User's Manual

M16C R8C Compact Emulator Debugger V.1.03

User's Manual

Renesas Microcomputer Development Environment System

Renesas Electronics
www.renesas.com

Rev.1.00 2007.07

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.

- M32C/80, M16C/80 Series

  Note: In this help, the information which depends on this CPU is described as "for M32C".
- M16C/Tiny, R8C/Tiny Series

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

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

(Blank Page)

Setup of Debugger

(Blank Page)

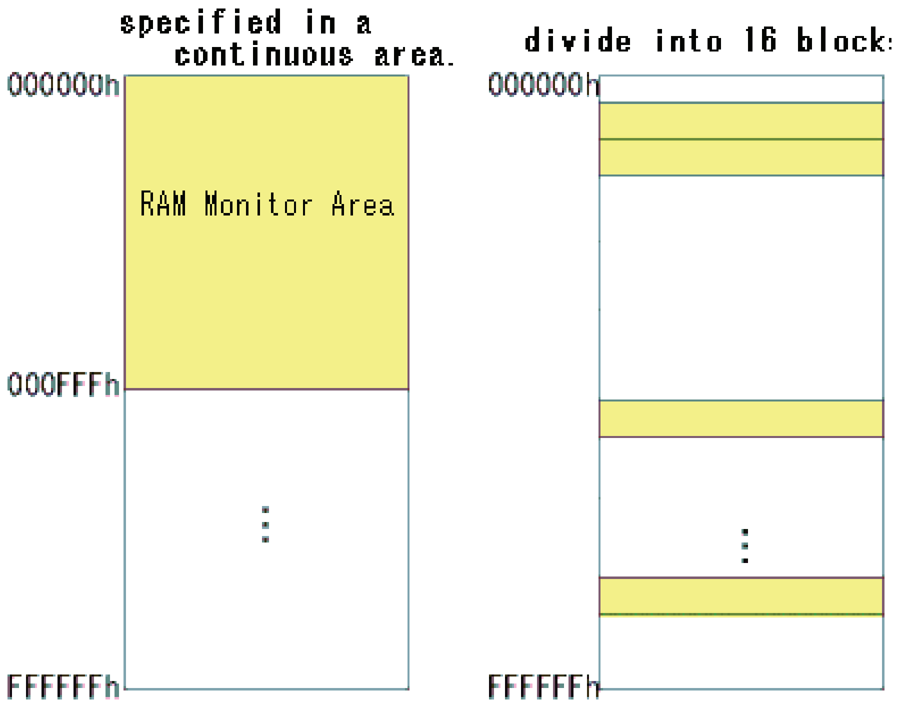
1. Features

1.1 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 has 1 Kbytes of RAM monitor area which can be located in any contiguous address location or in 4 separate blocks comprised of 256 bytes each.

1.1.1 RAM Monitor Area

This debugger has 1 Kbytes of RAM monitor area which can be located in any contiguous address location or in 4 separate blocks comprised of 256 bytes each.



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:
"12.1.2 Area where software breakpoint can be set"

1.2.2 Hardware Breaks Function

This function causes the target program to stop upon detecting a data read/write to memory, instruction execution, or the rising/falling edge of the input signal fed from an external trace cable.
The contents of events that can be set vary with each target MCU.
The following designations are available as break events:

- Address designation
  - Instruction fetch
  - Memory access
  - Bit access
- External trigger designation
- Interruption

The number of events that can be specified are two events of all. For the address designation method, instruction fetch and memory access allow the range designation and logical condition designation, in addition to the normal one-address designation. Moreover, instruction fetch allows you to specify the function name.
Memory access allows you to specify the comparison data to read/write data related to the specified address in the same manner as when setting the H/W breakpoint. It also allows mask designation to the comparison data.

These break events can be combined as below:

- Trace when all of the valid events are established (AND condition)
- Trace when all of the valid events are established at the same time (simultaneous AND condition)
- Trace when one of the valid events is established (OR condition)

1.2.3 Address Interrupt Breaks Function

This function stops the target program immediately before executing an instruction at a specified address. This function is realized by using the MCU's address match interrupt.
The address interrupt break function can only be used when the address match interrupt is not used in the user application. The count of breakpoints depends on the connected MCU.

Note

The address interrupt break function can only be used when the Enable the Address Match Interrupt Break Function check box on the Init dialog box MCU tab is selected. (Details).

1.2.3.1 Setting and Deleting a Break Points

The address interrupt beakpoint can be set by the following windows.

- Editor (Source) Window
- Address Interrupt Break Point Setting Window

You can double-click the mouse to set/reset the address interrupt breakpoint in the Editor (Source) window (same as software breakpoints).
You can also switch to temporarily disable/enable the address interrupt breakpoint in the Address Interrupt Break Point Setting Window.

1.3 Real-Time Trace Function

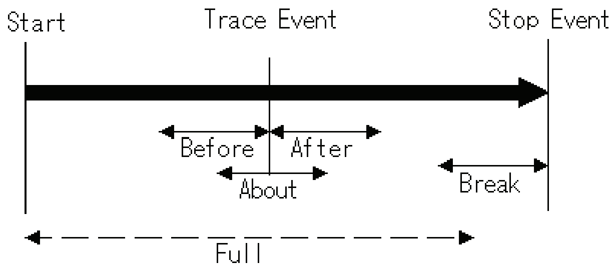
The real-time trace function records the execution history of the target program.
Up to 64K cycles of execution history can be recorded. This record allows inspecting the bus information, executed instructions, and source program execution path for each cycle.
The execution history is referred to in 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 64K cycles execution history can be referred to with this debugger.
The trace area of the following 5 mode is being supported.

- Break
  64K cycles before target program stops
- Before
  64K cycles before trace point
- About
  32K cycles either side of trace point
- After
  64K cycles after trace point
- Full
  Until 64K cycles are written in the trace memory



"Break" is set by default. To refer the execution history before stopping the target program, use "Break" (designation of trace event is not required).
To refer the execution history at any position, or to continue execution of the target program, specify the trace event and change the trance range.

1.3.2 Trace Condition Setting

The following designations are available as trace events:

- Address designation
  - Instruction fetch
  - Memory access
  - Bit access

The number of events that can be specified are two events of all. These break events can be combined as below:

- Trace when all of the valid events are established (AND condition)
- Trace when all of the valid events are established at the same time (And(same time) condition)
- Trace when one of the valid events is established (OR condition)

1.3.3 Trace Data Write Condition

Trace data write conditions can be specified.
You can specify the following write conditions:

- Write conditions unlimited (default)
- Cycles from the start event established to the end event established
- Only cycles where the start event is established
- Cycles from the start event established to the start event unestablished
- Other than cycles from the start event established to the end event established
- Other than cycles where the start event is established
- Other than cycles from the start event established to the start event unestablished

1.4 Real-Time OS Debugging Function

This function debugs the realtime OS-dependent parts of the target program that uses the realtime OS.
This function helps to show the status of the realtime OS.

1.5 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. About the Compact Emulator

The compact emulator is a small emulator equipped with the debugging function needed for full-scale development, such as real-time trace and a hardware break, though it is a handy price and a small body.

2.1 Communication method

The supported communication methods are as follows.

I/F	Emulator
	Compact Emulator
USB	Support

2.2 Function table

The supported functions are as follows.

Function	Compact Emulator
SW Break	64 points
HW Break	2 points
Address Interrupt Break	4 points*
Real-Time Trace	64K Cycles
RAM Monitor	1K bytes (256bytes x 4blocks) area
Time Measurement	Go to Stop

*Depends on the target MCU used.

3. Before starting the debugger

3.1 Communication method by emulator

The supported communication methods are as follows.

- USB

3.1.1 USB Interface

- 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.

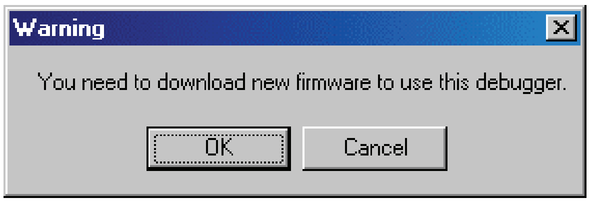
3.2 Download of Firmware

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

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

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.



You need to download new firmware to use this debugger.

3.3 Setting before emulator starts

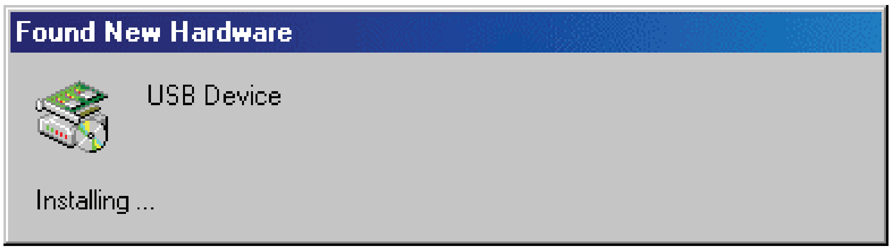
3.3.1 USB communication

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".

3.3.1.1 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.



Found New Hardware

USB Device

Installing ...

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

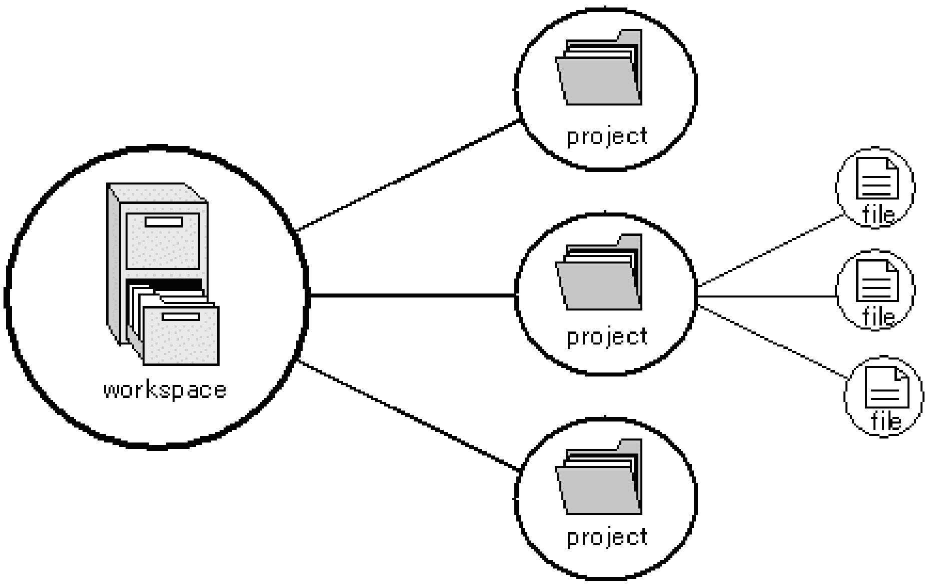
- Before the USB device drivers can be installed, the debugger you use must already be installed. Install this debugger first.
- A user who install the USB device driver need administrator rights.
- During installation, a message may be output indicating that the device driver proper musbdrv.sys cannot be found. In this case, specify the musbdrv.sys which is stored in the same directory as is the musbdrv.inf file.

4. 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.

4.1 Workspaces, Projects, and Files

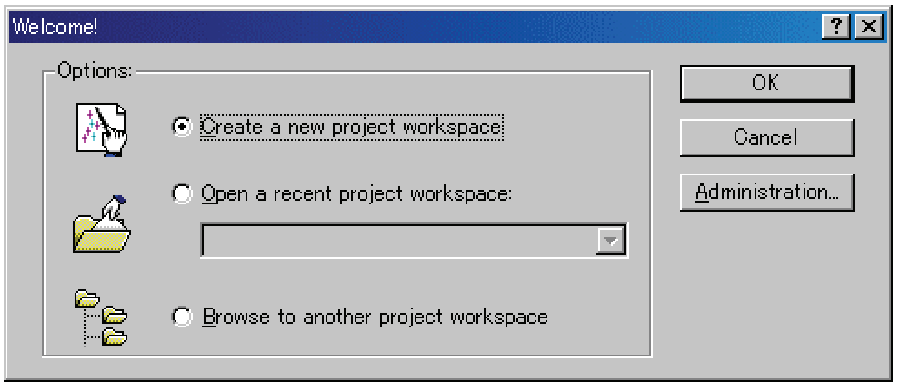
Just as a word processor allows you to create and modify documents, this product allows you to create and modify workspaces.
A workspace can be thought of as a container of projects and, similarly, a project can be thought of as a container of project files. Thus, each workspace contains one or more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

4.2 Starting the High-performance Embedded Workshop

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


Welcome!
Options:
Create a new project workspace
Open a recent project workspace:
Browse to another project workspace
OK
Cancel
Administration...

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).

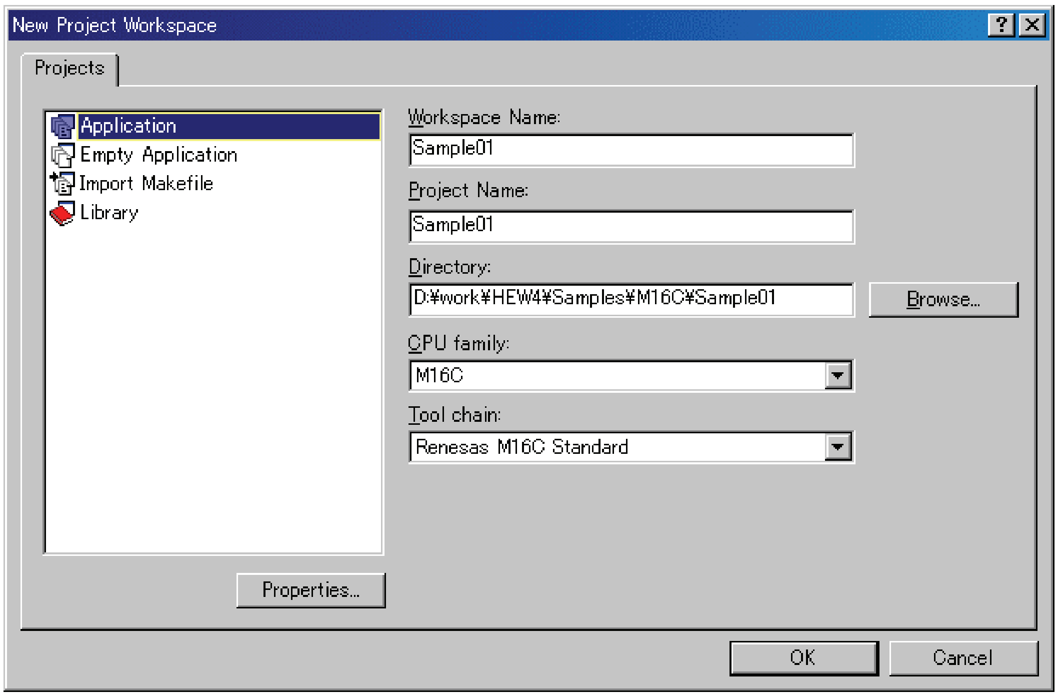
Please refer to the following about the method to create a new workspace.
Refer to "4.2.1 Creating a New Workspace (Toolchain Used)"
Refer to "4.2.2 Creating a New Workspace (Toolchain Not Used)"
* When debugging the existing load module file with this product, a workspace is created by this method.

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.

4.2.1 Creating a New Workspace (Toolchain Used)

4.2.1.1 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.



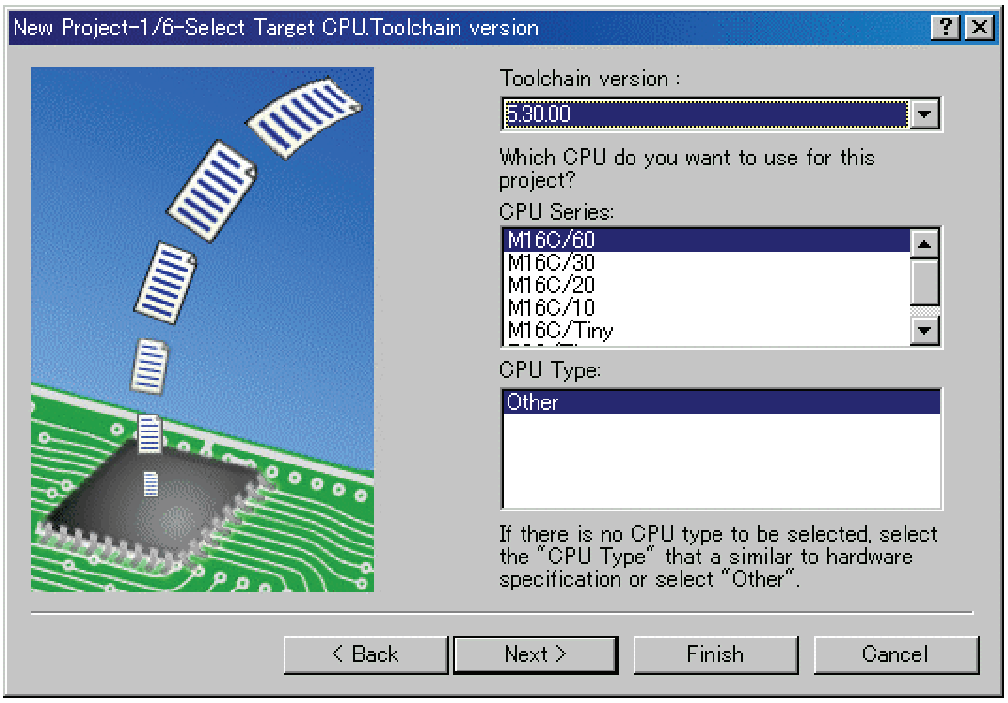
Workspace Name: Sample01
Project Name: Sample01
Directory: D:¥work¥HEW4¥Samples¥M16C¥Sample01
CPU family: M16C
Tool chain: Renesas M16C Standard

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.

4.2.1.2 Step2 : Setting for the Toolchain

A wizard for the project creation starts.



Toolchain version : 5.30.00

Which CPU do you want to use for this project?

CPU Series:
M16C/60
M16C/30
M16C/20
M16C/10
M16C/Tiny

CPU Type:
Other

If there is no CPU type to be selected, select the "CPU Type" that a similar to hardware specification or select "Other".

Here, the following contents are set.

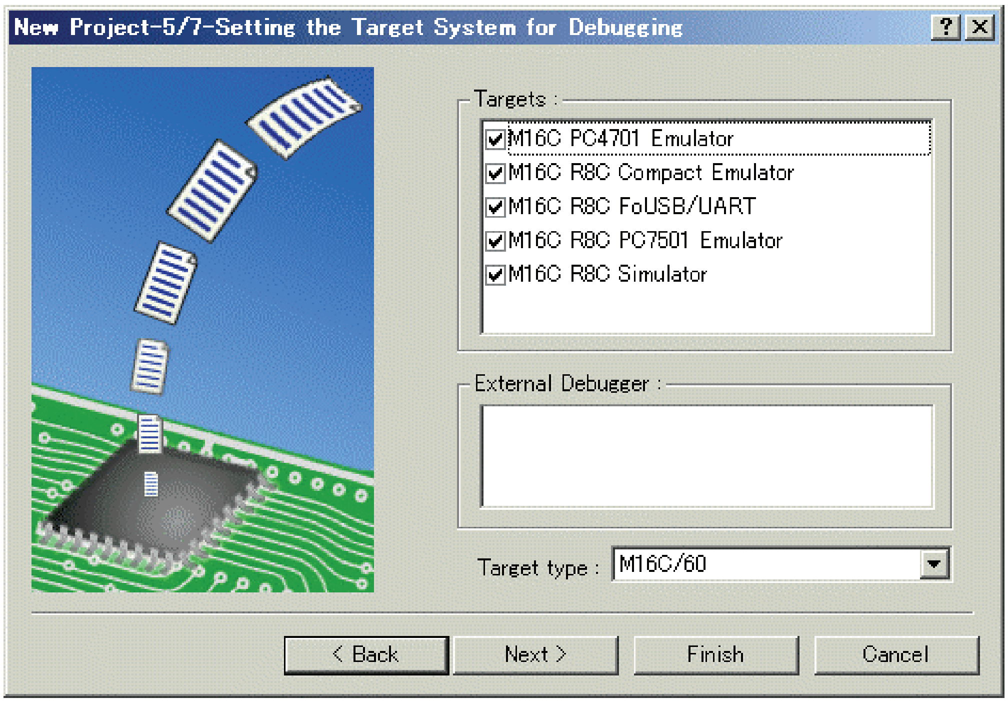
- toolchain
- the setting for the real-time OS (when using)
- the setting for the startup file, heap area, stack area, and so on

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.

4.2.1.3 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.



New Project-5/7-Setting the Target System for Debugging

Targets :
- ☑ M16C PC4701 Emulator
- ☑ M16C R8C Compact Emulator
- ☑ M16C R8C FoUSB/UART
- ☑ M16C R8C PC7501 Emulator
- ☑ M16C R8C Simulator

External Debugger :

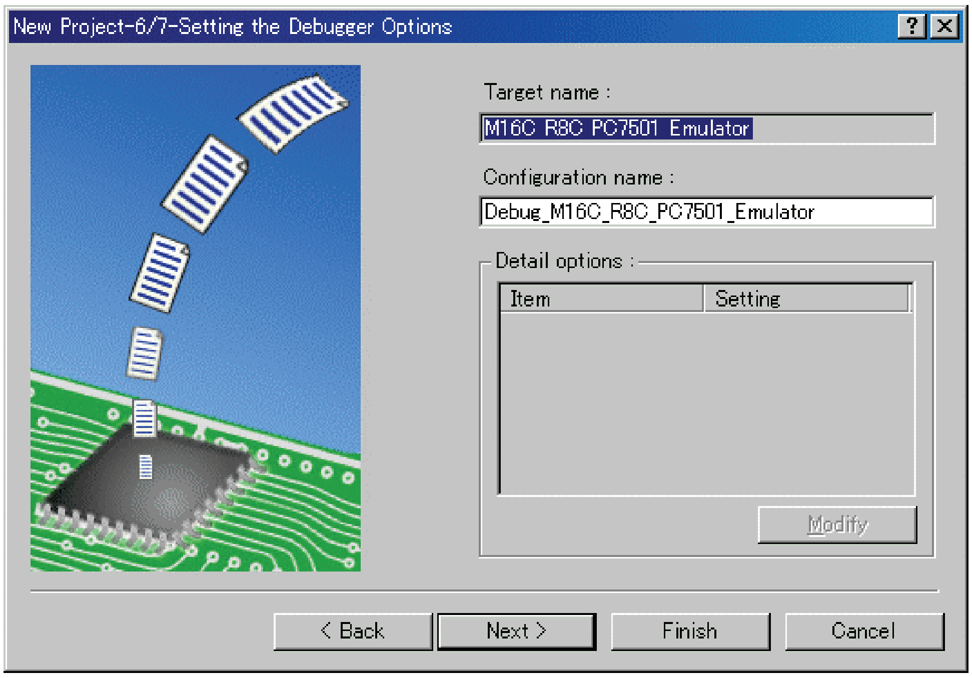
Target type : M16C/60

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.

4.2.1.4 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).


New Project-6/7-Setting the Debugger Options

Target name : M16C R8C PC7501 Emulator

Configuration name : Debug_M16C_R8C_PC7501_Emulator

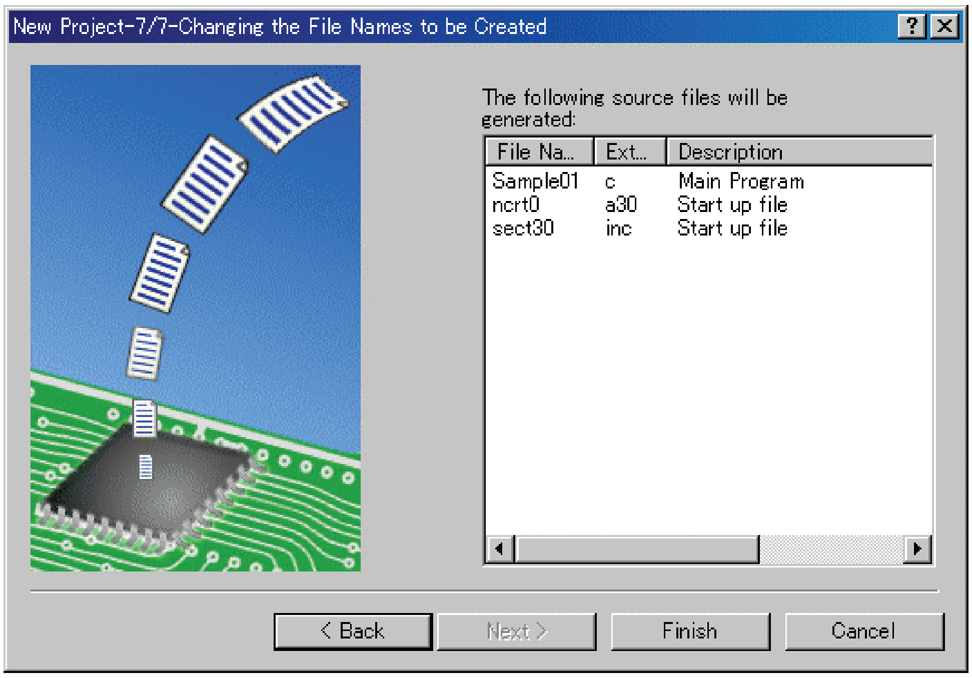
Detail options :

Item	Setting

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

4.2.1.5 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.


New Project-7/7-Changing the File Names to be Created

The following source files will be generated:

File Na...	Ext...	Description
Sample01	c	Main Program
ncrt0	a30	Start up file
sect30	inc	Start up file

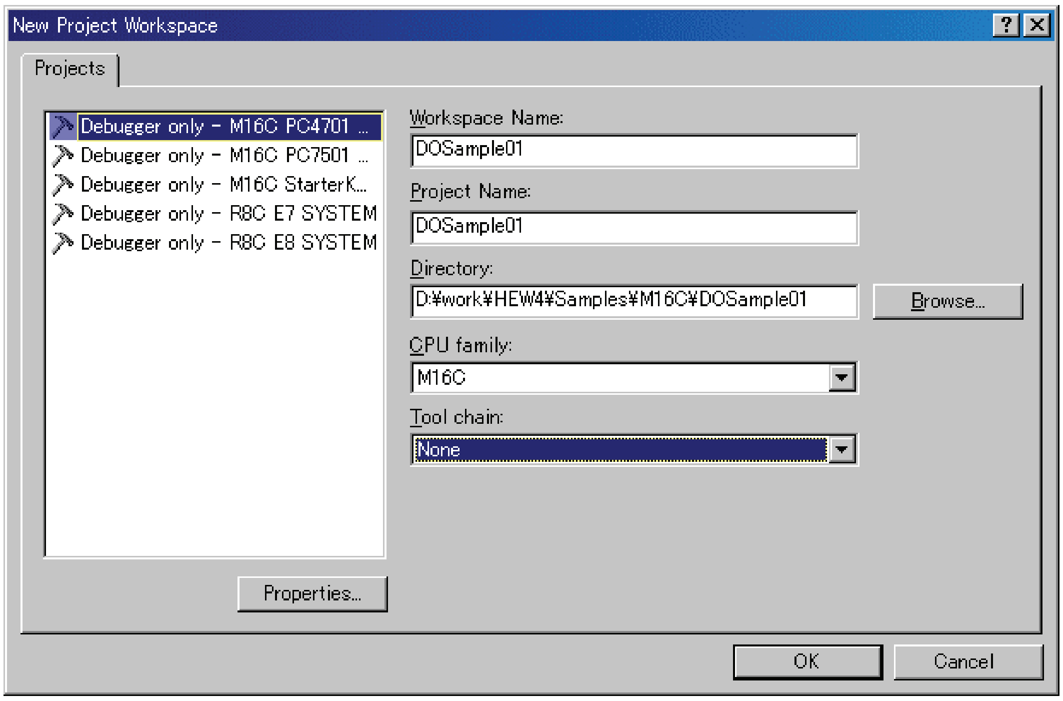
This is the end of the emulator settings.
Exit the Project Generator following the instructions on the screen.

4.2.2 Creating a New Workspace (Toolchain Not Used)

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

4.2.2.1 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.



Debugger only - M16C PC4701 ...
Debugger only - M16C PC7501 ...
Debugger only - M16C StarterK...
Debugger only - R8C E7 SYSTEM
Debugger only - R8C E8 SYSTEM

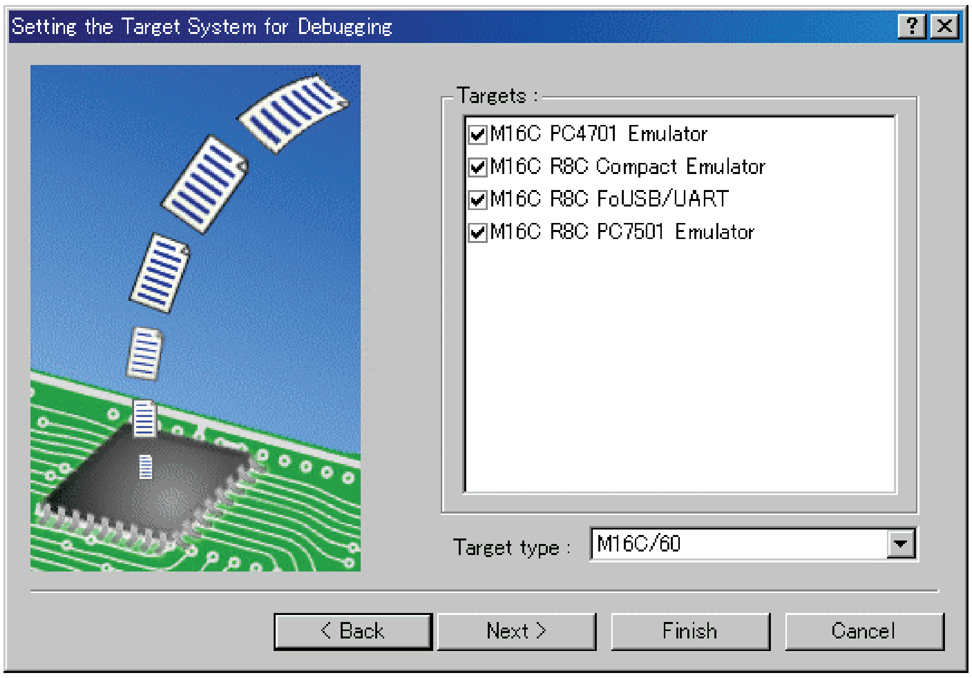
Workspace Name: DOSample01
Project Name: DOSample01
Directory: D:¥work¥HEW4¥Samples¥M16C¥DOSample01
CPU family: M16C
Tool chain: None

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.

4.2.2.2 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.


Setting the Target System for Debugging

Targets :
- [✓] M16C PC4701 Emulator
- [✓] M16C R8C Compact Emulator
- [✓] M16C R8C FoUSB/UART
- [✓] M16C R8C PC7501 Emulator

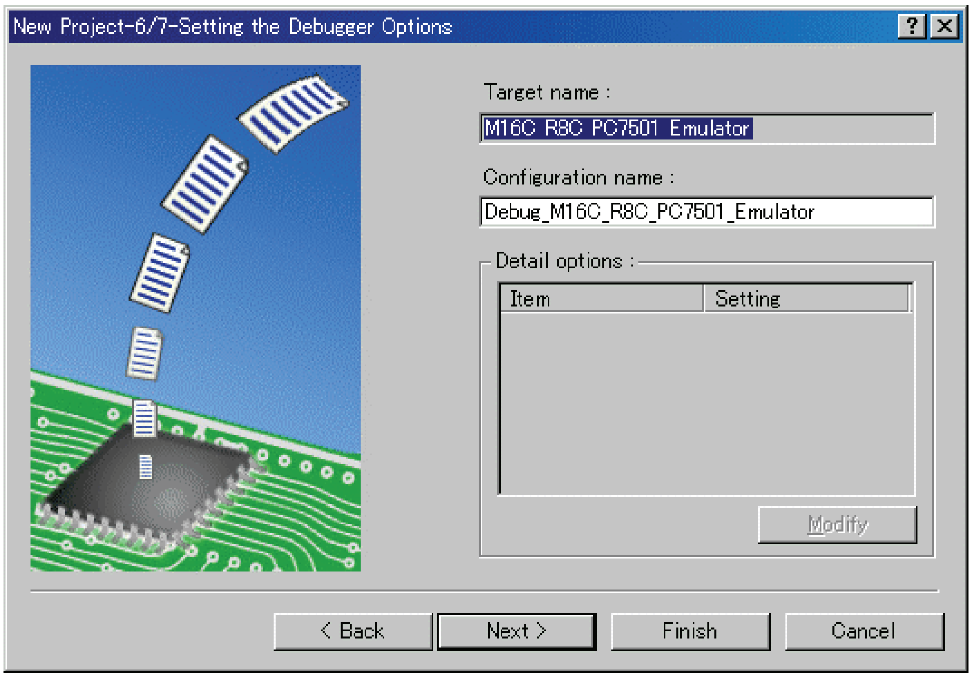
Target type : M16C/60

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.

4.2.2.3 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).


New Project-6/7-Setting the Debugger Options

Target name : M16C R8C PC7501 Emulator

Configuration name : Debug_M16C_R8C_PC7501_Emulator

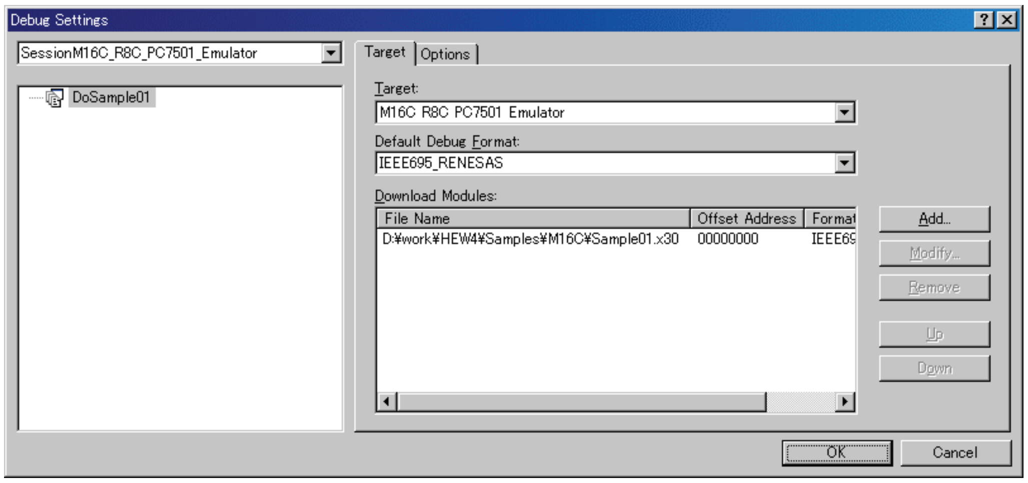
Detail options :

Item	Setting

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.

4.2.2.4 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.



Target: M16C R8C PC7501 Emulator

Default Debug Format: IEEE695_RENESAS

Download Modules:

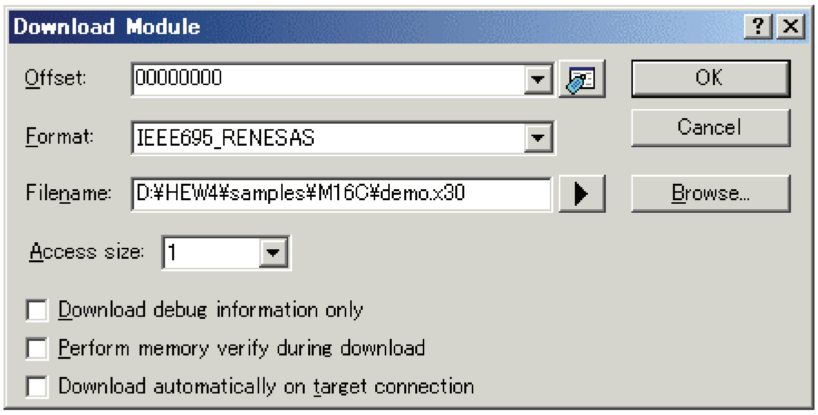
File Name	Offset Address	Format
D:¥work¥HEW4¥Samples¥M16C¥Sample01.x30	00000000	IEEE69

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

Format Name	Contents
IEEE695_RENESAS	IEEE-695 format file (When Using Renesas toolchain)
IEEE695_IAR	IEEE-695 format file (When Using IAR toolchain)
IEEE695_TASKING	IEEE-695 format file (When Using Tasking toolchain)
ELF/DWARF2	ELF/DWARF2 format file (When Using Renesas toolchain)
ELF/DWARF2_IAR	ELF/DWARF2 format file (When Using IAR toolchain)
ELF/DWARF2_TASKING	ELF/DWARF2 format file (When Using Tasking toolchain)
ELF/DWARF2_KPIT	ELF/DWARF2 format file (When Using KPIT toolchain)

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.



Offset: 00000000

Format: IEEE695_RENESAS

Filename: D:¥HEW4¥samples¥M16C¥demo.x30

Access size: 1

☐ Download debug information only

☐ Perform memory verify during download

☐ Download automatically on target connection

- 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

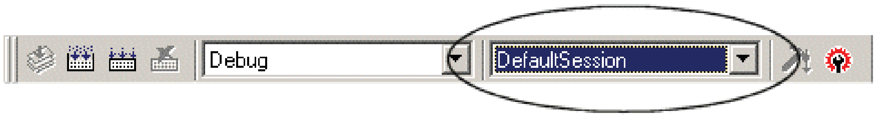
"Offset", "Access size" and "Perform memory verify during download" is ignored. The offset is always set to 0, the access size is always set to 1 and the verification does not work.

4.3 Starting the Debugger

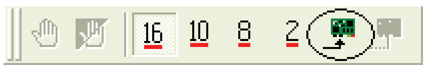
The debugging can be started by connecting with an emulator.

4.3.1 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.



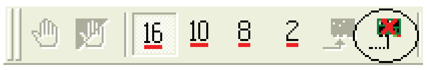
After the session name is selected, the dialog box for setting the debugger is displayed and the emulator will be connected.
When the dialog box is not displayed, select [Connect] from the [Debug] menu.



4.3.2 Ending the Emulator

The emulator can be exited by using the following methods:

1. Selecting the “Disconnect”
   Select [Disconnect] from the [Debug] menu.



2. Selecting the "DefaultSession"
   Select the "DefaultSession" in the list box that was used at the time of emulator connection.

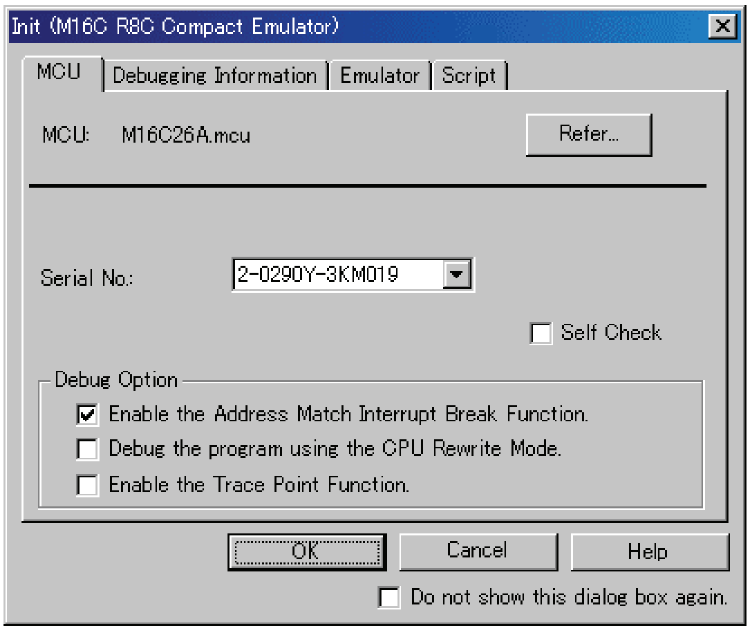
3. Exiting the High-performance Embedded Workshop
   Select [Exit] from the [File] menu. High-performance Embedded Workshop will be ended.

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.

5. Setup the Debugger

5.1 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.


Init (M16C R8C Compact Emulator)

MCU: M16C26A.mcu

Serial No.: 2-0290Y-3KM019

☐ Self Check

Debug Option
☑ Enable the Address Match Interrupt Break Function.
☐ Debug the program using the CPU Rewrite Mode.
☐ Enable the Trace Point Function.

☐ Do not show this dialog box again.

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

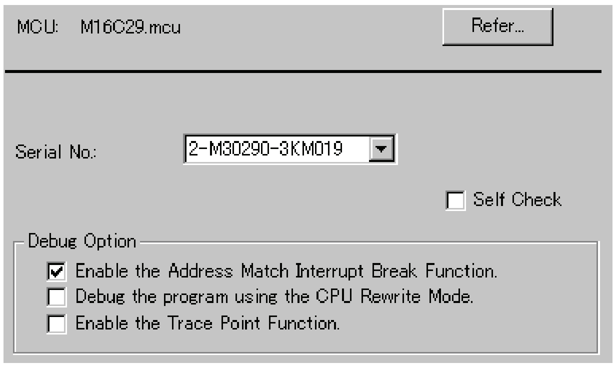
Tab Name	Product Name	
	The debugger for M32C	The debugger for M16C/R8C
MCU	exist	exist
Debugging Information	exist	exist
Emulator	exist	exist
Script	exist	exist

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

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

5.1.1 MCU Tab

The specified content becomes effective when the next being start.



MCU: M16C29.mcu

Serial No.: 2-M30290-3KM019

☐ Self Check

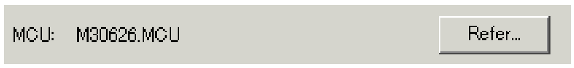
Debug Option

☑ Enable the Address Match Interrupt Break Function.

☐ Debug the program using the CPU Rewrite Mode.

☐ Enable the Trace Point Function.

5.1.1.1 Specifying the MCU file



MCU: M30626.MCU

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.

5.1.1.2 Setting of the Communication Interface

The displayed data varies depending on the specified communication interface.
The available communication interface varies depending on the products.
The following shows the setting for each communication interface.
Refer to "5.2.1 Setting of the USB Interface"

5.1.1.3 Executing Self-Check

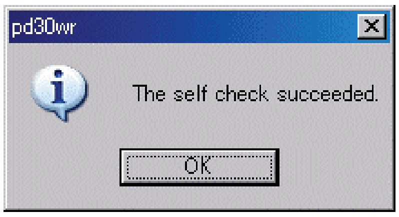
Specify this option to execute self-check* on the emulator when the debugger starts up.

☐ Self Check

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.

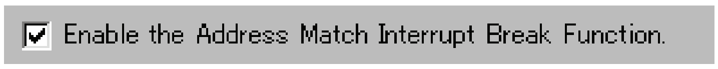

pd30wr
The self check succeeded.
OK

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.

5.1.1.4 Using/unusing the address interrupt break function

Specify whether or not to use the address interrupt break function.

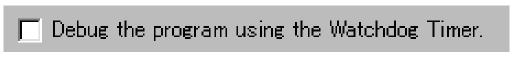

[✓] Enable the Address Match Interrupt Break Function.

- To use the address interrupt break function (default)
  Select the check box shown above.
  In this case, the address interrupt break function is used by the emulator, and cannot be used in the user program.
- Not to use the address interrupt break function
  Deselect the check box shown above.
  In this case, the address interrupt break function can be used in the user program.

The contents set here are reflected at only startup time.

5.1.1.5 Using/unusing the watchdog timer

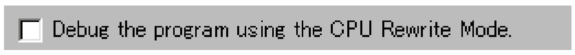
Specify whether or not to use the watchdog timer. (By default, the watchdog timer is unused.)
This specification exist for the M32C debugger only.


[ ] Debug the program using the Watchdog Timer.

When debugging the target system that uses a watchdog timer, select the check box shown above.

5.1.1.6 Choosing to use or not to use CPU rewrite mode

Specify whether or not you want to use CPU rewrite mode. (By default, CPU rewrite mode is unused.)


☐ Debug the program using the CPU Rewrite Mode.

Select the above check box when you are debugging the target system that uses CPU rewrite mode. This specification can only be set or changed when you start the debugger.

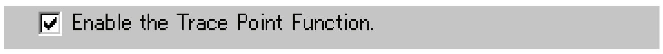
Supplementary explanation

When debugging in CPU rewrite mode is enabled, the following limitations apply:

- Address match breakpoints cannot be set.
- No software breaks can be set in the internal ROM area.
- The command Come cannot be executed in the internal ROM area.

5.1.1.7 Choosing to use or not to use the trace point setting function

Specify whether or not you want to use the trace point setting function. (By default, the trace point function is unused.)


☑ Enable the Trace Point Function.

Select the above check box when you use the event of Compact emulator as a trace point.

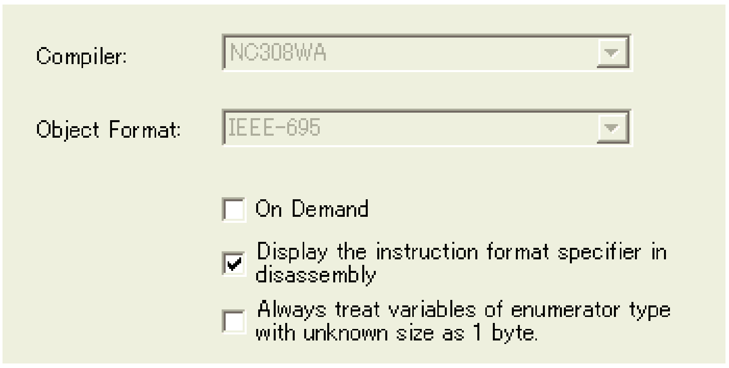
Supplementary explanation

When the trace point setting function is enabled, the following limitations apply:

- Hardware break function cannot use.

5.1.2 Debugging Information Tab

The specified content becomes effective when the next being start.



Compiler: NC308WA

Object Format: IEEE-695

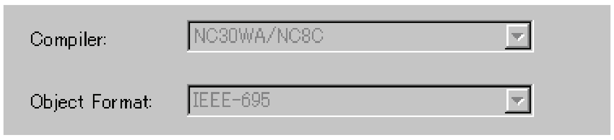
☐ On Demand

☑ Display the instruction format specifier in disassembly

☐ Always treat variables of enumerator type with unknown size as 1 byte.

5.1.2.1 display the compiler used and its object format

Display the compiler used and its object file format.



Compiler: NC30WA/NC8C

Object Format: IEEE-695

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

5.1.2.2 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.
The specified content becomes effective when the next being download.

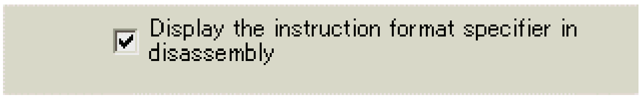
- On-memory method
  Debugging information is stored in the internal memory of your computer.
  Usually, select this method.
- On-demand method
  Debugging information is stored in a reusable temporary file on the hard disk of your computer.
  Because the stored debugging information is reused, the next time you download the same load module it can be downloaded faster.
  This method is suitable when it takes so long time to download the debugging information, because the PC has less memory against the load module file size.

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.

5.1.2.3 Specify whether to display the instruction format specifier

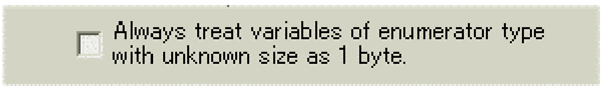
Specify whether to display the instruction format specifier in the disassembled display.


☑ Display the instruction format specifier in disassembly

Select the above check box when you display the instruction format specifier.
This specification can only be set or changed when you start the debugger.

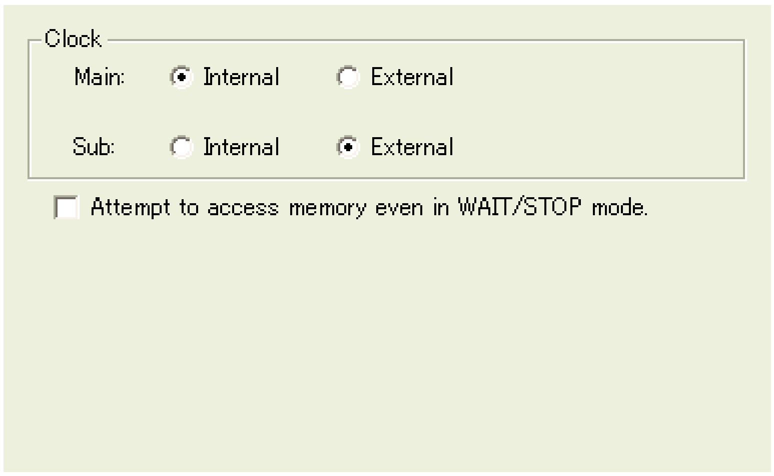
5.1.2.4 To treat size of enumeration type as 1 byte

You can specify whether your debugger treat all sizes of enumeration types whose size is unknown in the debugging information as 1 byte. For reducing memory consumption, NC30 and NC308 have an option to treat the sizes of enumerator types as 1 byte and not as same size of 'int'. Note that NC30 and NC308 don't output the sizes of enumerator types in debugging information and debuggers consider the size as same size of 'int'.
Therefore you may not correctly refer the values of enumeration types in the target programs which were compiled with the above option. This function is for resolving the above issue. See the users' manual of each compiler for details of the above option


☐ Always treat variables of enumerator type with unknown size as 1 byte.

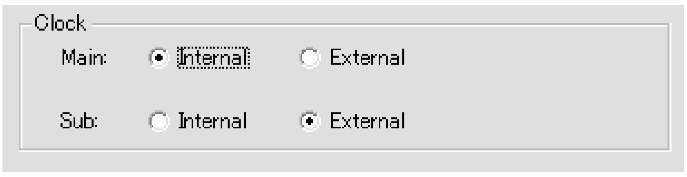
Check the above check box if you would like to treat all sizes of enumeration types as 1 byte. It is necessary to load the debugging information again in order to reflect this setting.

5.1.3 Emulator Tab


Clock
Main: (● Internal) (External)
Sub: (Internal) (● External)
☐ Attempt to access memory even in WAIT/STOP mode.

5.1.3.1 Specify the Target Clock

Change the setting by synchronizing with the clock used by the target microcomputer. (Internal is set by default.)


Clock
Main: (● Internal) (External)
Sub: (Internal) (● External)

Select Internal to set the internal clock, and External to set the external clock.
The specified content becomes effective when the next being start.

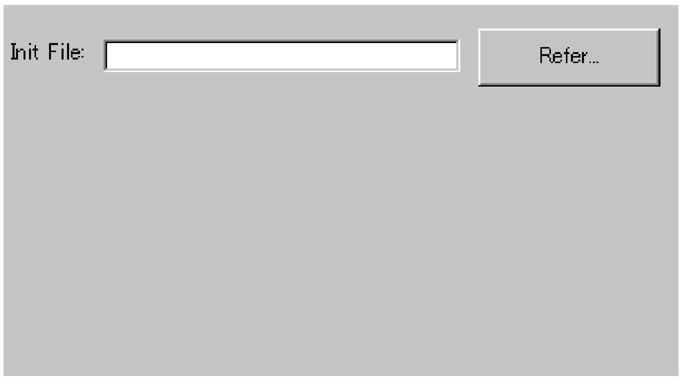
5.1.3.2 Attempt to access memory during WAIT/STOP mode

Set this check on, when the mcu needs to access memory during WAIT/STOP mode.

When this check is ON, debugger will attempt to access memory by waiting for about 5 seconds until the mcu returns from WAIT/STOP mode. If the mcu remains WAIT/STOP mode during this period, the operation will receive an error. When this check is OFF, debugger will receive an error without accessing to real memory.

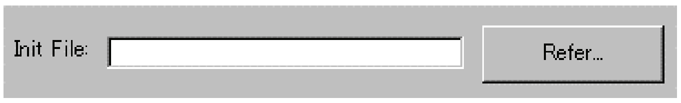
5.1.4 Script Tab

The specified content becomes effective when the next being start.



5.1.4.1 Automatically Execute the Script Commands

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



By clicking the "Refer" button, the File Selection dialog is opened.
The specified script file is displayed in the "Init File:" field.
To disable auto-execution of the script command, erase a character string displayed in the "Init File:" field.

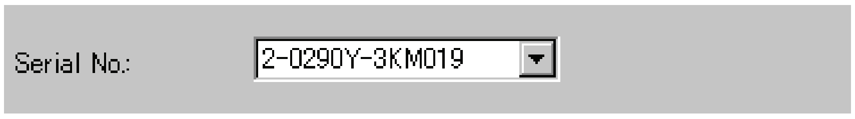
5.2 Setting of the Communication Interface

5.2.1 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 "3.3.1.1 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.

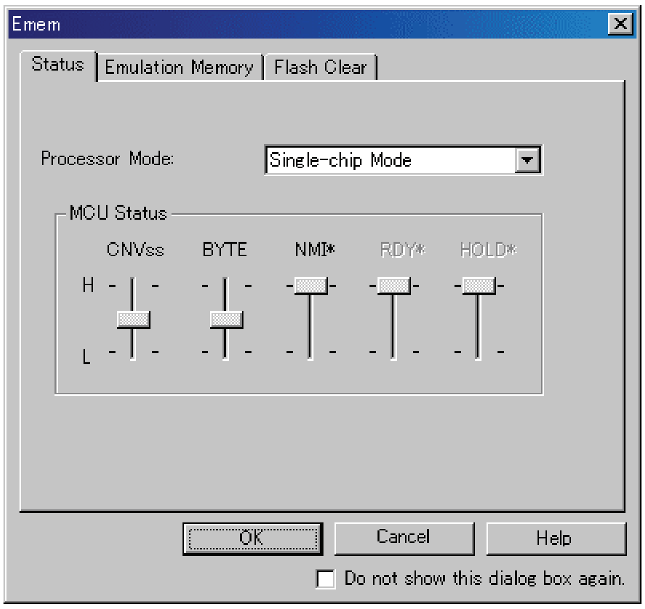


Serial No.: 2-0290Y-3KM019

5.3 Setup the Debugger for M32C

5.3.1 Emem Dialog

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



Processor Mode: Single-chip Mode

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

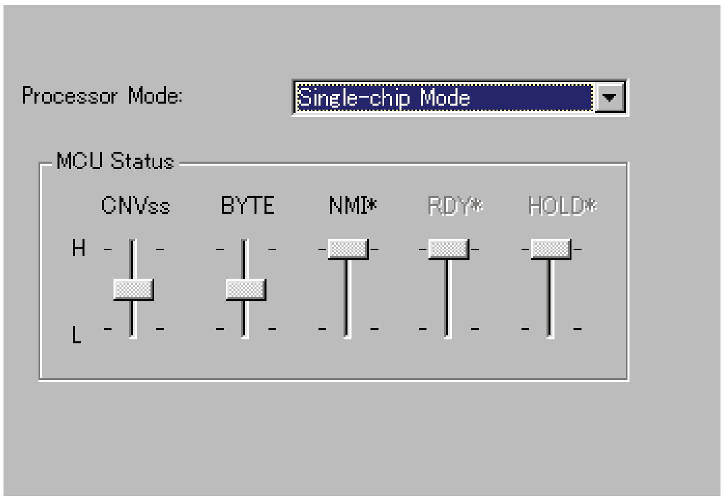
Tab Name	Contents
Status	Specify the processor mode.
Emulation Memory	Specify the emulation memory area.
Flash Clear	Specify whether or not to clear the contents of the MCU's internal flash ROM.

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

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

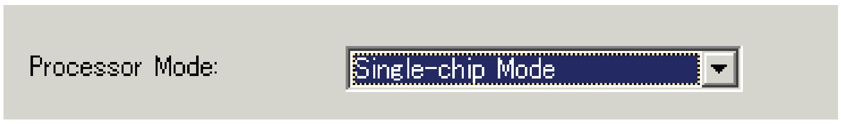
5.3.1.1 Status Tab

The specified content becomes effective when the next being start.


Processor Mode: Single-chip Mode

5.3.1.1.1. Select the Processor Mode

Specify the processor mode for the target system.

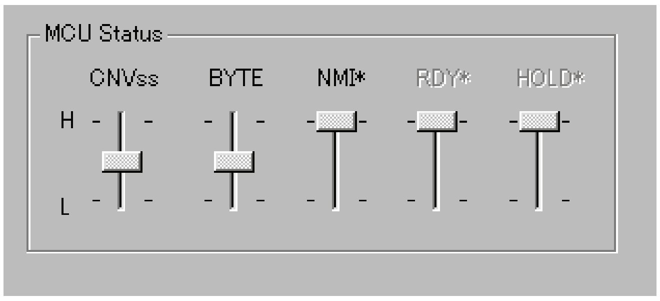

Processor Mode: Single-chip Mode

Either the following can be specified.

- Single-chip Mode
  Single-chip Mode
- Memory Expansion 8 Bit
  Memory Expansion Mode (8 bits bus width)
- Memory Expansion 16 Bit
  Memory Expansion Mode (16 bits bus width)

5.3.1.1.2. Inspecting the MCU status

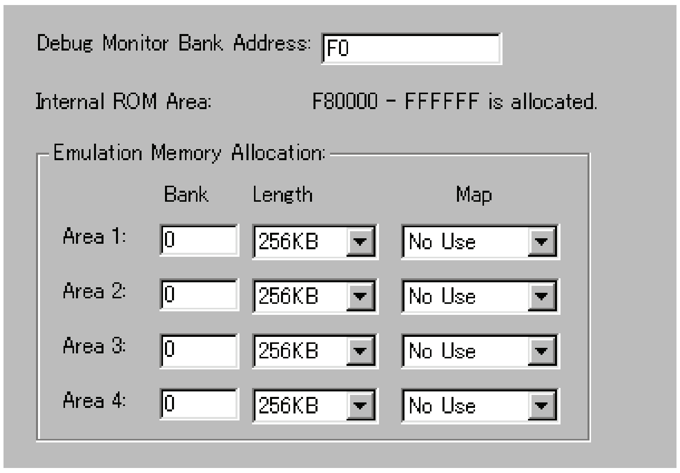
Clicking this tab displays the status of each MCU pin. It allows to check whether the MCU pin status matches the processor mode to be set.



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

5.3.1.2 Emulation Memory Tab

The specified content becomes effective when the next being start.



Debug Monitor Bank Address: F0

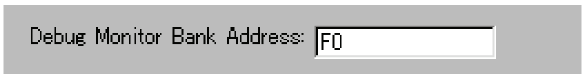
Internal ROM Area: F80000 - FFFFFF is allocated.

Emulation Memory Allocation:

	Bank	Length	Map
Area 1:	0	256KB	No Use
Area 2:	0	256KB	No Use
Area 3:	0	256KB	No Use
Area 4:	0	256KB	No Use

5.3.1.2.1. Debug monitor's bank address settings

This product allocates a 64-Kbyte contiguous address area as the emulator's work area for use by the debug monitor.
Specify any bank that the target system does not use. The debug monitor uses a 64-Kbyte area from the start address of the specified bank.
(Example: If the specified bank is "F0," then the debug monitor uses a 64-Kbyte area beginning with address F000000h.)



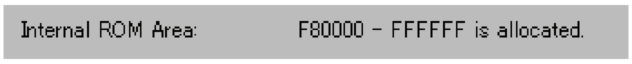
Debug Monitor Bank Address: F0

- The bank specified here cannot have its contents referenced or set.
  The contents of this area when displayed in the Memory window or the Program/Source window's disassemble display mode may not be correct.

- The following bank addresses cannot be specified:
  - MCU internal resources (e.g., SFR and RAM areas)
  - DRAM area and multiplexed area
  - Interrupt vector area

5.3.1.2.2. Automatic emulation memory allocation for the internal ROM

When single-chip or memory extension mode is selected, emulation memory is automatically allocated to the internal ROM area.
The automatically allocated internal ROM address range is displayed in this field.

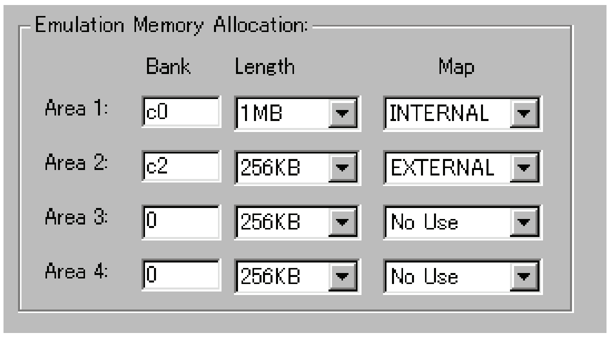


Internal ROM Area: F80000 - FFFFFF is allocated.

5.3.1.2.3. Emulation memory allocation for an extended area

When memory extension or microprocessor mode is selected, emulation memory can be allocated to the extended area to be debugged (in up to four areas).
(When the emulation memory board is not connected to the emulator, the emulation memory cannot be allocated.)

Here, allocate memory for the debug target area and specify its mapping information.


Emulation Memory Allocation:

	Bank	Length	Map
Area 1:	c0	1MB	INTERNAL
Area 2:	c2	256KB	EXTERNAL
Area 3:	0	256KB	No Use
Area 4:	0	256KB	No Use

Follow the procedure described below.

Bank (Set bank address)	Specify the bank address of the debug target area to be allocated in hexadecimal. If specified as C0, C00000h is the start address of the debug target area.
Length (Specify size of area)	Specify the size of the debug target area (256 bytes or 1 Mbytes). If Length is specified to be "256 bytes," banks 00, 04, 08, and up to FC (every four banks) are specified for Bank; if Length is specified to be "1 Mbytes," banks 00, 10, 20, and up to F0 (every 16 banks) are specified for Bank.
Map (Specify area map)	Specify the mapping information ("Internal" or "External") for the specified area. If no area is specified, select "No Use." Internal The area specified to be "Internal" is mapped into the internal area (emulation memory). External The area specified to be "External" is mapped into the external area (external resources in the target system).

- Areas for which "No Use" is selected for Map and those not specified here are mapped into external areas.
  If compared to the case where areas are explicitly specified to be "External," the only difference is a download speed. (Downloading into these areas is slower than downloading into the areas specified to be "External.")
- The internal ROM area is automatically mapped into the emulation memory. Therefore, there is no need to set here.
- Be careful that the debug areas will not overlap.
- Make sure the total size of the specified debug target areas does not exceed the emulation memory size of the emulation memory board used.

The setting of the emulation memory area varies depending on the specified processor mode.

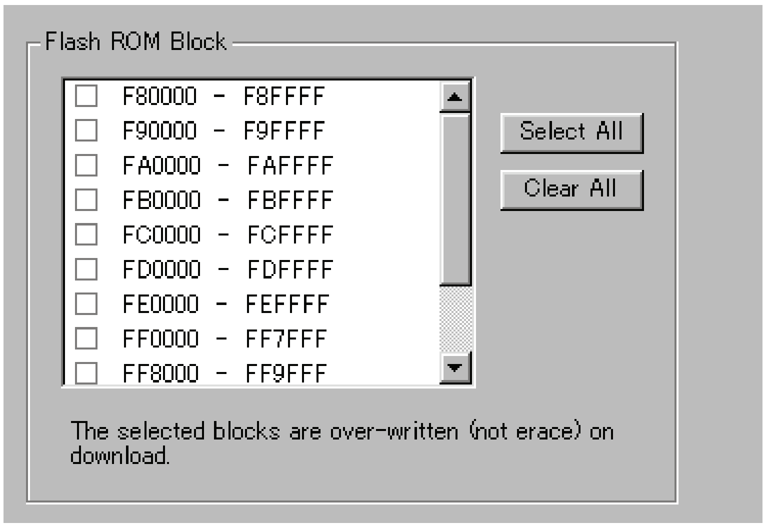
- Single-chip Mode
  You do not need to specify the area to be assigned as the emulation memory.
  The internal ROM area is automatically mapped into the emulation memory. The address range of the automatically mapped area is displayed in the Internal ROM Area: field.
- Memory Expansion Mode(8bit and 16bit)
  If you have an area to be assigned as the emulation memory in addition to internal ROM area, specify it specify it separately.
  The internal ROM area is automatically mapped into the emulation memory. The address range of the automatically mapped area is displayed in the Internal ROM Area: field.
- Microprocessor Mode(8bit and 16bit)
  Specify the area to be assigned separately. (There is no area which is automatically assigned.)

ATTENTION

- The mapping setting data specified using the Map command is not reflected to the Emem dialog box.
- et the emulation memory areas in the order of usage priority.
  The emulation memory areas to be set by the Map command are numbered, ignoring the unused (Not Use) areas.
  Accordingly, the emulation memory areas set in the Emem dialog box and the emulation memory area numbers set by the Map command will be mismatched.

5.3.1.3 Flash Clear Tab

The specified content becomes effective when the next being start.



☐ F80000 - F8FFFF
☐ F90000 - F9FFFF
☐ FA0000 - FAFFFF
☐ FB0000 - FBFFFF
☐ FC0000 - FCFFFF
☐ FD0000 - FDFFFF
☐ FE0000 - FEFFFF
☐ FF0000 - FF7FFF
☐ FF8000 - FF9FFF

The selected blocks are over-written (not erace) on download.

5.3.1.3.1. Setting to clear the MCU's internal flash ROM

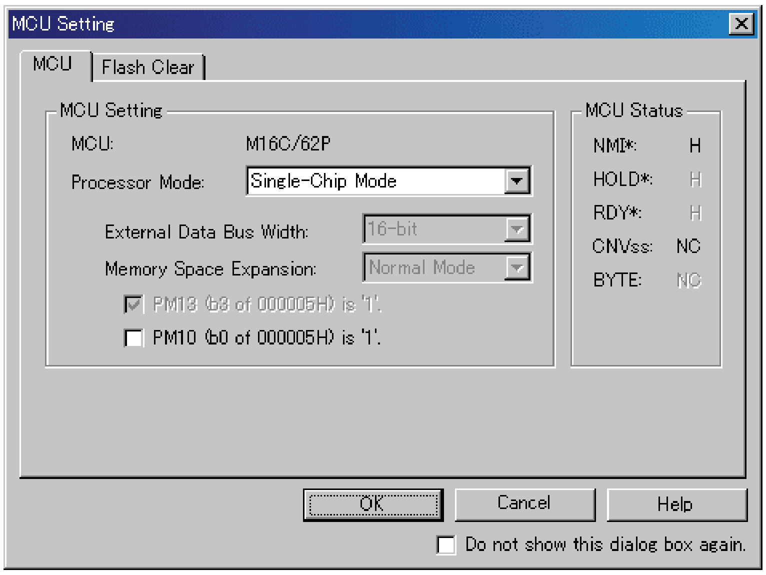
Specify whether or not to clear the contents of the MCU's internal flash ROM when downloading the target program or data.
The MCU's internal flash ROM is displayed block by block in the list view.

- The blocks whose check marks are turned on do not have their flash contents cleared when downloading. The memory contents in places not overwritten by downloading remain intact.
- The blocks whose check marks are turned off have their flash contents cleared when downloading.
- Pressing the Select All button keeps all blocks from being cleared when downloading.
- Pressing the Clear All button clears all blocks when downloading.

5.4 Setup the Debugger for M16C/R8C

5.4.1 MCU Setting Dialog

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



MCU: M16C/62P
Processor Mode: Single-Chip Mode
External Data Bus Width: 16-bit
Memory Space Expansion: Normal Mode
☑ PM13 (b3 of 000005H) is '1'.
☐ PM10 (b0 of 000005H) is '1'.
NMI*: H
HOLD*: H
RDY*: H
CNVss: NC
BYTE: NC
☐ Do not show this dialog box again.

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

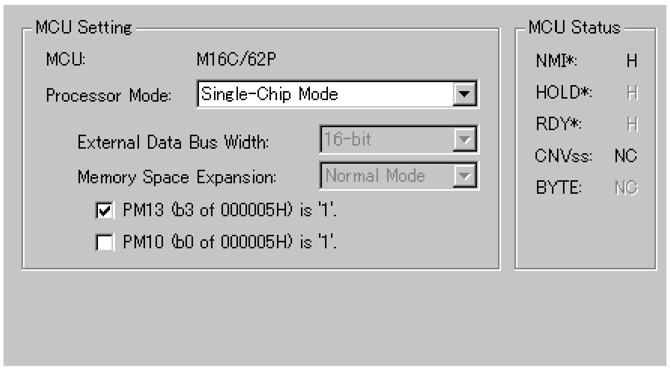
Tab Name	Contents
MCU	Specify the MCU's processor mode, debug options, etc.
MAP	Set memory areas into which emulation memory is mapped. (*The emulation memory board is necessary to use this tab.)
Flash Clear	Specify whether to clear the contents of the MCU's internal flash ROM.

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

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

5.4.1.1 MCU Tab

The specified content becomes effective when the next being start.

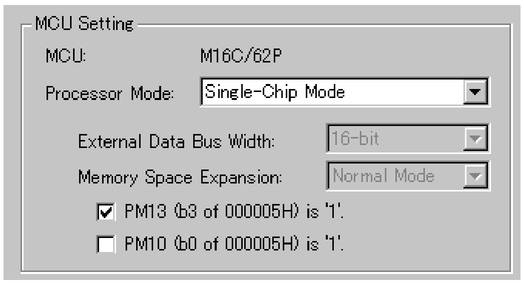


MCU Setting
MCU: M16C/62P
Processor Mode: Single-Chip Mode
External Data Bus Width: 16-bit
Memory Space Expansion: Normal Mode
[✓] PM13 (b3 of 000005H) is '1'.
[ ] PM10 (b0 of 000005H) is '1'.

MCU Status
NMI*: H
HOLD*: H
RDY*: H
CNVss: NC
BYTE: NC

5.4.1.1.1. Select the Processor Mode

Specify the processor mode for the target system.



MCU Setting
MCU: M16C/62P
Processor Mode: Single-Chip Mode
External Data Bus Width: 16-bit
Memory Space Expansion: Normal Mode
[✓] PM13 (b3 of 000005H) is '1'.
[ ] PM10 (b0 of 000005H) is '1'.

Either the following can be specified.

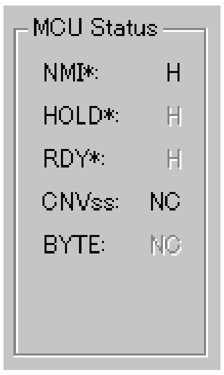
- Single-chip Mode
  Single-chip Mode
- Memory Expansion Mode
  Memory Expansion Mode
- Microprocessor Mode
  Microprocessor Mode

Also, you need to specify the following information according to the processor mode you've selected.

- External Data Bus Width
  If you selected memory extension or microprocessor mode, specify "16-bit" or "8-bit" for the external bus width. Make sure the specified external bus width matches settings of the BYTE pin.
- Memory Space Expansion
  If you selected memory extension or microprocessor mode, specify whether or not to use the memory space expansion facility. Select "4MB Mode" if you want to use the memory space expansion facility or "Normal Mode" if you do not.
- PM13(b3 of 000005H)
  Specify whether you set the bit PM13 (b3 of 000005H). When you use your target system with the setting that PM13 is 1, check this option.
- PM10(b0 of 000005H)
  Specify whether you set the bit PM10 (b0 of 000005H). When you use your target system with the setting that PM10 is 1, check this option.

5.4.1.1.2. Inspecting the MCU status

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



MCU Status

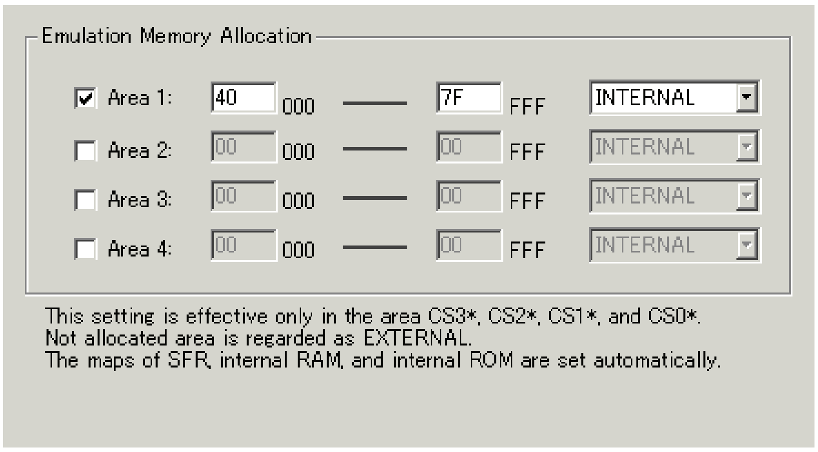
NMI*:	H
HOLD*:	H
RDY*:	H
CNVss:	NC
BYTE:	NC

"NC" means that the value is indeterminate.

5.4.1.2 MAP Tab

The emulation memory board is necessary to use this command.

The specified content becomes effective when the next being start.



Emulation Memory Allocation

✓ Area 1:	40 000	—	7F FFF	INTERNAL
Area 2:	00 000	—	00 FFF	INTERNAL
Area 3:	00 000	—	00 FFF	INTERNAL
Area 4:	00 000	—	00 FFF	INTERNAL

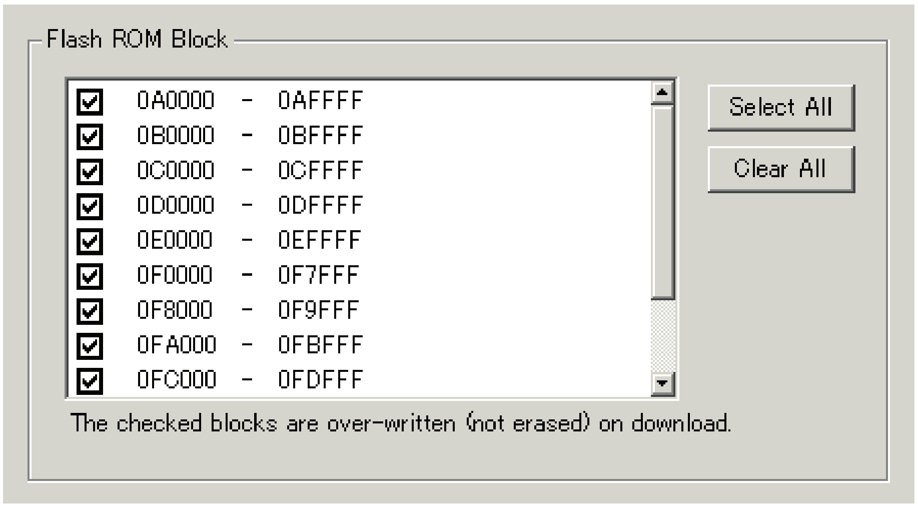
This setting is effective only in the area CS3*, CS2*, CS1*, and CS0*.
Not allocated area is regarded as EXTERNAL.
The maps of SFR, internal RAM, and internal ROM are set automatically.

5.4.1.2.1. Emulation memory allocation

Set the memory area in 4 KB units into which you want the emulation memory to be mapped. Four of such memory areas can be set.
The emulation memory is mapped into the areas marked "Internal." The unselected areas and the areas which have nothing specified are allocated to external areas.
Note that MAP settings are effective for only the areas CS3*, CS2*, CS1*, and CS0*. The SFR, internal ROM, and internal RAM areas are automatically mapped.

5.4.1.3 Flash Clear Tab

The specified content becomes effective when the next being start.



Flash ROM Block

✓	0A0000	-	0AFFFF
✓	0B0000	-	0BFFFF
✓	0C0000	-	0CFFFF
✓	0D0000	-	0DFFFF
✓	0E0000	-	0EFFFF
✓	0F0000	-	0F7FFF
✓	0F8000	-	0F9FFF
✓	0FA000	-	0FBFFF
✓	0FC000	-	0FDFFF

The checked blocks are over-written (not erased) on download.

5.4.1.3.1. Setting to clear the MCU's internal flash ROM

Specify whether or not to clear the contents of the MCU's internal flash ROM when downloading the target program or data. The MCU's internal flash ROM is displayed block by block in the list view.

- The blocks whose check marks are turned on do not have their flash contents cleared when downloading. The memory contents in places not overwritten by downloading remain intact.
- The blocks whose check marks are turned off have their flash contents cleared when downloading.
- Pressing the Select All button keeps all blocks from being cleared when downloading.
- Pressing the Clear All button clears all blocks when downloading.

Tutorial

(Blank Page)

6. Tutorial

6.1 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.

6.2 Usage

Please follow these instructions:

6.2.1 Step1 : Starting the Debugger

6.2.1.1 Preparation before Use

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

6.2.1.2 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 "5 Setup the Debugger ".
After the setup of a debugger, it will function as a debugger.

6.2.2 Step2：Checking the Operation of RAM

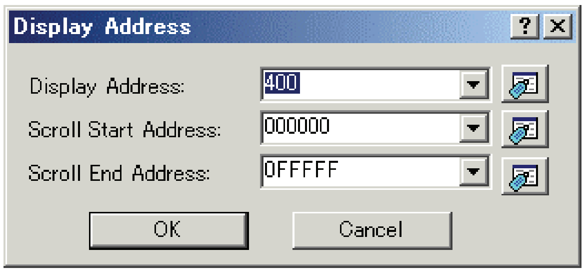
Check that RAM is operating correctly. Display and edit the contents of the memory in the [Memory] window to check that the memory is operating correctly.

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.

6.2.2.1 Checking the Operation of RAM

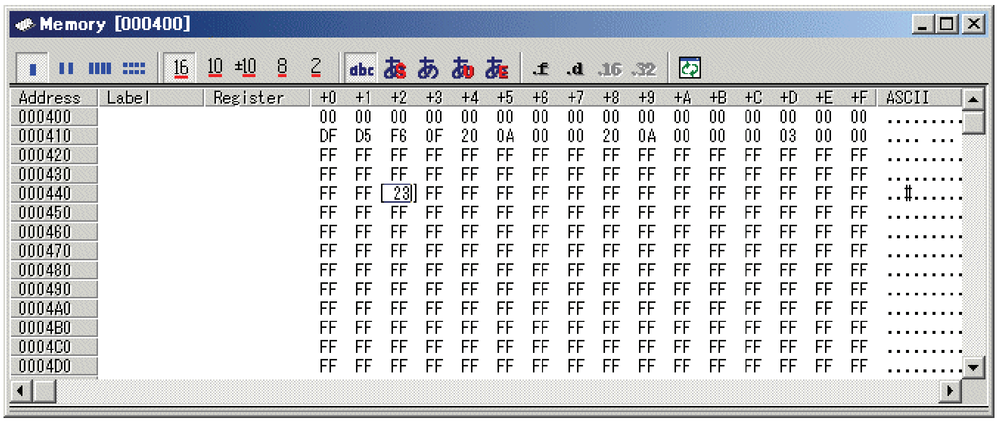
Select [Memory] from the [CPU] submenu of the [View] menu and enter the RAM address (Here, enter "400") in the [Display Address] edit boxes. The [Scroll Start Address] and [Scroll End Address] editing box is left to a default setting. (By default, the scroll range is set to 0h to the maximum address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



Display Address:	400
Scroll Start Address:	000000
Scroll End Address:	0FFFFF

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.



```
Address  Label  Register  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F  ASCII
000400                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........
000410                    DF D5 F6 0F 20 0A 00 00 20 0A 00 00 00 03 00 00  .... ...
000420                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
000430                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
000440                    FF FF 23 FF FF FF FF FF FF FF FF FF FF FF FF FF  ..‡......
000450                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
000460                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
000470                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
000480                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
000490                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
0004A0                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
0004B0                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
0004C0                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
0004D0                    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .........
```

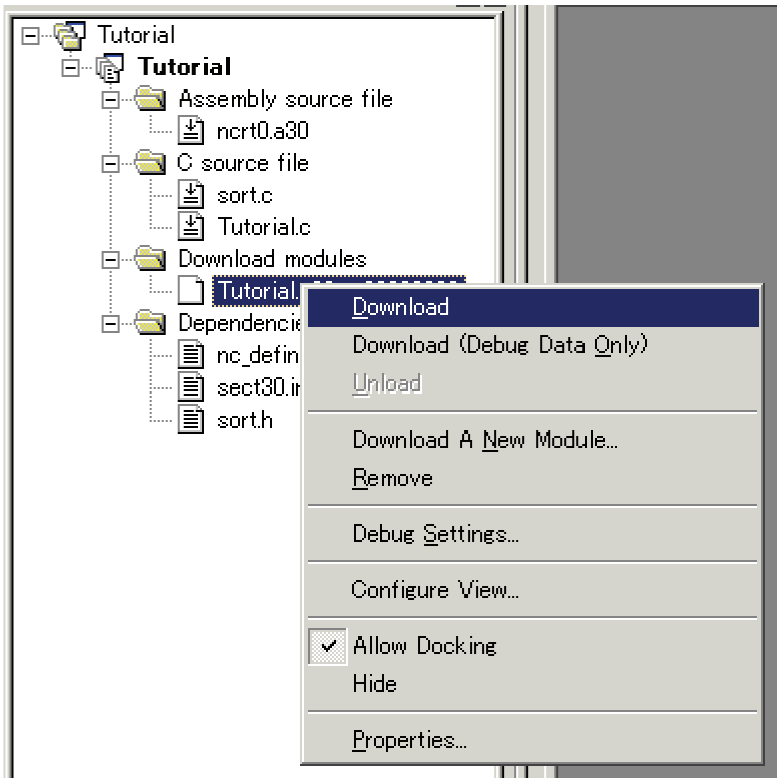
Placing the mouse cursor on a point in the display of data in the [Memory] window and double-clicking allows the values at that point to be changed.

6.2.3 Step3 : Downloading the Tutorial Program

6.2.3.1 Downloading the Tutorial Program

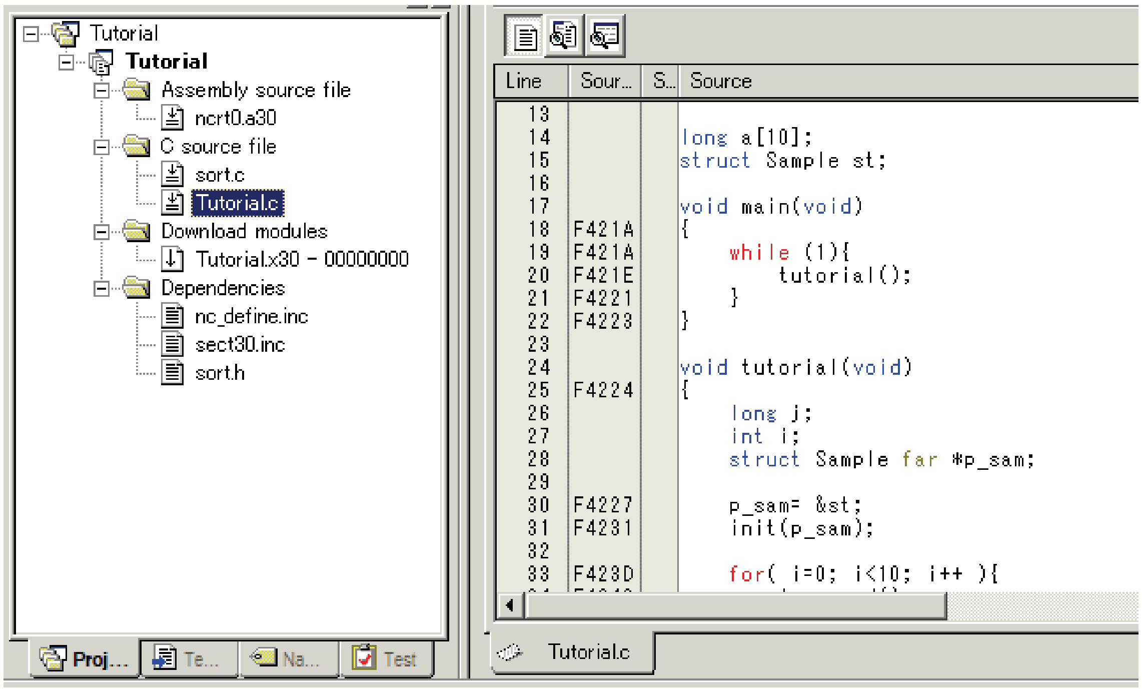
Download the object program to be debugged. The download file and the address to be downloaded will depends on the target mcu you uses. Please replace the screen image and addresses with corresponding one to your target mcu.

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



6.2.3.2 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.



```
          long a[10];
          struct Sample st;

          void main(void)
F421A     {
F421A         while (1){
F421E             tutorial();
F4221         }
F4223     }

          void tutorial(void)
F4224     {
              long j;
              int i;
              struct Sample far *p_sam;

F4227         p_sam= &st;
F4231         init(p_sam);

F423D         for( i=0; i<10; i++ ){
```

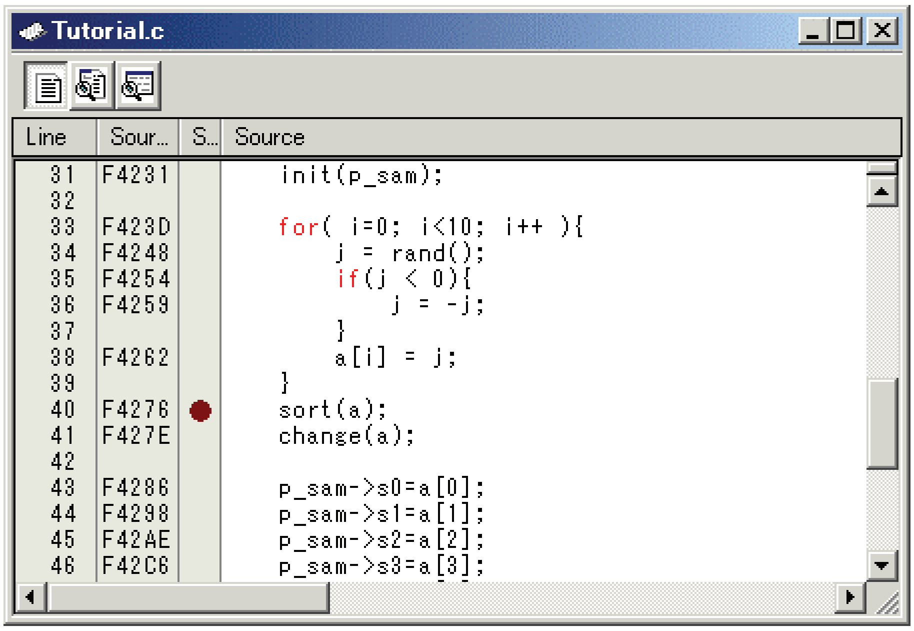
Select the [Format Views...] option from the [Setup] menu to set a font and size that are legible, if necessary.
Initially the [Editor(Source)] window shows the start of the user program, but the user can use the scroll bar to scroll through the user program and look at the other statements.

6.2.4 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.

6.2.4.1 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.



```
31  F4231       init(p_sam);
32
33  F423D       for( i=0; i<10; i++ ){
34  F4248           j = rand();
35  F4254           if(j < 0){
36  F4259               j = -j;
37                  }
38  F4262           a[i] = j;
39              }
40  F4276  ●    sort(a);
41  F427E       change(a);
42
43  F4286       p_sam->s0=a[0];
44  F4298       p_sam->s1=a[1];
45  F42AE       p_sam->s2=a[2];
46  F42C6       p_sam->s3=a[3];
```

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

6.2.5 Step5：Executing the Program

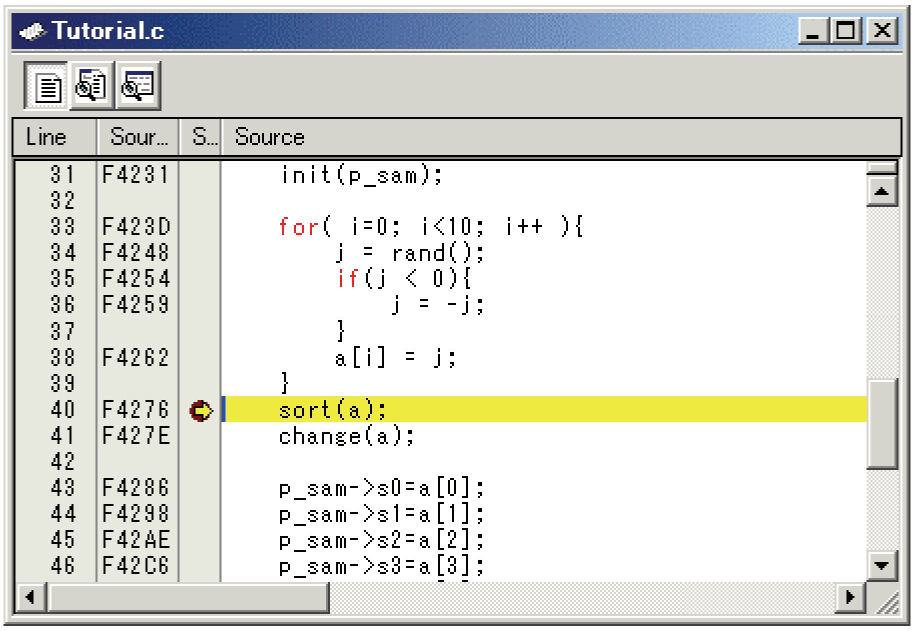
Execute the program as described in the following:

6.2.5.1 Resetting of CPU

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

6.2.5.2 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.



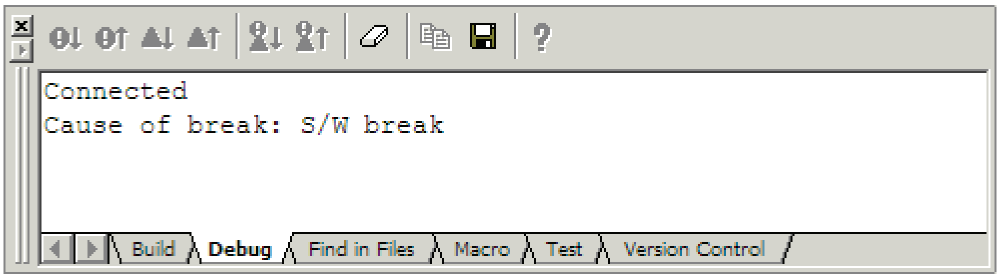
```
F4231       init(p_sam);

F423D       for( i=0; i<10; i++ ){
F4248           j = rand();
F4254           if(j < 0){
F4259               j = -j;
                }
F4262           a[i] = j;
            }
F4276       sort(a);
F427E       change(a);

F4286       p_sam->s0=a[0];
F4298       p_sam->s1=a[1];
F42AE       p_sam->s2=a[2];
F42C6       p_sam->s3=a[3];
```

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.

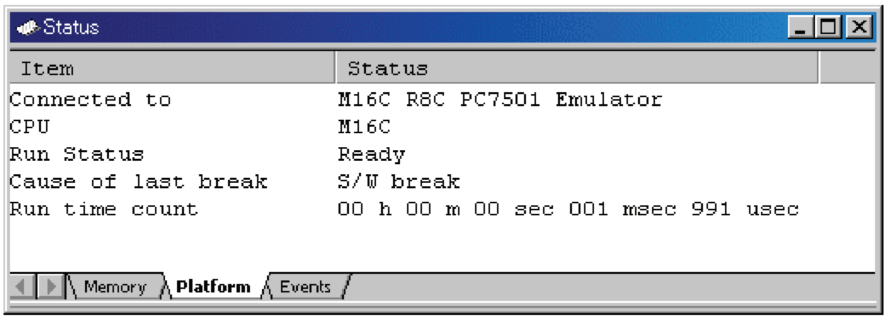
6.2.5.3 Reviewing Cause of the Break

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



```
Connected
Cause of break: S/W break
```

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.



Item	Status
Connected to	M16C R8C PC7501 Emulator
CPU	M16C
Run Status	Ready
Cause of last break	S/W break
Run time count	00 h 00 m 00 sec 001 msec 991 usec

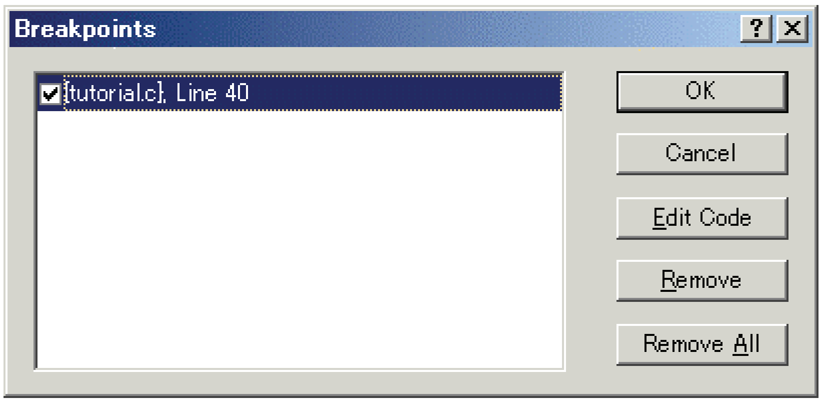
Please refer to "11 Display the Cause of the Program Stoppage " about the notation of a break factor.

6.2.6 Step6：Reviewing Breakpoints

The user can see all the breakpoints set in the program in the [Breakpoints] dialog box.

6.2.6.1 Reviewing Breakpoints

Push the key Ctrl+B, and the [Breakpoints] dialog box will be displayed.



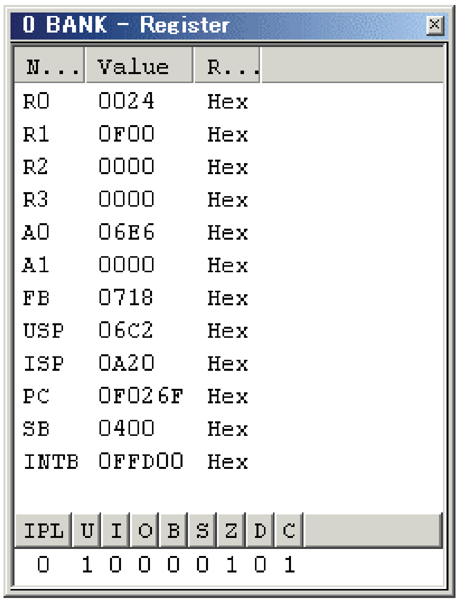
This window allows the user to delete, enable, or disable breakpoints.

6.2.7 Step7 : Viewing Register

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

6.2.7.1 Viewing Register

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

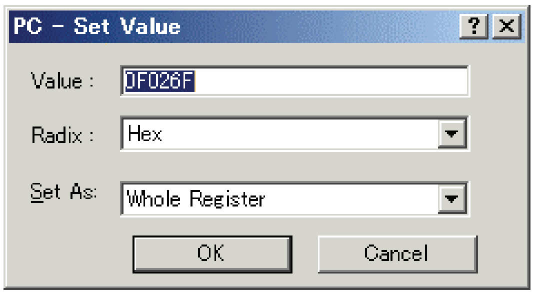


N...	Value	R...
R0	0024	Hex
R1	0F00	Hex
R2	0000	Hex
R3	0000	Hex
A0	06E6	Hex
A1	0000	Hex
FB	0718	Hex
USP	06C2	Hex
ISP	0A20	Hex
PC	0F026F	Hex
SB	0400	Hex
INTB	0FFD00	Hex

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

6.2.7.2 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.



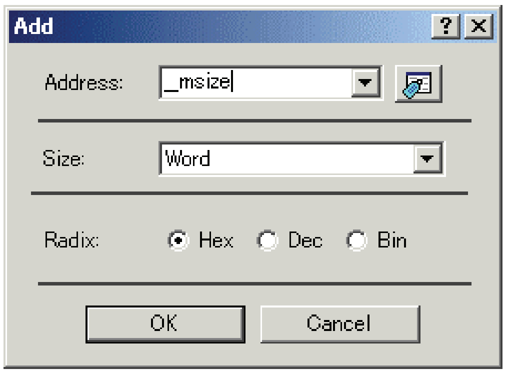
Value : 0F026F
Radix : Hex
Set As: Whole Register

6.2.8 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.

6.2.8.1 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.

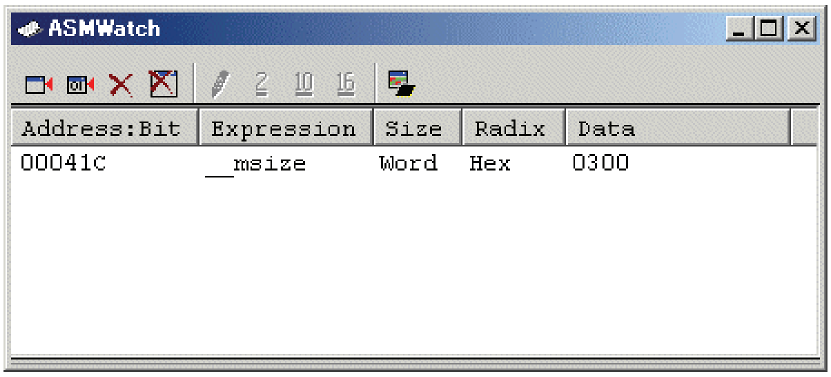

Add

Address: _msize

Size: Word

Radix: [✓ Hex] Dec Bin

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


ASMWatch

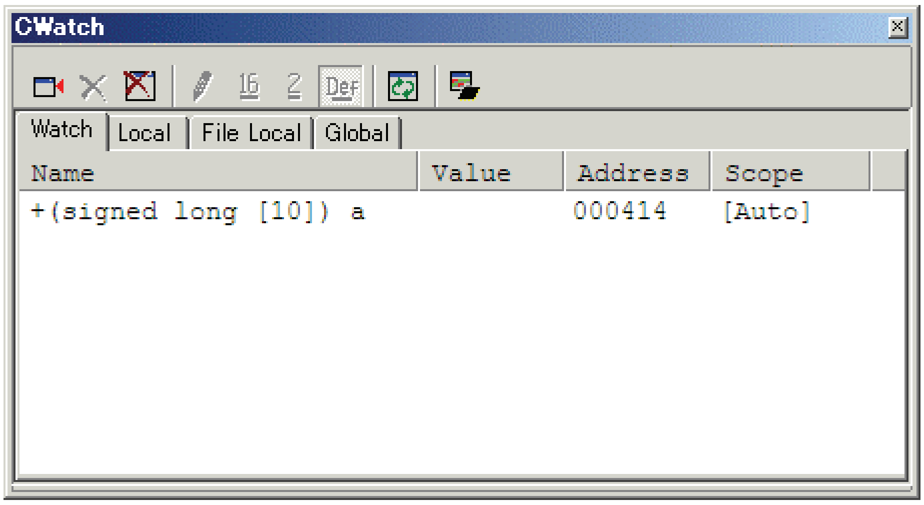
Address:Bit	Expression	Size	Radix	Data
00041C	__msize	Word	Hex	0300

6.2.9 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.

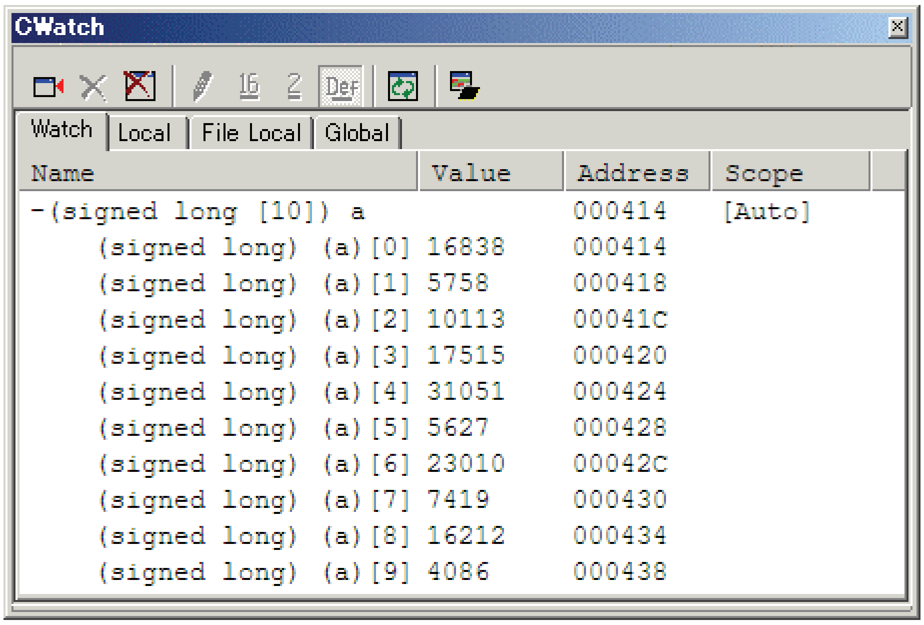
6.2.9.1 Watching Variables

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.



```
Name                    Value   Address   Scope
+(signed long [10]) a           000414    [Auto]
```

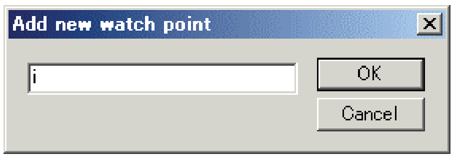
The user can click mark '+' at the left side of array a in the [C Watch] window to watch all the elements.



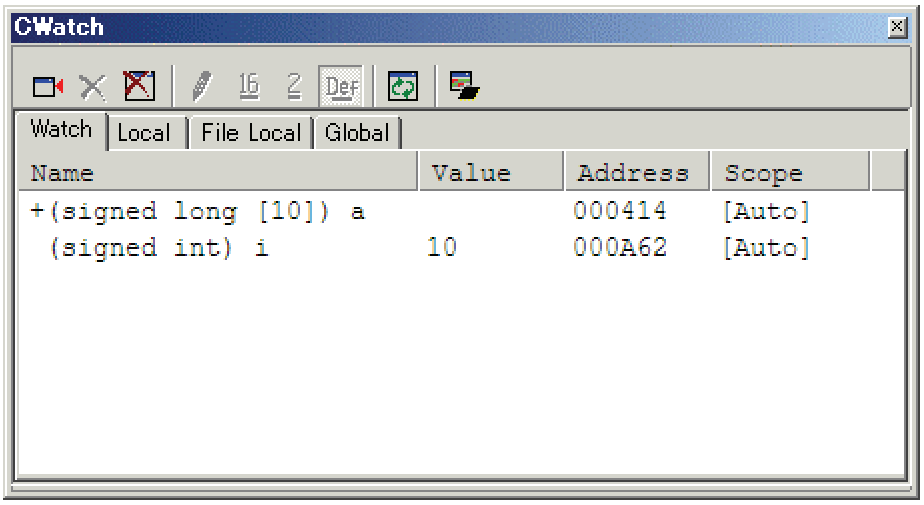
```
Name                            Value   Address   Scope
-(signed long [10]) a                   000414    [Auto]
    (signed long) (a)[0]        16838   000414
    (signed long) (a)[1]        5758    000418
    (signed long) (a)[2]        10113   00041C
    (signed long) (a)[3]        17515   000420
    (signed long) (a)[4]        31051   000424
    (signed long) (a)[5]        5627    000428
    (signed long) (a)[6]        23010   00042C
    (signed long) (a)[7]        7419    000430
    (signed long) (a)[8]        16212   000434
    (signed long) (a)[9]        4086    000438
```

6.2.9.2 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.



```
Name                        Value      Address    Scope
+(signed long [10]) a                  000414     [Auto]
 (signed int) i             10         000A62     [Auto]
```

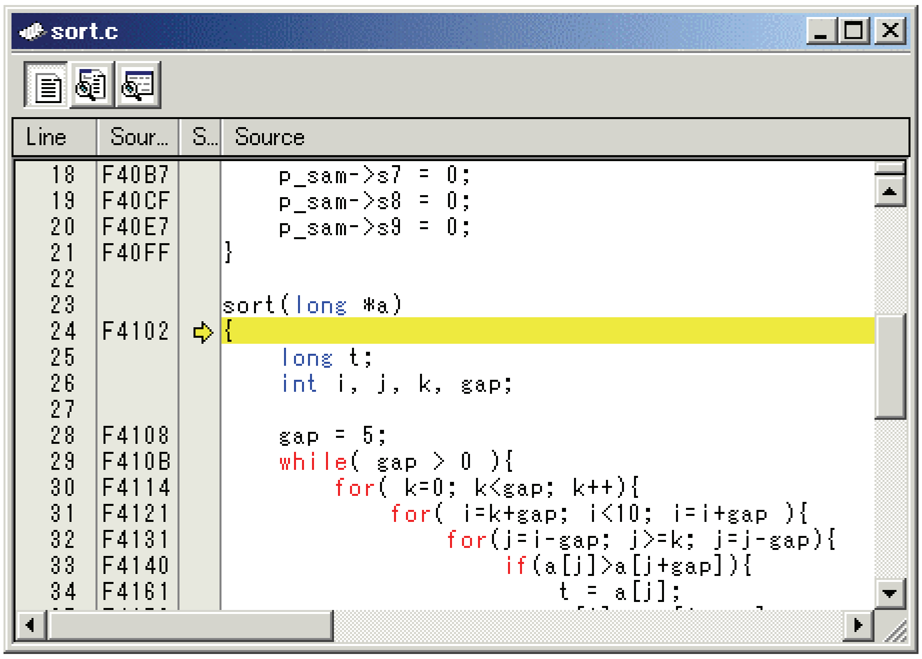
6.2.10 Step10 : Stepping Through a Program

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.

6.2.10.1 Executing [Step In] Command

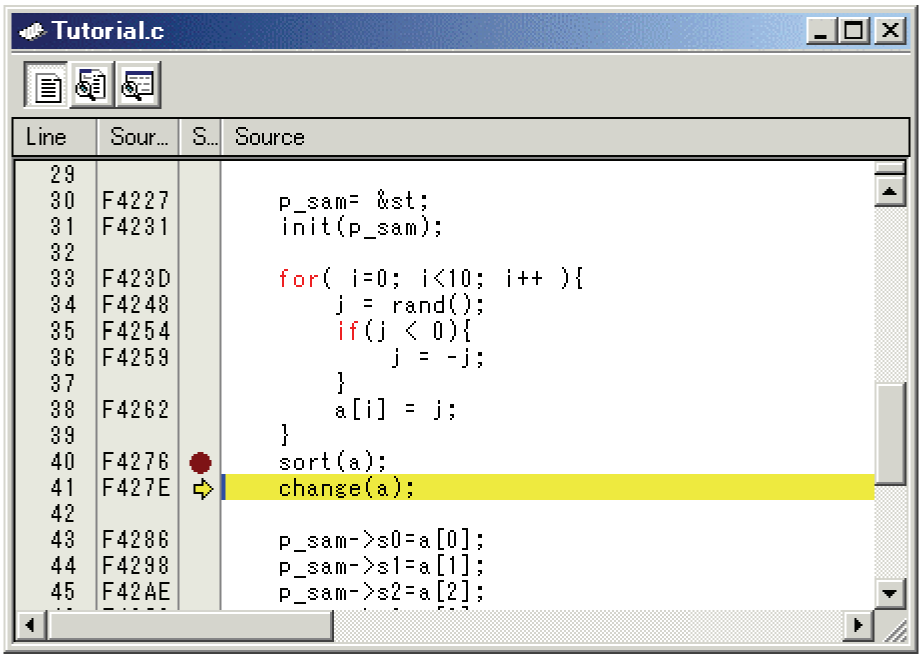
The [Step In] command steps into the called function(subroutine) and stops at the first statement of the called function(subroutine).
To step through the sort function, select [Step In] from the [Debug] menu, or click the [Step In] button on the toolbar.
The PC cursor moves to the first statement of the sort function in the [Editor(Source)] window.



```
F40B7          p_sam->s7 = 0;
F40CF          p_sam->s8 = 0;
F40E7          p_sam->s9 = 0;
F40FF      }

           sort(long *a)
F4102   ⇨  {
               long t;
               int i, j, k, gap;

F4108          gap = 5;
F410B          while( gap > 0 ){
F4114              for( k=0; k<gap; k++){
F4121                  for( i=k+gap; i<10; i=i+gap ){
F4131                      for(j=i-gap; j>=k; j=j-gap){
F4140                          if(a[j]>a[j+gap]){
F4161                              t = a[j];
```

6.2.10.2 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.



```
            
F4227       p_sam= &st;
F4231       init(p_sam);

F423D       for( i=0; i<10; i++ ){
F4248           j = rand();
F4254           if(j < 0){
F4259               j = -j;
                }
F4262           a[i] = j;
            }
F4276  ●    sort(a);
F427E  ⇨    change(a);

F4286       p_sam->s0=a[0];
F4298       p_sam->s1=a[1];
F42AE       p_sam->s2=a[2];
```

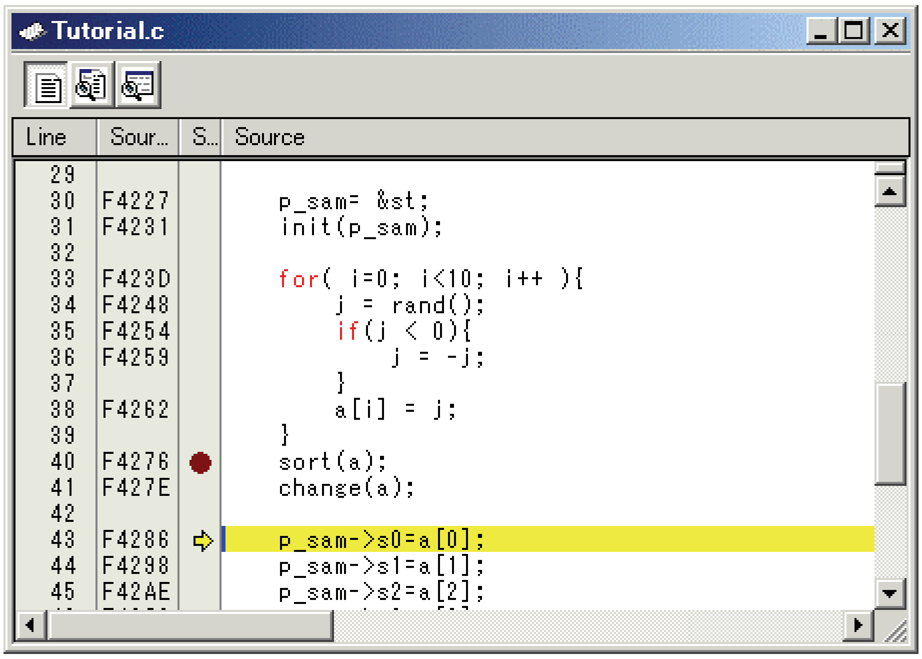
Note
It takes time to execute this function. When the calling source is clarified, use [Go To Cursor].

6.2.10.3 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.



```
29
30  F4227      p_sam= &st;
31  F4231      init(p_sam);
32
33  F423D      for( i=0; i<10; i++ ){
34  F4248          j = rand();
35  F4254          if(j < 0){
36  F4259              j = -j;
37                 }
38  F4262          a[i] = j;
39             }
40  F4276  ●   sort(a);
41  F427E      change(a);
42
43  F4286  ⇨   p_sam->s0=a[0];
44  F4298      p_sam->s1=a[1];
45  F42AE      p_sam->s2=a[2];
```

6.2.11 Step11：Forced Breaking of Program Executions

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

6.2.11.1 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.

6.2.12 Step12 : Displaying Local Variables

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

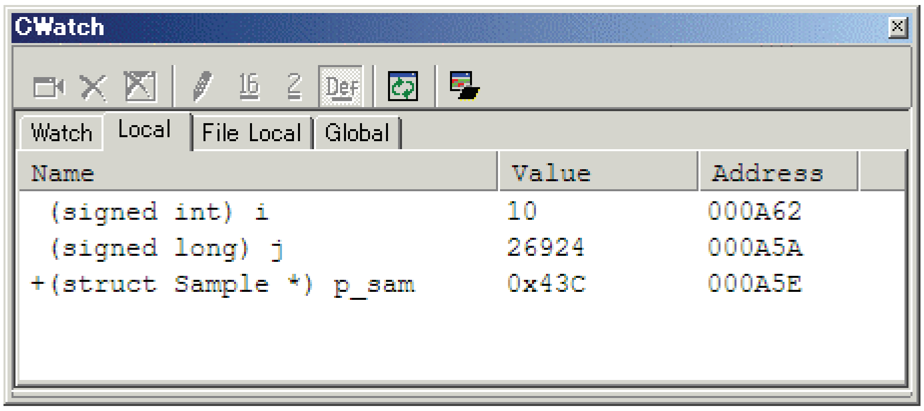
6.2.12.1 Displaying Local Variables

For example, we will examine the local variables in the tutorial function, which declares three local variables: i, j, and p_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.



Name	Value	Address
(signed int) i	10	000A62
(signed long) j	26924	000A5A
+(struct Sample *) p_sam	0x43C	000A5E

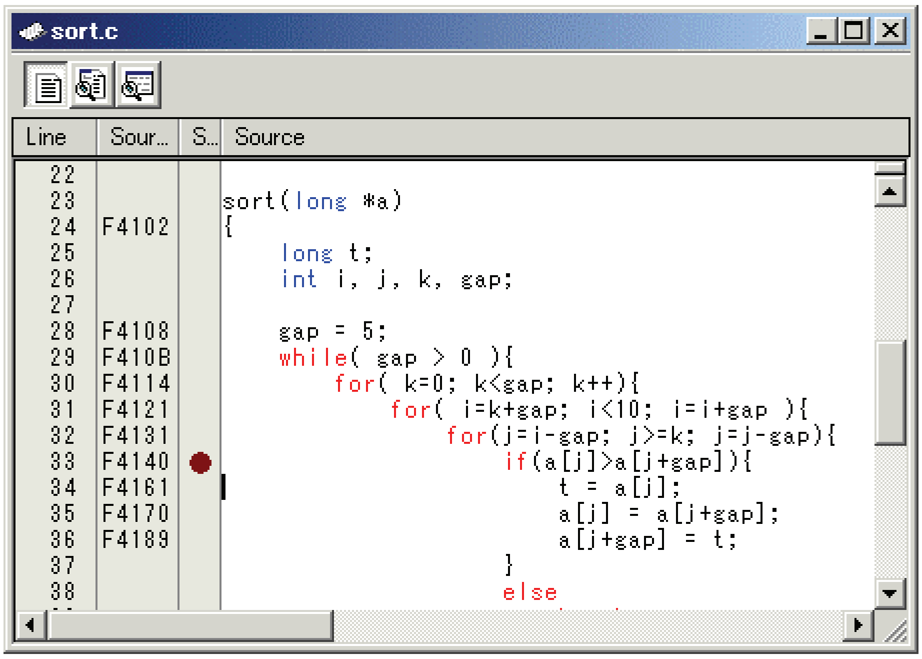
Double-click the mark '+' at the left side of pointer p_sam in the [Locals] window to display the structure *(p_sam).
When the user refers to the members of the structure at the end of the Tutorial function, it is clarified that random data is sorted in descending order.

6.2.13 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.

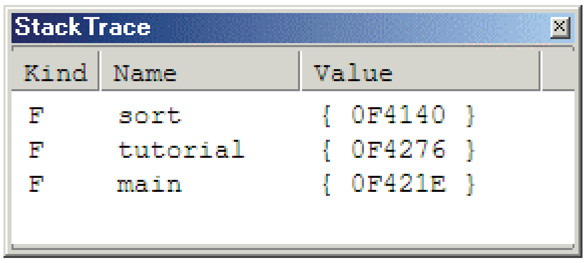
6.2.13.1 Reference the function call status

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



```
Sour...  S...  Source

                sort(long *a)
F4102           {
                    long t;
                    int i, j, k, gap;

F4108               gap = 5;
F410B               while( gap > 0 ){
F4114                   for( k=0; k<gap; k++){
F4121                       for( i=k+gap; i<10; i=i+gap ){
F4131                           for(j=i-gap; j>=k; j=j-gap){
F4140     ●                         if(a[j]>a[j+gap]){
F4161                                   t = a[j];
F4170                                   a[j] = a[j+gap];
F4189                                   a[j+gap] = t;
                                    }
                                    else
```

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.



```
Kind  Name      Value
F     sort      { 0F4140 }
F     tutorial  { 0F4276 }
F     main      { 0F421E }
```

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.

6.2.14 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)

7. Windows/Dialogs

The window of this debugger is shown below.

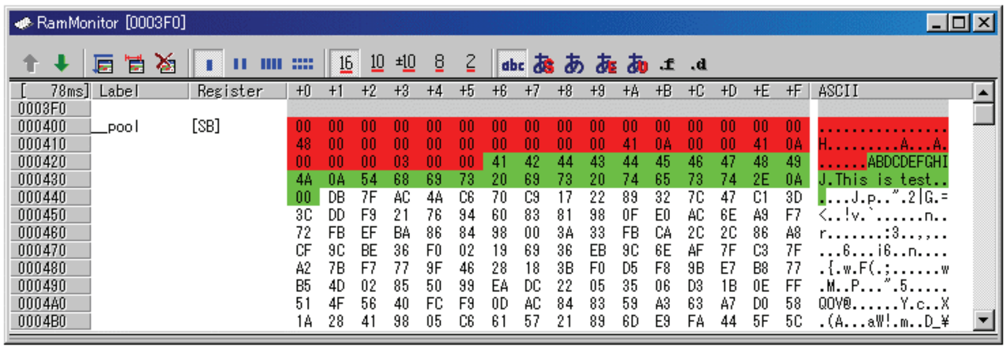
Window Name	View Menu
RAM Monitor Window	[View]->[CPU]->[RamMonitor]
ASM Watch Window	[View]->[Symbol]->[ASMWatch]
C Watch Window	[View]->[Symbol]->[CWatch]
Script Window	[View]->[Script]
S/W Break Point Setting Window	[View]->[Break]->[S/W Break Points]
H/W Break Point Setting Window	[View]->[Break]->[H/W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

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

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

7.1 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.



```
RamMonitor [0003F0]
[    78ms] Label      Register  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F  ASCII
0003F0
000400    __pool      [SB]      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000410                          48 00 00 00 00 00 00 00 00 00 41 0A 00 00 41 0A  H.........A...A.
000420                          00 00 00 03 00 00 41 42 44 43 44 45 46 47 48 49  ......ABDCDEFGHI
000430                          4A 0A 54 68 69 73 20 69 73 20 74 65 73 74 2E 0A  J.This is test..
000440                          00 DB 7F AC 4A C6 70 C9 17 22 89 32 7C 47 C1 3D  ....J.p..".2|G.=
000450                          3C DD F9 21 76 94 60 83 81 98 0F E0 AC 6E A9 F7  <..!v.`......n..
000460                          72 FB EF BA 86 84 98 00 3A 33 FB CA 2C 2C 86 A8  r.......:3..,,..
000470                          CF 9C BE 36 F0 02 19 69 36 EB 9C 6E AF 7F C3 7F  ...6...i6..n....
000480                          A2 7B F7 77 9F 46 28 18 3B F0 D5 F8 9B E7 B8 77  .{.w.F(.;......w
000490                          B5 4D 02 85 50 99 EA DC 22 05 35 06 D3 1B 0E FF  .M..P...".5.....
0004A0                          51 4F 56 40 FC F9 0D AC 84 83 59 A3 63 A7 D0 58  QOV@......Y.c..X
0004B0                          1A 28 41 98 05 C6 61 57 21 89 6D E9 FA 44 5F 5C  .(A...aW!.m..D_¥
```

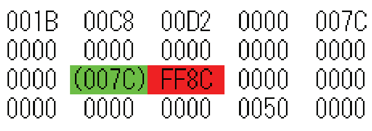
- This system has 1 Kbyte of RAM monitor area which can be located in any contiguous address location or in 4 separate blocks comprised of 256 bytes each.
- The RAM monitor area can be changed to any desired address range.
  Refer to "7.1.2 Setting the RAM monitor area" for details on how to change the 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.



```
001B  00C8  00D2  0000  007C
0000  0000  0000  0000  0000
0000 (007C) FF8C  0000  0000
0000  0000  0000  0050  0000
```

- 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

7.1.1 Extended Menus

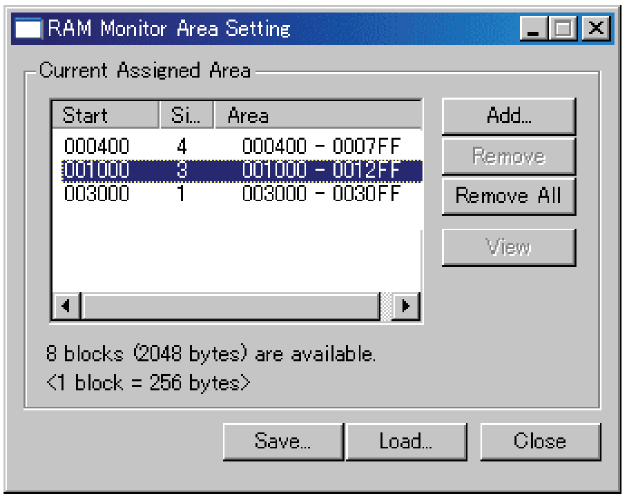
This window has the following popup menus that can be brought up by right-clicking in the window.

Menu		Function
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.

7.1.2 Setting the RAM monitor area

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

The RAM monitor area setup window shown below will appear. The currently set RAM monitor areas are listed in this window.



Start	Si...	Area
000400	4	000400 - 0007FF
001000	3	001000 - 0012FF
003000	1	003000 - 0030FF

8 blocks (2048 bytes) are available.
<1 block = 256 bytes>

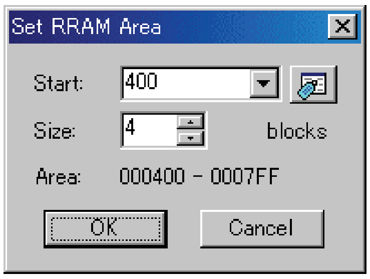
Use this window to add, delete or change RAM monitor areas.

- Specify a RAM monitor area by its start address and size (the latter by a number of blocks.)
- The start address can be specified in 0x100 byte units.
  If you specify a non-aligned address value, it is rounded off to the nearest address value in 0x100 byte units before being set.
- Specify the size of the RAM monitor area by a number of blocks.
  For the Compact Emulator, one block is 256 bytes in size. Up to 4 blocks can be specified.
- RAM monitor areas can be added until the total number of blocks used reaches 4.
  (The number of blocks (and the size) that are currently available to use are displayed below the list.)

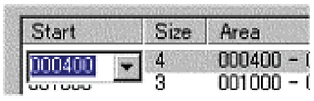
7.1.2.1 Changing the RAM Monitor Area

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

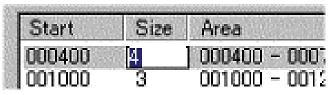
- Changing from a dialog box
  Select the RAM monitor area you want to change from a list of RAM monitor areas and double-click on it.
  The Set RRAM Area dialog box shown below will appear. Specify the start address and the size (by a number of blocks) of the RAM monitor area in the Start and the Size fields of this dialog box.



- Changing directly in the window
  Select the RAM monitor area you want to change from a list of RAM monitor areas and click again in its Start display column or Size display column.
  Specify a new start address or a new size with which you want to be changed in the ensuing edit box. Press the Enter key to confirm what you've entered, or the Esc key to cancel.



Changing the address



Changing the size

7.1.2.2 Adding RAM Monitor Areas

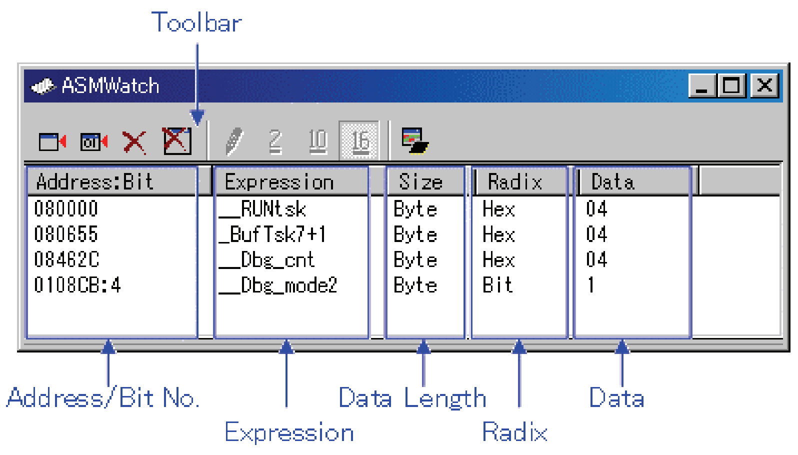
Click the [Add...] button.
The Set RRAM Area dialog box will appear. Specify the start address and the size (by a number of blocks) of a new RAM monitor area in the Start and the Size fields of this dialog box.

7.1.2.3 Deleting RAM Monitor Areas

Select the RAM monitor area you want to delete from a list of RAM monitor areas and click the [Remove] button.
To delete all RAM monitor areas, click the [Remove All] button.

7.2 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.



Address:Bit	Expression	Size	Radix	Data
080000	__RUNtsk	Byte	Hex	04
080655	_BufTsk7+1	Byte	Hex	04
08462C	__Dbg_cnt	Byte	Hex	04
0108CB:4	__Dbg_mode2	Byte	Bit	1

- 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.

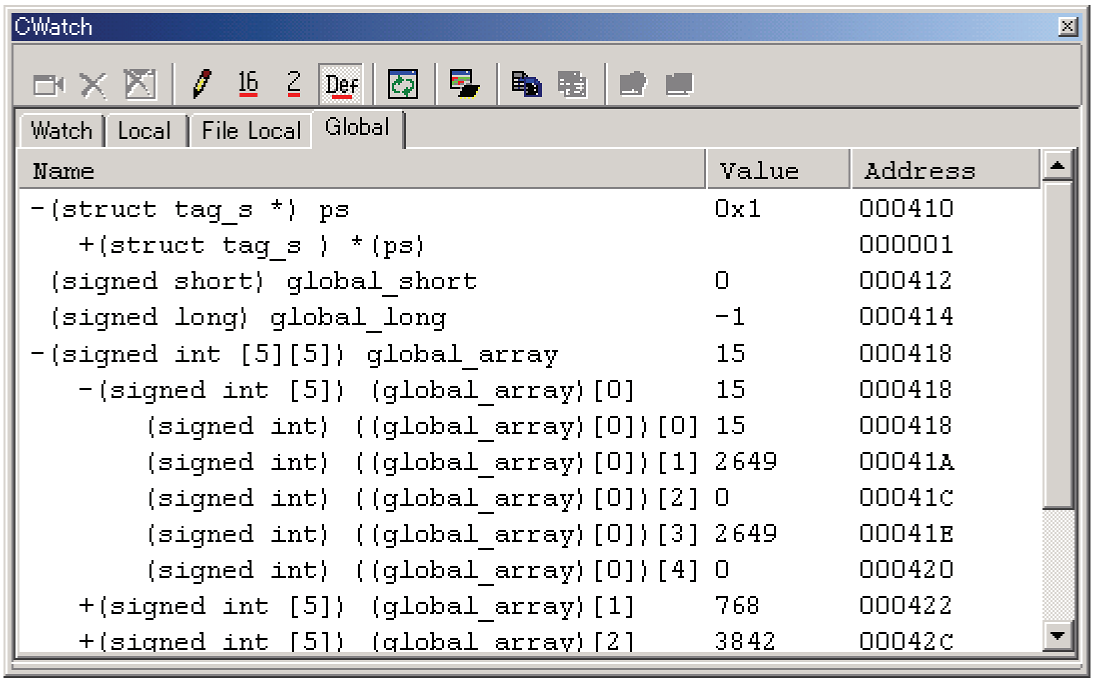
7.2.1 Extended Menus

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.

7.3 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.



```
Name                                          Value     Address
-(struct tag_s *) ps                          0x1       000410
   +(struct tag_s ) *(ps)                               000001
 (signed short) global_short                  0         000412
 (signed long) global_long                    -1        000414
-(signed int [5][5]) global_array             15        000418
   -(signed int [5]) (global_array)[0]        15        000418
       (signed int) ((global_array)[0])[0]    15        000418
       (signed int) ((global_array)[0])[1]    2649      00041A
       (signed int) ((global_array)[0])[2]    0         00041C
       (signed int) ((global_array)[0])[3]    2649      00041E
       (signed int) ((global_array)[0])[4]    0         000420
   +(signed int [5]) (global_array)[1]        768       000422
   +(signed int [5]) (global_array)[2]        3842      00042C
```

- 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.
  - The initial display radix can be changed.
  - Leading-zero suppression is selectable in hexadecimal display.
- 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 reference scope of the variable is selectable from current scope, global scope and each file's scopes.
- 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.
- The RAM monitor can be allocated to the address of specified variable

ATTENTION

- You cannot change the values of the C watch points listed below:
  - 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.

About more information for C variables, please refer to "12.1.3 Get or set C variables"

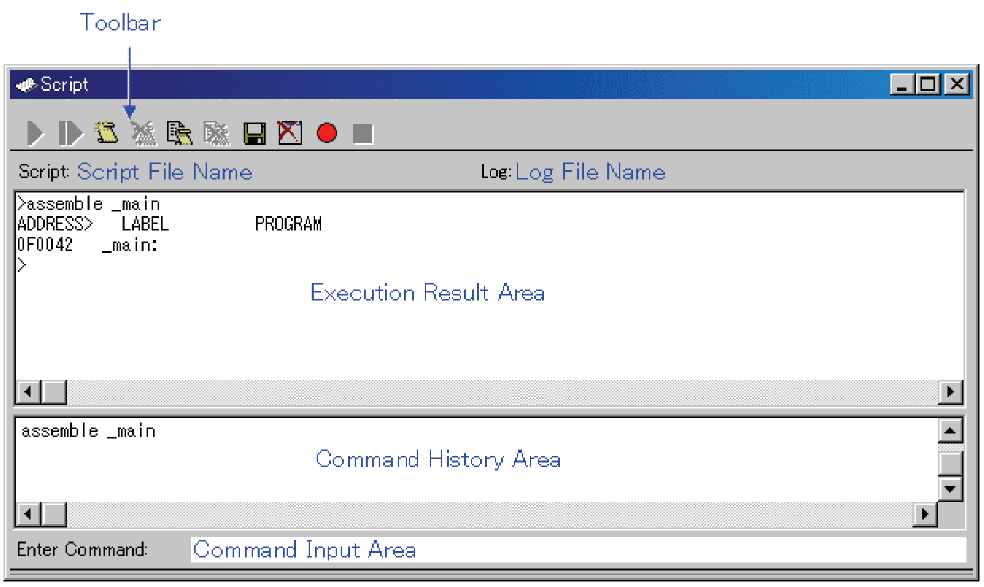
7.3.1 Extended Menus

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).
	Set initial...	Set initial radix.
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.
Zero suppress in Hex display		Suppress zero in Hex display.
Sort	Sort by Name	Sort variables by its name.
	Sort by Address	Sort variables by its address.
RAM Monitor	Enable RAM Monitor	Switch enable or disable RAM monitor function.
	Sampling Period...	Set RAM monitor sampling period.
	Arrange a RAM monitor area around this variable	Arrange a RAM monitor area around this variable.
	Start Recording...	Start to record the updated values.
	Stop Recording	Stop recording the updated values.
Add New Tab...		Add new tab.
Remove Tab		Remove the selected tab.
Copy		Copy the selected item to the clipboard.
Copy All		Copy the all items in the sheet to the clipboard.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

7.4 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.



```
>assemble _main
ADDRESS>   LABEL         PROGRAM
0F0042    _main:
>
```

```
assemble _main
```

- 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.

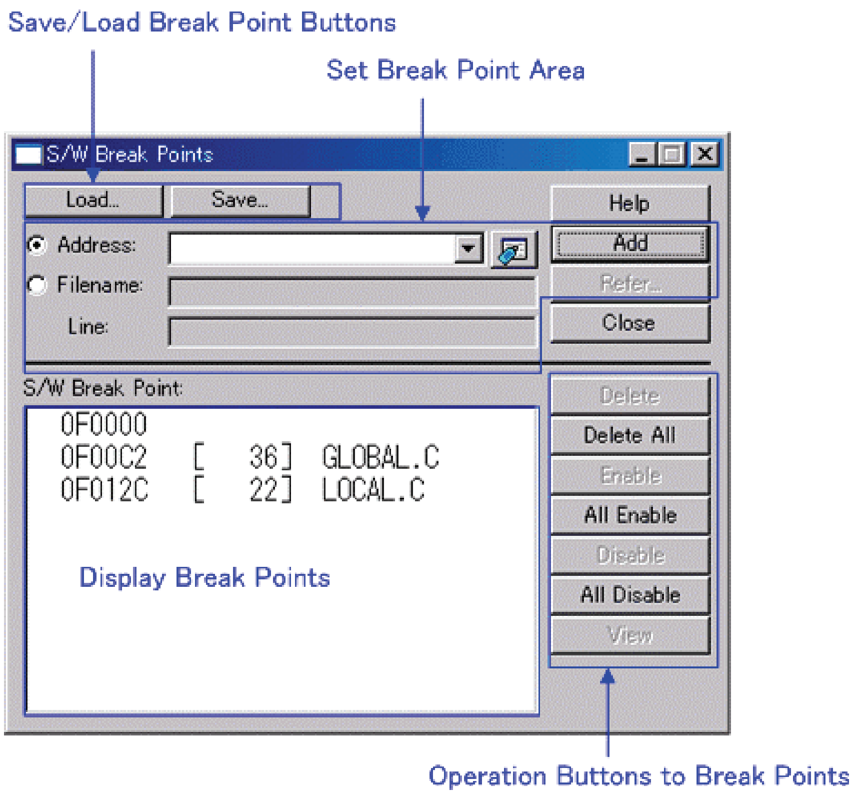
7.4.1 Extended Menus

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.

7.5 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.



```
0F0000
0F00C2    [   36]  GLOBAL.C
0F012C    [   22]  LOCAL.C
```

- 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.

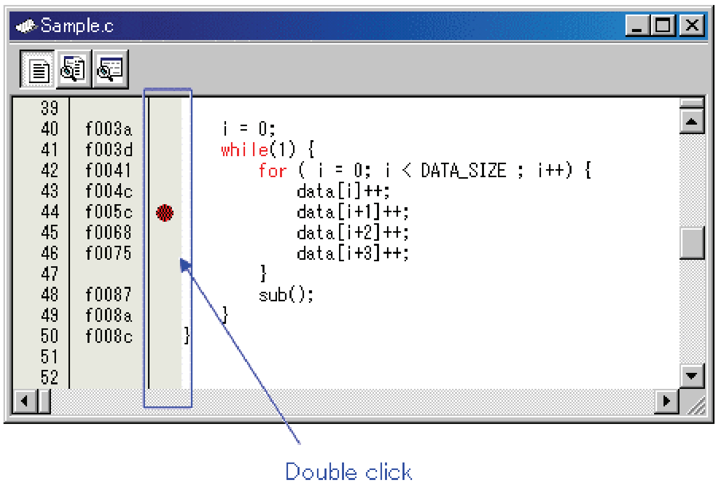
7.5.1 Command Button

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.

7.5.2 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 "12.1.2 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.)



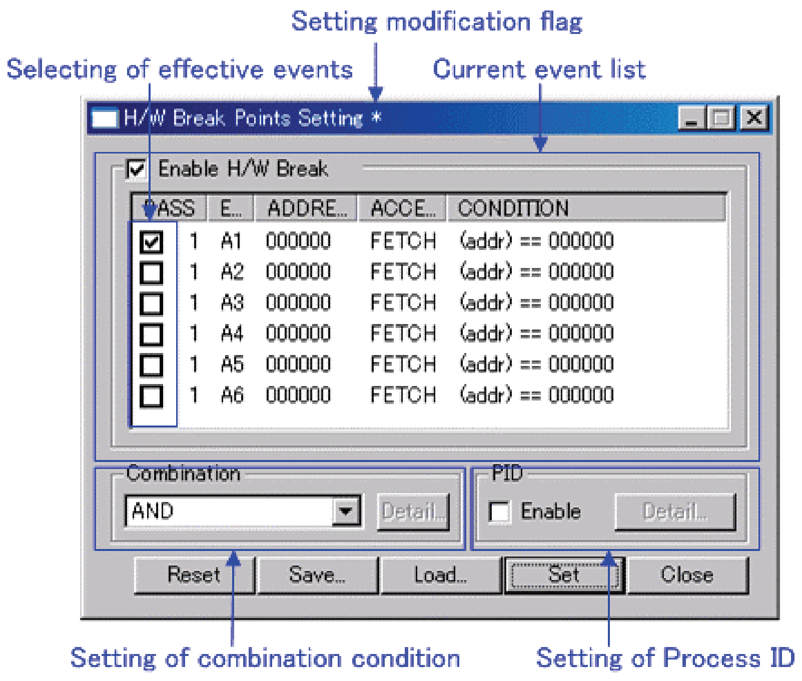
```
f003a          i = 0;
f003d          while(1) {
f0041              for ( i = 0; i < DATA_SIZE ; i++) {
f004c                  data[i]++;
f005c  ●               data[i+1]++;
f0068                  data[i+2]++;
f0075                  data[i+3]++;
                   }
f0087              sub();
f008a          }
f008c      }
```

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.

7.6 H/W Break Point Setting Window

The H/W Breakpoint Setting window is used to set hardware breakpoints for the Emulators.



PASS	E...	ADDRE...	ACCE...	CONDITION
☑ 1	A1	000000	FETCH	(addr) == 000000
☐ 1	A2	000000	FETCH	(addr) == 000000
☐ 1	A3	000000	FETCH	(addr) == 000000
☐ 1	A4	000000	FETCH	(addr) == 000000
☐ 1	A5	000000	FETCH	(addr) == 000000
☐ 1	A6	000000	FETCH	(addr) == 000000

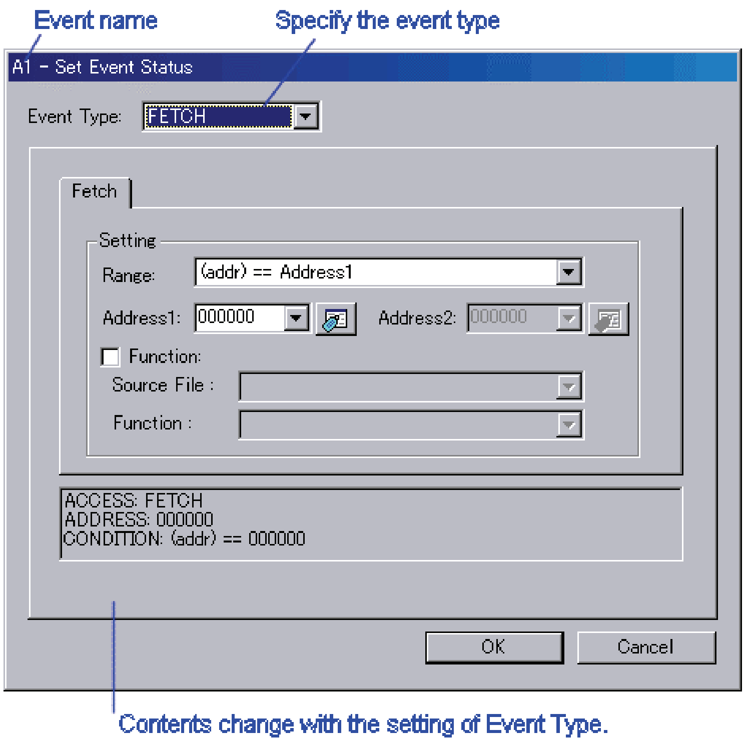
- The events listed below can be specified as break events. If the contents of events are altered, they are marked by an asterisk (*) on the title bar. The asterisks (*) are not displayed after setting up the emulator.

  Fetch, Memory Access, Bit Access

- Events at up to two points can be used.
  The H/W Break Point Setting window and the Trace Point Setting windows use the same resource of the emulator. Use the MCU tab in the Init dialog box, in order to specify for which function the resources are used. On this tab, deselect the Enable the Trace Point Function check box.
- These events can be combined in one of the following ways:
  - Break when all of the valid events are established (AND condition)
  - Break when all of the valid events are established at the same time (simultaneous AND condition)
  - Break when one of the valid events is established (OR condition)
- At the time the debugger starts up, the hardware breaks have no effect.

7.6.1 Specify the Break Event

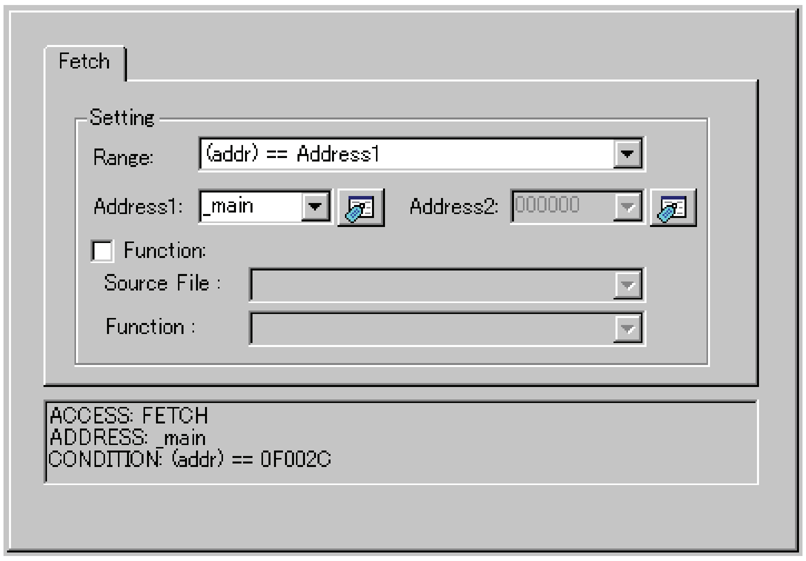
To set events, double-click to select the event you want to set from the event setting area of the H/W Break Point Setting Window. This opens the dialog box shown below.



ACCESS: FETCH
ADDRESS: 000000
CONDITION: (addr) == 000000

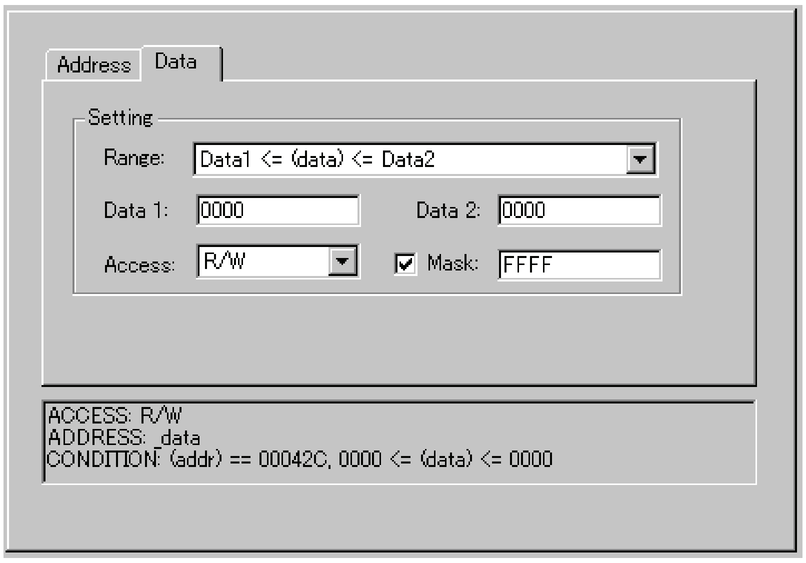
Following events can be set by specifying Event Type in this dialog box.

- When FETCH is selected
  Breaks for the instruction fetch.



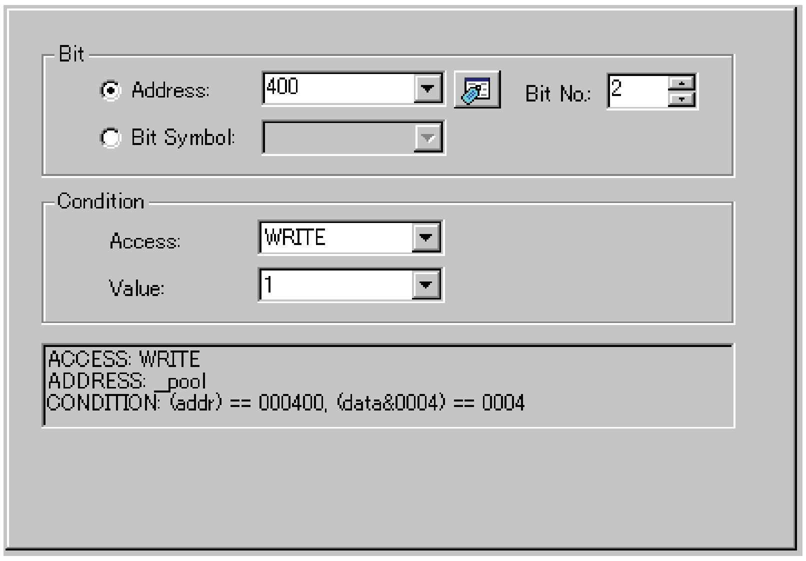
ACCESS: FETCH
ADDRESS: _main
CONDITION: (addr) == 0F002C

- When DATA ACCESS is selected
  Breaks for the memory access.



ACCESS: R/W
ADDRESS: _data
CONDITION: (addr) == 00042C, 0000 <= (data) <= 0000

- When BIT SYMBOL is selected
  Breaks for the bit access.



Address: 400

Bit No.: 2

Access: WRITE

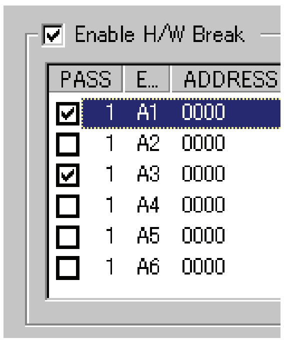
Value: 1

```
ACCESS: WRITE
ADDRESS: _pool
CONDITION: (addr) == 000400, (data&0004) == 0004
```

7.6.2 Specify the Combinatorial Condition

To specify a combinatorial condition, specify the desired condition from the combinatorial condition specification area.

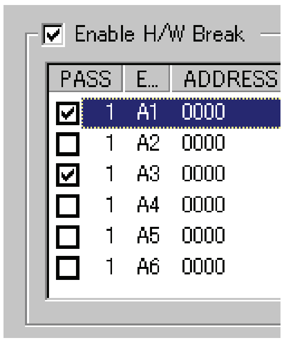
- When AND or OR is selected
  In the event specification area, the event used and a pass count for that event can be specified. To alter the pass count, while the event to alter is being selected, click the pass count value of that event.



☑ Enable H/W Break

PASS	E...	ADDRESS
☑ 1	A1	0000
☐ 1	A2	0000
☑ 1	A3	0000
☐ 1	A4	0000
☐ 1	A5	0000
☐ 1	A6	0000

- When AND (Same Time) is selected
  In the event specification area, the event used can be specified. No pass counts can be specified.



☑ Enable H/W Break

PASS	E...	ADDRESS
☑ 1	A1	0000
☐ 1	A2	0000
☑ 1	A3	0000
☐ 1	A4	0000
☐ 1	A5	0000
☐ 1	A6	0000

7.6.3 Command Button

The buttons on this window has the following meanings.

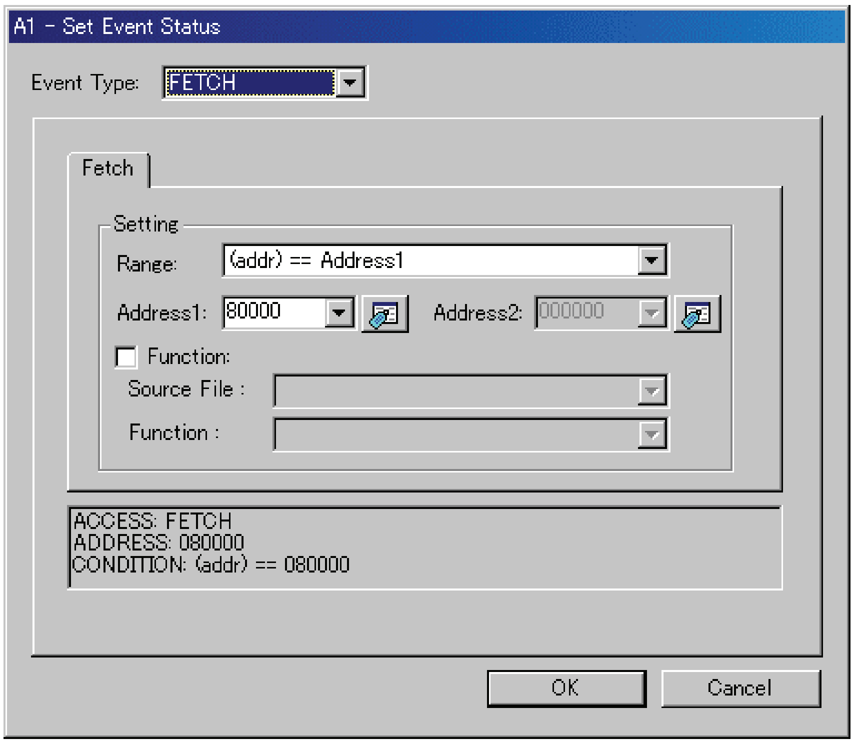
Button	Function
Reset	Discards the contents being displayed in the window and loads contents from the emulator in which they were set.
Save...	Saves the contents set in the window to a file.
Load...	Loads event information from a file in which it was saved.
Set	Sends the contents set in the window to the emulator.
Close	Closes the window.

7.6.4 Specify the Events (Instruction Fetch)

To specify an instruction fetch event, change the event select dialog box's Event Type to "FETCH". The event is established when instruction is fetched from the specified address or any address in the specified address range.

7.6.4.1 Instruction Fetch of Specified Address

Set as below.
Example) Instruction fetch at address 80000h


A1 - Set Event Status

Event Type: FETCH

Fetch

Setting

Range: (addr) == Address1

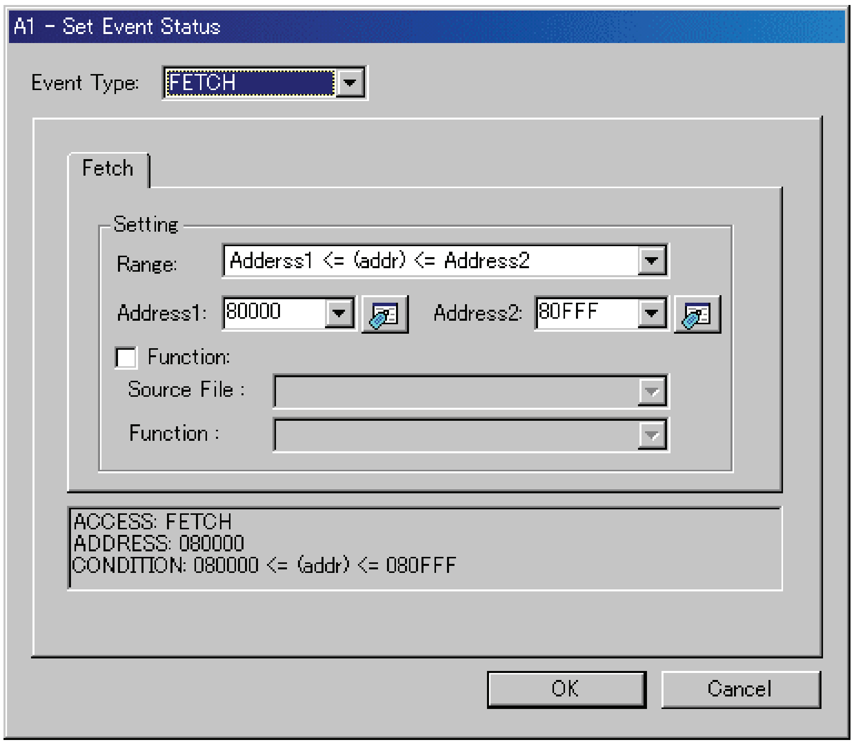
Address1: 80000 Address2: 000000

Function:
Source File :
Function :

```
ACCESS: FETCH
ADDRESS: 080000
CONDITION: (addr) == 080000
```

7.6.4.2 Instruction Fetch of Specified Address Area(In)

Set as below.
Example) Instruction fetch at address 80000h to 80FFFh



A1 - Set Event Status

Event Type: FETCH

Fetch

Setting

Range: Adderss1 <= (addr) <= Address2

Address1: 80000 Address2: 80FFF

Function:

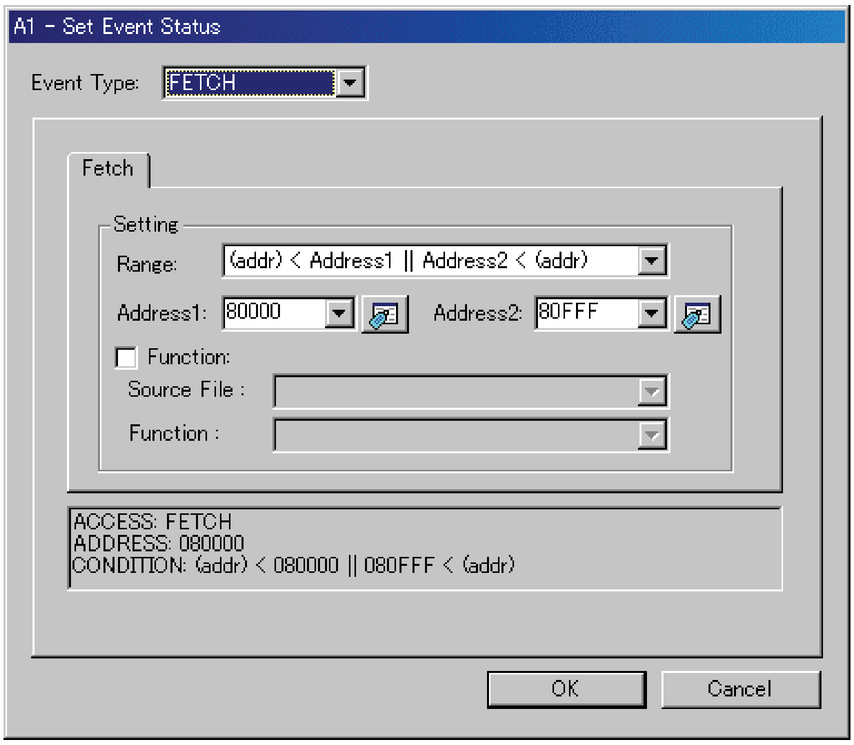
Source File :

Function :

ACCESS: FETCH
ADDRESS: 080000
CONDITION: 080000 <= (addr) <= 080FFF

7.6.4.3 Instruction Fetch of Specified Address Area(Out)

Set as below.
Example) Instruction fetch at any address other than the range 80000h to 80FFFh

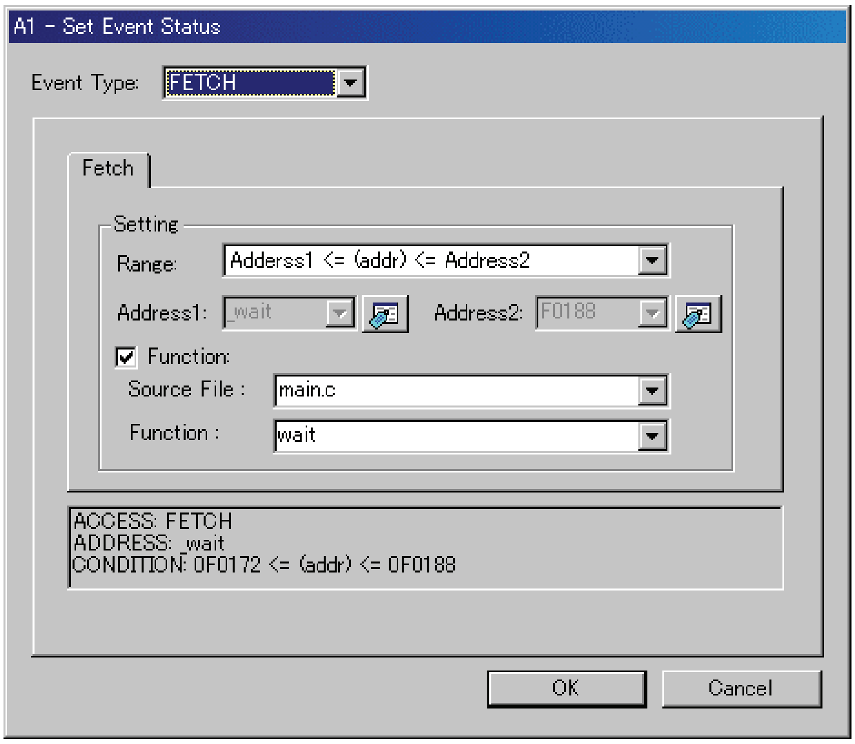


Event Type: FETCH
Range: (addr) < Address1 || Address2 < (addr)
Address1: 80000
Address2: 80FFF
Function:
Source File :
Function :

```
ACCESS: FETCH
ADDRESS: 080000
CONDITION: (addr) < 080000 || 080FFF < (addr)
```

7.6.4.4 Entering/exiting to specified function

Set as below.
Example) Entering a break to function name "wait"



A1 - Set Event Status

Event Type: FETCH

Fetch

Setting

Range: Adderss1 <= (addr) <= Address2

Address1: _wait   Address2: F0188

☑ Function:

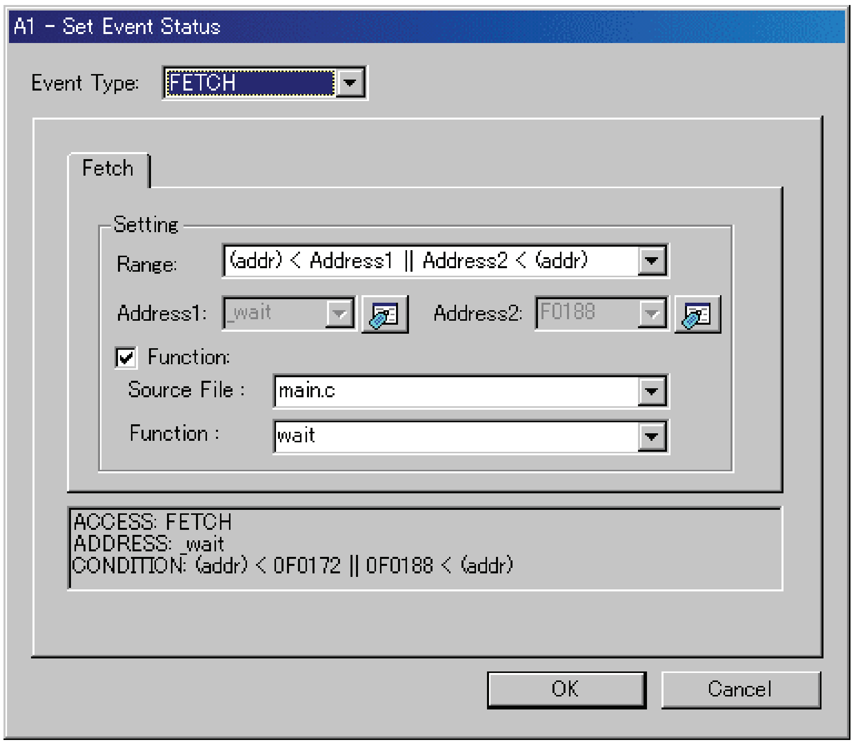
Source File : main.c

Function : wait

```
ACCESS: FETCH
ADDRESS: _wait
CONDITION: 0F0172 <= (addr) <= 0F0188
```

OK   Cancel

Example) Exiting from function name "wait"



A1 - Set Event Status

Event Type: FETCH

Fetch

Setting

Range: (addr) < Address1 || Address2 < (addr)

Address1: _wait   Address2: F0188

☑ Function:

Source File : main.c

Function : wait

```
ACCESS: FETCH
ADDRESS: _wait
CONDITION: (addr) < 0F0172 || 0F0188 < (addr)
```

OK   Cancel

7.6.5 Specify the Events (Memory Access)

To specify a memory access event, change the event select dialog box's Event Type to "DATA ACCESS". The event is established when memory is accessed at the specified address or under conditions set for the specified address range.

7.6.5.1 Memory Access(The debugger for M32C)

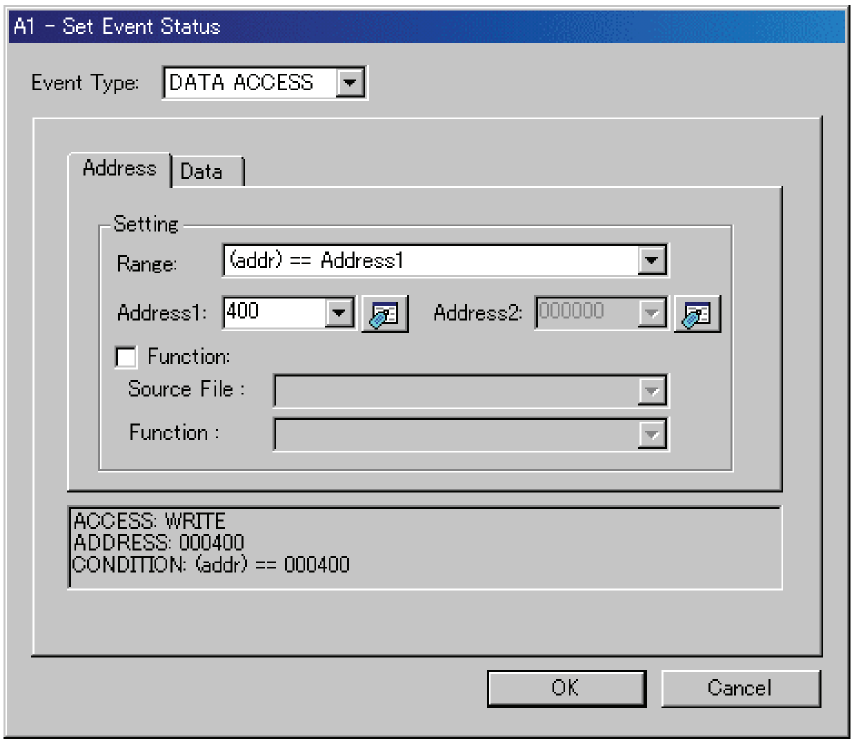
ATTENTION

It is not detected that data access to the odd addresses in word-size access.

7.6.5.1.1. Writing/Reading a Specified Address

Set as below.

Example) Writing to even address 400h



A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (addr) == Address1

Address1: 400

Address2: 000000

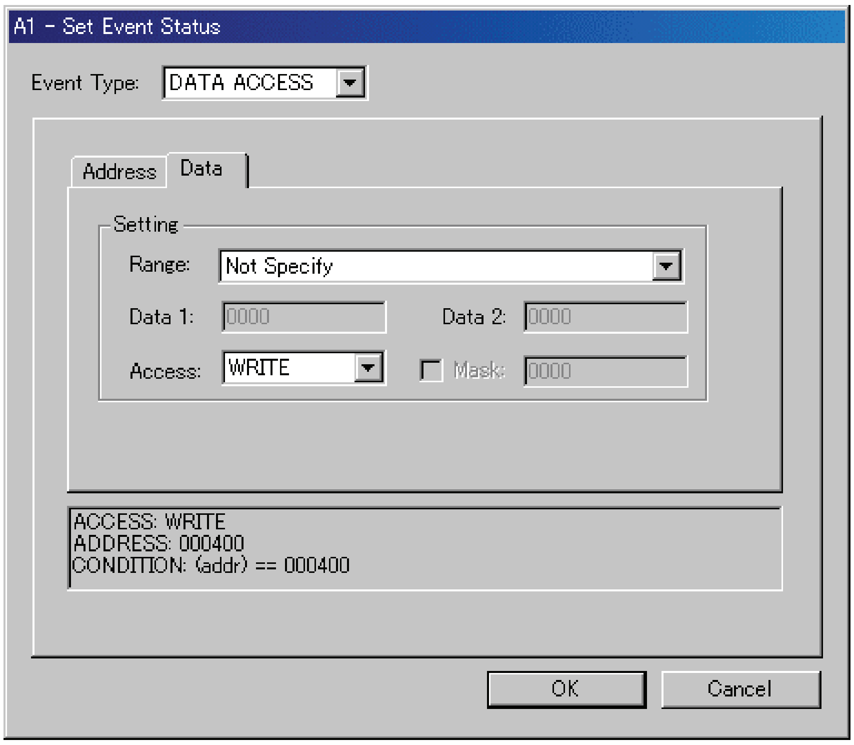
☐ Function:

Source File :

Function :

ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400

OK Cancel



A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: Not Specify

Data 1: 0000

Data 2: 0000

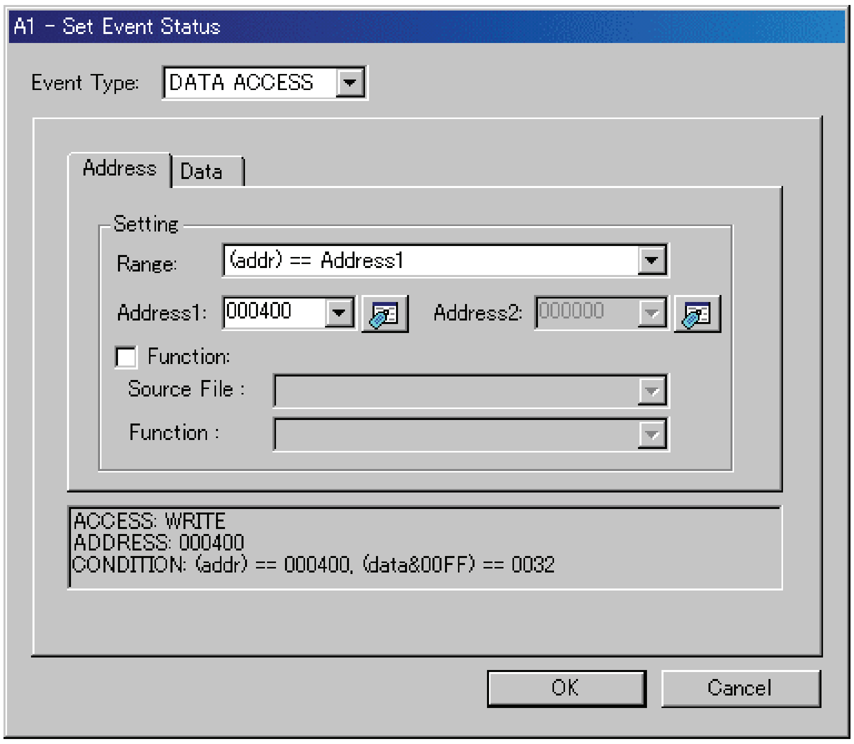
Access: WRITE

☐ Mask: 0000

ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400

OK Cancel

Example) Writing byte length data 32h to even address 400h



Event Type: DATA ACCESS

Range: (addr) == Address1

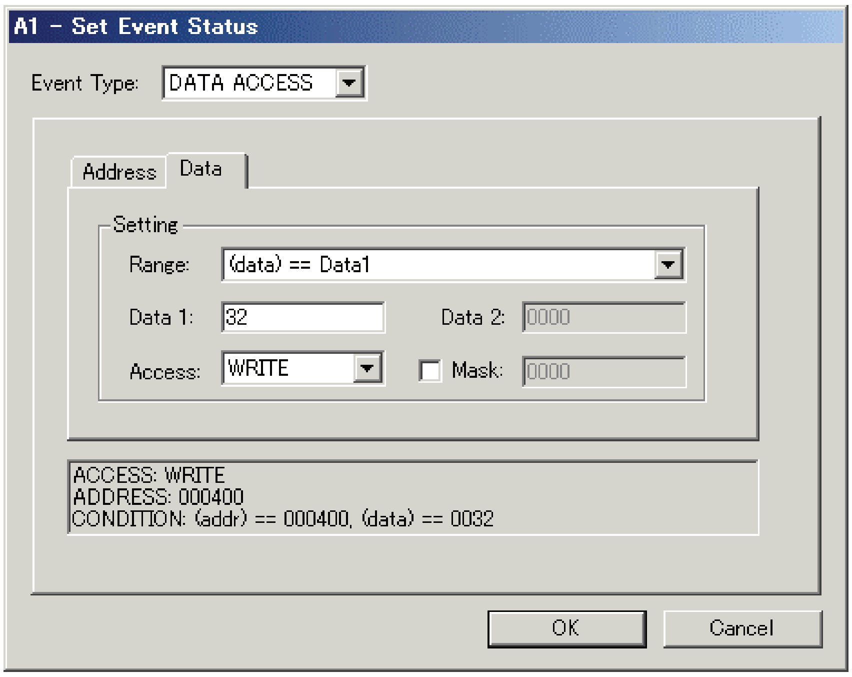
Address1: 000400

Address2: 000000

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0032
```



Event Type: DATA ACCESS

Range: (data) == Data1

Data 1: 32

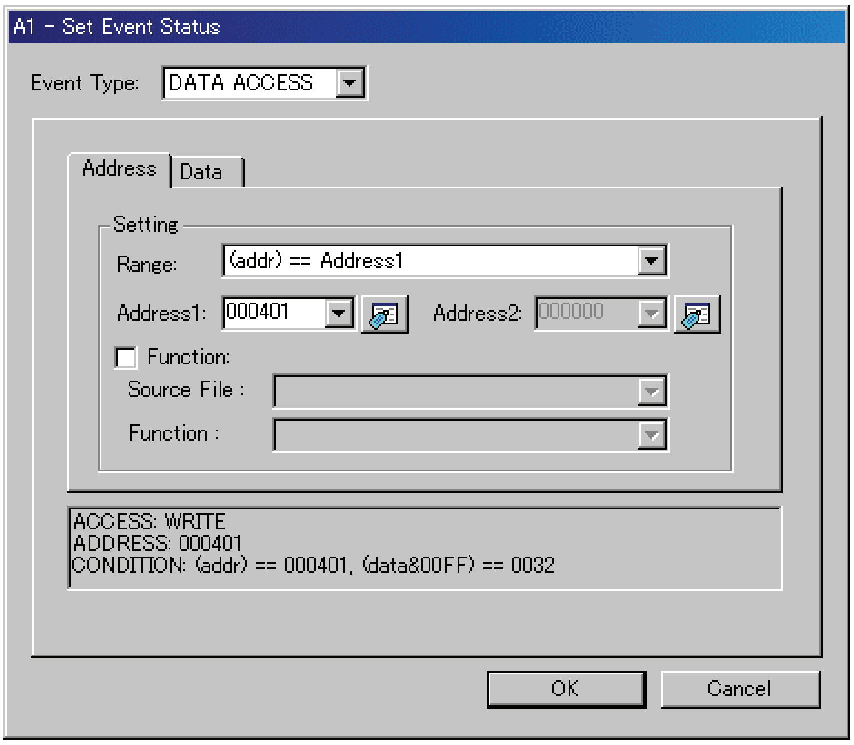
Data 2: 0000

Access: WRITE

Mask: 0000

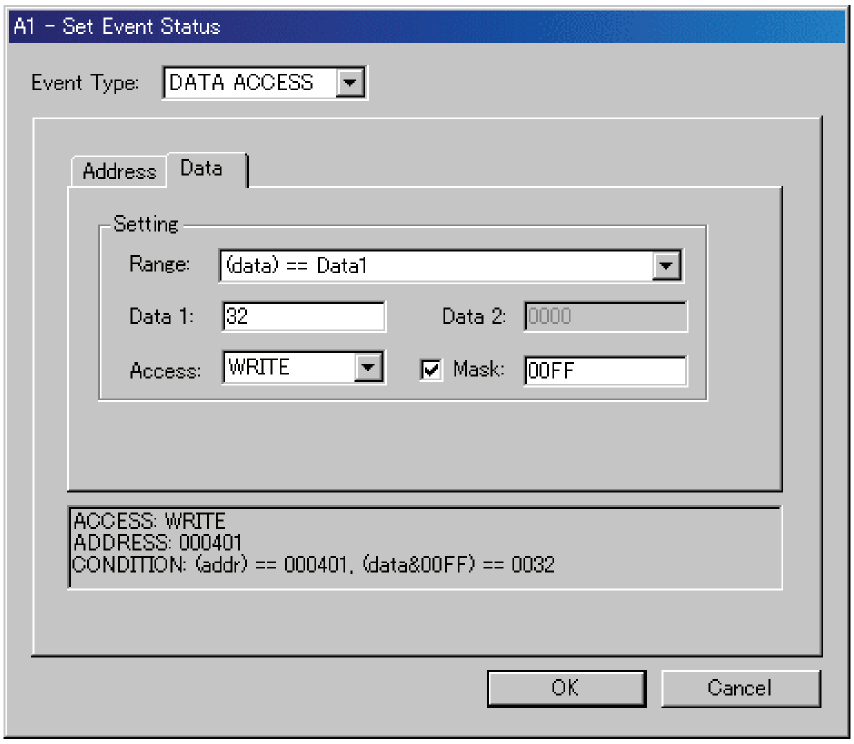
```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 0032
```

Example) Writing byte length data 32h to odd address 401h
Contents of settings vary with each product and bus width.
(8 bits bus width)



Event Type: DATA ACCESS
Range: (addr) == Address1
Address1: 000401
Address2: 000000
☐ Function:

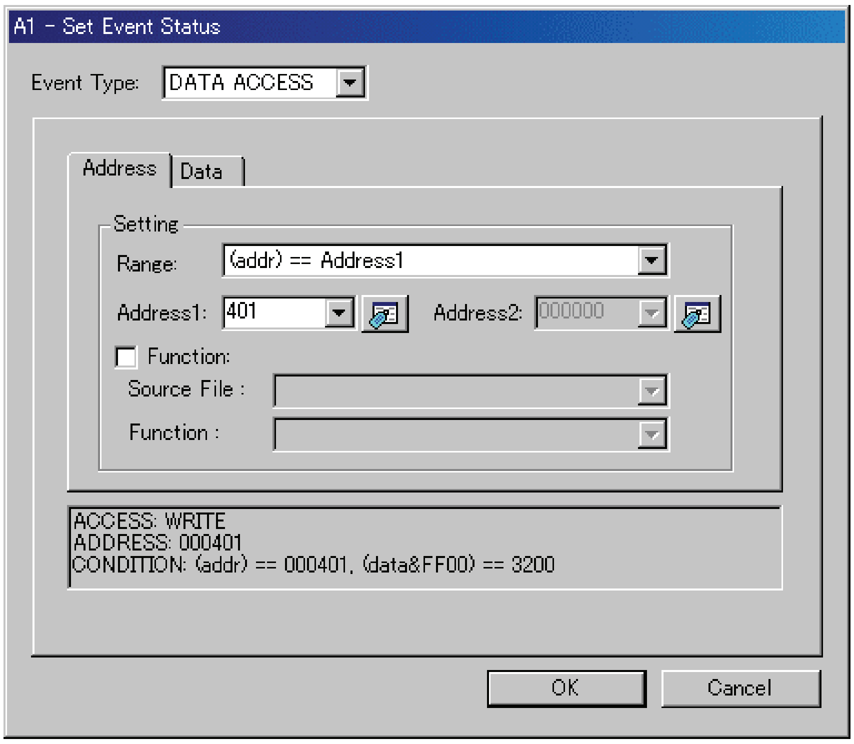
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&00FF) == 0032



Event Type: DATA ACCESS
Range: (data) == Data1
Data 1: 32
Data 2: 0000
Access: WRITE
☑ Mask: 00FF

ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&00FF) == 0032

(16 bits bus width)


A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (addr) == Address1

Address1: 401 Address2: 000000

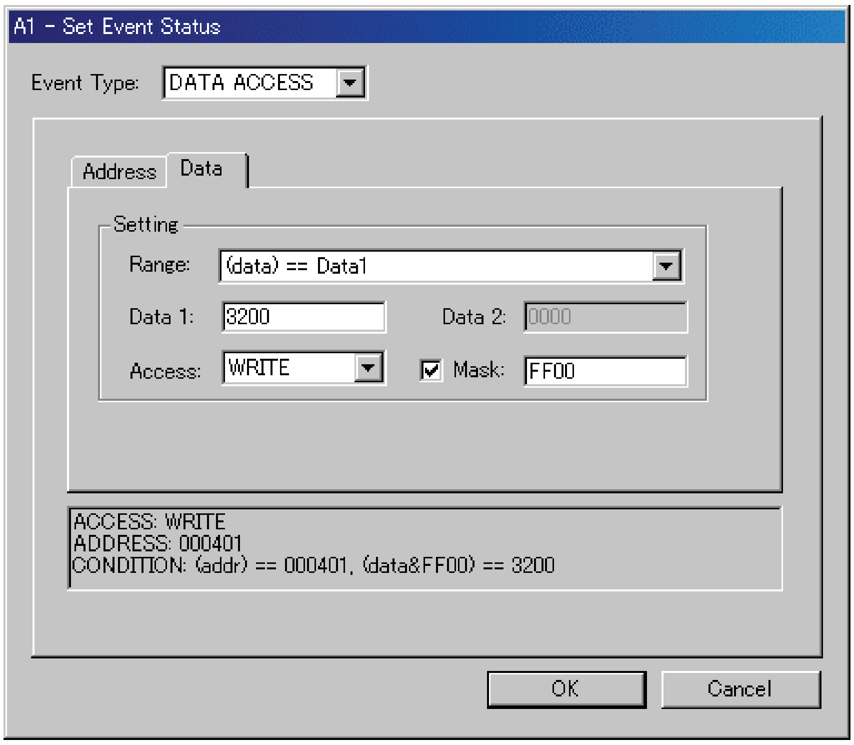
☐ Function:

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
```

OK Cancel


A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (data) == Data1

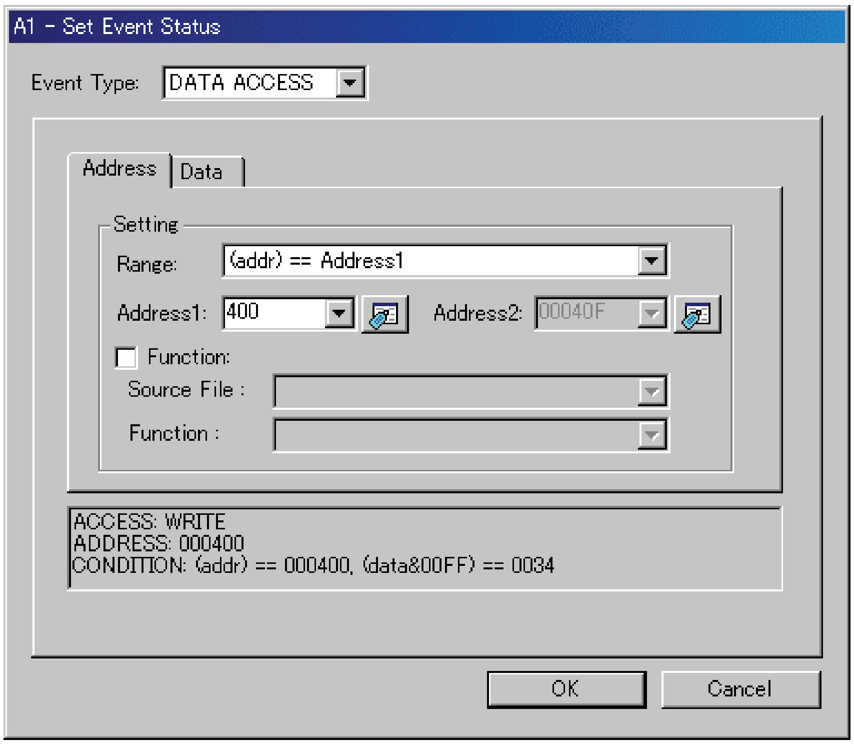
Data 1: 3200 Data 2: 0000

Access: WRITE ☑ Mask: FF00

```
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
```

OK Cancel

Example) Writing word length data 1234h to even address 400h
Contents of settings vary with each product and bus width.
(8 bits bus width)
Please specify "And" of the 1st point and the 2nd point for a combination condition.
1st point

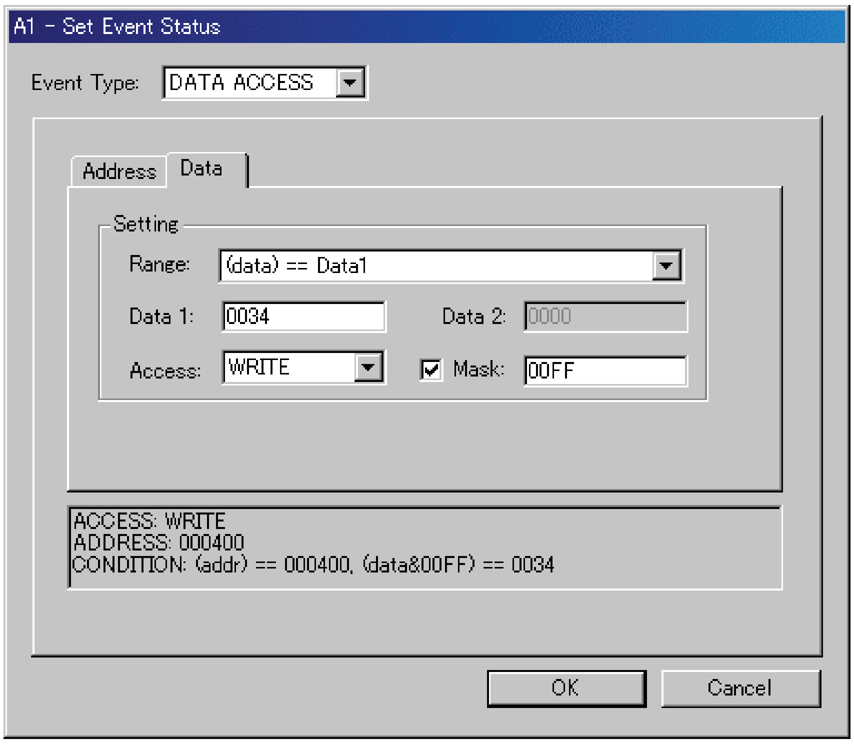

A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting
Range: (addr) == Address1
Address1: 400 Address2: 00040F
☐ Function:
Source File :
Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0034
```


A1 - Set Event Status

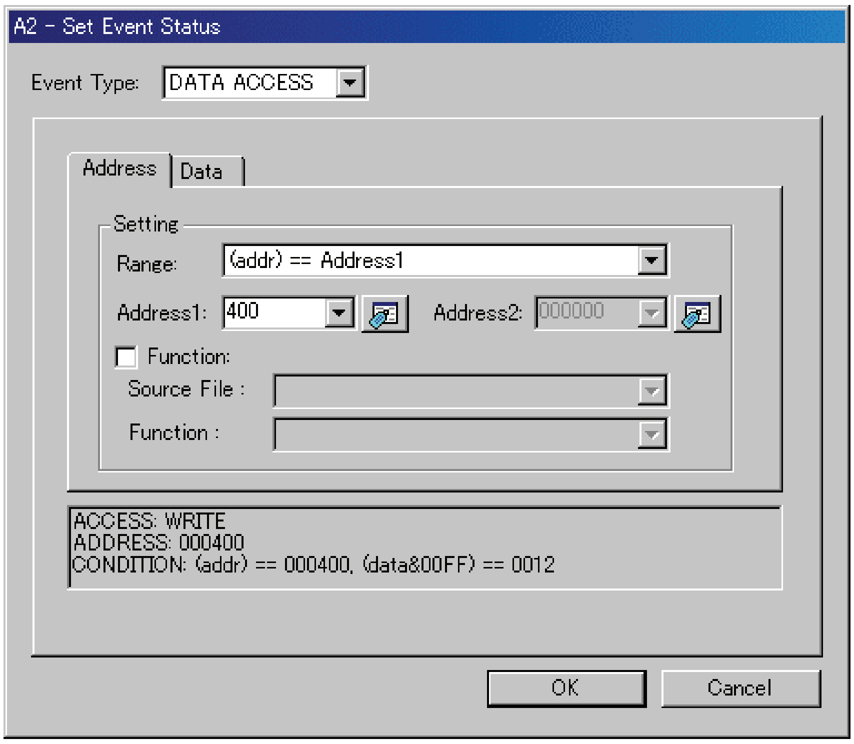
Event Type: DATA ACCESS

Address | Data

Setting
Range: (data) == Data1
Data 1: 0034 Data 2: 0000
Access: WRITE ☑ Mask: 00FF

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0034
```

2nd point



Event Type: DATA ACCESS

Setting

Range: (addr) == Address1

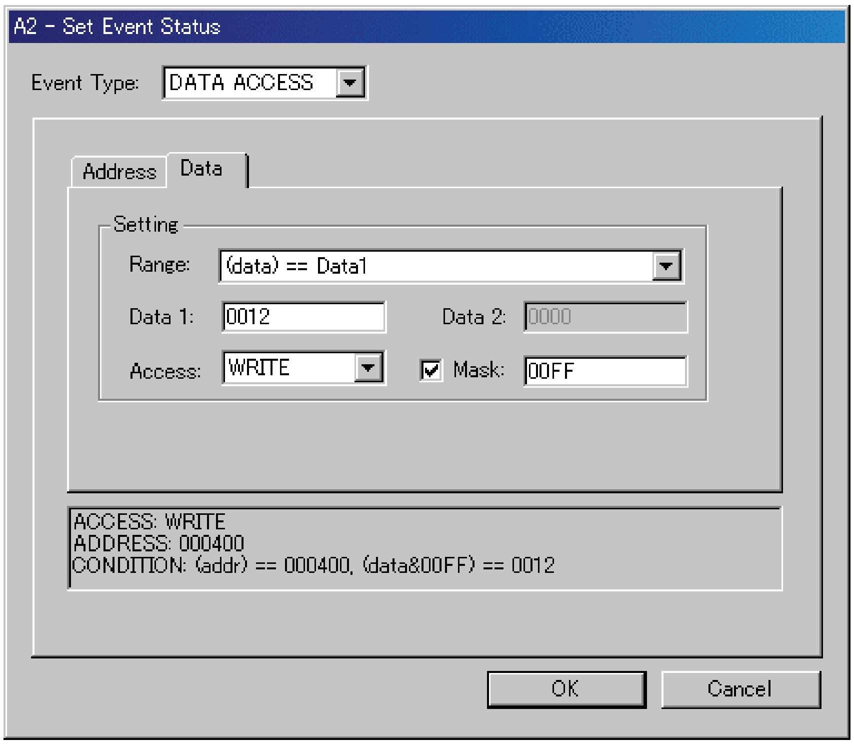
Address1: 400 Address2: 000000

☐ Function:

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
```



Event Type: DATA ACCESS

Setting

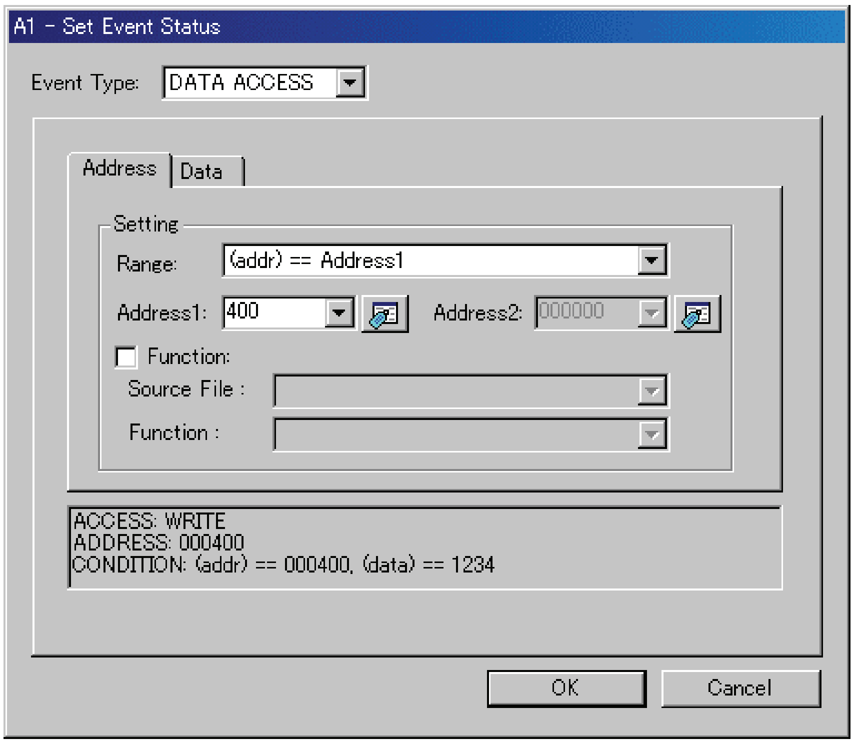
Range: (data) == Data1

Data 1: 0012 Data 2: 0000

Access: WRITE ☑ Mask: 00FF

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
```

(16 bits bus width)


A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (addr) == Address1

Address1: 400

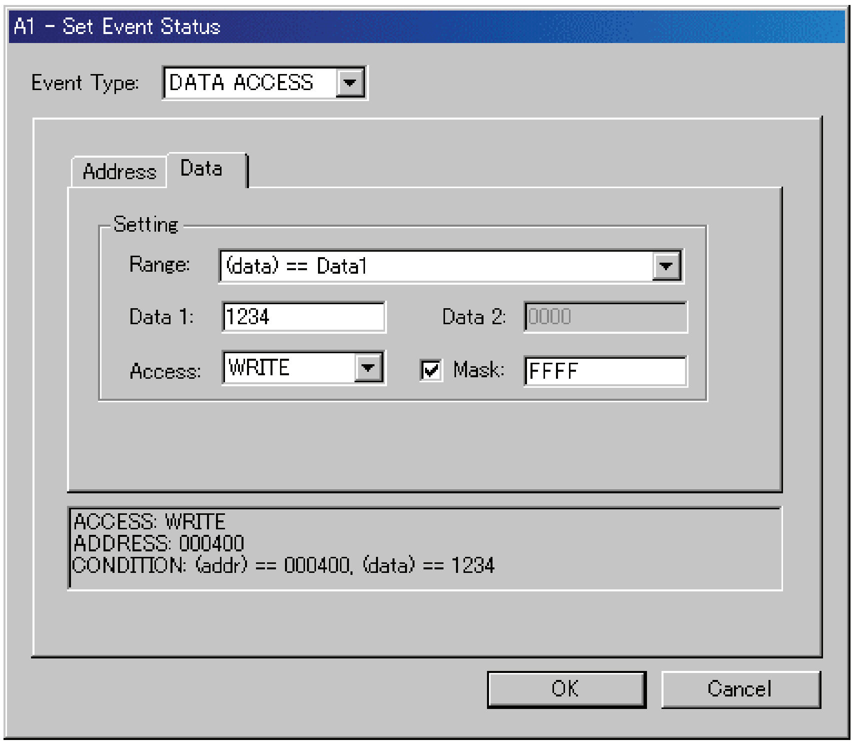
Address2: 000000

☐ Function:

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234
```


A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (data) == Data1

Data 1: 1234

Data 2: 0000

Access: WRITE

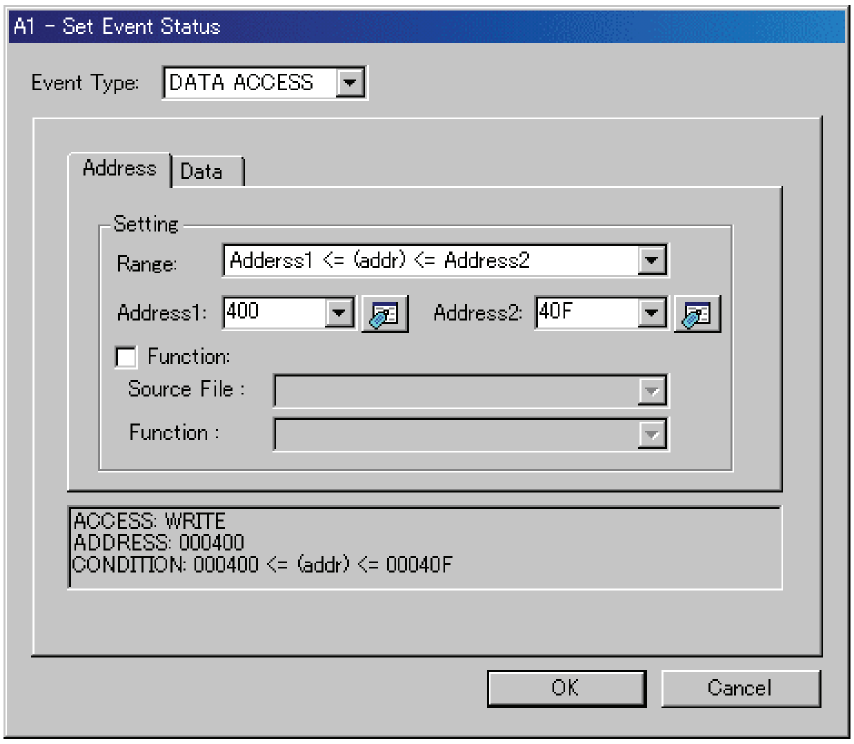
☑ Mask: FFFF

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234
```

7.6.5.1.2. Reading/writing data to the specified address range

Set as below.

Example) Writing data to addresses ranging from 400h to 40Fh


A1 - Set Event Status

Event Type: DATA ACCESS

Range: Aderss1 <= (addr) <= Address2

Address1: 400

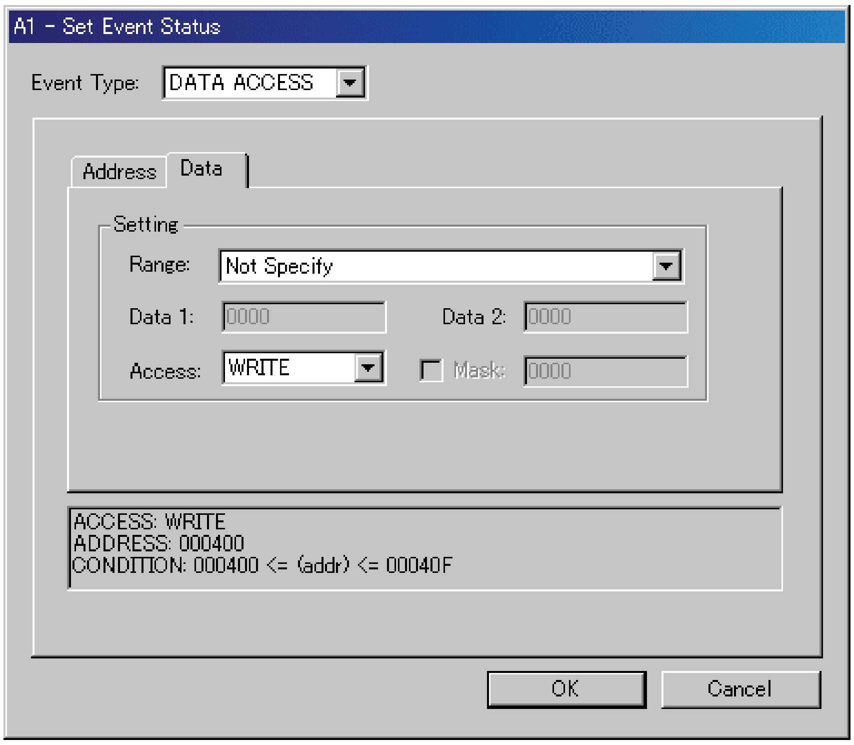
Address2: 40F

Function:

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: 000400 <= (addr) <= 00040F
```


A1 - Set Event Status

Event Type: DATA ACCESS

Range: Not Specify

Data 1: 0000

Data 2: 0000

Access: WRITE

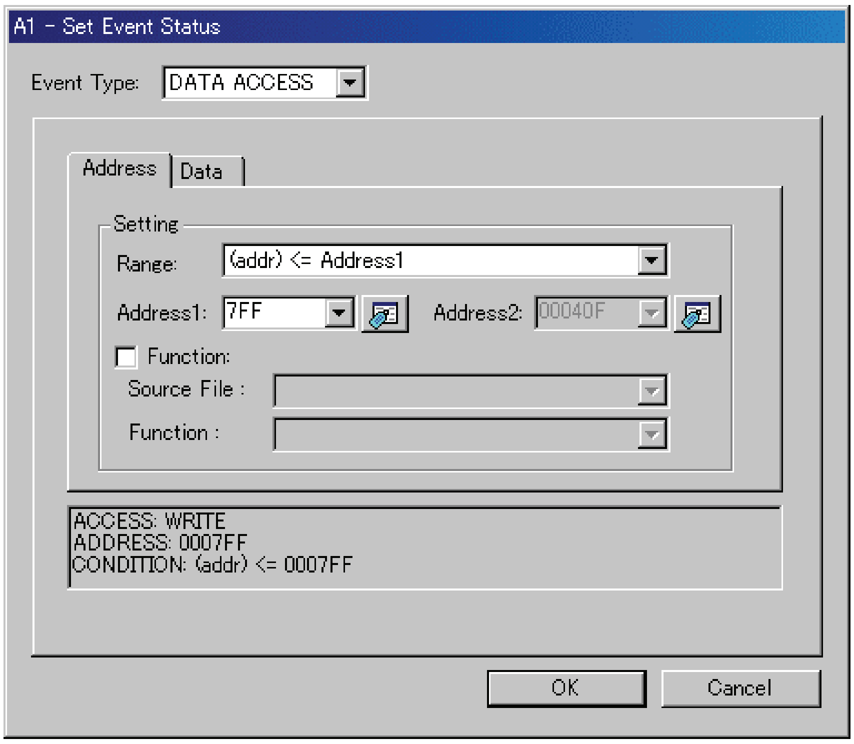
Mask: 0000

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: 000400 <= (addr) <= 00040F
```

7.6.5.1.3. Reading/writing data to addresses outside the specified range

Set as below.

Example) Writing data to addresses below 7FFh


A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (addr) <= Address1

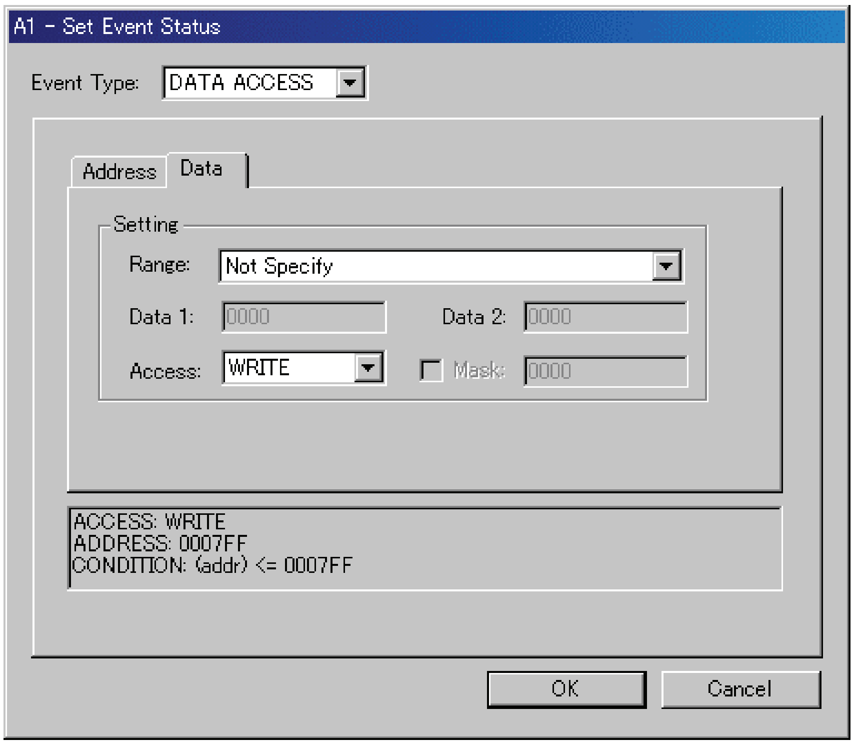
Address1: 7FF   Address2: 00040F

Function:

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 0007FF
CONDITION: (addr) <= 0007FF
```


A1 - Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: Not Specify

Data 1: 0000   Data 2: 0000

Access: WRITE   Mask: 0000

```
ACCESS: WRITE
ADDRESS: 0007FF
CONDITION: (addr) <= 0007FF
```

7.6.5.2 Memory Access(The debugger for M16C/R8C)

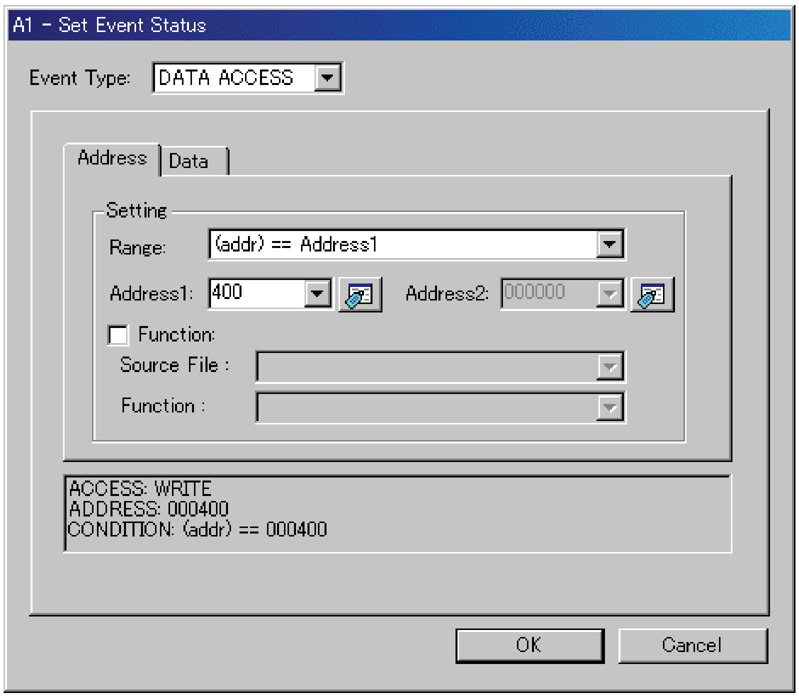
ATTENTION

It is not detected that data access to the odd addresses in word-size access.

7.6.5.2.1. Writing/Reading a Specified Address

Set as below.

Example) Writing to even address 400h



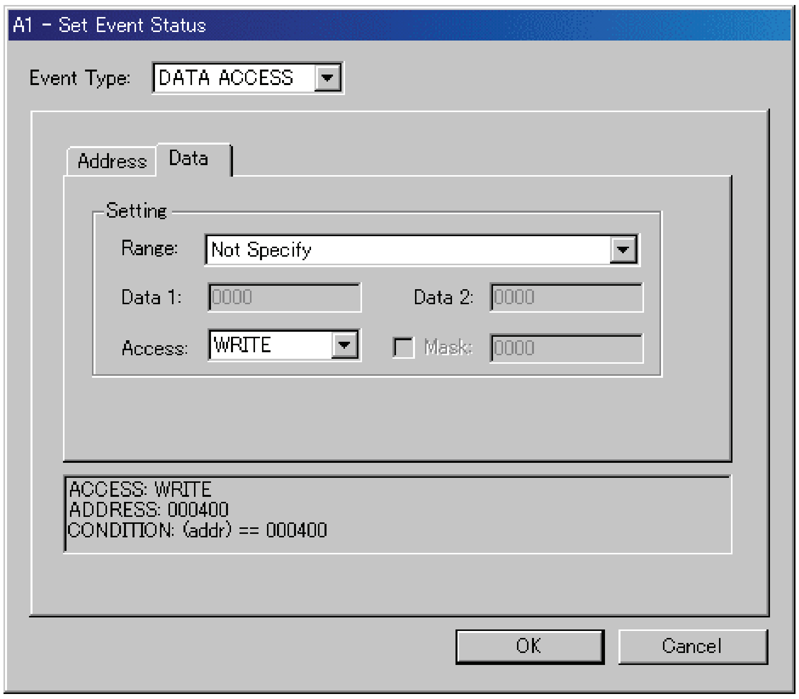
Event Type: DATA ACCESS

Range: (addr) == Address1

Address1: 400 Address2: 000000

Function: ☐

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400
```



Event Type: DATA ACCESS

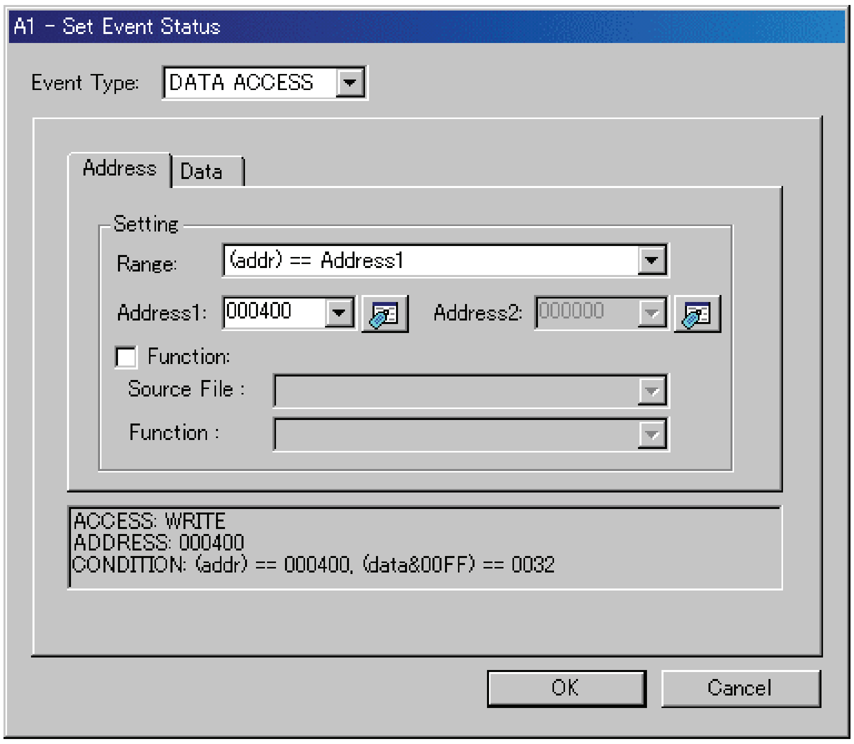
Range: Not Specify

Data 1: 0000 Data 2: 0000

Access: WRITE ☐ Mask: 0000

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400
```

Example) Writing byte length data 32h to even address 400h



Event Type: DATA ACCESS

Range: (addr) == Address1

Address1: 000400

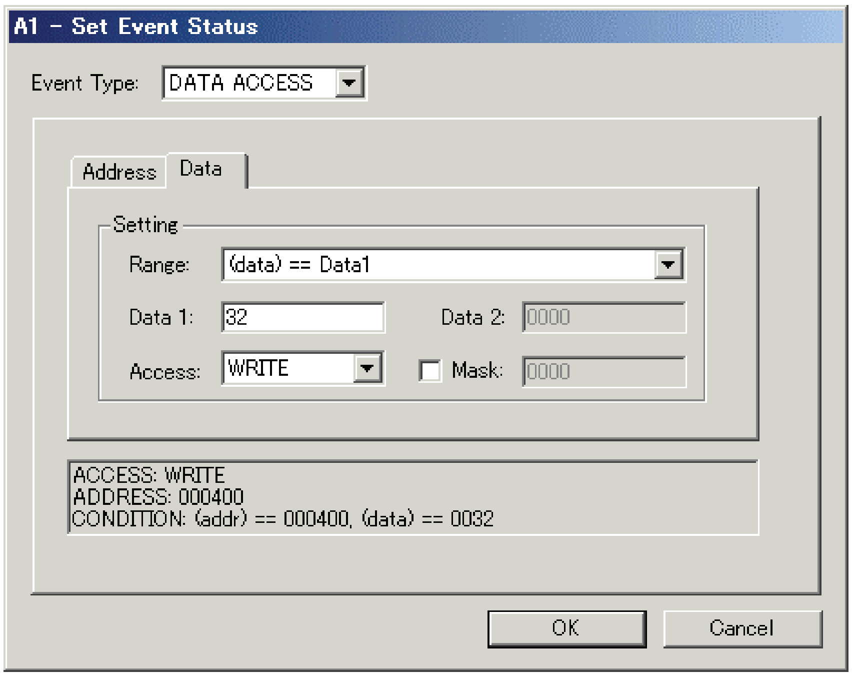
Address2: 000000

Function:

Source File :

Function :

ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0032



Event Type: DATA ACCESS

Range: (data) == Data1

Data 1: 32

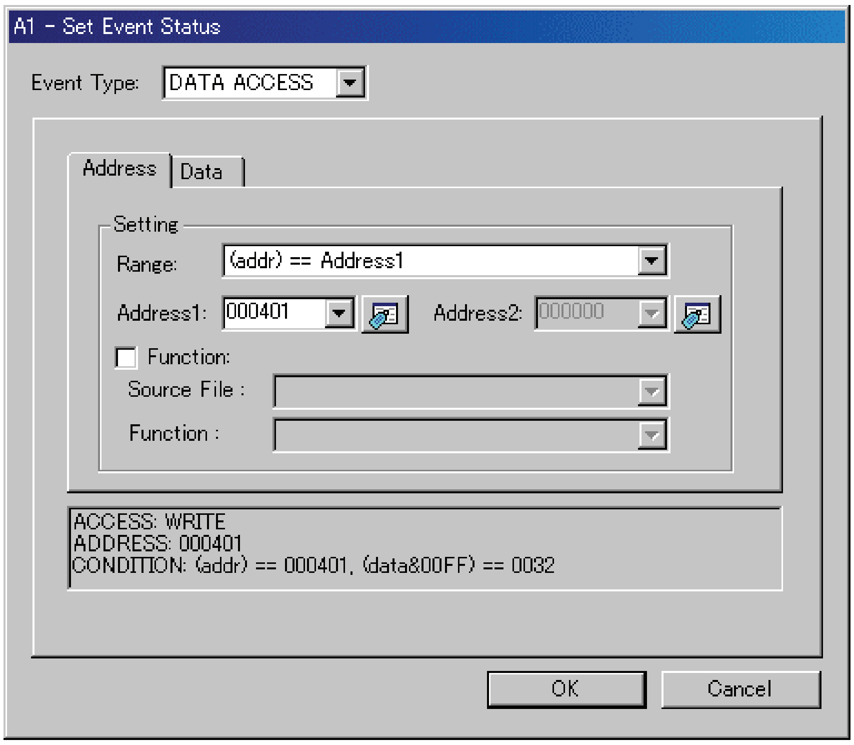
Data 2: 0000

Access: WRITE

Mask: 0000

ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 0032

Example) Writing byte length data 32h to odd address 401h
Contents of settings vary with each product and bus width.
(8 bits bus width)


A1 - Set Event Status

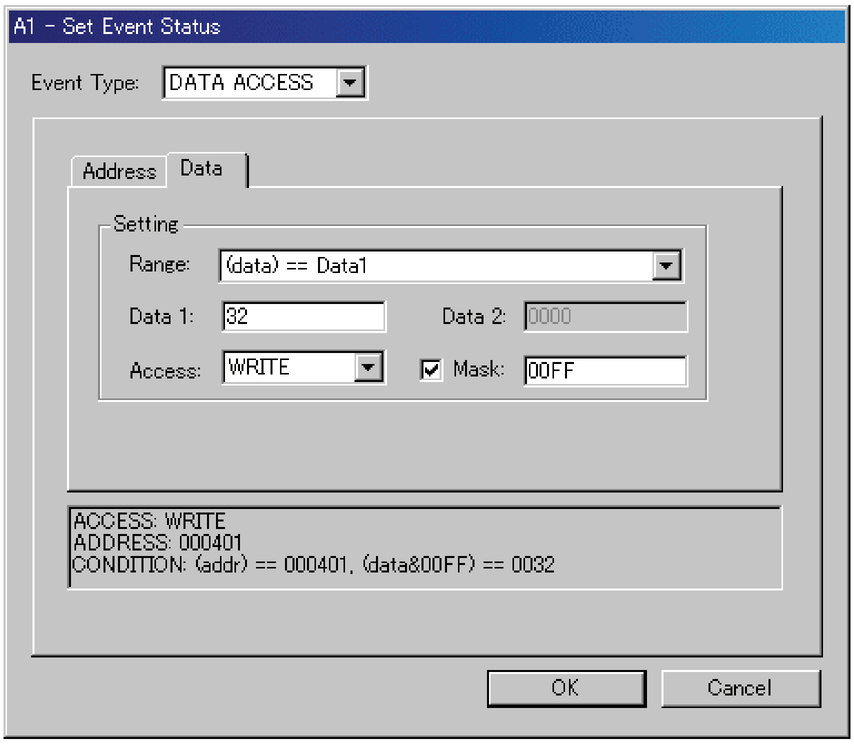
Event Type: DATA ACCESS

Address | Data

Setting
Range: (addr) == Address1
Address1: 000401   Address2: 000000
☐ Function:
Source File :
Function :

```
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&00FF) == 0032
```

OK   Cancel


A1 - Set Event Status

Event Type: DATA ACCESS

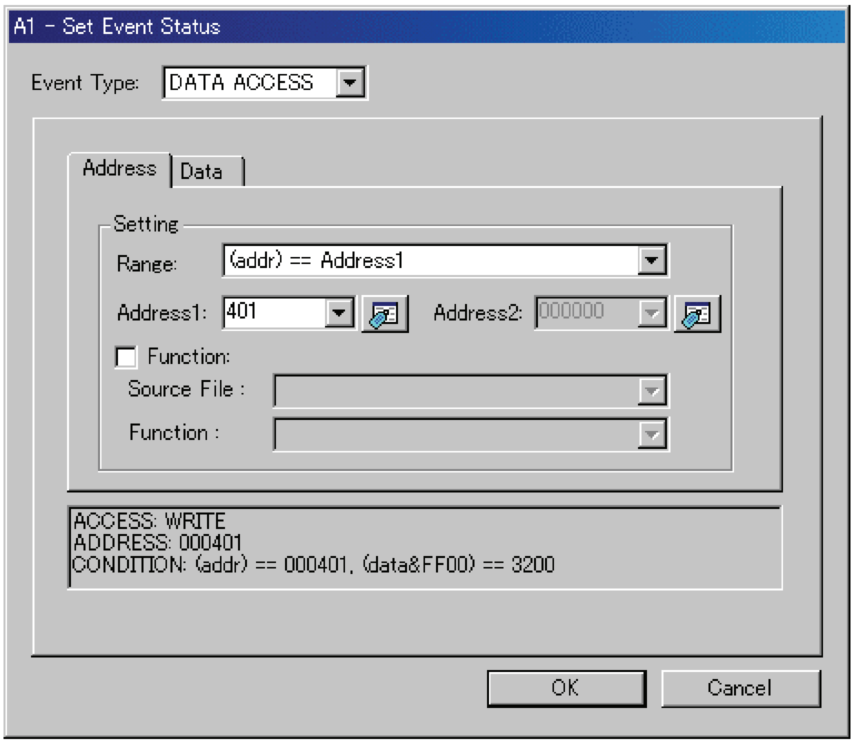
Address | Data

Setting
Range: (data) == Data1
Data 1: 32   Data 2: 0000
Access: WRITE   ☑ Mask: 00FF

```
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&00FF) == 0032
```

OK   Cancel

(16 bits bus width)



Event Type: DATA ACCESS

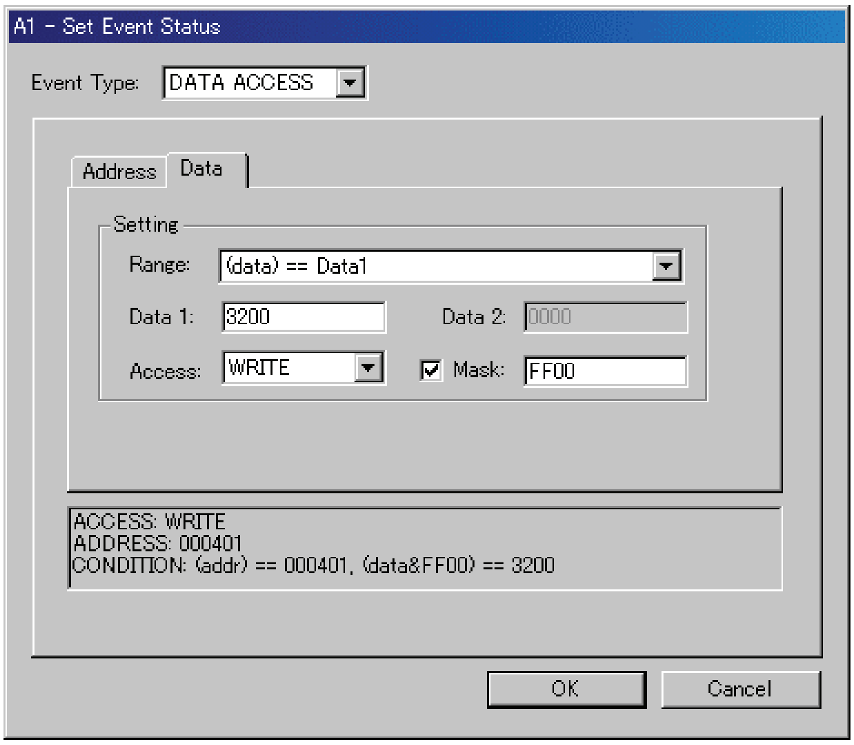
Range: (addr) == Address1

Address1: 401 Address2: 000000

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
```



Event Type: DATA ACCESS

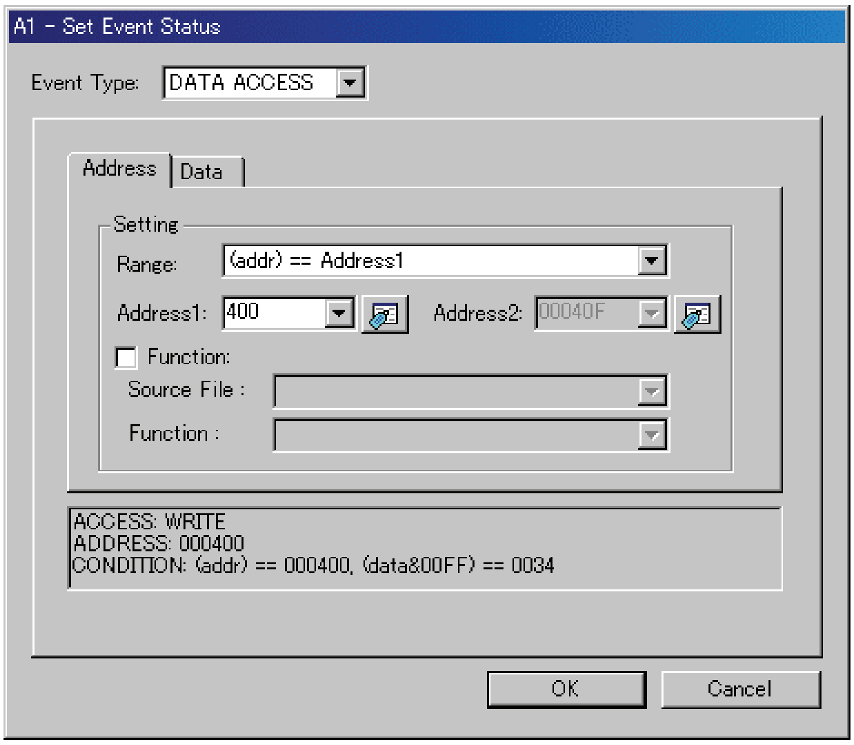
Range: (data) == Data1

Data 1: 3200 Data 2: 0000

Access: WRITE ☑ Mask: FF00

```
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
```

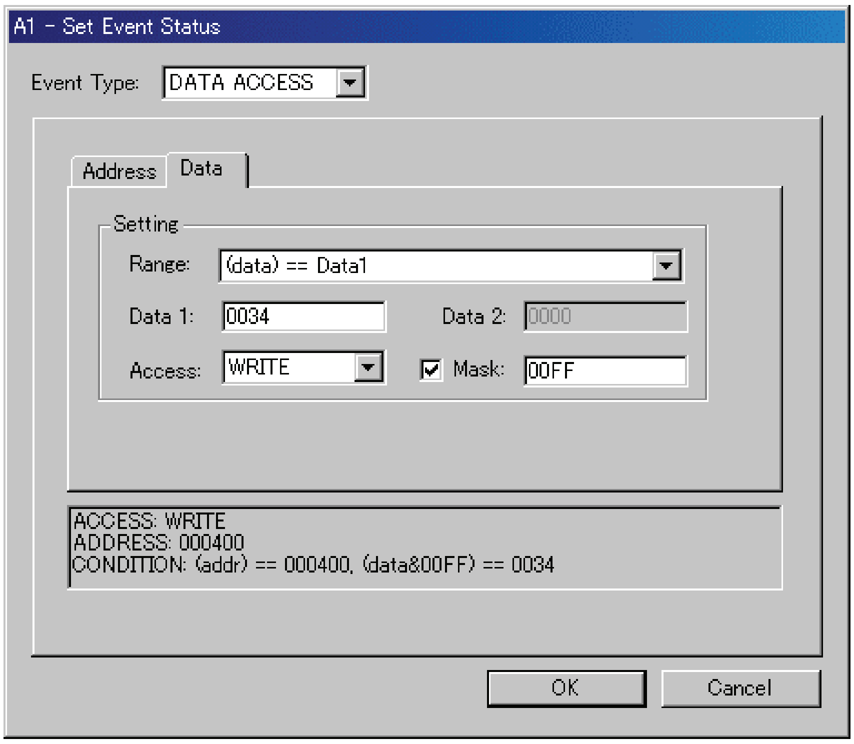
Example) Writing word length data 1234h to even address 400h
Contents of settings vary with each product and bus width.
(8 bits bus width)
Please specify "And" of the 1st point and the 2nd point for a combination condition.
1st point



Event Type: DATA ACCESS

Setting
Range: (addr) == Address1
Address1: 400
Address2: 00040F
☐ Function:
Source File :
Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0034
```

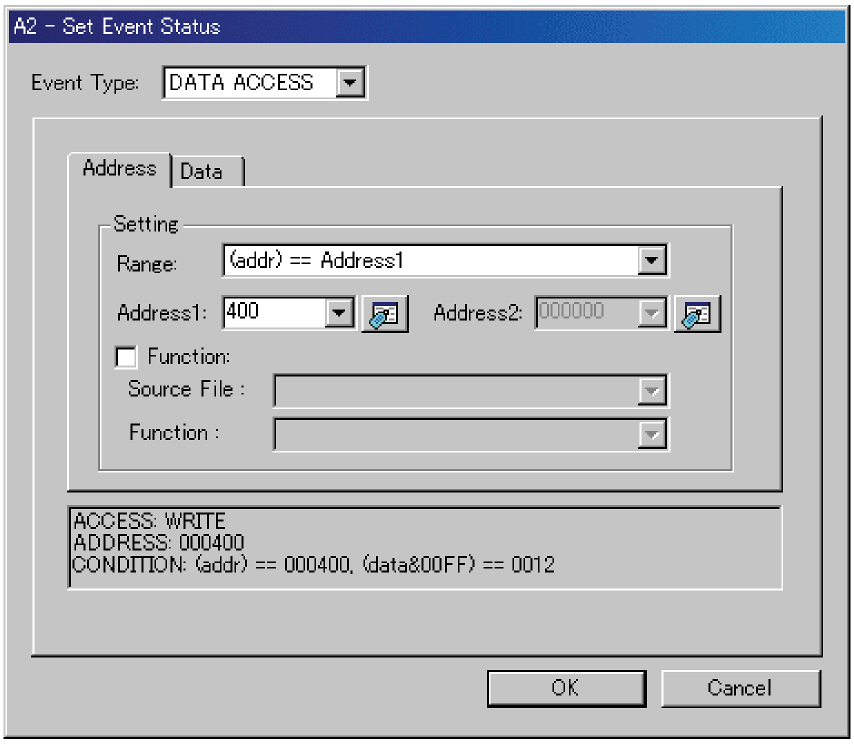


Event Type: DATA ACCESS

Setting
Range: (data) == Data1
Data 1: 0034
Data 2: 0000
Access: WRITE
☑ Mask: 00FF

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0034
```

2nd point


A2 – Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (addr) == Address1

Address1: 400

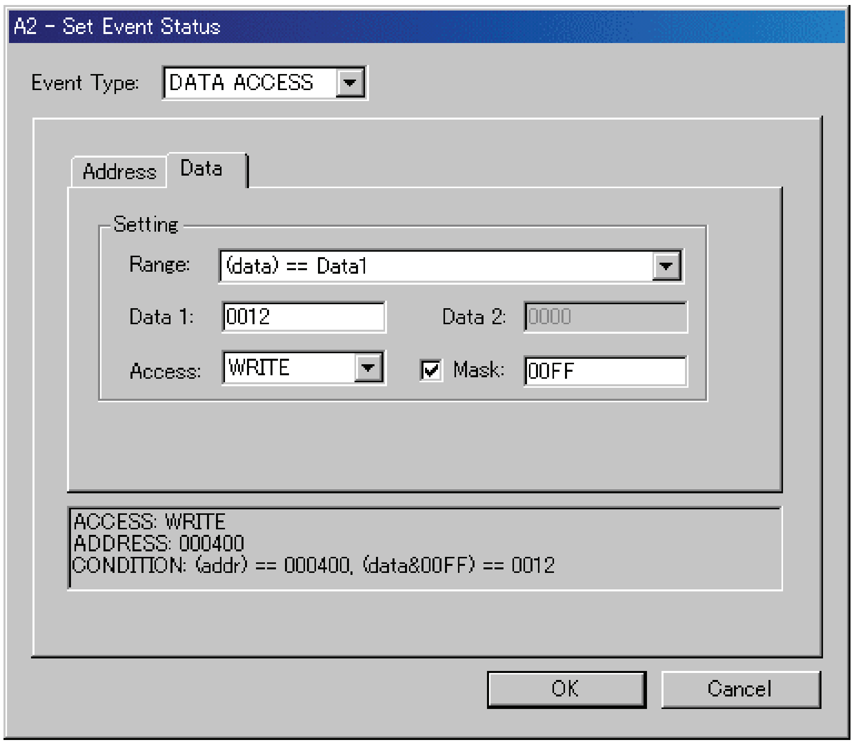
Address2: 000000

Function:

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
```


A2 – Set Event Status

Event Type: DATA ACCESS

Address | Data

Setting

Range: (data) == Data1

Data 1: 0012

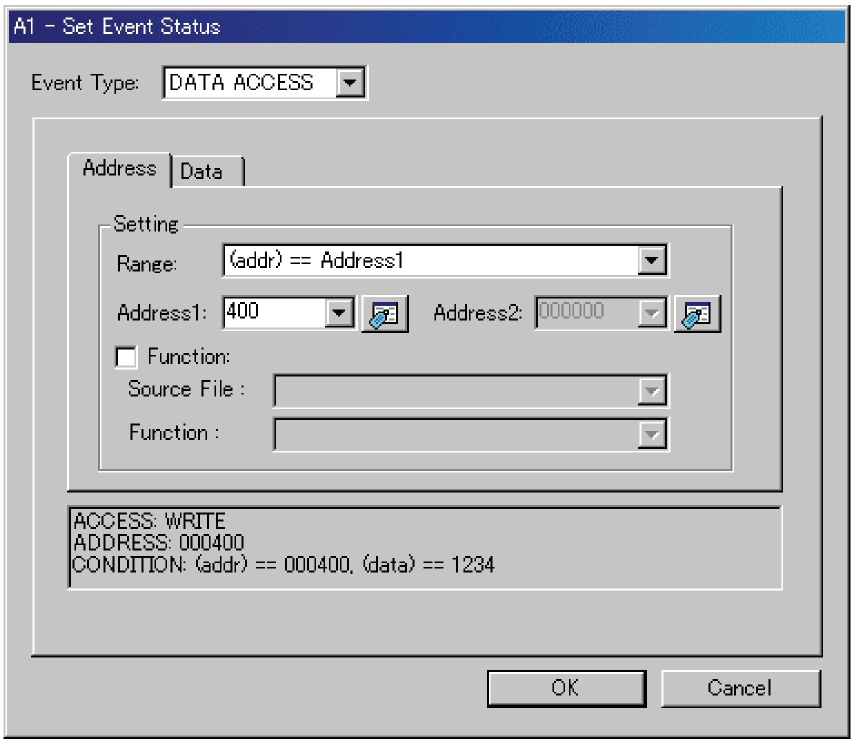
Data 2: 0000

Access: WRITE

[✓ Mask:] 00FF

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
```

(16 bits bus width)



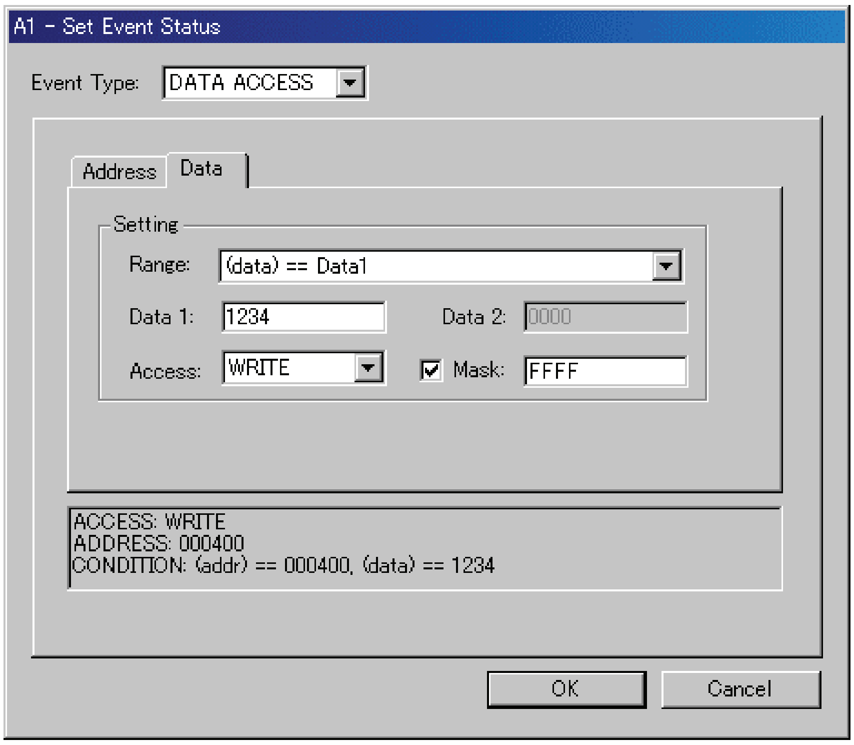
Event Type: DATA ACCESS

Range: (addr) == Address1

Address1: 400

Address2: 000000

ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234



Event Type: DATA ACCESS

Range: (data) == Data1

Data 1: 1234

Data 2: 0000

Access: WRITE

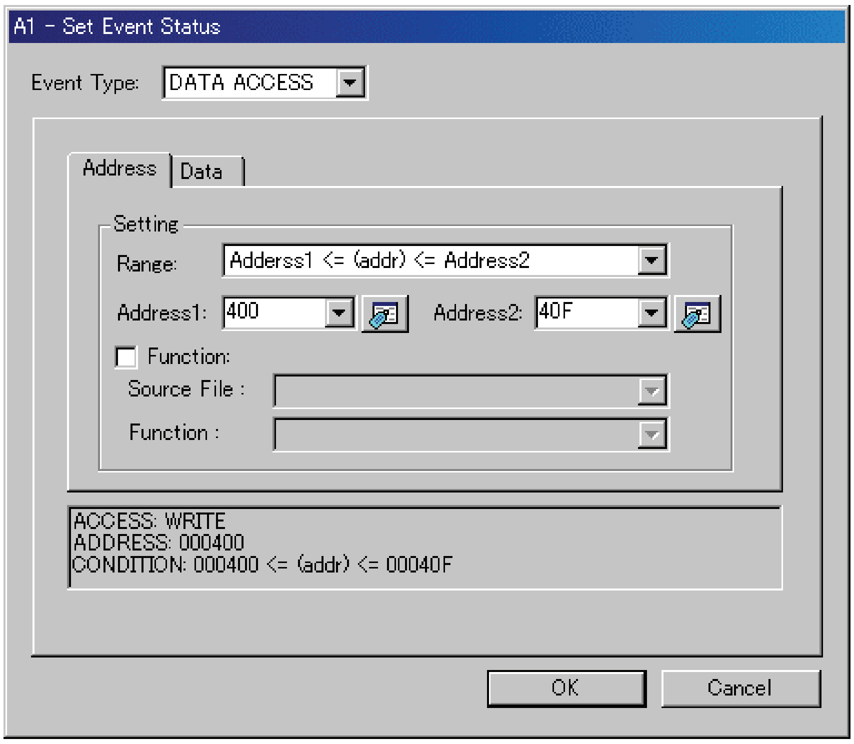
☑ Mask: FFFF

ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234

7.6.5.2.2. Reading/writing data to the specified address range

Set as below.

Example) Writing data to addresses ranging from 400h to 40Fh



Event Type: DATA ACCESS

Range: Aderss1 <= (addr) <= Address2

Address1: 400

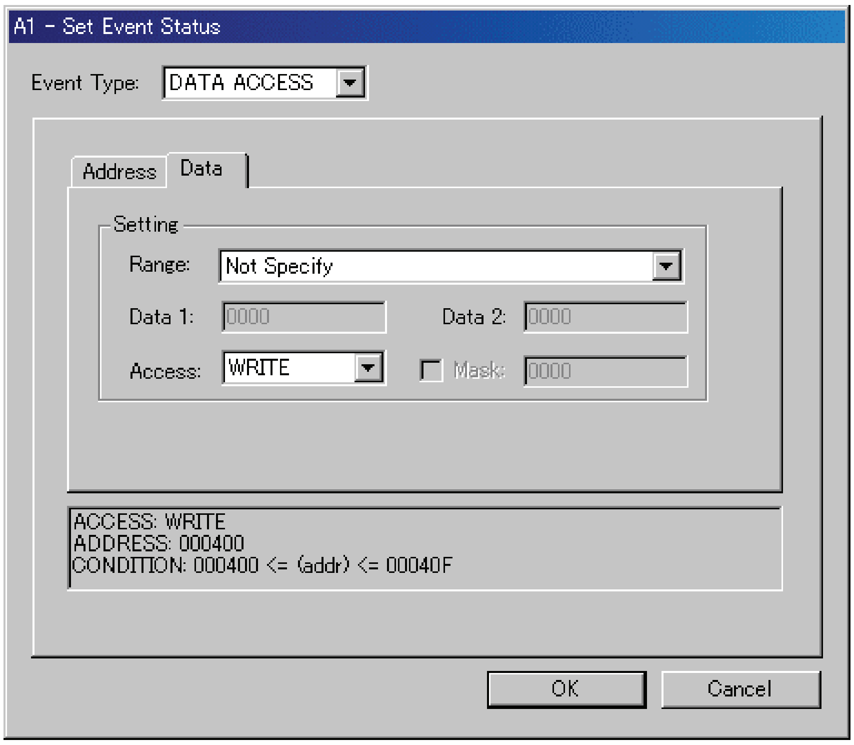
Address2: 40F

Function: (unchecked)

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: 000400 <= (addr) <= 00040F
```



Event Type: DATA ACCESS

Range: Not Specify

Data 1: 0000

Data 2: 0000

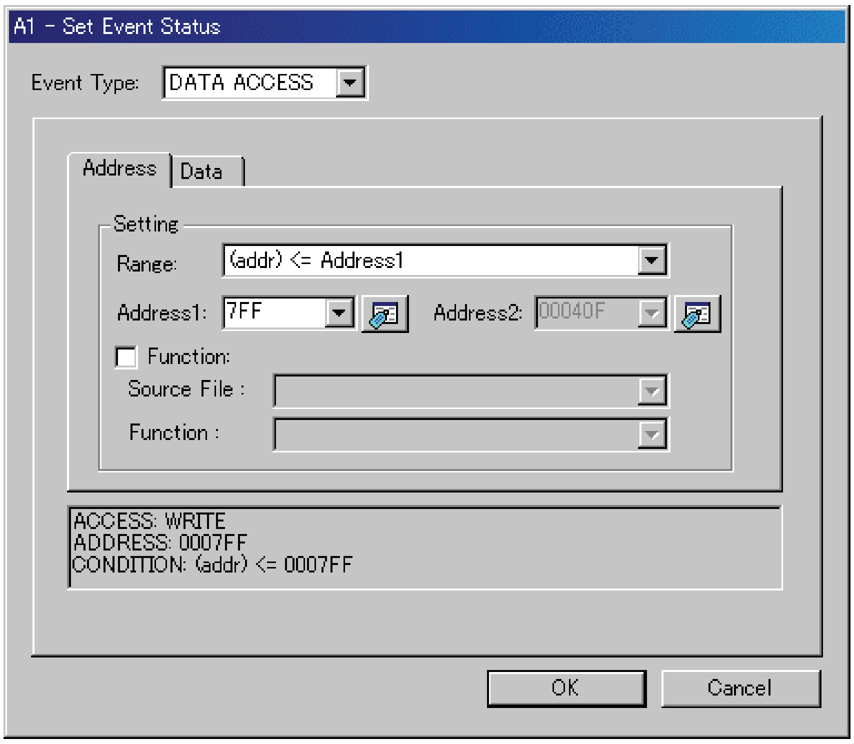
Access: WRITE

Mask: 0000

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: 000400 <= (addr) <= 00040F
```

7.6.5.2.3. Reading/writing data to addresses outside the specified range

Set as below.
Example) Writing data to addresses below 7FFh


A1 - Set Event Status

Event Type: DATA ACCESS

Address tab — Setting

Range: (addr) <= Address1

Address1: 7FF

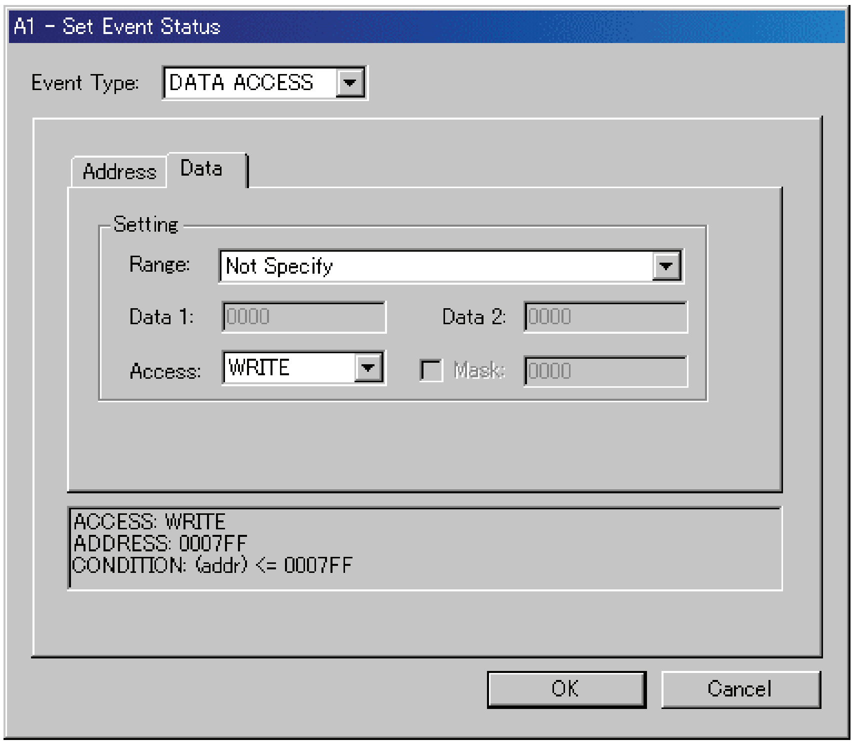
Address2: 00040F

Function: (unchecked)

Source File :

Function :

```
ACCESS: WRITE
ADDRESS: 0007FF
CONDITION: (addr) <= 0007FF
```


A1 - Set Event Status

Event Type: DATA ACCESS

Data tab — Setting

Range: Not Specify

Data 1: 0000

Data 2: 0000

Access: WRITE

Mask: 0000 (unchecked)

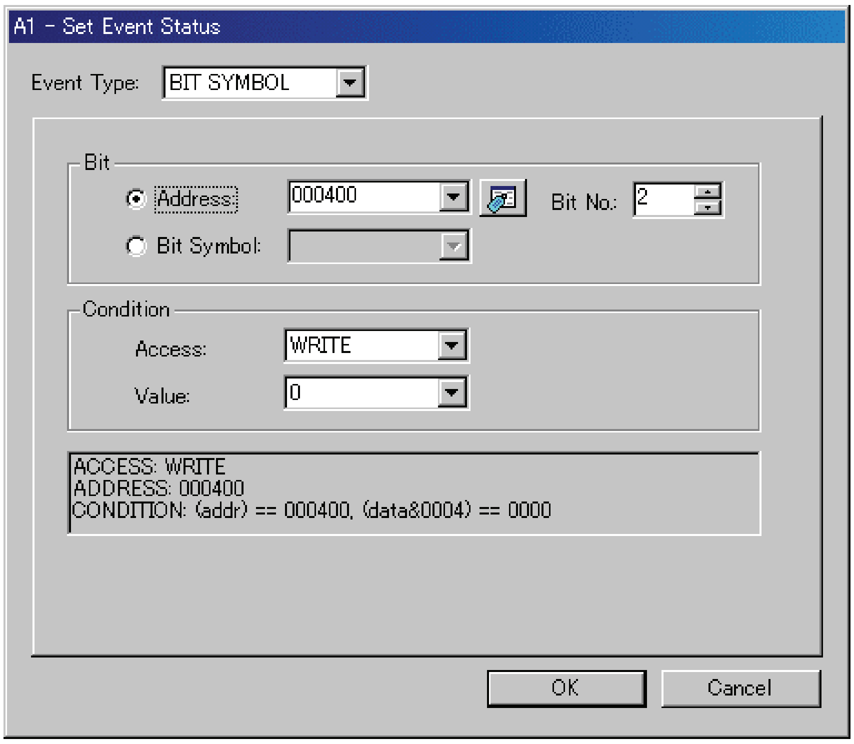
```
ACCESS: WRITE
ADDRESS: 0007FF
CONDITION: (addr) <= 0007FF
```

7.6.6 Specify the Events (Bit Access)

To specify a bit access event, change the event select dialog box's Event Type to "BIT SYMBOL". The event is established when the specified bit at the specified address or specified bit symbol is accessed under specified conditions.

7.6.6.1 Writing/Reading a Specified Bit

Set as below.
Example) Writing "0" to bit 2 at address 400h



A1 - Set Event Status

Event Type: BIT SYMBOL

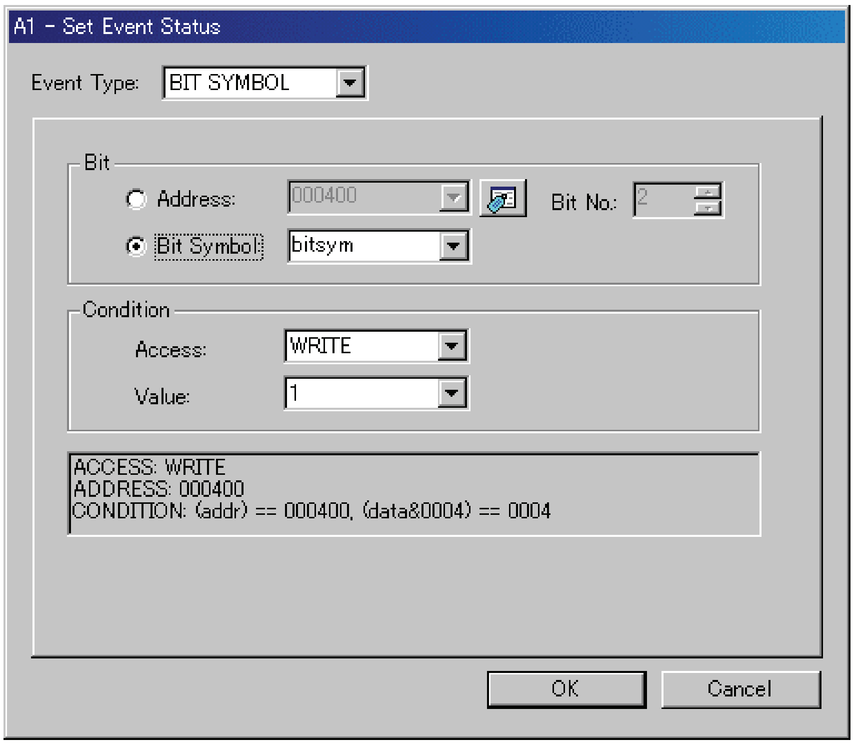
Bit
[✓ Address:] 000400 Bit No.: 2
Bit Symbol:

Condition
Access: WRITE
Value: 0

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&0004) == 0000
```

7.6.6.2 Writing/Reading a Specified Bit Symbol

Set as below.
Example) Writing "1" to bit symbol "bitsym"



A1 - Set Event Status

Event Type: BIT SYMBOL

Bit
( ) Address: 000400 Bit No.: 2
(●) Bit Symbol: bitsym

Condition
Access: WRITE
Value: 1

```
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&0004) == 0004
```

7.6.7 Specify the Event Combination Condition

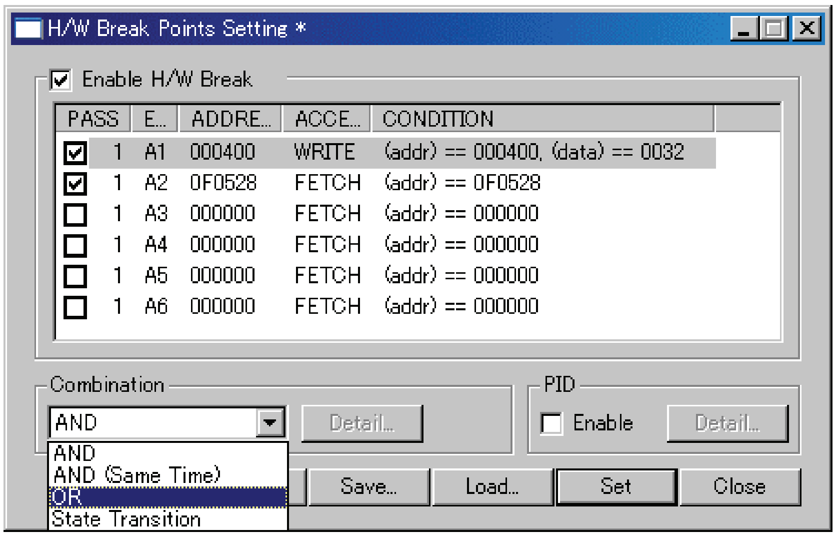
Use the Combination group of the event setting windows to specify the combinatorial conditions of events.
The combination of two or more events can be used.
One of the following combinatorial conditions can be selected.

AND	All of the specified events are established
AND(Same Time)	The specified events are established at the same time
OR	One of the specified events is established

Pass counts (number of times passed) can be specified for each event (1-255). If the specified combinatorial condition is AND (Same Time), no pass counts can be set (fixed to 1).

7.6.7.1 Select AND, OR

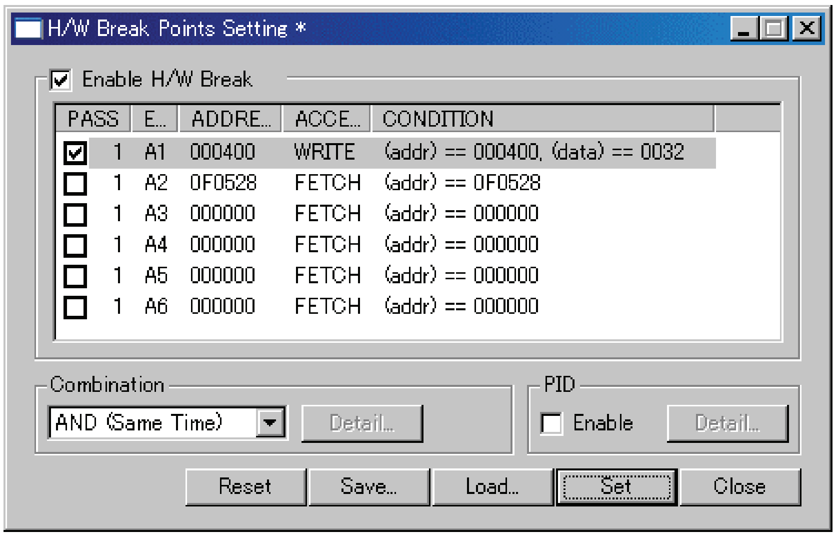
Change the Combination group to "AND" to specify AND for the combinatorial condition, or "OR" to specify OR for the combinatorial condition. Next, check (turn on) an event in the event specification area that you want to use, and specify a pass count for that event. To alter the pass count, while the event to alter is being selected, click the pass count value of that event.



PASS	E...	ADDRE...	ACCE...	CONDITION
✓ 1	A1	000400	WRITE	(addr) == 000400, (data) == 0032
✓ 1	A2	0F0528	FETCH	(addr) == 0F0528
☐ 1	A3	000000	FETCH	(addr) == 000000
☐ 1	A4	000000	FETCH	(addr) == 000000
☐ 1	A5	000000	FETCH	(addr) == 000000
☐ 1	A6	000000	FETCH	(addr) == 000000

7.6.7.2 Select AND(Same Time)

Change the Combination group to "AND (Same Time)". Next, check (turn on) an event in the event specification area that you want to use. No pass counts can be specified (fixed to 1).


H/W Break Points Setting *

☑ Enable H/W Break

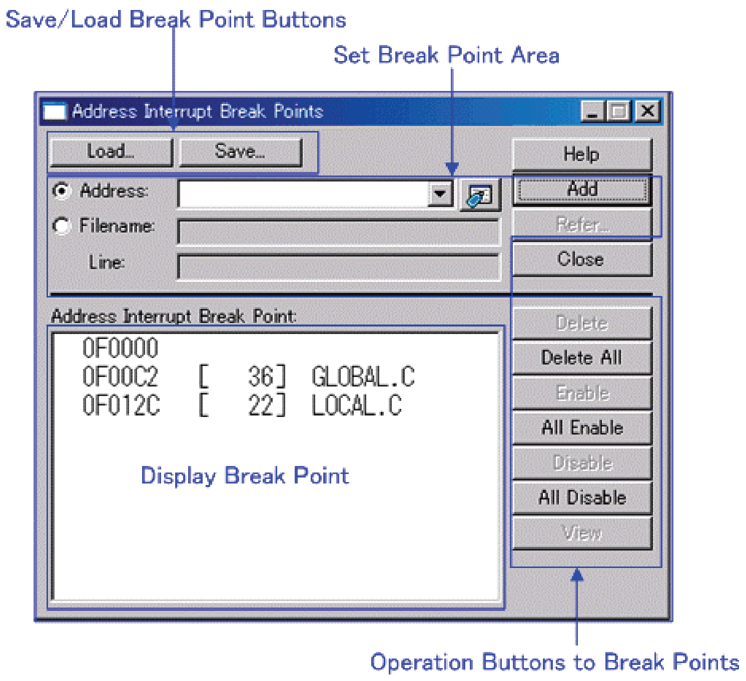
PASS	E...	ADDRE...	ACCE...	CONDITION
☑ 1	A1	000400	WRITE	(addr) == 000400, (data) == 0032
☐ 1	A2	0F0528	FETCH	(addr) == 0F0528
☐ 1	A3	000000	FETCH	(addr) == 000000
☐ 1	A4	000000	FETCH	(addr) == 000000
☐ 1	A5	000000	FETCH	(addr) == 000000
☐ 1	A6	000000	FETCH	(addr) == 000000

Combination: AND (Same Time)

PID: ☐ Enable

7.7 Address Interrupt Break Point Setting Window

The Address Interrupt Break Point Setting window allows you to set address interrupt break points. This function stops the target program immediately before executing an instruction at a specified address. This function is realized by using the MCU's address match interrupt. So that the address interrupt break function can only be used when the address match interrupt is not used in the user application.



```
0F0000
0F00C2   [   36]  GLOBAL.C
0F012C   [   22]  LOCAL.C
```

- This window is available only when the address interrupt break function is used. Use the MCU tab in the Init dialog box to specify whether or not to use the address interrupt break function. On this tab, select the Enable the Address Match Interrupt Break Function check box.(Details).
- The number of address interrupt breakpoints that can be set varies with each product.
- Breakpoints can be specified by "Address" or "Filename + Line No.".
- If you have set multiple breakpoints, program execution stops when any one break address is encountered (OR conditions).
- You can clear, enable or disable breakpoints selected by clicking in the breakpoint display area. You can also enable and disable 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 breakpoints from a file, they are added to any existing break points.

7.7.1 Command Button

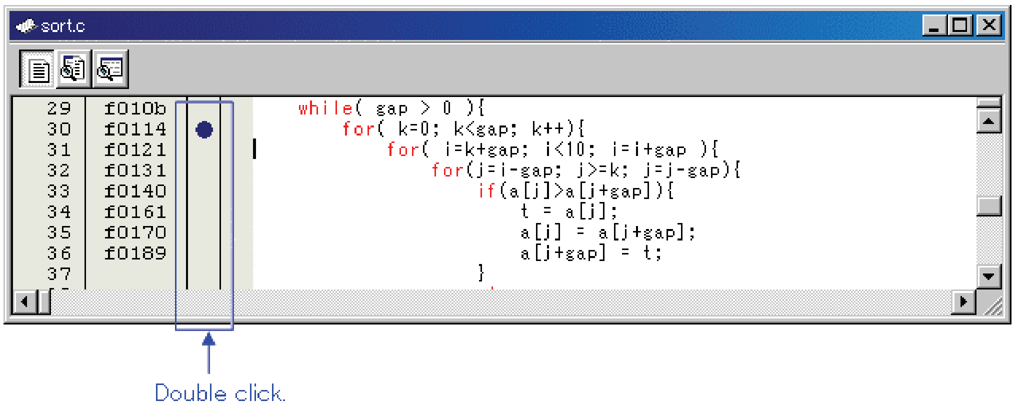
The buttons on this window has the following meanings.

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.

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

The address interrupt break function can only be used when the address match interrupt is not used in the user application.
For details, refer to "1.2.3 Address Interrupt Breaks Function"

You can set break points in the Editor(Source) Window. To do so, double-click the break point setting area ("Address Match Interrupt Break" column) for the line in which you want to set the break. (A blue marker is displayed on the line to which the break point was set.)



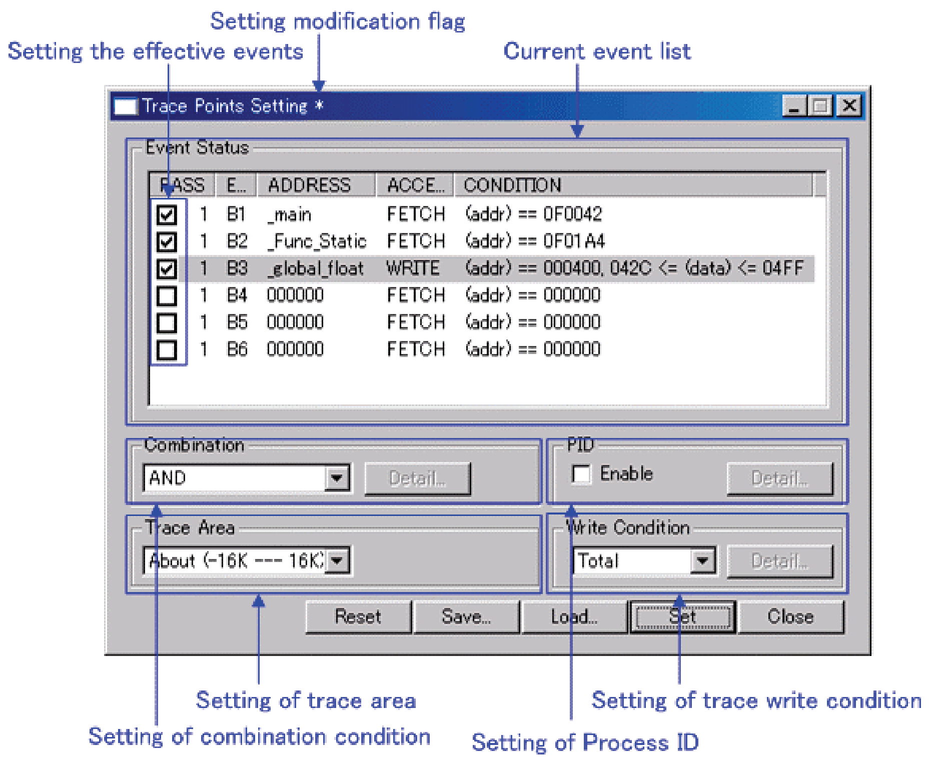
```
f010b       while( gap > 0 ){
f0114   ●       for( k=0; k<gap; k++){
f0121               for( i=k+gap; i<10; i=i+gap ){
f0131                   for(j=i-gap; j>=k; j=j-gap){
f0140                       if(a[j]>a[j+gap]){
f0161                           t = a[j];
f0170                           a[j] = a[j+gap];
f0189                           a[j+gap] = t;
                            }
```

You can delete the break point by double-clicking again in the break point setting area ("Address Match Interrupt Break" column).

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

7.8 Trace Point Setting Window

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



PASS	E...	ADDRESS	ACCE...	CONDITION
☑ 1	B1	_main	FETCH	(addr) == 0F0042
☑ 1	B2	_Func_Static	FETCH	(addr) == 0F01A4
☑ 1	B3	_global_float	WRITE	(addr) == 000400, 042C <= (data) <= 04FF
☐ 1	B4	000000	FETCH	(addr) == 000000
☐ 1	B5	000000	FETCH	(addr) == 000000
☐ 1	B6	000000	FETCH	(addr) == 000000

Combination: AND

PID: ☐ Enable

Trace Area: About (-16K --- 16K)

Write Condition: Total

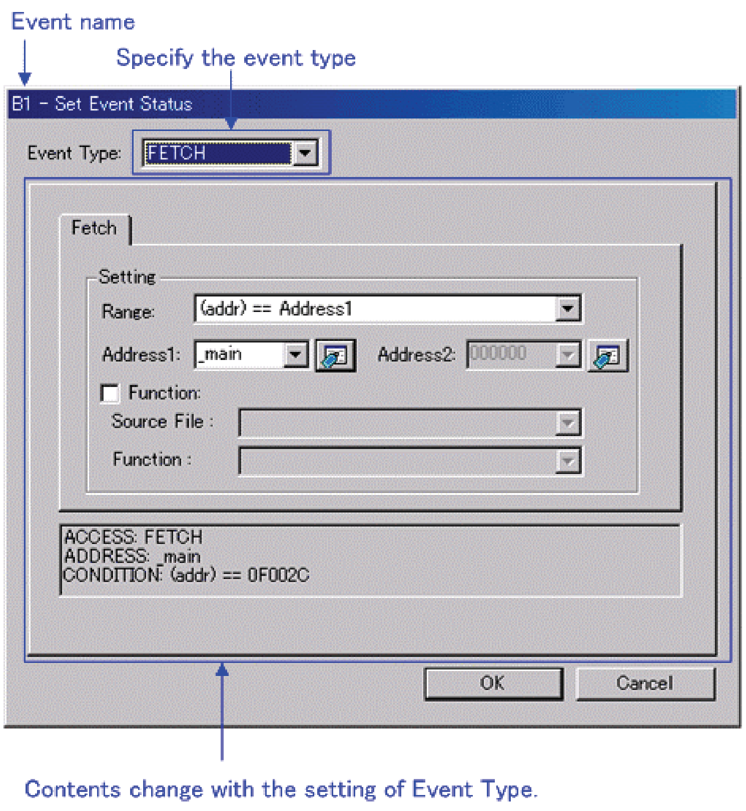
- The events listed below can be specified as trace events. If the contents of events are altered, they are marked by an asterisk (*) on the title bar. The asterisks (*) are not displayed after setting up the emulator.

  Fetch, Memory Access, Bit Access

- Events at up to two points can be used.
  The H/W Break Point Setting window and the Trace Point Setting windows use the same resource of the emulator. Use the MCU tab in the Init dialog box, in order to specify for which function the resources are used. On this tab, select the Enable the Trace Point Function check box.
- These events can be combined in one of the following ways:
  - Trace when all of the valid events are established (AND condition)
  - Trace when all of the valid events are established at the same time (simultaneous AND condition)
  - Trace when one of the valid events is established (OR condition)

7.8.1 Specify the Trace Event

To set events, double-click to select the event you want to set from the event setting area of the Trace Point Setting Window. This opens the dialog box shown below.

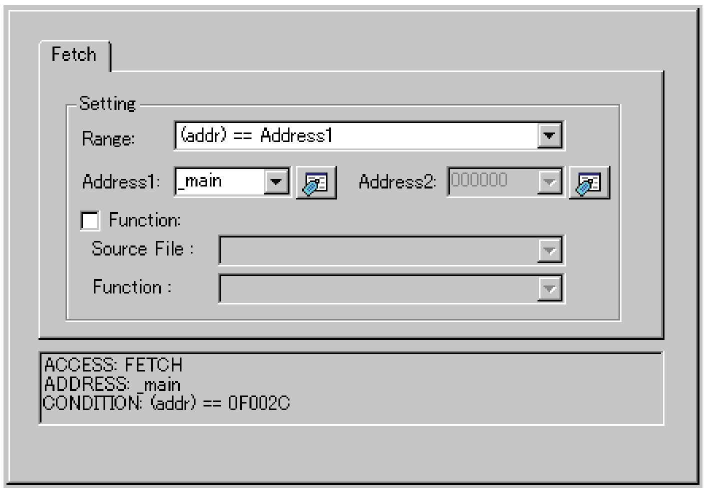


Range: (addr) == Address1
Address1: _main
Address2: 000000
Function:
Source File :
Function :

ACCESS: FETCH
ADDRESS: _main
CONDITION: (addr) == 0F002C

Following events can be set by specifying Event Type in this dialog box.

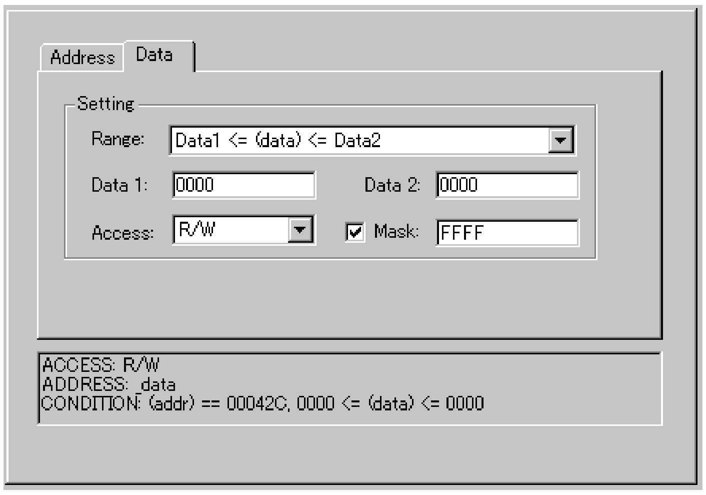
- When FETCH is selected
  Traces for the instruction fetch.



Range: (addr) == Address1
Address1: _main
Address2: 000000
Function:
Source File :
Function :

ACCESS: FETCH
ADDRESS: _main
CONDITION: (addr) == 0F002C

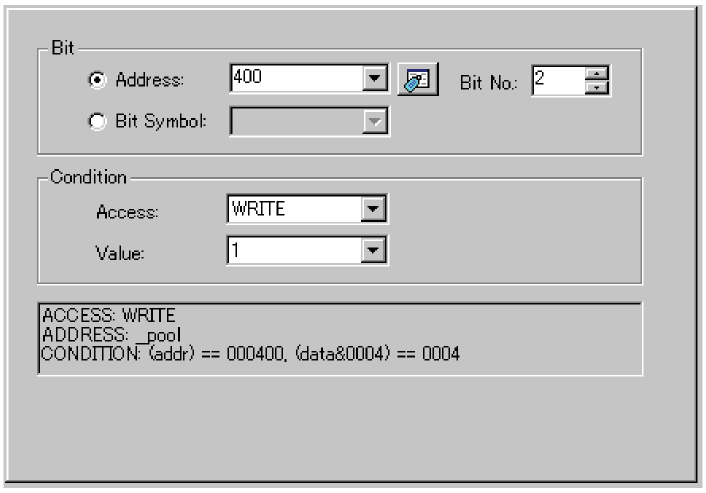
- When DATA ACCESS is selected
  Traces for the memory access.



Range: Data1 <= (data) <= Data2
Data 1: 0000
Data 2: 0000
Access: R/W
✓ Mask: FFFF

```
ACCESS: R/W
ADDRESS: _data
CONDITION: (addr) == 00042C, 0000 <= (data) <= 0000
```

- When BIT SYMBOL is selected
  Traces for the bit access.



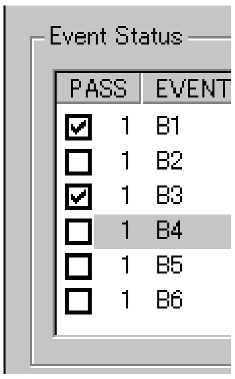
● Address: 400
Bit No.: 2
○ Bit Symbol:
Access: WRITE
Value: 1

```
ACCESS: WRITE
ADDRESS: _pool
CONDITION: (addr) == 000400, (data&0004) == 0004
```

7.8.2 Specify the Combinatorial Condition

To specify a combinatorial condition, specify the desired condition from the combinatorial condition specification area.

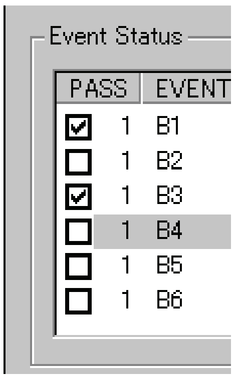
- When AND or OR is selected
  In the event specification area, the event used and a pass count for that event can be specified. To alter the pass count, while the event to alter is being selected, click the pass count value of that event.



Event Status

PASS		EVENT
✓	1	B1
☐	1	B2
✓	1	B3
☐	1	B4
☐	1	B5
☐	1	B6

- When AND (Same Time) is selected
  In the event specification area, the event used can be specified. No pass counts can be specified.

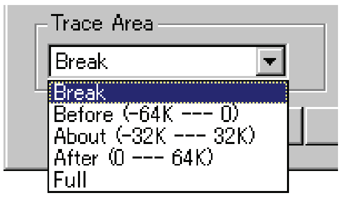


Event Status

PASS		EVENT
✓	1	B1
☐	1	B2
✓	1	B3
☐	1	B4
☐	1	B5
☐	1	B6

7.8.3 Specify the Trace Range

For the compact emulator debugger, 64K cycles equivalent of data can be recorded.


Trace Area

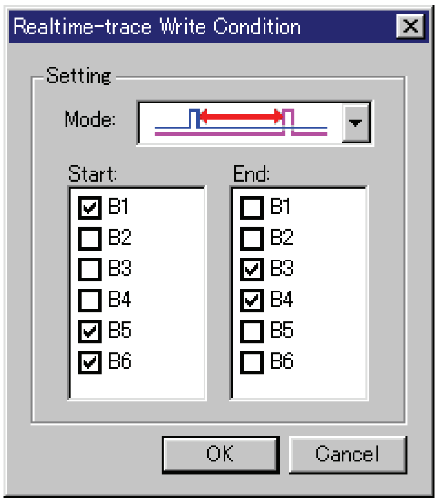
Break

Break
Before (-64K --- 0)
About (-32K --- 32K)
After (0 --- 64K)
Full

Break	Stores the 64K cycles (-64K to 0 cycles) to the point at which the target program stops.
Before	Stores the 64K cycles (-64K to 1 cycles) to the point at which the trace point is passed.
About	Stores the 64K cycles (-32K to 32K cycles) either side of the trace point.
After	Stores the 64K cycles (0 to 64K cycles) of trace data after the trace point.
Full	Stores the 64K cycles (-64K to 0 cycles) of trace data after the trace starts.

7.8.4 Specify the Trace Write Condition

Conditions for cycles to be written to trace memory can be specified.


Realtime-trace Write Condition

Setting

Mode:

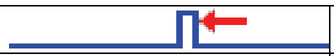
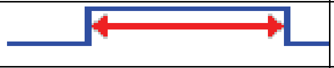
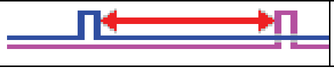
Start:
- [x] B1
- [ ] B2
- [ ] B3
- [ ] B4
- [x] B5
- [x] B6

End:
- [ ] B1
- [ ] B2
- [x] B3
- [x] B4
- [ ] B5
- [ ] B6

OK Cancel

Total	Writes all cycles.
Pick up	Writes only the cycles where specified condition holds true.
Exclude	Writes only the cycles where specified condition does not hold true.

Also, following three write modes are supported.

	Only cycles where specified event is established
	Cycles from where specified event is established to where specified event is not established
	Cycles from where start event is established to where end event is established

7.8.5 Command Button

The buttons on this window has the following meanings.

Button	Function
Reset	Discards the contents being displayed in the window and loads contents from the emulator in which they were set.
Save...	Saves the contents set in the window to a file.
Load...	Loads event information from a file in which it was saved.
Set	Sends the contents set in the window to the emulator.
Close	Closes the window.

7.8.6 Specify the Events (Instruction Fetch)

How to set events for fetch is same as the way for H/W Break Point Setting Window.
For detail about the setting, refer to "7.6.4 Specify the Events (Instruction Fetch)."

7.8.7 Specify the Events (Memory Access)

How to set events for memory access is same as the way for H/W Break Point Setting Window.
For detail about the setting, refer to "7.6.5 Specify the Events (Memory Access)."

7.8.8 Specify the Events (Bit Access)

How to set events for bit access is same as the way for H/W Break Point Setting Window.
For detail about the setting, refer to "7.6.5 Specify the Events (Memory Access)."

7.8.9 Specify the Event Combination Condition

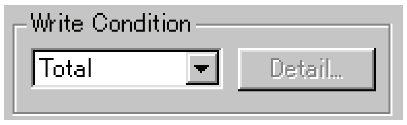
How to set combination of events is same as the way for H/W Break Point Setting Window.
For detail about the setting, refer to "7.6.7 Specify the Event Combination Condition."

7.8.10 Specify the write condition

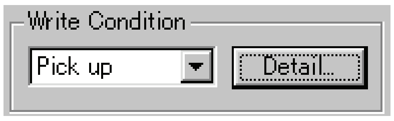
Trace data write conditions can be specified.
You can specify the following write conditions:

1. Write conditions unlimited (default)
2. Cycles from the start event established to the end event established
3. Only cycles where the start event is established
4. Cycles from the start event established to the start event unestablished
5. Other than cycles from the start event established to the end event established
6. Other than cycles where the start event is established
7. Other than cycles from the start event established to the start event unestablished

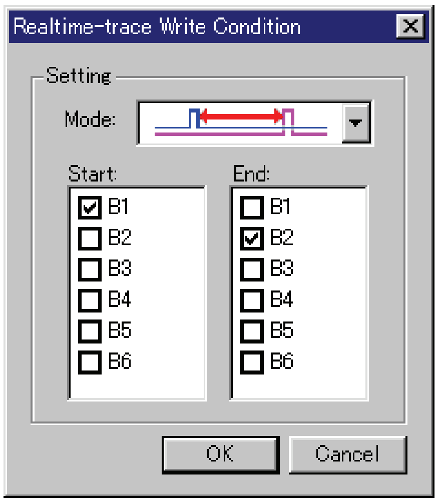
To specify condition 1, choose "Total" from the list box of the window's "Write Condition" item.



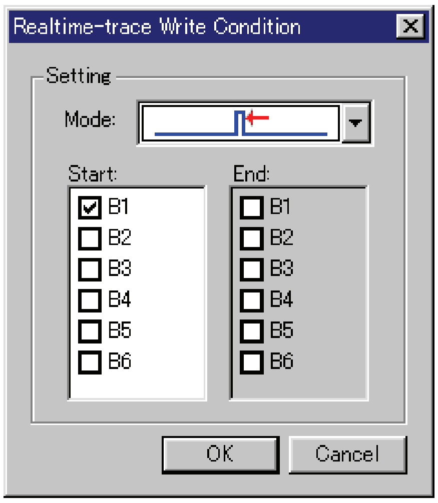
To specify conditions 2 to 4, choose "Pick Up" and click the "Detail..." button to open the "Realtime-trace Write Condition" dialog box.



- For condition 2, choose the Mode shown below and set the Start and End events.



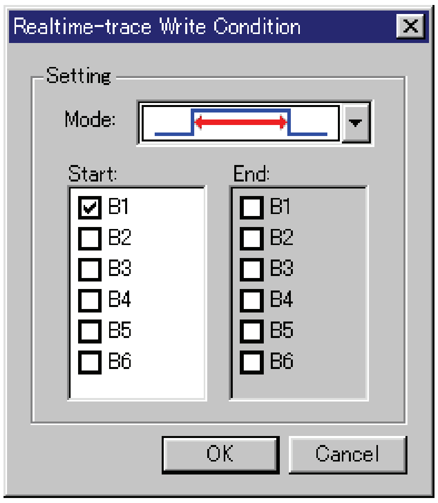
- For condition 3, choose the Mode shown below and set the Start event.



Start: [✓ B1] B2 B3 B4 B5 B6

End: B1 B2 B3 B4 B5 B6

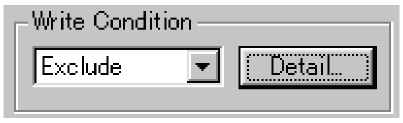
- For condition 4, choose the Mode shown below and set the Start event.



Start: [✓ B1] B2 B3 B4 B5 B6

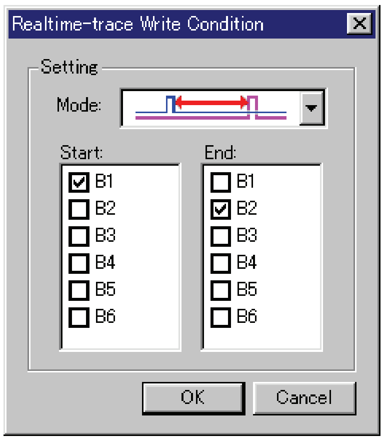
End: B1 B2 B3 B4 B5 B6

Similarly, when specifying conditions 5 to 7, choose "Exclude" and click the "Detail..." button to open the Realtime-trace Write Condition dialog box.



Exclude

- For condition 5, choose the Mode shown below and set the Start and End events.


Realtime-trace Write Condition

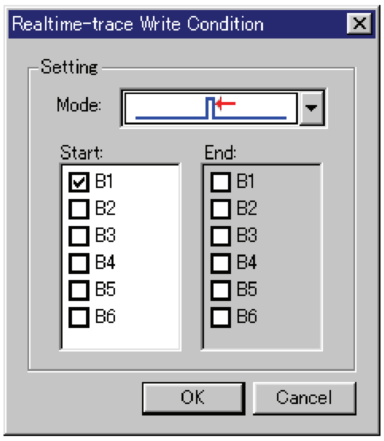
Setting

Mode:

Start:	End:
☑ B1	☐ B1
☐ B2	☑ B2
☐ B3	☐ B3
☐ B4	☐ B4
☐ B5	☐ B5
☐ B6	☐ B6

OK Cancel

For condition 6, choose the Mode shown below and set the Start event.


Realtime-trace Write Condition

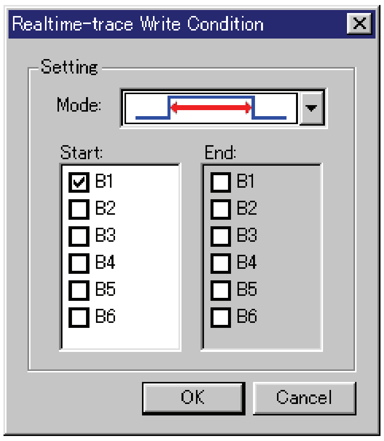
Setting

Mode:

Start:	End:
☑ B1	☐ B1
☐ B2	☐ B2
☐ B3	☐ B3
☐ B4	☐ B4
☐ B5	☐ B5
☐ B6	☐ B6

OK Cancel

- For condition 7, choose the Mode shown below and set the Start event.


Realtime-trace Write Condition

Setting

Mode:

Start:	End:
☑ B1	☐ B1
☐ B2	☐ B2
☐ B3	☐ B3
☐ B4	☐ B4
☐ B5	☐ B5
☐ B6	☐ B6

OK Cancel

7.9 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.

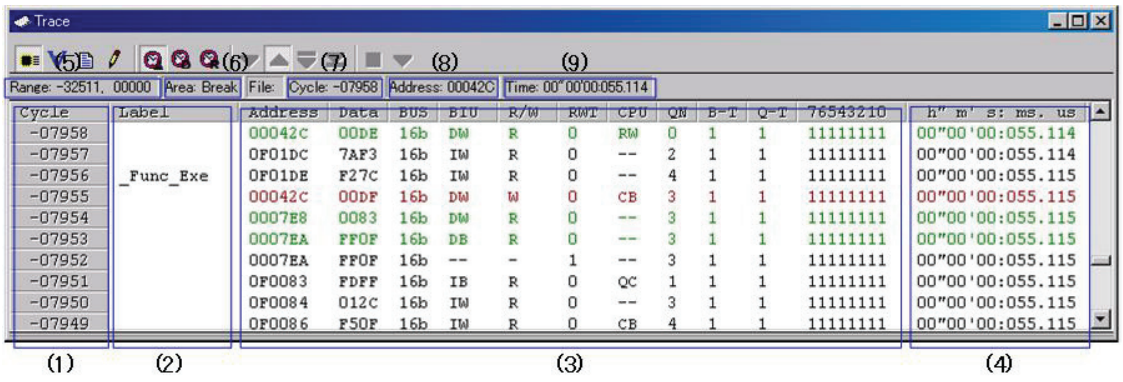
- Bus mode
  This mode allows you to inspect cycle-by-cycle bus information. The display content depends on the MCU and emulator system used. In addition to bus information, this mode allows disassemble, source line or data access information to be displayed in combination.
- Disassemble mode
  This 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 "Referencing the Trace Point Setting Window." With default settings, the trace information immediately before the program has stopped is recorded.

7.9.1 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.



Range: -32511, 00000 | Area: Break | File: | Cycle: -07958 | Address: 00042C | Time: 00"00'00:055.114

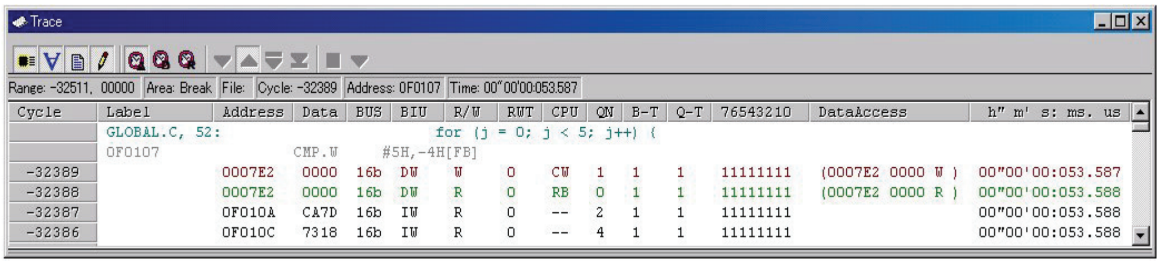
Cycle	Label	Address	Data	BUS	BIU	R/W	RWT	CPU	QN	B-T	Q-T	76543210	h" m' s: ms. us
-07958		00042C	00DE	16b	DW	R	0	RW	0	1	1	11111111	00"00'00:055.114
-07957		0F01DC	7AF3	16b	IW	R	0	--	2	1	1	11111111	00"00'00:055.114
-07956	_Func_Exe	0F01DE	F27C	16b	IW	R	0	--	4	1	1	11111111	00"00'00:055.115
-07955		00042C	00DF	16b	DW	W	0	CB	3	1	1	11111111	00"00'00:055.115
-07954		0007E8	0083	16b	DW	R	0	--	3	1	1	11111111	00"00'00:055.115
-07953		0007EA	FF0F	16b	DB	R	0	--	3	1	1	11111111	00"00'00:055.115
-07952		0007EA	FF0F	16b	--	-	1	--	3	1	1	11111111	00"00'00:055.115
-07951		0F0083	FDFF	16b	IB	R	0	QC	1	1	1	11111111	00"00'00:055.115
-07950		0F0084	012C	16b	IW	R	0	--	3	1	1	11111111	00"00'00:055.115
-07949		0F0086	F50F	16b	IW	R	0	CB	4	1	1	11111111	00"00'00:055.115

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"7.9.6 Display of bus information on the M32C Debugger"
- Refer to"7.9.7 Display of bus information on the M16C/R8C Debugger"r

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.

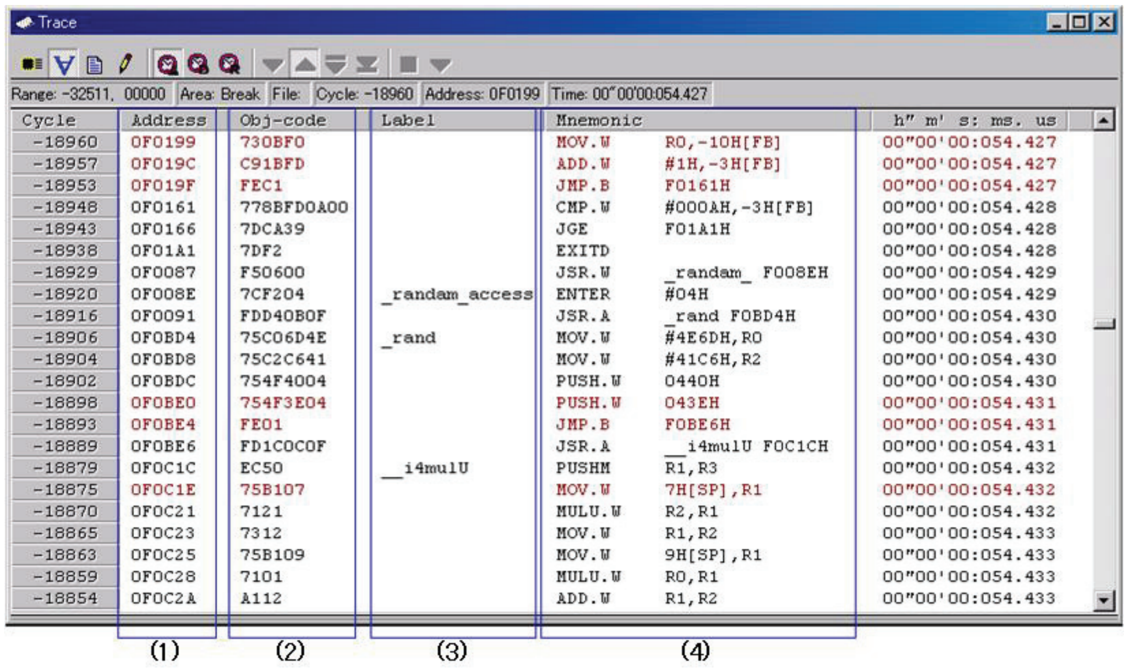
In addition to bus information, the window can display disassemble, source line or data access information in combination. In this case, the display will be similar to the one shown below.



```
Range: -32511, 00000  Area: Break  File:  Cycle: -32389  Address: 0F0107  Time: 00"00'00:053.587
Cycle   Label          Address  Data  BUS  BIU  R/W  RWT  CPU  QN  B-T  Q-T  76543210  DataAccess           h" m' s: ms. us
        GLOBAL.C, 52:                            for (j = 0; j < 5; j++) {
        0F0107                  CMP.W      #5H,-4H[FB]
-32389                 0007E2   0000  16b  DW   W    0    CW   1   1    1    11111111  (0007E2 0000 W )     00"00'00:053.587
-32388                 0007E2   0000  16b  DW   R    0    RB   0   1    1    11111111  (0007E2 0000 R )     00"00'00:053.588
-32387                 0F010A   CA7D  16b  IW   R    0    --   2   1    1    11111111                       00"00'00:053.588
-32386                 0F010C   7318  16b  IW   R    0    --   4   1    1    11111111                       00"00'00:053.588
```

7.9.2 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.

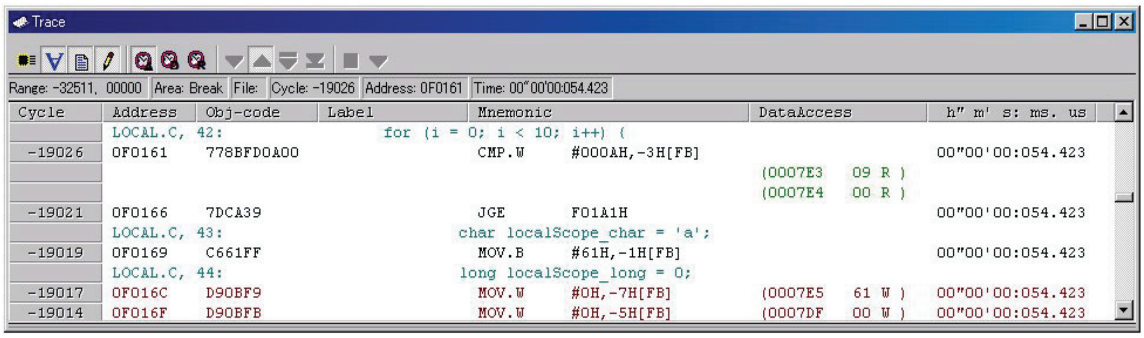


```
Trace
Range: -32511, 00000  Area: Break  File:  Cycle: -18960  Address: 0F0199  Time: 00"00'00:054.427
Cycle    Address  Obj-code    Label           Mnemonic                       h" m' s: ms. us
 -18960  0F0199   730BF0                      MOV.W    R0,-10H[FB]           00"00'00:054.427
 -18957  0F019C   C91BFD                      ADD.W    #1H,-3H[FB]           00"00'00:054.427
 -18953  0F019F   FEC1                        JMP.B    F0161H                00"00'00:054.427
 -18948  0F0161   778BFD0A00                  CMP.W    #000AH,-3H[FB]        00"00'00:054.428
 -18943  0F0166   7DCA39                      JGE      F01A1H                00"00'00:054.428
 -18938  0F01A1   7DF2                        EXITD                          00"00'00:054.428
 -18929  0F0087   F50600                      JSR.W    _randam_ F008EH       00"00'00:054.429
 -18920  0F008E   7CF204      _randam_access  ENTER    #04H                  00"00'00:054.429
 -18916  0F0091   FDD40B0F                    JSR.A    _rand F0BD4H          00"00'00:054.430
 -18906  0F0BD4   75C06D4E    _rand           MOV.W    #4E6DH,R0             00"00'00:054.430
 -18904  0F0BD8   75C2C641                    MOV.W    #41C6H,R2             00"00'00:054.430
 -18902  0F0BDC   754F4004                    PUSH.W   0440H                 00"00'00:054.430
 -18898  0F0BE0   754F3E04                    PUSH.W   043EH                 00"00'00:054.431
 -18893  0F0BE4   FE01                        JMP.B    F0BE6H                00"00'00:054.431
 -18889  0F0BE6   FD1C0C0F                    JSR.A    __i4mulU F0C1CH       00"00'00:054.431
 -18879  0F0C1C   EC50        __i4mulU        PUSHM    R1,R3                 00"00'00:054.432
 -18875  0F0C1E   75B107                      MOV.W    7H[SP],R1             00"00'00:054.432
 -18870  0F0C21   7121                        MULU.W   R2,R1                 00"00'00:054.432
 -18865  0F0C23   7312                        MOV.W    R1,R2                 00"00'00:054.433
 -18863  0F0C25   75B109                      MOV.W    9H[SP],R1             00"00'00:054.433
 -18859  0F0C28   7101                        MULU.W   R0,R1                 00"00'00:054.433
 -18854  0F0C2A   A112                        ADD.W    R1,R2                 00"00'00:054.433
```

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.

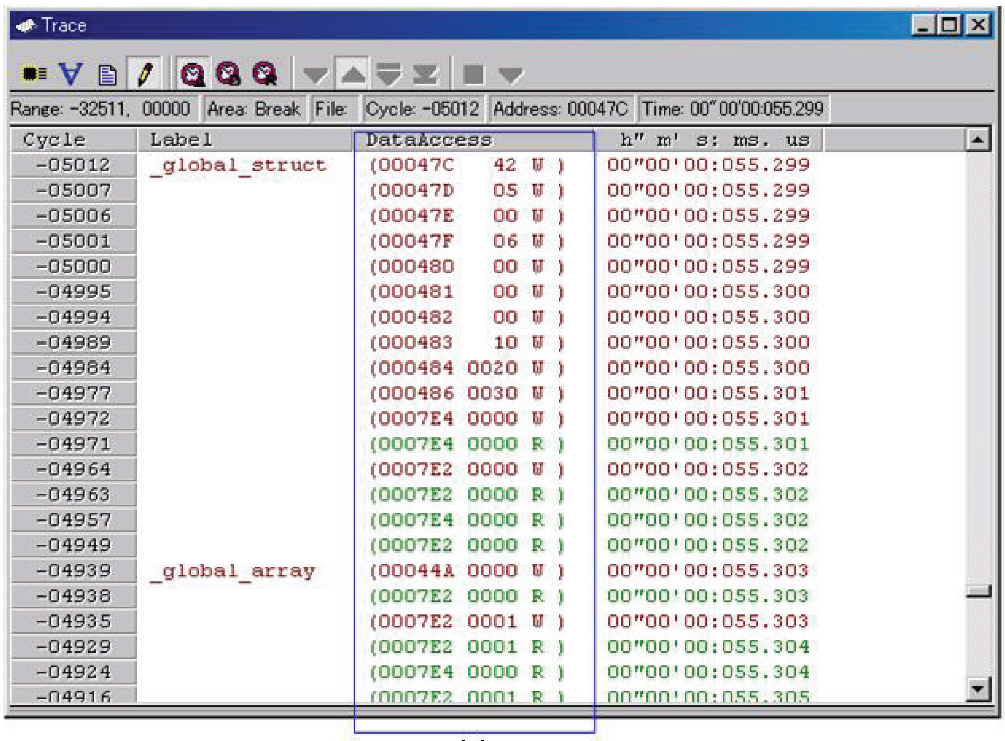
In addition to disassemble information, the window can display source line or data access information in combination. In this case, the display will be similar to the one shown below.



```
Trace
Range: -32511, 00000  Area: Break  File:  Cycle: -19026  Address: 0F0161  Time: 00"00'00:054.423
Cycle    Address  Obj-code    Label     Mnemonic                  DataAccess        h" m' s: ms. us
         LOCAL.C, 42:                for (i = 0; i < 10; i++) {
 -19026  0F0161   778BFD0A00            CMP.W   #000AH,-3H[FB]                      00"00'00:054.423
                                                                  (0007E3   09 R )
                                                                  (0007E4   00 R )
 -19021  0F0166   7DCA39                JGE     F01A1H                              00"00'00:054.423
         LOCAL.C, 43:                  char localScope_char = 'a';
 -19019  0F0169   C661FF                MOV.B   #61H,-1H[FB]                        00"00'00:054.423
         LOCAL.C, 44:                  long localScope_long = 0;
 -19017  0F016C   D90BF9                MOV.W   #0H,-7H[FB]       (0007E5   61 W )  00"00'00:054.423
 -19014  0F016F   D90BFB                MOV.W   #0H,-5H[FB]       (0007DF   00 W )  00"00'00:054.423
```

7.9.3 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.

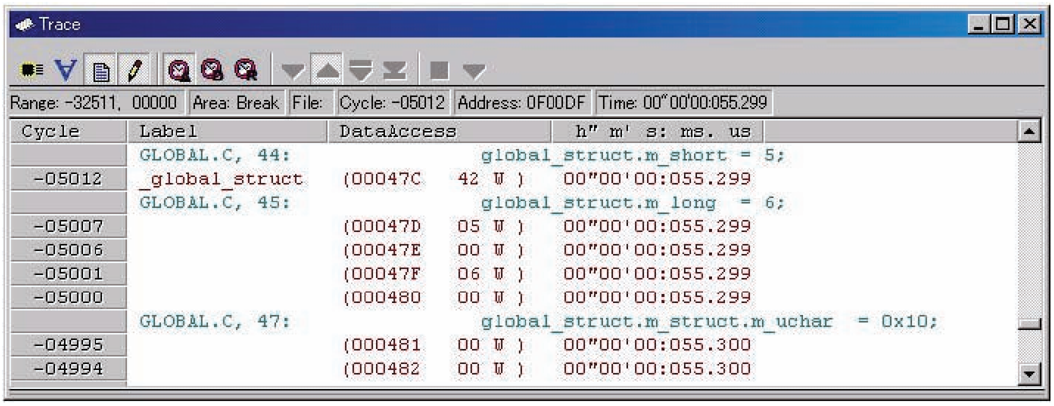


```
Range: -32511, 00000 | Area: Break | File: | Cycle: -05012 | Address: 00047C | Time: 00"00'00:055.299

Cycle     Label            DataAccess           h" m' s: ms. us
-05012    _global_struct   (00047C   42 W )     00"00'00:055.299
-05007                     (00047D   05 W )     00"00'00:055.299
-05006                     (00047E   00 W )     00"00'00:055.299
-05001                     (00047F   06 W )     00"00'00:055.299
-05000                     (000480   00 W )     00"00'00:055.299
-04995                     (000481   00 W )     00"00'00:055.300
-04994                     (000482   00 W )     00"00'00:055.300
-04989                     (000483   10 W )     00"00'00:055.300
-04984                     (000484 0020 W )     00"00'00:055.300
-04977                     (000486 0030 W )     00"00'00:055.301
-04972                     (0007E4 0000 W )     00"00'00:055.301
-04971                     (0007E4 0000 R )     00"00'00:055.301
-04964                     (0007E2 0000 W )     00"00'00:055.302
-04963                     (0007E2 0000 R )     00"00'00:055.302
-04957                     (0007E4 0000 R )     00"00'00:055.302
-04949                     (0007E2 0000 R )     00"00'00:055.302
-04939    _global_array    (00044A 0000 W )     00"00'00:055.303
-04938                     (0007E2 0000 R )     00"00'00:055.303
-04935                     (0007E2 0001 W )     00"00'00:055.303
-04929                     (0007E2 0001 R )     00"00'00:055.304
-04924                     (0007E4 0000 R )     00"00'00:055.304
-04916                     (0007E2 0001 R )     00"00'00:055.305
```

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.

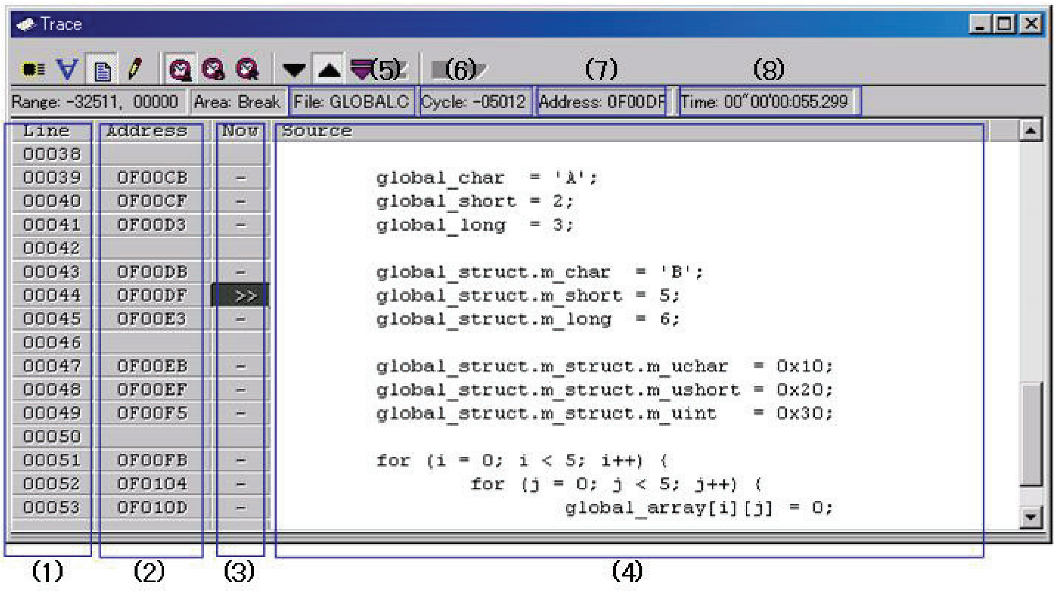
In addition to data access information, the window can display source line information in combination. In this case, the display will be similar to the one shown below.



```
Range: -32511, 00000 | Area: Break | File: | Cycle: -05012 | Address: 0F00DF | Time: 00"00'00:055.299

Cycle     Label            DataAccess           h" m' s: ms. us
          GLOBAL.C, 44:                global_struct.m_short = 5;
-05012    _global_struct   (00047C   42 W )     00"00'00:055.299
          GLOBAL.C, 45:                global_struct.m_long  = 6;
-05007                     (00047D   05 W )     00"00'00:055.299
-05006                     (00047E   00 W )     00"00'00:055.299
-05001                     (00047F   06 W )     00"00'00:055.299
-05000                     (000480   00 W )     00"00'00:055.299
          GLOBAL.C, 47:                global_struct.m_struct.m_uchar  = 0x10;
-04995                     (000481   00 W )     00"00'00:055.300
-04994                     (000482   00 W )     00"00'00:055.300
```

7.9.4 Configuration of Source Mode

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



```
Range: -32511, 00000 | Area: Break | File: GLOBALC | Cycle: -05012 | Address: 0F00DF | Time: 00"00'00:055.299

Line   Address  Now  Source
00038
00039  0F00CB   -            global_char  = 'A';
00040  0F00CF   -            global_short = 2;
00041  0F00D3   -            global_long  = 3;
00042
00043  0F00DB   -            global_struct.m_char  = 'B';
00044  0F00DF   >>           global_struct.m_short = 5;
00045  0F00E3   -            global_struct.m_long  = 6;
00046
00047  0F00EB   -            global_struct.m_struct.m_uchar  = 0x10;
00048  0F00EF   -            global_struct.m_struct.m_ushort = 0x20;
00049  0F00F5   -            global_struct.m_struct.m_uint   = 0x30;
00050
00051  0F00FB   -            for (i = 0; i < 5; i++) {
00052  0F0104   -                    for (j = 0; j < 5; j++) {
00053  0F010D   -                            global_array[i][j] = 0;
```

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.

7.9.5 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.

7.9.6 Display of bus information on the M32C Debugger

From left to right, the contents are as follows:

- Address
  The status of the address bus
- Data
  The status of the data bus
- BUS
  The width of the external data bus ("8b" for an 8-bit data bus, and "16b" for a 16-bit data bus)
- BHE
  Indicates the status (0 or 1) of the BHE (Byte High Enable) signal.
  If BHE = 0, it means that the CPU is accessing an odd address.
- BIU
  This shows the status between the BIU (bus interface unit) and memory, and BIU and I/O.

Representation	BIU status
-	No access
WAIT	Executing wait instruction
RBML	Read access (bytes, ML on)
F	Fetch access
QC	Discontinuous Fetch access (queue buffer)
RWML	Read access (words, ML on)
INT	Interrupt acknowledge
RB	Read access (bytes)
WB	Write access (bytes)
DRB	Read access by DMA (bytes)
DWB	Write access by DMA (bytes)
RW	Read access (words)
WW	Write access (words)
DRW	Read access by DMA (words)
DWW	Write access by DMA (words)

- R/W
  Shows the status of the data bus ("R" for r ead, "W" for wr it e, "-" for no access).
- RWT
  This signal shows the effective position in the bus cycle ("0" when effective. Address, Data, and BIU signals are valid when RWT is "0".
- CPU, OPC, OPR
  This shows the signal between CPU and BIU. In the column "CPU", the data shows whether CPU accesses BIU or not . In the Column "OPC", the data shows the byte size of read operat ion code. In the Column "OPR", the data shows the byte size of read operand.

Representation			**Status**	
CPU	**OPC**	**OPR**	**Operation code size**	**Operand size**
-	-	-	No accessing	
CPU	0	1	0byte	1byte
CPU	0	2	0byte	2bytes
CPU	0	3	0byte	3bytes
CPU	1	0	1byte	0byte
CPU	1	1	1byte	1byte
CPU	1	2	1byte	2bytes
CPU	1	3	1byte	3bytes
CPU	2	0	2bytes	0byte
CPU	2	1	2bytes	1byte
CPU	2	2	2bytes	2bytes
CPU	3	0	3bytes	0byte
CPU	3	1	3bytes	1byte
DMA	-	-	DMA accessing	
DMAT	-	-	DMA accessing(terminal count)	

7.9.7 Display of bus information on the M16C/R8C Debugger

From left to right, the contents are as follows:

- Address
  The status of the address bus
- Data
  The status of the data bus
- BUS
  The width of the external data bus ("8b" for an 8-bit data bus, and "16b" for a 16-bit data bus)
- BHE
  Indicates the status (0 or 1) of the BHE (Byte High Enable) signal.
  If BHE = 0, it means that the CPU is accessing an odd address.
- BIU
  This shows the status between the BIU (bus interface unit) and memory, and BIU and I/O.

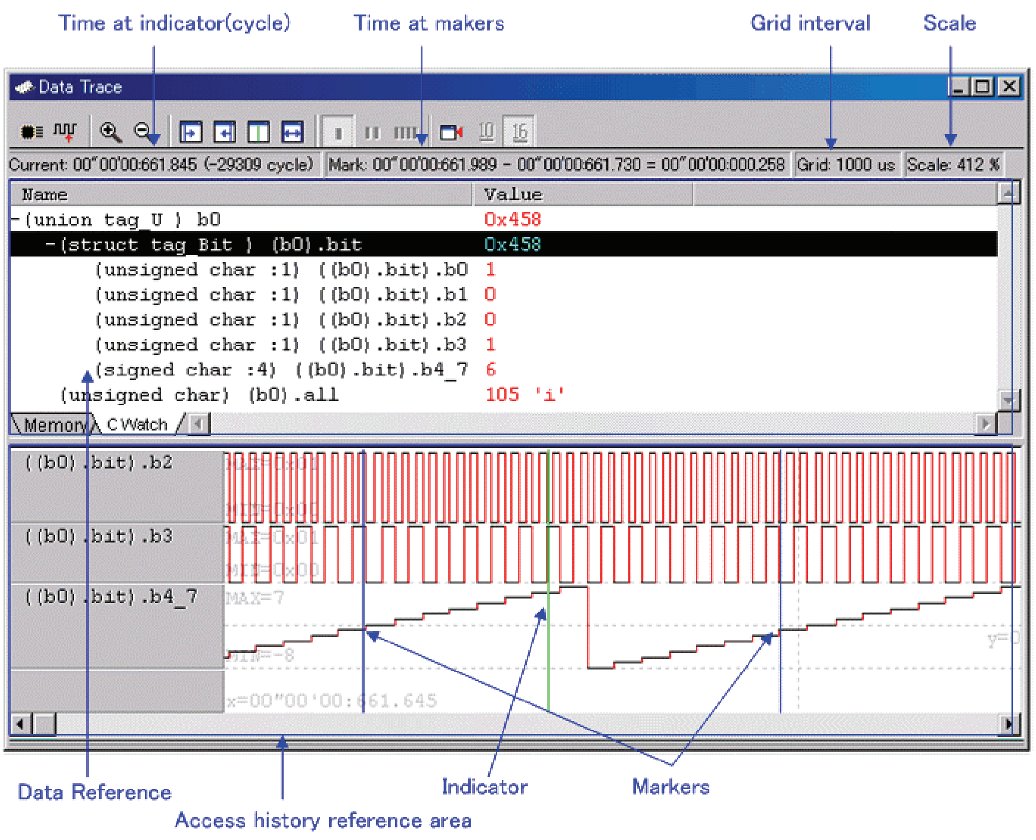
Representation	BIU status
-	No access
DMA	Data access other than a CPU cause such as DMA
INT	Start of INTACK sequence
IB	Instruction code read due to CPU cause (bytes)
DB	data access due to CPU cause (bytes)
IW	Instruction code read due to CPU cause (words)
DW	data access due to CPU cause (words)

- R/W
  Shows the status of the data bus ("R" for r ead, "W" for wr it e, "-" for no access).
- RWT
  This signal shows the effective position in the bus cycle ("0" when effective. Address, Data, and BIU signals are valid when RWT is "0".
- CPU
  This shows the signal between CPU and BIU.

Representation	Status
-	No accessing
CB	Opecode read (bytes)
RB	Operand read (bytes)
QC	Instruction queue buffer clear
CW	Opecode read (words)
RW	Operand read (words)

7.10 Data Trace Window

The Data Trace Window is used to analyze the results of real-time trace measurements and graphically show data access information.
It operates in conjunction with Trace Window.
The debugger for compact emulators don't support time information.



```
Current: 00"00'00:661.845 (-29309 cycle) | Mark: 00"00'00:661.989 - 00"00'00:661.730 = 00"00'00:000.258 | Grid: 1000 us | Scale: 412 %
```

```
Name                                                  Value
-(union tag_U ) b0                                    0x458
  -(struct tag_Bit ) (b0).bit                         0x458
      (unsigned char :1)  ((b0).bit).b0               1
      (unsigned char :1)  ((b0).bit).b1               0
      (unsigned char :1)  ((b0).bit).b2               0
      (unsigned char :1)  ((b0).bit).b3               1
      (signed char :4)  ((b0).bit).b4_7               6
   (unsigned char) (b0).all                           105 'i'
```

```
((b0).bit).b2      MAX=0x01  MIN=0x00
((b0).bit).b3      MAX=0x01  MIN=0x00
((b0).bit).b4_7    MAX=7  MIN=-8
                   x=00"00'00:661.645
```

- In the data reference area, you can inspect memory values at the point of a cycle currently in interest or the values of registered C variables.
- In the access history reference area, you can see the history of accesses to registered addresses in chart form.
- In conjunction with the Trace Window, you can inspect memory values at the point of a cycle you are watching in the Trace Window. Conversely, you can show the cycle in the Trace Window which you are watching in the Data Trace Window.

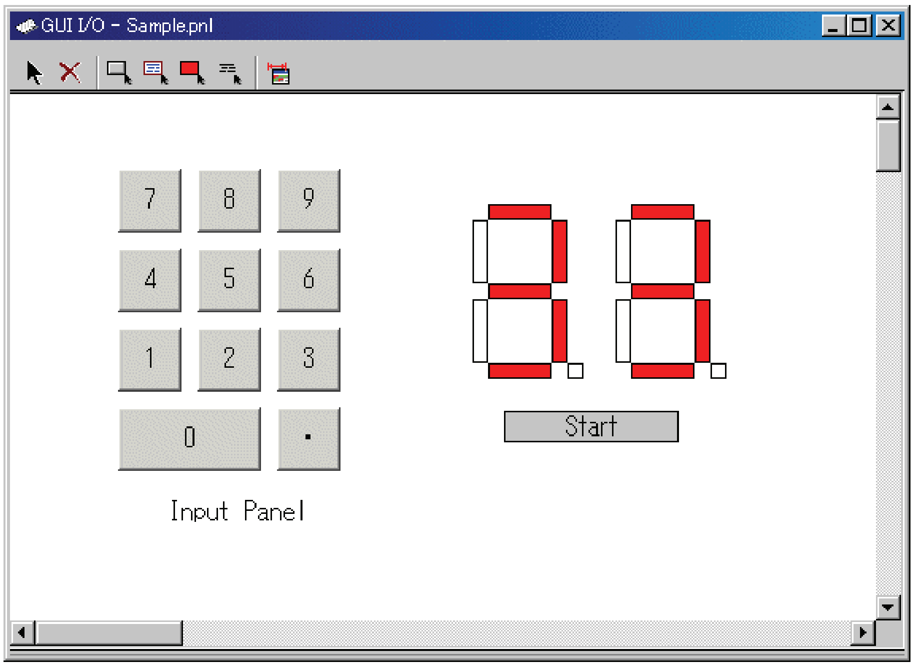
7.10.1 Extended Menus

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

Menu		Function
Analyze Trace Data		Analyze the realtime-trace data.
Set Cycle...		Specify the display cycle.
Sync with Trace Window		Synchronize with Trace Window.
Data Length	1byte	Display in 1Byte unit.
	2bytes	Display in 2Byte unit.
	4bytes	Display in 4Byte unit.
Radix	Hex	Display in Hexadecimal.
	Dec	Display in Decimal.
Address...		Display from specified address.
Add C Watch		Add C watchpoint.
Remove C Watch		Remove the selected C watchpoint.
Hide Type Name		Hide type names from variables.
Add...		Adds new watch item into Access History Reference Area.
Remove		Removes the selected watch item from Access History Reference Area.
Zoom	Zoom In	Increase the display scale.
	Zoom Out	Decrease the display scale.
	Zoom...	Specify the display scale.
Marker	Start Marker	Move the start marker in the display area.
	End Marker	Move the end marker in the display area.
	Indicator	Move the indicator in the display area.
	Adjust	Set cycle range between markers.
Change Grid Interval...		Change the grid interval.
Change Row Setting...		Change setting of the selected row.
Color...		Change the display color.
Toolbar display		Display toolbar.
Customize toolbar...		Open toolbar customize dialog box.
Allow Docking		Allow window docking.
Hide		Hide window.

7.11 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.

7.11.1 Extended Menus

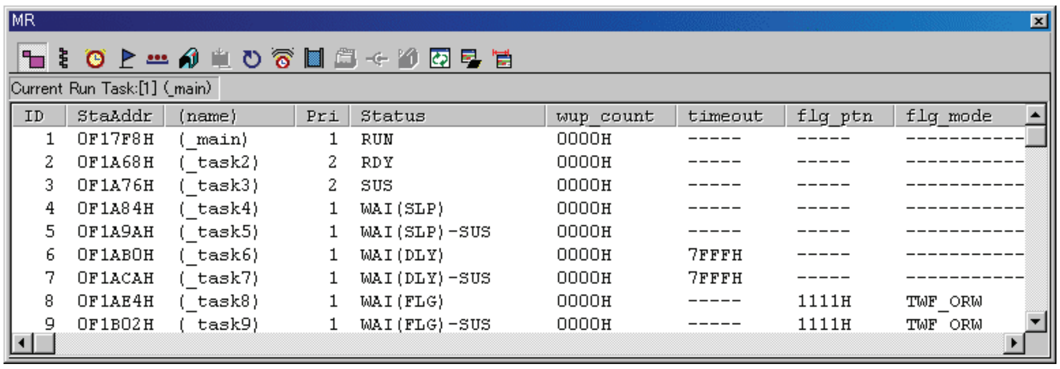
This window has the following popup menus that can be brought up by right-clicking in the window.

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.

7.12 MR Window

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

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



Current Run Task:[1] (_main)

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

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

ATTENTION

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

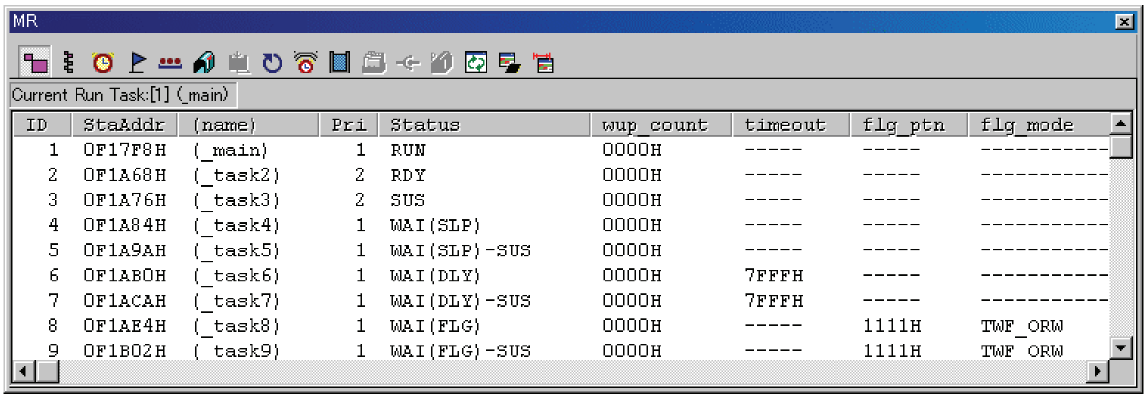
7.12.1 Extended Menus

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

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

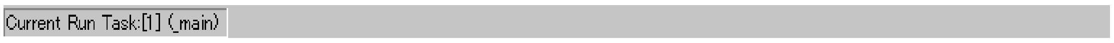
7.12.2 Display the Task Status

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



```
Current Run Task:[1] (_main)

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

By double-clicking any line, the information on the task context is displayed in the Context dialog. For details on the Context dialog, see "7.12.12 Display the Task Context"
The following data is displayed in the status bar.



```
Current Run Task:[1] (_main)
```

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

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

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

- *1Task Status

Display	**Status**
RUN	RUNNING state
RDY	READY state
SUS	SUSPENDED state
DMT	DORMANT state
WAI(SLP)	Sleeping state
WAI(SLP)-SUS	Sleeping state (suspended)
WAI(SLP-TMO)	Sleeping state with time-out
WAI(SLP-TMO)-SUS	Sleeping state with time-out (suspended)
WAI(DLY)	Delayed state due to dly_tsk
WAI(DLY)-SUS	Delayed state due to dly_tsk (suspended)
WAI(FLG)	Waiting state for an eventflag
WAI(FLG)-SUS	Waiting state for an eventflag (suspended)
WAI(FLG-TMO)	Waiting state for an eventflag with time-out
WAI(FLG-TMO)-SUS	Waiting state for an eventflag with time-out (suspended)
WAI(SEM)	Waiting state for a semaphore resource
WAI(SEM)-SUS	Waiting state for a semaphore resource (suspended)
WAI(SEM-TMO)	Waiting state for a semaphore resource with time-out
WAI(SEM-TMO)-SUS	Waiting state for a semaphore resource with time-out (suspended)
WAI(MBX)	Receiving waiting state for a mailbox
WAI(MBX)-SUS	Receiving waiting state for a mailbox (suspended)
WAI(MBX-TMO)	Receiving waiting state for a mailbox with time-out
WAI(MBX-TMO)-SUS	Receiving waiting state for a mailbox with time-out (suspended)

- *2Display the Wait Cancellation Condition of Event Flag

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

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

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

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

- *1Task Status

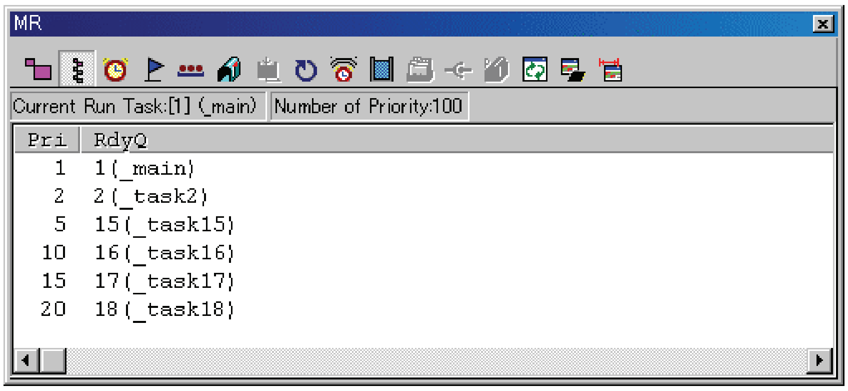
Display	Status
RUN	RUNNING state
RDY	READY state
SUS	SUSPENDED state
DMT	DORMANT state
WAI(SLP)	Sleeping state
WAI(SLP)-SUS	Sleeping state (suspended)
WAI(SLP-TMO)	Sleeping state with time-out
WAI(SLP-TMO)-SUS	Sleeping state with time-out (suspended)
WAI(DLY)	Delayed state due to dly_tsk
WAI(DLY)-SUS	Delayed state due to dly_tsk (suspended)
WAI(FLG)	Waiting state for an eventflag
WAI(FLG)-SUS	Waiting state for an eventflag (suspended)
WAI(FLG-TMO)	Waiting state for an eventflag
WAI(FLG-TMO)-SUS	Waiting state for an eventflag (suspended)
WAI(SEM)	Waiting state for a semaphore resource
WAI(SEM)-SUS	Waiting state for a semaphore resource (suspended)
WAI(SEM-TMO)	Waiting state for a semaphore resource with time-out
WAI(SEM-TMO)-SUS	Waiting state for a semaphore resource with time-out (suspended)
WAI(MBX)	Receiving waiting state for a mailbox
WAI(MBX)-SUS	Receiving waiting state for a mailbox (suspended)
WAI(MBX-TMO)	Receiving waiting state for a mailbox with time-out
WAI(MBX-TMO)-SUS	Receiving waiting state for a mailbox with time-out (suspended)
WAI(SDTQ)	Sending waiting state for a data queue
WAI(SDTQ)-SUS	Sending waiting state for a data queue (suspended)
WAI(SDTQ-TMO)	Sending waiting state for a data queue with time-out
WAI(SDTQ-TMO)-SUS	Sending waiting state for a data queue with time-out (suspended)
WAI(RDTQ)	Receiving waiting state for a data queue
WAI(RDTQ)-SUS	Receiving waiting state for a data queue (suspended)
WAI(RDTQ-TMO)	Receiving waiting state for a data queue with time-out
WAI(RDTQ-TMO)-SUS	Receiving waiting state for a data queue with time-out (suspended)
WAI(VSDTQ)	Sending waiting state for an extended data queue
WAI(VSDTQ)-SUS	Sending waiting state for an extended data queue (suspended)
WAI(VSDTQ-TMO)	Sending waiting state for an extended data queue with time-out
WAI(VSDTQ-TMO)-SUS	Sending waiting state for an extended data queue with time-out (suspended)
WAI(VRDTQ)	Receiving waiting state for an extended data queue
WAI(VRDTQ)-SUS	Receiving waiting state for an extended data queue (suspended)
WAI(VRDTQ-TMO)	Receiving waiting state for an extended data queue with time-out
WAI(VRDTQ-TMO)-SUS	Receiving waiting state for an extended data queue with time-out (suspended)
WAI(MPF)	Waiting state for a fixed-sized memory block
WAI(MPF)-SUS	Waiting state for a fixed-sized memory block (suspended)
WAI(MPF-TMO)	Waiting state for a fixed-sized memory block with time-out
WAI(MPF-TMO)-SUS	Waiting state for a fixed-sized memory block with time-out (suspended)

- *2Display the Wait Cancellation Condition of Event Flag

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

7.12.3 Display the Ready Queue Status

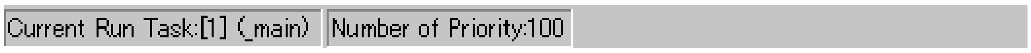
In the MR window, select Popup Menu - [Mode] -> [Ready Queue].



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

```
 Pri  RdyQ
   1  1(_main)
   2  2(_task2)
   5  15(_task15)
  10  16(_task16)
  15  17(_task17)
  20  18(_task18)
```

The following data is displayed in the status bar.



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

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

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

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

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

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

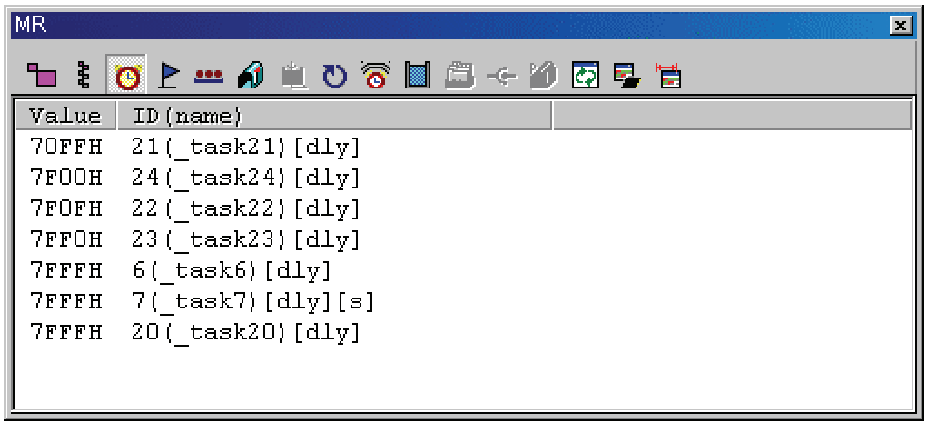
The function of each item is as described below. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

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

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

7.12.4 Display the Timeout Queue Status

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



```
Value   ID(name)
70FFH   21(_task21)[dly]
7F00H   24(_task24)[dly]
7F0FH   22(_task22)[dly]
7FF0H   23(_task23)[dly]
7FFFH   6(_task6)[dly]
7FFFH   7(_task7)[dly][s]
7FFFH   20(_task20)[dly]
```

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

The function of each item is as described below.
Tasks waiting at present are displayed in the descending order of timeout value. (When the realtime OS is MRxx conformed to uITRON specifications V.3.0.)

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

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

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

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

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

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

The function of each item is as described below.
Tasks waiting at present are displayed in the descending order of timeout value. (When the realtime OS is MRxx conformed to uITRON specifications V.4.0.)

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

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

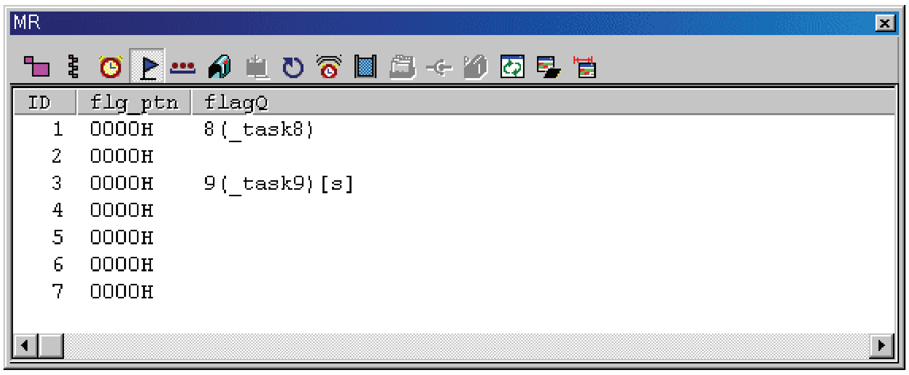
Character string	Wait state
[slp]	Wait due to tslp_tsk
[dly]	Wait due to dly_tsk
[flg]	Wait due to twai_flg
[sem]	Wait due to twai_sem
[mbx]	Wait due to trcv_mbx
[mpf]	Wait due to tget_mpf
[sdtq]	Wait due to tsnd_dtq
[rdtq]	Wait due to trcv_dtq
[vsdtq]	Wait due to vtsnd_dtq
[vrdtq]	Wait due to vtrcv_dtq

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

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

7.12.5 Display the Event Flag Status

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



```
ID  flg_ptn  flagQ
 1  0000H    8(_task8)
 2  0000H
 3  0000H    9(_task9)[s]
 4  0000H
 5  0000H
 6  0000H
 7  0000H
```

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

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

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

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

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

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

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

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

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

- The following are displayed in the Flgatr area:

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

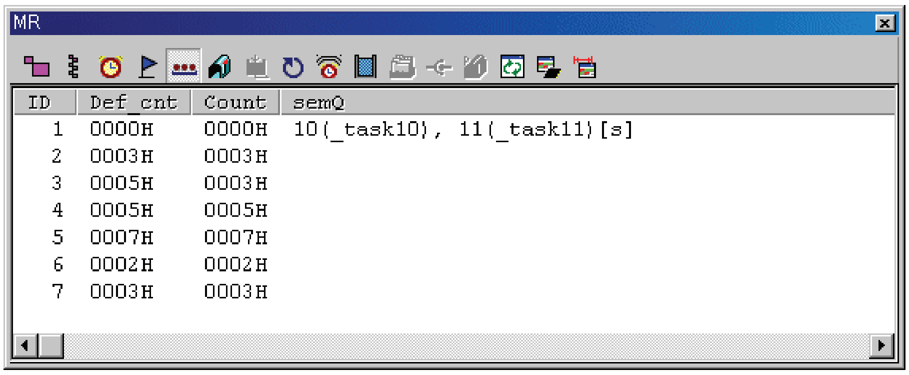
- When a task connected to the event flag queue is in the state of waiting with timeout enabled (waiting in twai_flg), a string "[tmo]", which indicates a state of waiting with timeout enabled, is appended to a string displayed in the Flag Queue field.
  When a task connected to the event flag queue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

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

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

7.12.6 Display the Semaphore Status

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



```
ID  Def_cnt  Count  semQ
 1  0000H    0000H  10(_task10), 11(_task11)[s]
 2  0003H    0003H
 3  0005H    0003H
 4  0005H    0005H
 5  0007H    0007H
 6  0002H    0002H
 7  0003H    0003H
```

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

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

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

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

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

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

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

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

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

- The following are displayed in the Sematr area:

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

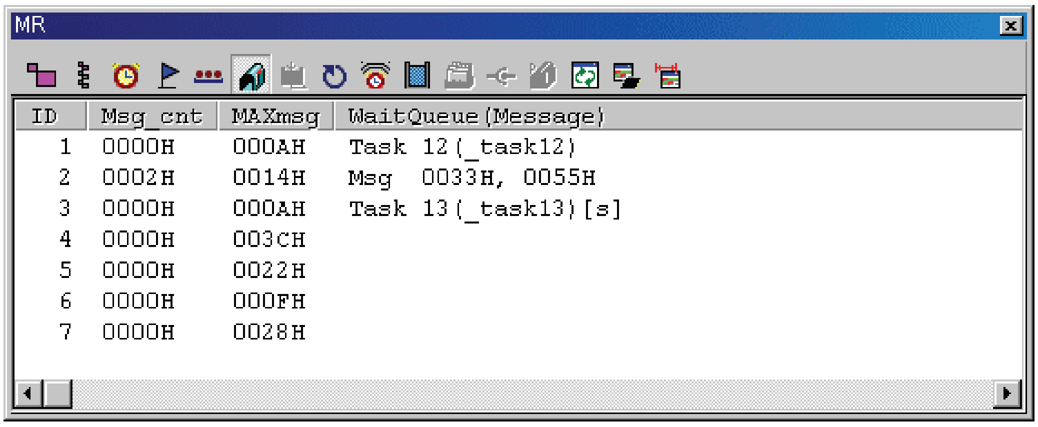
- When a task connected to the SEM queue is in the state of waiting with timeout enabled (waiting in twai_sem), a string "[tmo]", which indicates a state of waiting with timeout enabled, is appended to a string displayed in the Semaphore Queue field.
  When a task connected to the SEM queue is in the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

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

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

7.12.7 Display the Mailbox Status

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



```
ID   Msg_cnt  MAXmsg  WaitQueue(Message)
  1  0000H    000AH   Task 12(_task12)
  2  0002H    0014H   Msg  0033H, 0055H
  3  0000H    000AH   Task 13(_task13)[s]
  4  0000H    003CH
  5  0000H    0022H
  6  0000H    000FH
  7  0000H    0028H
```

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

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

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

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

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

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

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

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

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

- The following are displayed in the Mbxatr area:

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

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

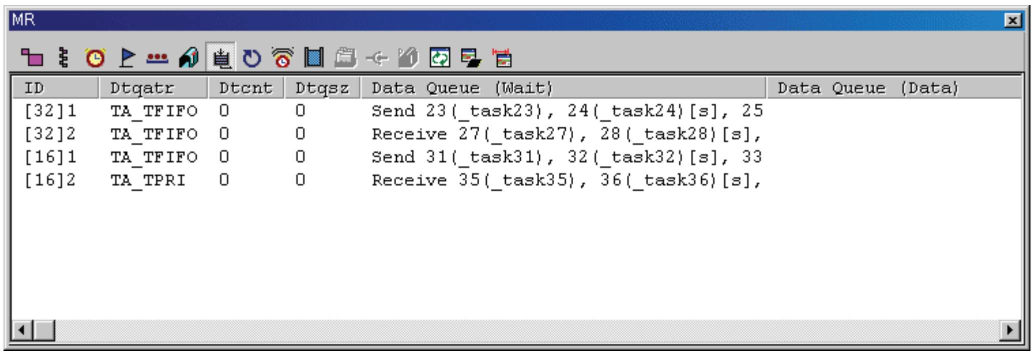
When a task connected to the mail box queue is in the state of forced waiting (Double waiting), a string "[s]", which indicates the state of double waiting, is appended to a string displayed in the Mailbox Queue (Wait) field.

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

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

7.12.8 Display the Data Queue Status

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



```
ID     Dtqatr    Dtcnt  Dtqsz  Data Queue (Wait)                      Data Queue (Data)
[32]1  TA_TFIFO  0      0      Send 23(_task23), 24(_task24)[s], 25
[32]2  TA_TFIFO  0      0      Receive 27(_task27), 28(_task28)[s],
[16]1  TA_TFIFO  0      0      Send 31(_task31), 32(_task32)[s], 33
[16]2  TA_TPRI   0      0      Receive 35(_task35), 36(_task36)[s],
```

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

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

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

- The display of the ID field varies depending on which one is specified, the standard data(32 bits) or the extended data(16 bits).
  MR308/4
  - If the standard data(32 bits), the ID field displays a string "[32]" and data queue ID number.
  - If the extended data(16 bits), the ID field displays a string "[16]" and data queue ID number.

  MR30/4
  - If the standard data(16 bits), the ID field displays a string "[16]" and data queue ID number.
  - If the extended data(32 bits), the ID field displays a string "[32]" and data queue ID number.

- The following are displayed in the Dtqatr area:

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

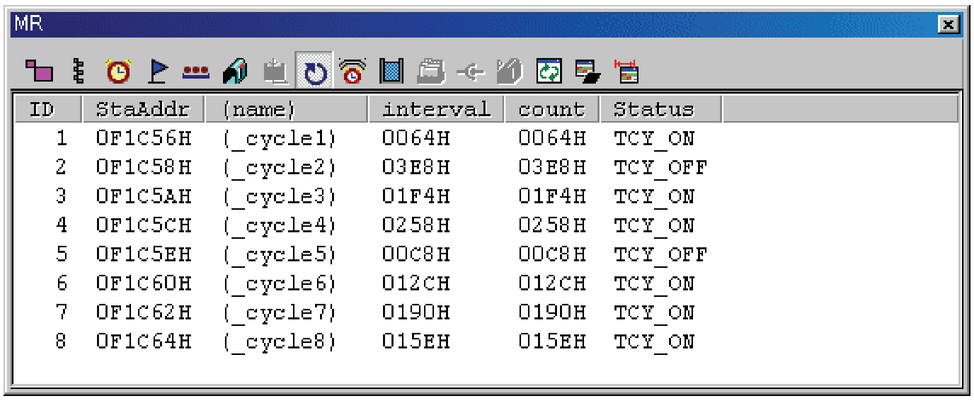
- The Data Queue (Wait) field displays a string "Send" if a task waiting for a message sending, and then displays the ID number and name of the task waiting for a message sending. Also, if a task waiting for a message receiving, displays a string "Receive" and then displays the ID number and name of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

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

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

7.12.9 Display the Cycle Handler Status

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



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

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

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

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

- The following are displayed in the Status area:

TCY_ON	Cycle handler enabled
TCY_OFF	Cycle handler disabled

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

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

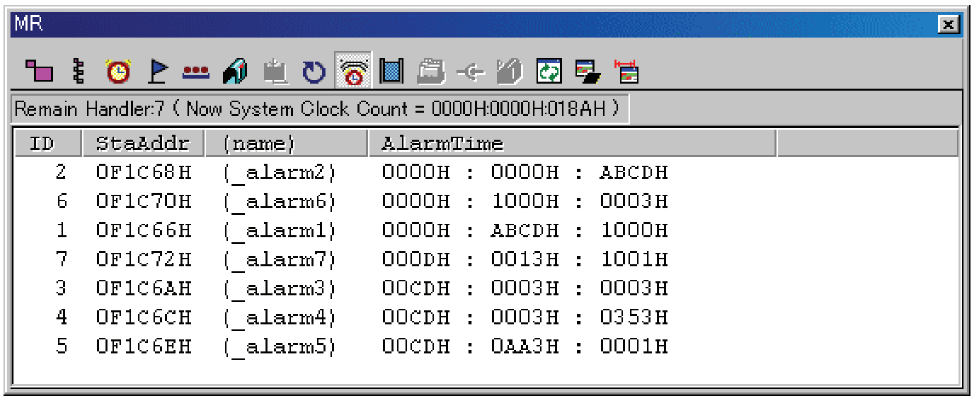
Item	Contents
ID	ID No. of cycle handler
Name	Name of cycle handler
Cycphs	The activation phase (by the millisecond)
Cyctim	The activation cycle time (by the millisecond)
Tmout	The amount of time by the millisecond remaining before the cyclic handler's next activation time
Status	Activity status of cycle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

- The following are displayed in the Status area:

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

7.12.10 Display the Alarm Handler Status

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



```
Remain Handler:7 ( Now System Clock Count = 0000H:0000H:018AH )
 ID  StaAddr   (name)     AlarmTime
  2  0F1C68H  (_alarm2)   0000H : 0000H : ABCDH
  6  0F1C70H  (_alarm6)   0000H : 1000H : 0003H
  1  0F1C66H  (_alarm1)   0000H : ABCDH : 1000H
  7  0F1C72H  (_alarm7)   000DH : 0013H : 1001H
  3  0F1C6AH  (_alarm3)   00CDH : 0003H : 0003H
  4  0F1C6CH  (_alarm4)   00CDH : 0003H : 0353H
  5  0F1C6EH  (_alarm5)   00CDH : 0AA3H : 0001H
```

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

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

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

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

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

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

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

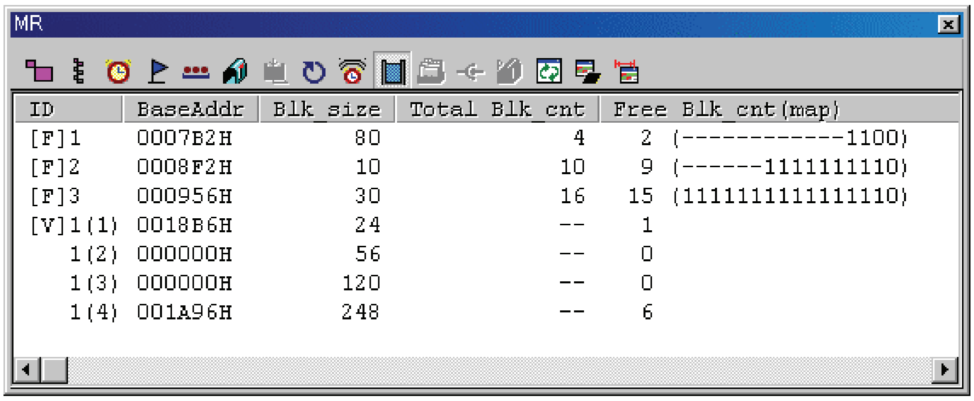
Item	Contents
ID	ID No. of alarm handler
Name	Name of alarm handler
Almtim	The amount of time by the millisecond remaining before the alarm handler's activation time
Status	Activity status of alarm handler

The following are displayed in the Status area:

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

7.12.11 Display the Memory Pool Status

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


MR

```
ID        BaseAddr  Blk_size  Total Blk_cnt  Free Blk_cnt(map)
[F]1      0007B2H        80              4   2  (------------1100)
[F]2      0008F2H        10             10   9  (------1111111110)
[F]3      000956H        30             16  15  (1111111111111110)
[V]1(1)   0018B6H        24             --   1
   1(2)   000000H        56             --   0
   1(3)   000000H       120             --   0
   1(4)   001A96H       248             --   6
```

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

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

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

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

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

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

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

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

- The following are displayed in the Mplatr area:

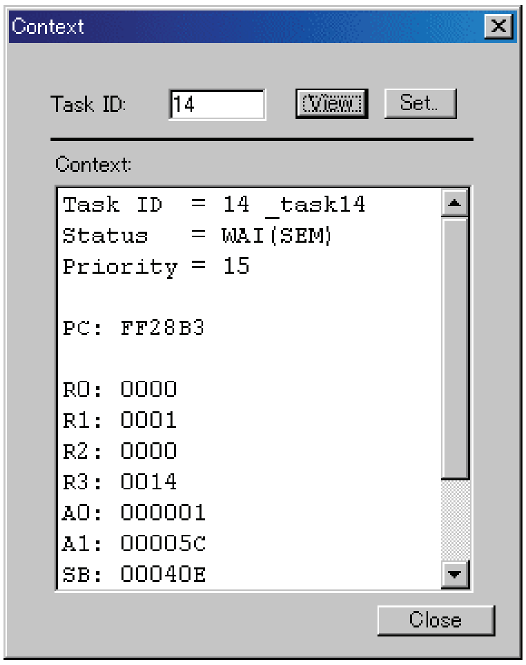
TA_TFIFO	Task wait queue is in FIFO order
TA_TPRI	Task wait queue is in task priority order

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

7.12.12 Display the Task Context

7.12.12.1 Display the Task Context

In the MR window, select Popup Menu - [Context...].
The Context dialog box is opened. The Context dialog box is used to reference/specify the context information of the specified task.
You can also open the Context dialog box by double-clicking the data display area in the task state display mode .



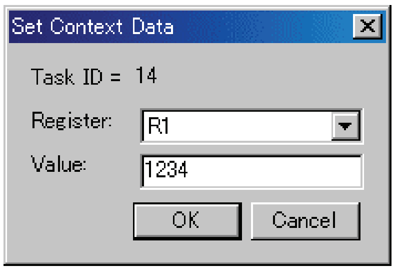
```
Task ID  = 14 _task14
Status   = WAI(SEM)
Priority = 15

PC: FF28B3

R0: 0000
R1: 0001
R2: 0000
R3: 0014
A0: 000001
A1: 00005C
SB: 00040E
```

Enter the task ID number in the Task ID field and click the View button (or press the Enter key).
The context of the specified task appears in the Context field.

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

7.12.12.2 Change the task context

Enter the task ID number in the Task ID field in the Context dialog and click the Set button. The Set Context dialog is opened.
The Set Context dialog is used to set the specified context register value of the specified task.



Task ID = 14
Register: R1
Value: 1234

Specify the register to be changed in the Register field list box and enter the value to be set in the "Value:" field.
If an expression description set in the "Value:" field is wrong, or if the specified value is outside the allowable range set for the specified register, an error occurs.

8. 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.

8.1 Table of Script Commands (classified by function)

8.1.1 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

8.1.2 File Operation Commands

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

8.1.3 Register Operation Commands

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

8.1.4 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)

8.1.5 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

8.1.6 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

8.1.7 Address Interrupt Break Setting Commands

Command Name	Short Name	Contents
ADdressInterruptBreak	ADIB	Sets and checks the address interrupt break

8.1.8 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

8.1.9 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

8.1.10 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

8.1.11 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

8.1.12 Map Commands

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

8.1.13 Clock Command

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

8.1.14 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

8.1.15 Real-time OS Command

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

8.1.16 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

8.2 Table of Script Commands (alphabetical order)

Command Name	Short Name	Contents
AddPath	-	Adds the search path
ADdressInterruptBreak	ADIB	Sets and checks the address interrupt break
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
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)
MR*	-	Displays status of realtime OS (MRxx)
OverStep	O	Overstep execution of source lines
OverStepInstruaction	OI	Overstep execution of instructions
Path	-	Sets and checks the search path

Pause	-	Waits for user input
Print	-	Check value of specified C variable expression.
Radix	-	Sets and checks the radix for numerical input
Register	R	Checks and sets a register value
Reload	-	Re-downloads the target program
Reset	-	Resets the target MCU
Return	RET	Executes a source line return
ReturnInstruction	RETI	Executes an instruction return
Scope	-	Sets and checks the effective local symbol scope
Script	-	Opens and executes a script file
Section	SEC	Checks section information
Set	-	Set specified data in specified C variable expression
SetMemoryByte	MB	Checks and changes memory contents (in 1-byte units)
SetMemoryLword*	ML	Checks and changes memory contents (in 4-byte units)
SetMemoryWord*	MW	Checks and changes memory contents (in 2-byte units)
Sleep	-	Halts for user input until the specified time has elapsed
SoftwareBreak	SB	Sets and checks software breaks
SoftwareBreakClear	SBC	Clears software breaks
SoftwareBreakClearAll	SBCA	Clears software breaks
SoftwareBreakDisable	SBD	Disables software breakpoints
SoftwareBreakDisableAll	SBDA	Disables all software breaks
SoftwareBreakEnable	SBE	Enables software breakpoints
SoftwareBreakEnableAll	SBEA	Enables all software breaks
Status	-	Checks the operating status of the MCU
Step	S	Step execution of source line
StepInstruction	SI	Step execution of instructions
Stop	-	Stops program execution
Symbol	SYM	Checks assembler symbols
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

9. 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.

9.1 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.

9.1.1 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).

9.1.2 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.

9.1.3 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).

9.1.4 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.

9.1.5 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.

9.2 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

9.2.1 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

9.2.2 Symbols and labels

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

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

Product Name	Notes
The debugger for M32R, The debugger for M32C, The debugger for M16C/R8C,	• 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.

9.2.2.1 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.

9.2.2.2 Priority levels of labels and symbols

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

- Conversion of address values

1. Local labels
2. Global labels
3. Local symbols
4. Global symbols
5. Local labels outside scope
6. Local symbols outside scope

- Conversion of data values

1. Local symbols
2. Global symbols
3. Local labels
4. Global labels
5. Local labels outside scope
6. Local symbols outside scope

- Conversion of bit values

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

9.2.3 Macro Variables

Macro variables are defined by assign statements in the script file. See Section"9.1.2Assign 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.

9.2.4 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.

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

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

9.2.5 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.

9.2.6 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.

9.2.7 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.

9.2.8 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

10. C/C++ Expressions

10.1 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.", ...

10.1.1 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.

10.1.2 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
```

10.1.3 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.

10.1.4 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.

10.1.5 Reference

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

10.1.6 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.

10.1.7 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.

10.1.8 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.

10.1.9 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.

10.1.10 Parentheses

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

10.1.11 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.

10.1.12 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.

10.1.13 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.

10.1.14 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.

10.1.15 Function Name

Function names that begin with English alphabets as required
under C conventions can be used.
In the case of C++, no function names can be used.

10.1.16 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).

10.1.17 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.

10.2 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.

10.2.1 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)
```

10.2.2 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)
```

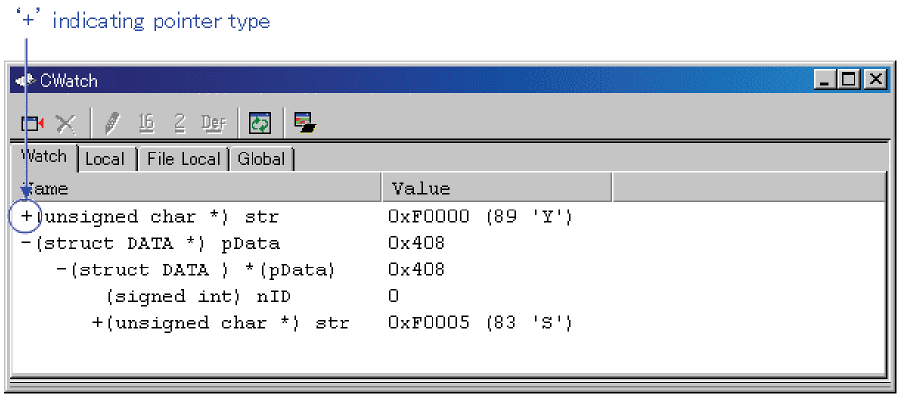
10.2.3 Pointer Types

- 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.



```
Name                              Value
+(unsigned char *) str            0xF0000 (89 'Y')
-(struct DATA *) pData            0x408
   -(struct DATA ) *(pData)       0x408
       (signed int) nID           0
      +(unsigned char *) str      0xF0005 (83 'S')
```

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 '-'.

10.2.4 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)
```

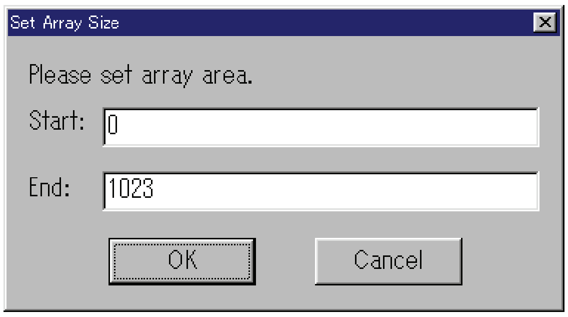
- When the result of calculation is a char [ ] type, it is displayed as follows:

```
        (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 "10.2.3 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.


Set Array Size

Please set array area.

Start: 0

End: 1023

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.

10.2.5 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)
```

10.2.6 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)
```

10.2.7 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)
```

10.2.8 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)
```

10.2.9 Syntax Errors

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

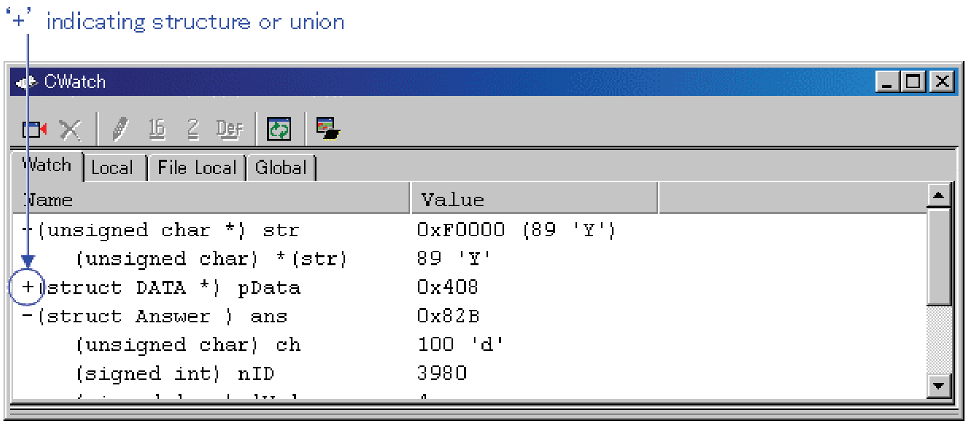
```
        () str*(p = <syntax error>(all Radices)
        (where str*(p is the syntax error)
```

10.2.10 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).



```
Name                              Value
-(unsigned char *) str            0xF0000 (89 'Y')
    (unsigned char) *(str)        89 'Y'
+(struct DATA *) pData            0x408
-(struct Answer ) ans             0x82B
    (unsigned char) ch            100 'd'
    (signed int) nID              3980
```

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
```

11. 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 "---".

12. Attention

12.1 Common Attention

12.1.1 File operation on Windows

The following points should be noted:

1. File Name and Directory Name
   - 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.

12.1.2 Area where software breakpoint can be set

The area which can be set for software breakpoint varies depending on the type of MCU.

12.1.2.1 The debugger for M32C

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

Processor Mode	Can be set area
Single Chip	Internal RAM, Internal ROM
Memory Extension	Internal RAM, Internal ROM Emulation memory (Only the Internal area)
Micro Processor	Internal RAM Emulation memory (Only the Internal area)

To stop the target program outside the emulation memory area, use the Come execution.

ATTENTION

- The setting for emulation memory is available when an emulation memory board is connected to the emulator.
- Software breaks can not be set to the internal ROM, when the target program is running.

12.1.2.2 The debugger for M16C/R8C

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

Processor Mode	Can be set area
Single Chip	Internal RAM, Internal ROM
Memory Extension	Internal RAM, Internal ROM Emulation memory
Micro Processor	Internal RAM Emulation memory

ATTENTION

- The setting for emulation memory is available when an emulation memory board is connected to the emulator.
- Software breaks can not be set to the internal ROM, when the target program is running.

12.1.3 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.
- 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.
- The following values can be substituted for the bit-fields.
  - integer constants, character constants, and enumerators
  - variables of bool types, characters types, integers types, and enumeration types
  - bit-field

  When the substituted value is larger than the size of the bit-field, it will be truncated.
- The bit-field member allocated in the SFR area might not be transformed into a correct value.
- While the target program is running, values of local variables and bit-fields cannot be modified.

12.1.4 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.
- The pointers for a member function can not be referred correctly in C watch window.

12.1.5 Option settings for download modules

These options, which can be set in "Debug Settings" dialog box, are invalid for this debugger:

- Offset : specified value is regarded as '0'
- Access size : specified value is regarded as '1'
- Perform memory verify during download : Not supported.

12.1.6 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.

12.1.7 Synchronized debugging

Synchronized debugging function is not available.

12.1.8 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.

12.2 Attention of the M32C Debugger

12.2.1 Stack area used by the emulator

The emulator uses the interrupt stack area as its work area (20 bytes).
When debugging, allocate a sufficient interrupt stack area consisting of the regularly used size plus 20 bytes.

12.2.2 Interrupt stack pointer when resetting the target program

The emulator sets the interrupt stack pointer (ISP) to 0500h when resetting the target program.
Remember that the interrupt stack pointer (ISP) is set to 0000h on a unit at the production stage.

12.2.3 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.
Refer to "12.4 Options for compiler, assembler, and linker"

The compiler that can be used by M32C debugger:

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

12.2.4 Target MCU HOLD terminal

When the target MCU HOLD terminal is set to LOW, you cannot stop execution of the target program.
Set the HOLD terminal to High and try to stop the target program again.
Even if the HOLD terminal is set to LOW for a short period of time, you may stop the target program with the HOLD terminal set to LOW.
If this happens, try to stop the target program again.

12.2.5 Hardware Event

If you specify word-length (2-byte length) data from an odd address as an event in the following data accesses, the event is not detected. Also, even when any other bit of the address that contains a specified bit is accessed during bit access, the event may become effective.

- Hardware Break Event
- Real-time Trace Event

- Examples to specify the event for data access with its value comparison.

Address	Access size	16bit data bus	8bit data bus
Even Address	WORD ex.: mov.w #1234h, 8000h	Address: 8000h Data: 1234h Data Mask: Don't care	<-
Even Address	BYTE ex.: mov.b #34h, 8000h	Address: 8000h Data: 34h Data Mask: 00FFh	<-
Odd Address	WORD ex.: mov.w #1234h, 8001h	Not supported	<-
Odd Address	BYTE ex.: mov.b #34h, 8001h	Address: 8001h Data: 3400h Data Mask: FF00h	Address: 8001h Data: 34h Data Mask: 00FFh

12.2.6 CPU rewrite

- Please don't rewrite the block 0 (FFC000h - FFFFFFh). If you did, emulator gets out of control.
- When you enabled CPU rewrite mode, you can not use the debug functions below:
  - Setting address match interrupt break points
  - Setting software break points to the internal flash ROM
  - COME execution to the internal flash ROM
- You can not use the functions below to the CPU rewrite control program (from setting CPU rewrite mode select bit to clearing it). If you use these functions, it may be unable to shift to CPU rewrite mode, or the contents of a flash ROM may be unable to be read to normalcy.
  - Single step
  - Software break points
  - Hardware break points
  - COME execution

To see the data which were rewritten by the program, break the program out of the CPU rewrite control program and see by dump window, memory window, and so on.

12.3 Attention of the M16C/R8C Debugger

12.3.1 Map of stack area used by the compact emulator

The compact emulator uses the interrupt stack area as its work area (20 bytes).
When debugging, allocate a sufficient interrupt stack area consisting of the regularly used size plus 20 bytes.

12.3.2 Interrupt stack pointer when resetting the target program

The emulator sets the interrupt stack pointer (ISP) to 0500h when resetting the target program.
Remember that the interrupt stack pointer (ISP) is set to 0000h on a unit at the production stage.

12.3.3 Options for compiler, assembler, and linker

The information may not be downloaded/debugged normally depending on the option designation of the compiler, assembler, and linker.
Please refer to the following for the option specification.
Refer to "12.4 Options for compiler, assembler, and linker"

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

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

12.3.4 TASKING C Compiler

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

12.3.5 Target MCU HOLD terminal

When the target MCU HOLD terminal is set to LOW, you cannot stop execution of the target program. Set the HOLD terminal to High and try to stop the target program again. Even if the HOLD terminal is set to LOW for a short period of time, you may stop the target program with the HOLD terminal set to LOW. If this happens, try to stop the target program again.

12.3.6 Hardware break function

While running program, the following operations are not performed:

- execute BreakMode command
- open H/W break point setting window

12.3.7 Hardware Event

If you specify word-length (2-byte length) data from an odd address as an event in the following data accesses, the event is not detected. Also, even when any other bit of the address that contains a specified bit is accessed during bit access, the event may become effective.

- Hardware Break Event
- Real-time Trace Event

- Examples to specify the event for data access with its value comparison.

Address	Access size	16bit data bus	8bit data bus
Even Address	WORD ex.: mov.w #1234h, 8000h	Address: 8000h Data: 1234h Data Mask: Don't care	<-
Even Address	BYTE ex.: mov.b #34h, 8000h	Address: 8000h Data: 34h Data Mask: 00FFh	<-
Odd Address	WORD ex.: mov.w #1234h, 8001h	Not supported	<-
Odd Address	BYTE ex.: mov.b #34h, 8001h	Address: 8001h Data: 3400h Data Mask: FF00h	Address: 8001h Data: 34h Data Mask: 00FFh

12.3.8 Memory space expansion

- Memory mapping
  If you select "4MB Mode" for Memory Space Expansion in MCU setting dialog, the memory map depends on the other setting of mcu:

Processor mode	PM13	OFS	Access area of the MCU	Bank0 - Bank5	Bank6	Bank7
Memory Expansion Mode	1	0	40000h - 7FFFFh	EXT	EXT	MAP
		1	40000h - 7FFFFh	EXT	EXT	MAP
	0	0	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	EXT	MAP
		1	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	MAP	---
Microprocessor Mode	---	0	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	EXT	---
			C0000h - FFFFFh	---	---	MAP
		1	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	MAP	---

PM13 : bit3 of processor mode register1 (00005h)
OFS : offset bit (bit2 of data bank register (0000Bh))
EXT : memory access to the target system
MAP : memory access to the area which depends on MAP setting in MCU Settingts dialog (INT means emulation memory in the compact emulator, EXT means target system)

Memory access to the expanded area
C watch window, Memory window, and other debugging windows can not show correct values, when it shows the expanded area of the memory space expansion fuction. Debugging windows does not consider the bank register.
To see this area, please use script commands listed below and specify the bank, the status of offest bit and the address for M16C:

```
DumpByte2, DumpWord2, or DumpLword2
```

You can also use the commands below:

```
DumpByte2, DumpWord2, DumpLword2, SetMemoryByte2, SetMemoryWord2,
SetMemoryLword2, FillByte2, FillWord2, FillLword2, Move2, MoveWord2
```

Caution

The function below may not work as expected when these functions access to the expanded area. Debbugger can not distinguish the bank information from BUS information.
- RAM monitor function (RAM Monitor window, C Watch window, etc)
- Coverage measurement function (Coverage window, Coverage command, etc)
- Memory protect function (Protect window, Protect command, etc)
- Hardware event (H/W break event, Real-time trace event, Time measurement event, etc)

About hardware event, you can specify SameAND combination for the event you need and the status of bank select register, to detect the access to collect bank.

12.3.9 CPU rewrite

- Please don't rewrite the last block. If you did, emulator gets out of control.
- When you enabled CPU rewrite mode, you can not use the debug functions below:
  - Setting software break points to the internal flash ROM
  - COME execution to the internal flash ROM
- You can not use the functions below to the CPU rewrite control program (from setting CPU rewrite mode select bit to clearing it). If you use these functions, it may be unable to shift to CPU rewrite mode, or the contents of a flash ROM may be unable to be read to normalcy.
  - Single step
  - Software break points
  - Address match interrupt break points
  - Hardware break points
  - COME execution

  To see the data which were rewritten by the program, break the program out of the CPU rewrite control program and see by dump window, memory window, and so on.

12.4 Options for compiler, assembler, and linker

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

12.4.1 When Using NCxx

When -O, -OR or -OS option is specified at compilation, the source line information may not be generated normally due to optimization, causing step execution to be operated abnormally.
To avoid this problem, specify -ONBSD (or -Ono_Break_source_debug) option together with -O, -OR or -OS option.

12.4.2 When Using the IAR C Compiler (EW)

Please specify the project setting by following process.

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

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

3. Build your program after the setting above.

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

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

Please specify the project setting by following process.

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

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

3. Build your program after the setting above.

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

12.4.4 When Using the IAR C Compiler (ICC)

12.4.4.1 Specify the Option

Please compile according to the following procedures and link.

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

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

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

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

12.4.4.2 Command Execution Examples

The following shows examples of entering commands depending on the product

- The debugger for M32C

```
>ICCMC80 -r file1.c<Enter>
>ICCMC80 -r file2.c<Enter>
>XLINK -o filename.695 -f lnkm80.xcl file1 file2<Enter>
```

- The debugger for M16C/R8C

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

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

12.4.5 When Using the TASKING C Compiler (EDE)

Please specify the project setting by following process.

1. Select menu - [EDE]->[C Compiler Option]->[Project Options...]. The "M16C C Compiler Options [Project Name]" dialog opens.
   Please set as follows by this dialog.
   - Optimeze Tab
     Please specify "No optimization" by Optimization level.
   - Debug Tab
     Please check only ""Enable generation of any debug information(including type checkeing)"" and "Genarate symbolic debug information".

2. Select menu - [EDE]->[Linker/Locator Options...]. The "M16C Linker/Locator Options [Project Name]" dialog opens.
   Please set as follows by this dialog.
   - Format Tab
     Please specify "IEEE 695 for debuggers(abs)" by Output Format.
3. Build your program after the setting above.

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

12.4.6 When Using the TASKING C Compiler (CM)

12.4.6.1 Specify the Option

Please specify "-g" and "- O0" options when compiling.
In the options other than the above-mentioned, the operation check is not done. Please acknowledge that the options other than the above-mentioned cannot be recommended.

12.4.6.2 Command Execution Examples

The following shows examples of entering commands.

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

[MEMO]

M16C R8C Compact Emulator Debugger V.1.03
User's Manual

Publication Date:	Jul. 01, 2007	Rev.1.00
Published by:	Sales Strategic Planning Div. Renesas Technology Corp.	
Edited by:	Microcomputer Tool Development Department Renesas Solutions Corp.	

M16C R8C Compact Emulator Debugger V.1.03 User's Manual

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

REJ10J1609-0100