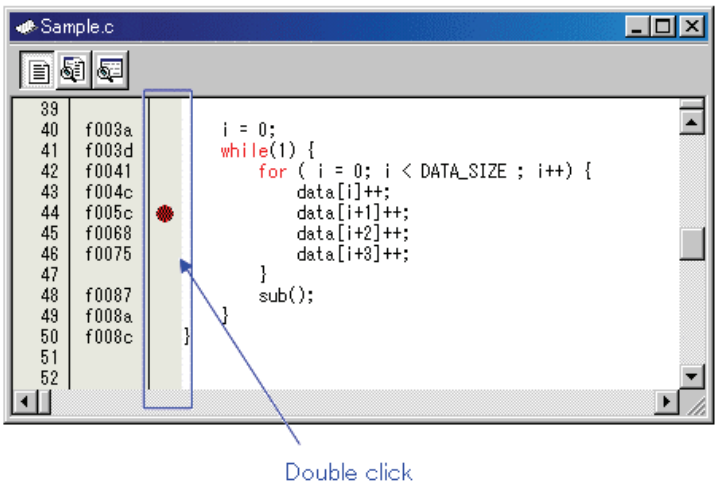
6.5.1 Command Button

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.

6.5.2 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 "10711.1.2 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.)

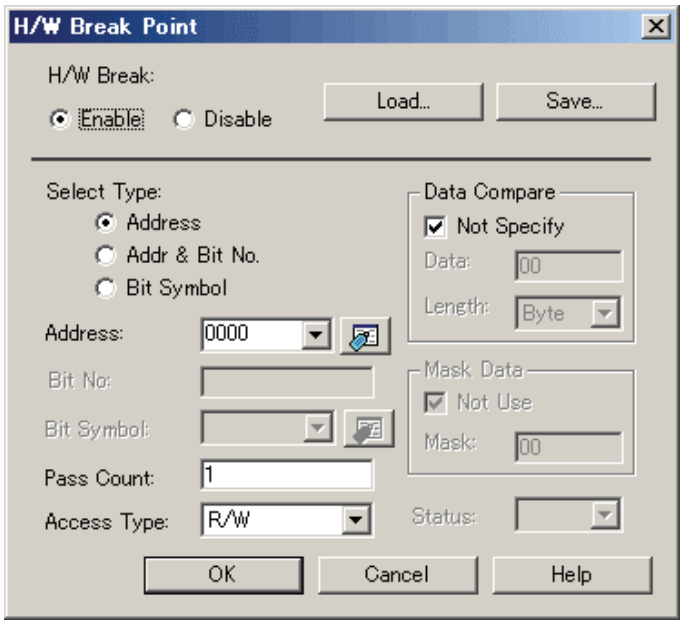


You can delete the break point by double-clicking again in the break point setting area ("S/W breakpoints" column).

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.

6.6 H/W Break Point Setting Dialog

The H/W Break Point Setting dialog box allows you to set hardware break points.
If you are using the emulator, you can set one address breakpoints with pass counts.



- As address break point access types, you can specify writing data to the address break point (Write), reading data from the address break point (Read), reading or writing data (R/W), and fetching instructions (Fetch).
- You can also specify that execution breaks if the data read from or written to the address break point has a specific value. Moreover, you can specify valid and invalid bits for the specific value.
- Hardware breakpoints can be saved to a file by clicking "Save". To read hardware breakpoint settings from the saved file, click "Load".

6.6.1 Specify the Break Event

Specify the H/W Break Point Setting Dialog

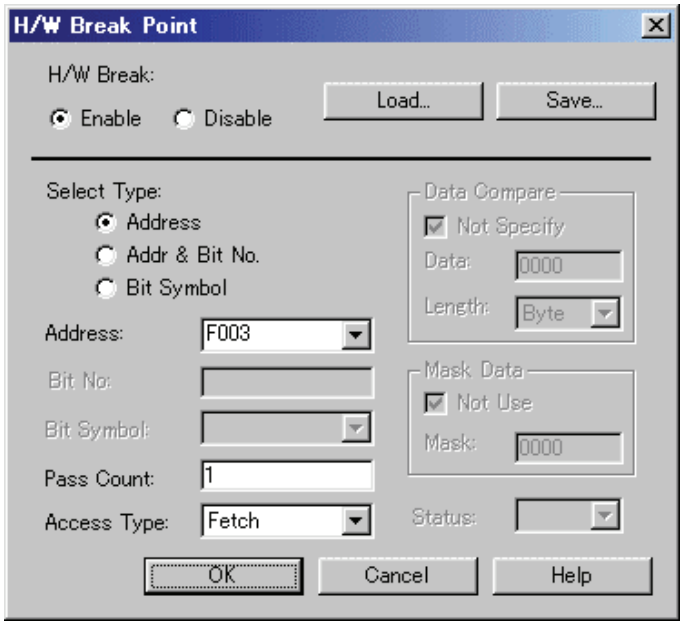
- You can specify the access type (Read, Write, R/W, Fetch).
- You can specify the data to be read/written. You can also specify "Enable Bit/Disable Bit" to the data (mask designation).
- You can specify the pass count (1 to 255).

ATTENTION

A write of word length data to the odd address cannot be detected.
For 740, a write of word length data to the even address cannot be detected, either.

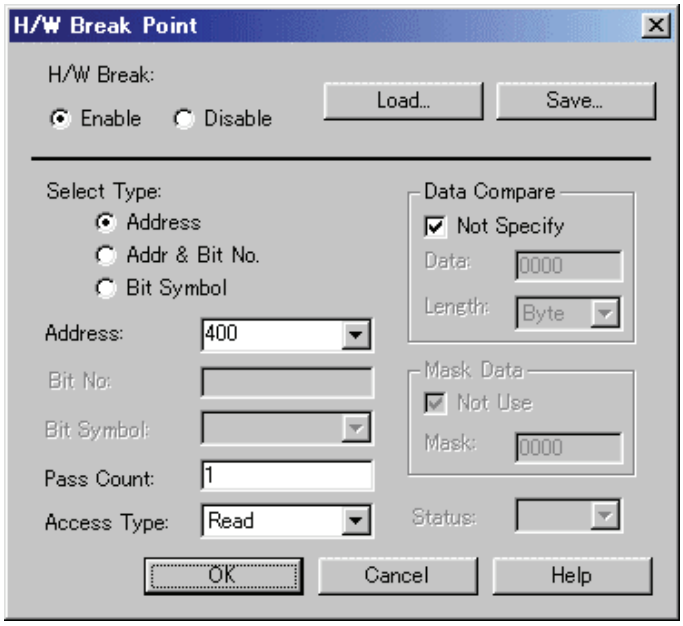
6.6.1.1 Instruction Fetch

Set as below.
Example) Executing a instruction at address F0003h

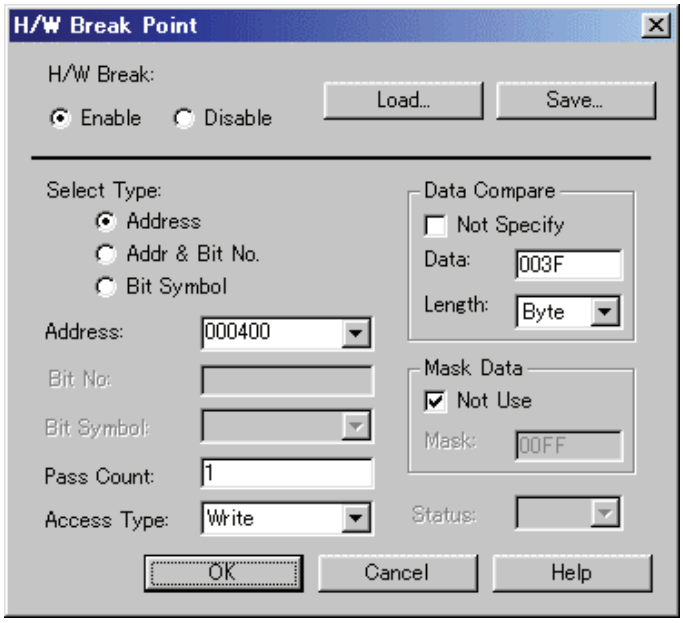


6.6.1.2 Memory Access

Set as below.
Example) Reading to even address 400h



Example) Writing byte length data 3Fh to even address 400h



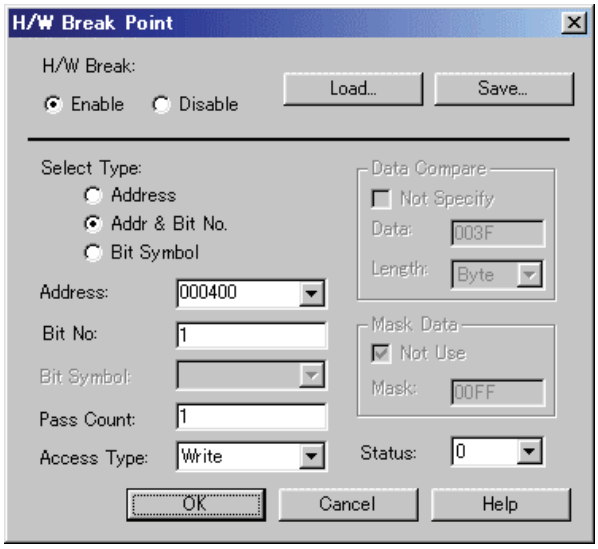
ATTENTION

When specifying a breakpoint of byte access type at an odd address to an area with 16-bit memory access bus width, the following settings are required:

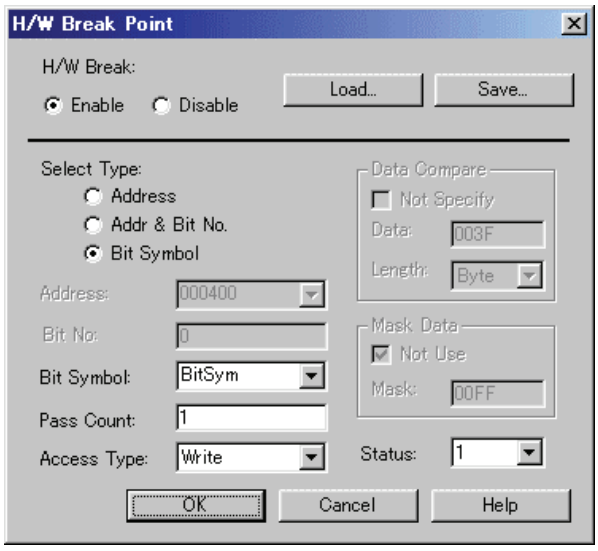
- Set the data to be compared in the high-order 8 bits of the comparison data.
- Set "Word" in the Length list box.
- Set "00" in the low-order bytes of the mask data.

6.6.1.3 Bit Access

Set as below.
Example) Writing data 0 to bit 1 of address 400h.



Example) Writing data 0 to bitsymbol "BitSym".

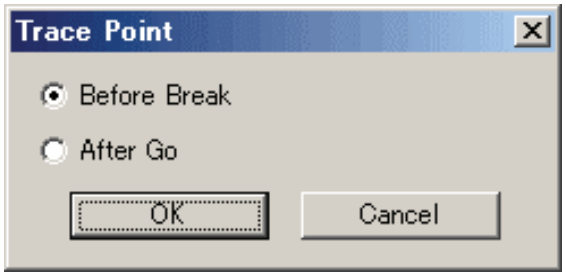


6.6.1.4 Disable the Hardware Break

Click the Disable button.

6.7 Trace Point Setting Dialog

The Trace Point Setting dialog is used to set trace points.



6.7.1 Specify the Trace Range

For the compact emulator debugger, 32K cycles equivalent of data can be recorded.

Before Break	Stores the 32K cycles (-32K to 0 cycles) to the point at which the target program stops.
After Go	Stores the 32K cycles (-32K to 0 cycles) of trace data after the trace starts.

6.8 Trace Window

The Trace Window is used to display the results of real-time trace measurement. The measurement result can be displayed in the following display modes.

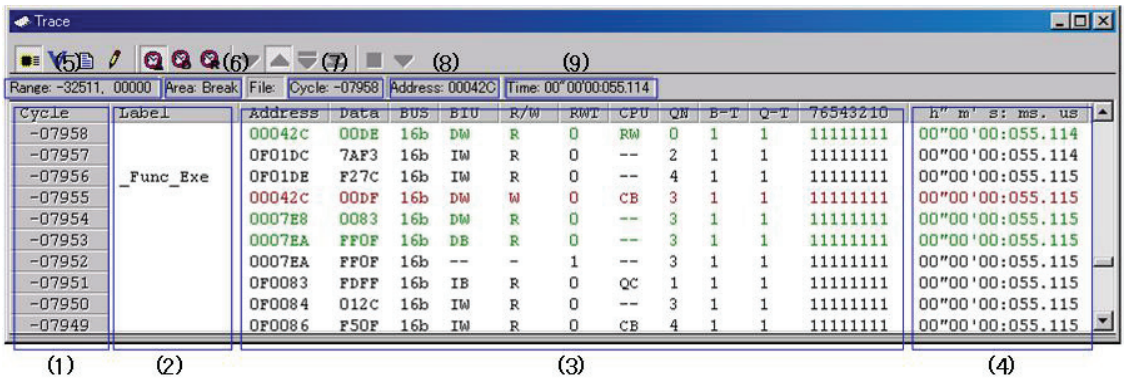
- Bus mode
  This mode allows you to inspect cycle-by-cycle bus information. The display content depends on the MCU and emulator system used. In addition to bus information, this mode allows disassemble, source line or data access information to be displayed in combination.
- Disassemble mode
  This mode allows you to inspect the executed instructions. In addition to disassemble information, this mode allows source line or data access information to be displayed in combination.
- Data access mode
  This mode allows you to inspect the data read/write cycles. In addition to data access information, this mode allows source line information to be displayed in combination.
- Source mode
  This mode allows you to inspect the program execution path in the source program.

The measurement result is displayed when a trace measurement has finished. When a trace measurement restarts, the window display is cleared.

The range of a trace measurement can be altered in the Trace Point Setting Window. For details about this window, refer to "6.7 Trace Point Setting Dialog" With default settings, the trace information immediately before the program has stopped is recorded.

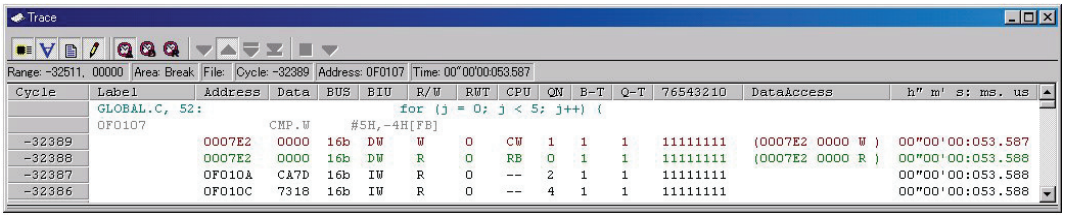
6.8.1 Configuration of Bus Mode

When bus mode is selected, trace information is displayed in bus mode. Bus mode is configured as shown below.
The display content in bus mode differs depending on the MCU or emulator system used.



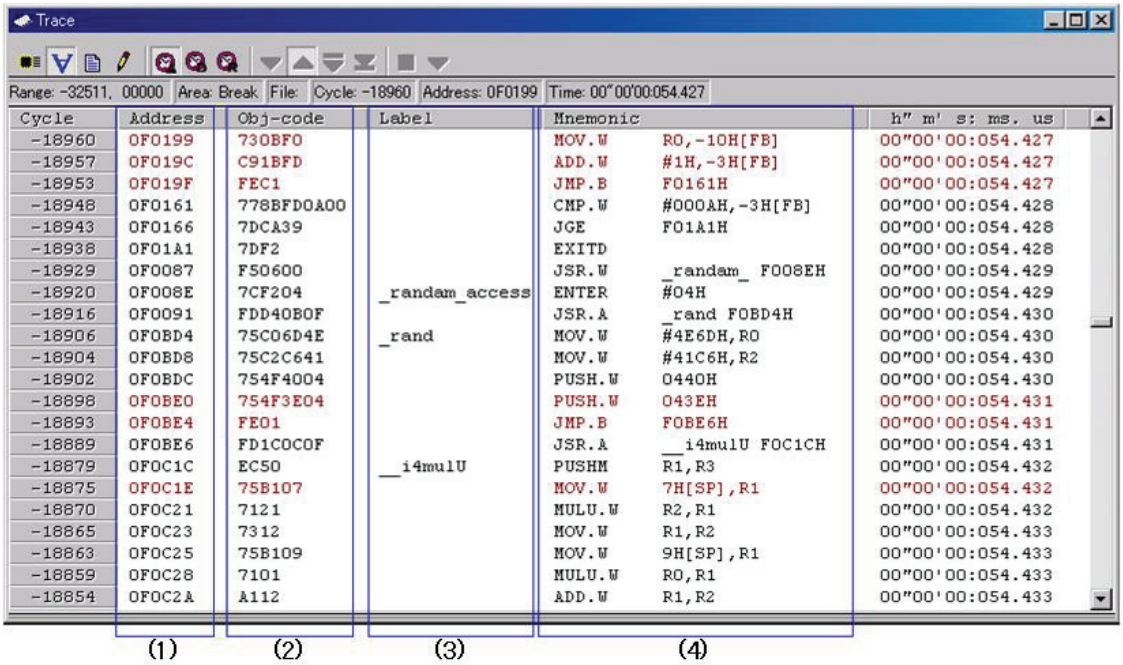
1. Cycle display area:
   Shows trace cycles. Double-click here to bring up a dialog box to change the displayed cycle.
2. Label display area:
   Shows labels corresponding to address bus information. Double-click here to bring up a dialog box to search for addresses.
3. Bus information display area:
   The content displayed here differs depending on the MCU or emulator system used.
   - Refer to "6.8.6 Display of bus information on the 740 Debugger"
4. Time information display area:
   Shows time information of trace measurement result. One of the following three modes can be selected from the menu. (The compact emulator debugger is not supported.)
   - Absolute Time:Shows an elapsed time from the time the program started running up to now in terms of absolute time (default).
   - Differences:Shows a differential time from the immediately preceding cycle.
   - Relative Time:Shows a relative time from the selected cycle. Note, however, that this mode changes to the absolute time display mode when the trace measurement result is updated.
5. Acquired range of trace measurement result:
   Shows the currently acquired range of trace measurement result.
6. Trace measurement range:
   Shows the currently set range of trace measurement.
7. First line cycle:
   Shows the cycle of the first line displayed.
8. First line address:
   Shows the address of the first line displayed.
9. First line time:
   First line time:
   Shows the time information of the first line displayed.
10. Window splitting box:
    Double-clicking this box splits the window into parts.

In addition to bus information, the window can display disassemble, source line or data access information in combination. In this case, the display will be similar to the one shown below.



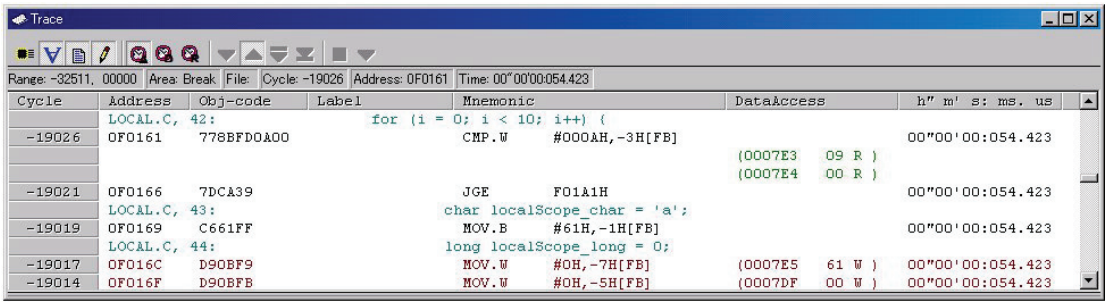
6.8.2 Configuration of Disassemble Mode

When disassemble mode is selected while bus mode is unselected, trace information is displayed in disassemble mode. Disassemble mode is configured as shown below.



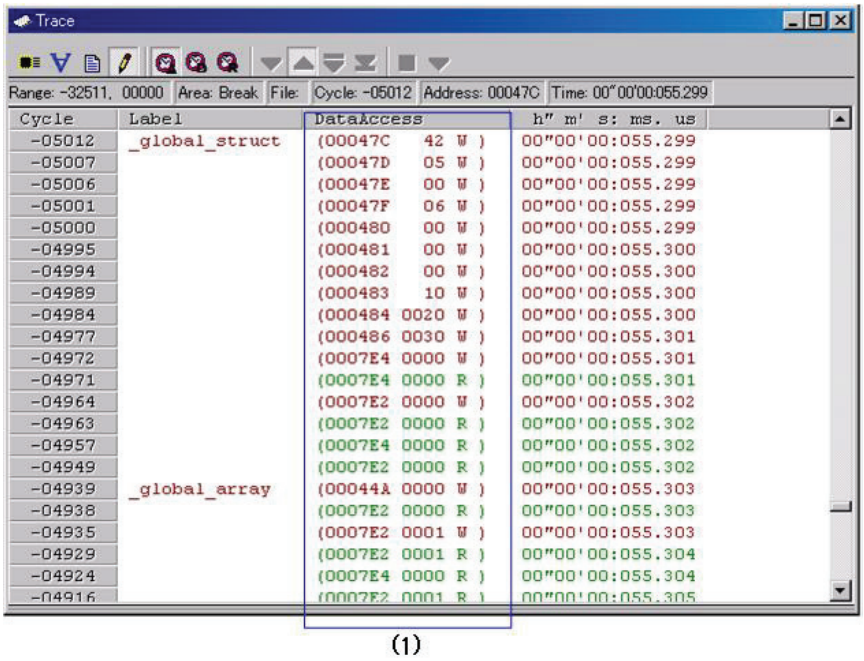
1. Address display area:
   Shows addresses corresponding to instructions. Double-click here to bring up a dialog box to search for addresses.
2. Object code display area:
   Shows the object codes of instructions.
3. Label display area:
   Shows labels corresponding to instruction addresses. Double-click here to bring up a dialog box to search for addresses.
4. Mnemonic display area:
   Shows the mnemonics of instructions.

Other display areas are the same as in bus mode.
In addition to disassemble information, the window can display source line or data access information in combination. In this case, the display will be similar to the one shown below.



6.8.3 Configuration of Data Access Mode

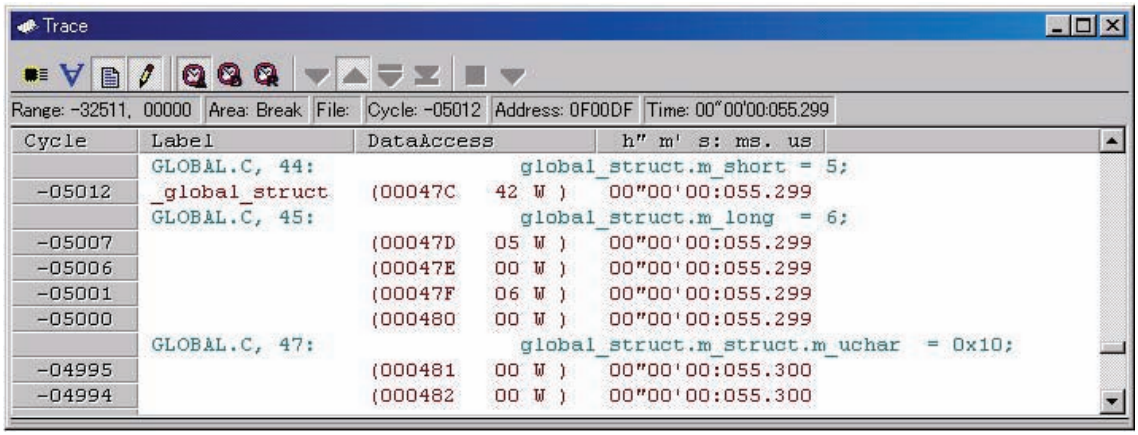
When data access mode is selected while bus mode and disassemble mode are unselected, trace information is displayed in data access mode. Data access mode is configured as shown below.



1. Data access display area:
   Shows data access information. If the information displayed here is "000400 1234 W," for example, it means that data "1234H" was written to the address 000400H in 2-byte width.

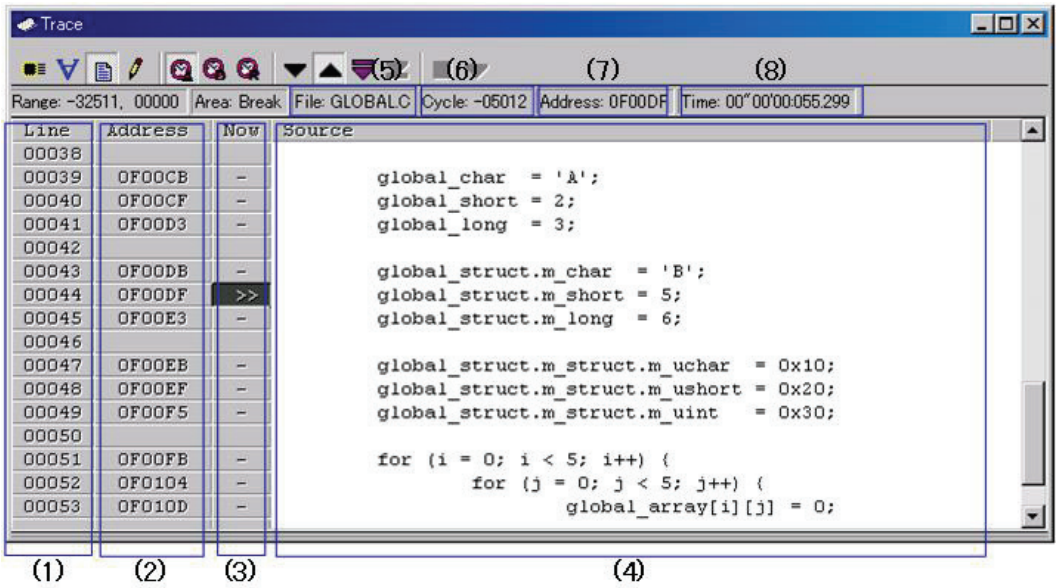
Other display areas are the same as in bus mode.

In addition to data access information, the window can display source line information in combination. In this case, the display will be similar to the one shown below.



6.8.4 Configuration of Source Mode

When only source mode is selected, trace information is displayed in source mode. Source mode is configured as shown below.



1. Line number display area:
   Shows the line number information of the displayed file. Double-click here to bring up a dialog box to change the displayed file.
2. Address display area:
   Shows addresses corresponding to source lines. Double-click here to bring up a dialog box to search for addresses.
3. Referenced cycle display area:
   Shows the currently referenced cycle that is marked by ">>." Furthermore, the addresses corresponding to source lines, if any, are marked by "-."
4. Source display area:
   Shows the content of the source file.
5. File name:
   Shows the file name of the currently displayed source file.
6. Referenced cycle:
   Shows the currently referenced cycle.
7. Referenced address:
   Shows the address corresponding to the currently referenced cycle.
8. Referenced time:
   Shows the time information corresponding to the currently referenced cycle.

Other display areas are the same as in bus mode.

6.8.5 Extended Menus

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

Menu		Function
BUS		Display the information of BUS mode.
DIS		Display the information of Disassemble mode.
SRC		Display the information of Source mode.
DATA		Display the information of Data access mode.
View	Cycle...	Changes the displayed position by specifying a cycle.
	Address...	Changes the displayed position by searching an address.
	Source...	Display a selected source file.
Time	Absolute Time	Shows elapsed time from the time the program started running up to now in terms of absolute time.
	Differences	Shows a differential time from the immediately preceding displayed cycle.
	Relative Time	Shows a relative time from the currently selected cycle.
Trace	Forward	Changes the direction of search to forward direction.
	Backward	Changes the direction of search to reverse direction.
	Step	Searches in Step mode in the specified direction of search.
	Come	Searches in Come mode in the specified direction of search.
	Stop	Stops trace measurement in the middle and displays the measured content at the present point of time.
	Restart	Restarts trace measurement.
Layout...		Change layout of the corrent view.
Copy		Copy selected lines.
Save...		Save trace data to file.
Load...		Load trace data from file.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

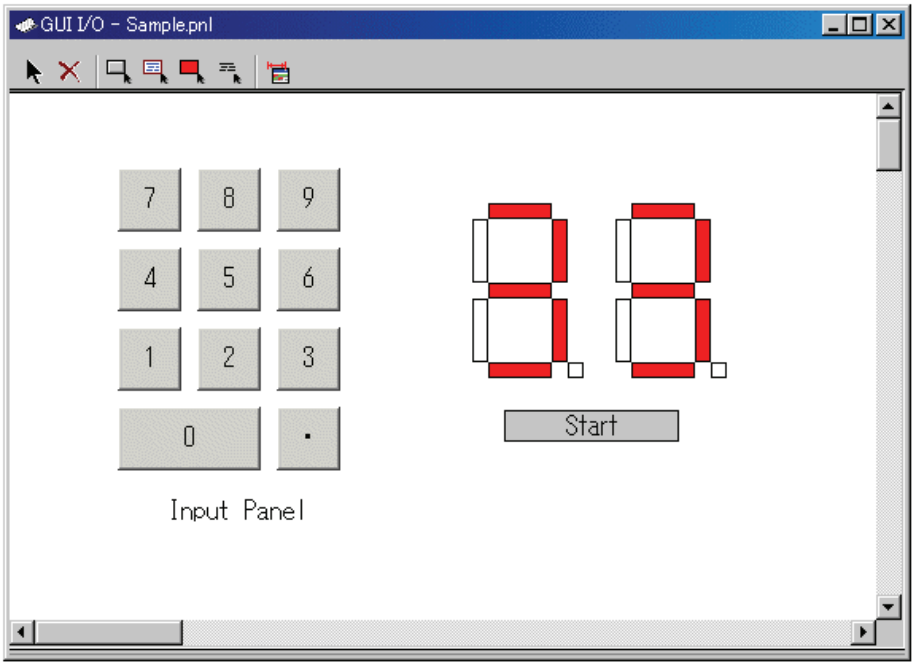
6.8.6 Display of bus information on the 740 Debugger

From left to right, the contents are as follows:

- Address
  The status of the address bus
- Data
  The status of the data bus
- Sync
  This signal is output when fetching an instruction op-code. When an op-code is being fetched, this signal indicates a logic 1.This Sync value is sometimes displayed as '(1)'.In this case, it denotes a dummy Sync meaning that the instruction on the line is not actually executed.
- Read
  This signal determines the direct ion of the data bus. When data is to be read, this signal indicates a logic 0.
- Write
  This signal determines the direct ion of the data bus. When data is to be written, this signal indicates a logic 0.

6.9 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.

6.9.1 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.

7. 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.

7.1 Table of Script Commands (classified by function)

7.1.1 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
OverStepInstruaction	OI	Overstep execution of instructions
Return	RET	Executes a source line return
ReturnInstruction	RETI	Executes an instruction return
Reset	-	Resets the target MCU
Time	-	Sets the run time display and checks the current setting

7.1.2 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
LoadIeee*	LI	Downloads IEEE-695 absolute-format files
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

7.1.3 Register Operation Commands

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

7.1.4 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)

7.1.5 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

7.1.6 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

7.1.7 Hardware Break Setting Commands

Command Name	Short Name	Contents
HardwareBreak	HB	Sets and checks a hardware break
BreakMode	BM	Sets and checks hardware break mode

7.1.8 Real-time Trace Commands

Command Name	Short Name	Contents
TracePoint	TP	Sets and checks a trace points
TraceData*	TD	Realtime trace data display
TraceList*	TL	Displays disassembled realtime trace data

7.1.9 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

7.1.10 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

7.1.11 Map Commands

Command Name	Short Name	Contents
Map*	-	Checks and sets mapping data

7.1.12 Clock Command

Command Name	Short Name	Contents
Clock	CLK	Checks and changes the clock

7.1.13 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

7.1.14 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

7.2 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
BreakMode	BM	Sets and checks hardware break mode
CD	-	Specifies and checks the current directory
Clock	CLK	Checks and changes the clock
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
HardwareBreak	HB	Sets and checks a hardware break
Label	-	Checks assembler labels
Load	L	Downloads the target program
LoadHex	LH	Downloads an Intel HEX-format file
LoadIeee*	LI	Downloads IEEE-695 absolute-format files
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
Map*	-	Checks and sets mapping data
Module	MOD	Displays modules names
Move	-	Moves memory blocks
MoveWord*	MOVEW	Moves memory blocks(in 2-byte units)
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
Time	-	Sets the run time display and checks the current setting
TraceData*	TD	Realtime trace data display
TraceList*	TL	Displays disassembled realtime trace data
TracePoint	TP	Sets and checks a trace points
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

8. 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.

8.1 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.

8.1.1 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).

8.1.2 Assign Statement

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

```
%macro-variable = expression
```

- 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.

8.1.3 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.

```
if ( expression )
    statement 1
 else
    statement 2
 endi
```

- 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).

8.1.4 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.

```
while ( expression )
    statement
 endw
```

- 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.

8.1.5 Comment statements

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

```
        ;character string
```

- 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.

8.2 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

8.2.1 Constants

You can use binary, octal, decimal, or hexadecimals. 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

8.2.2 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 740	• You cannot use the register name.(A,X,Y,S,PC,PS,P) • You cannot include the assembler structured instructions, pseudo instructions, macro instructions, operation code, or reserved words (.SECTION, .BYTE, switch, if, etc.). • You cannot use strings that start with two periods (..) for symbols or labels. .D0 to .D65535, .F0 to .F65535, .I0 to .I56635, .S0 to .S65535, ..0 to ..65535, ??0 to ??65535

8.2.2.1 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.

8.2.2.2 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 symbols outside scope
6. Local labels outside scope

- Conversion of bit values

1. Local bit symbols
2. Global bit symbols
3. Local bit symbols outside scope

8.2.3 Macro Variables

Macro variables are defined by assign statements in the script file. See Section "8.1.2 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 star t 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.

8.2.4 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 740	PC, A, X, Y, S, PS

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

8.2.5 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.

8.2.6 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.

8.2.7 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.

8.2.8 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

9. C/C++ Expressions

9.1 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.", ...

9.1.1 Immediate Values

You can use hexadecimals, decimals, octals as immediate values. Values starting with 0x are processed as hexadecimals, 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.

9.1.2 Scope Resolution

The scope resolution operator :: is available as following.
Global scope: ::valiable name

```
::x, ::val
```

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

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

9.1.3 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.

9.1.4 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.

9.1.5 Reference

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

9.1.6 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.

9.1.7 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.

9.1.8 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.

9.1.9 Pointers to Members

Pointers to members using the ".*" or "->*" operator can be refered 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 considered as the variable of pointer-to-member type.

9.1.10 Parentheses

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

9.1.11 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.

9.1.12 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.

9.1.13 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.

9.1.14 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.

9.1.15 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.

9.1.16 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).

9.1.17 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.

9.2 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.

9.2.1 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)
  ```

9.2.2 Basic Types

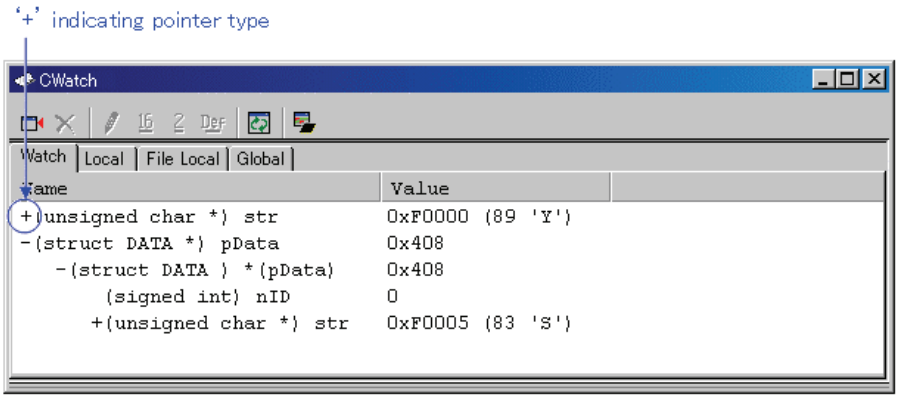
- When the result of calculation is a basic type other than a char type or floating point type, it is displayed as follows:
  ```
  (unsigned int) i = 65280(when Radix is in initial state)
  (unsigned int) i = 0xFF00(when Radix is hex)
  (unsigned int) i = 1111111100000000B(when Radix is binary)
  ```
- 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)
  ```

9.2.3 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)

l 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 '-'.

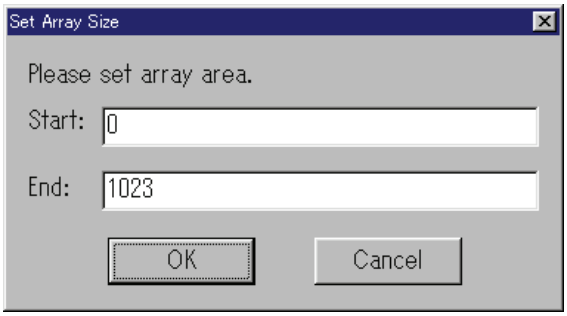
9.2.4 Array Types

- 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 "9.2.3 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.

9.2.5 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)
  ```

9.2.6 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)
  ```

9.2.7 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)
  ```

9.2.8 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)
```

9.2.9 Syntax Errors

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

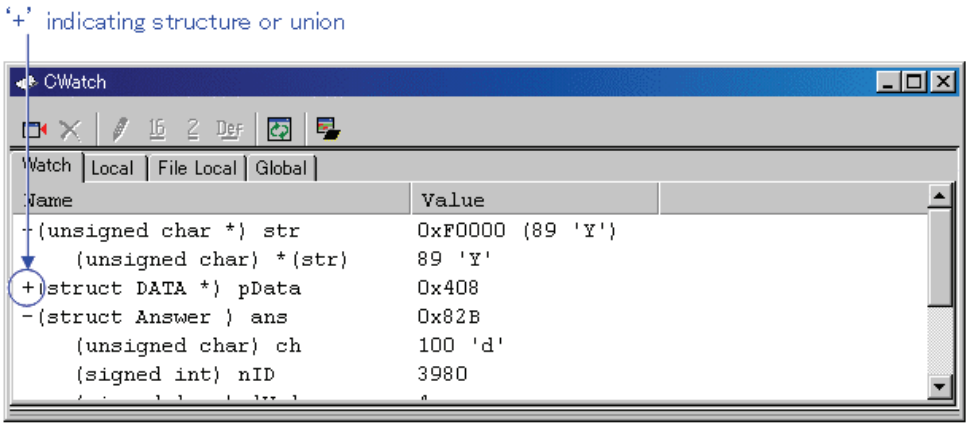
```
() str*(p = <syntax error>(all Radices)
(where str*(p is the syntax error)
```

9.2.10 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
```

10. Display the Cause of the Program Stoppage

If the program is stoped 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 "---".

11. Attention

11.1 Common Attention

11.1.1 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.

11.1.2 Area where software breakpoint can be set

The area which can be set for software breakpoint varies depending on the type of MCU.
Only the ROM area memory-mapped in the area set as Internal can be designated for software breakpoint.
You cannot set software breakpoint in ROM areas memory-mapped in the SFR area, RAM area or other areas set as External.

11.1.3 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.

11.1.4 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.

11.1.5 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.

11.1.6 Syncronized debugging

Syncronized debugging function is not available.

11.1.7 Compact Emulator reset switch

If system reset of the compact emulator does not function normally, terminate the debugger, turn ON the compact emulator again, and restart the debugger.
Then re-download the program.

11.2 Attention of the 740 Debugger

11.2.1 Setting of Memory Map

The map attributes immediately after the emulator has started up are External for 0h-3FFFh and Internal for 4000h-FFFFh. The memory map information must be altered to suit the target microcomputer's memory space. It is similar in the case of the microcomputer that inside ROM area begins from the 4000h past.

See Section "4.3 Setup the Debugger for 740 " in the Reference part for details.

11.2.2 Stack area used by the emulator

The emulator uses the user stack area as a work area (3 bytes).
Before starting debugging, be sure to reserve the user stack area + 3 byte area.

11.2.3 Watch dog timer

When the watch dog timer is enabled, operations other than free-run of the target program are inhibited. Before starting debugging, disable the watch dog timer.

11.2.4 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 "11.3 Option of C Compiler/Assembler/Linker"

The compiler that can be used by 740 debugger:

- the Assembler Package for 740 Family　SRA74
- the IAR C Compiler

11.2.5 Debugging in the 16-Timer functions

The microcomputer (38B5 group etc.) which supports 16 bit timer has undermentioned limitations.

- [Precaution 1]
  Outputs for the 16-bit timer may be invalid in the Dump Window etc. when the program execution is suspended by breaking or single-stepping during writing to the higher/lower order byte of the 16-bit timer.
  For example:

```
[TIMER_LOW] = [DATA1]
[TIMER_HIGH] = [DATA2] <- when a break occurs here
```

Note) The debugger suspends program execution immediately before the address to which a software breakpoint is set.

- [Precaution 2]
  Writing a value (with the Dump Window) into the 16-bit timer will be fail when the program execution is suspended by breaking or single-stepping during writing to the higher/lower order byte of the 16-bit timer.
  For example:

```
[DATA1] = [TIMER_LOW]
[DATA2] = [TIMER_HIGH] <- when a break occurs here
```

These problems caused by the microcomputer's specifications. In a microcomputer with the 16-bit timer, the timer is written in the order of the lower byte and the higher byte. And reading from the timer in reverse order.
Therefore, if a break occurs as shown in the Precaution 1 or 2, an incorrect value will be read or written by displaying or setting a value of the 16-bit timer in the Dump Window.

11.2.6 About the single-step execution and the program break function in the internal RAM area of the mcu

When debugging with the emulation pod M38000L2-FPD, single-step execution and program break function in the internal RAM area are not available.
When debugging the program transferred to the internal RAM area, please use the free-run execution and the trace function.

11.2.7 Hardware Event

If you specify any other bit of the address that contains a specified bit is accessed during bit access, the event may become effective in the following data accesses.
Hardware Break Event

11.3 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.

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.

11.3.1 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 740	-ylmba

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.

11.3.2 When Using the IAR C Compiler (ICC)

11.3.2.1 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 740	-ylmba

- 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.

11.3.2.2 Command Execution Examples

The following shows examples of entering commands depending on the product

- The debugger for 740

```
>ICC740 -r file1.c<Enter>
>ICC740 -r file2.c<Enter>
>XLINK -o filename.695 -f lnk7400t.xcl file1 file2<Enter>
```

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

11.3.3 When Using the Assembler Package for 740 Family

Please assemble according to the following procedures and link.

- At assemble
  - "-c" option
    outputs debugging information concerned with source line to a relocatable file.
    Note
    When the directive comand .FUNC is specified to a function in a source file, if "-c" option is used, the name of the function will be not available. Please do not use the option to make the name available.

  - "-s" option
    outputs local labels, local .equ symbols and local .bequ symbols to a relocatable file.

- At link
  - "-s" option
    generates a symbol file.

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.

11.3.3.1 Command Execution Examples

The following shows examples of entering commands depending on the product

- The Debugger for 740

```
>sra74 -c -s main.a74<Enter>
>sra74 -c -s sub.a74<Enter>
>link74 main sub ,,,-s<Enter>
```

[MEMO]

740 Compact Emulator 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.

740 Compact Emulator Debugger V.1.02
User's Manual

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

REJ10J1385-0100