RENESAS

High-performance Embedded Workshop

Real-time OS aware debugging

Summary

This document describes how to use the real-time OS aware debugging making use of High-performance Embedded Workshop.

The real-time OS aware debugging can be used for the following real-time OSes.

Corresponding Real-Time OS:

- HI7200/MP
- HI7750/4
- HI7700/4
- HI7000/4
- RI600/4
- M3T-MR30/4
- HI1000/4
- Smalight OS
- ThreadX
- NORTi
- osCAN
- µC/OS-II
- embOS
- µT-Kernel
- TOPPERS/ASP
- TOPPERS/JSP

Contents

1. Introduction 3

2. Overview of Real-time OS aware debugging 4

2.1 Operating Environment 4

3. Quick Start Guide 7

3.1 Downloading a Program using Real-time OS 7
3.2 Selecting the OS Definition File 8
3.3 Executing the Program 9
3.4 Referring to the Task State 10
3.5 Displaying the Task Execution History 11
3.6 Displaying the Task Execution Time 11

4. Functions 12

4.1 List of Functions 12
4.2 Functional Descriptions 14

5. Notes 28

6. FAQ 30

7. Website and Support 31

1. Introduction

This application note is for customers using the real-time OS aware debugging of High-performance Embedded Workshop.

This document describes the functions using the following environment. It is assumed that the real-time OS and High-performance Embedded Workshop have been installed in the host PC in advance. For the installation method, refer to the manual for each product.

[Environment in which operation was confirmed]

(1) Host PC: Windows Vista®, Windows®XP, Windows®2000

(2) High-performance Embedded Workshop V.4.06.00

(3) Real-time OS for SuperH Family manufactured by Renesas Technology Corp.
[HI7000/4 V.2.02 Release 05]

(4) SuperH Family RISC engine simulator/debugger V.9.09.00

2. Overview of the Real-time OS aware debugging

The real-time OS aware debugging are functions for supporting task-level debugging of the real-time OS application on High-performance Embedded Workshop. High-performance Embedded Workshop incorporates these functions as standard functions. The real-time OS does not have to be built again and it does not have an overhead. (Debug patches are unnecessary.)

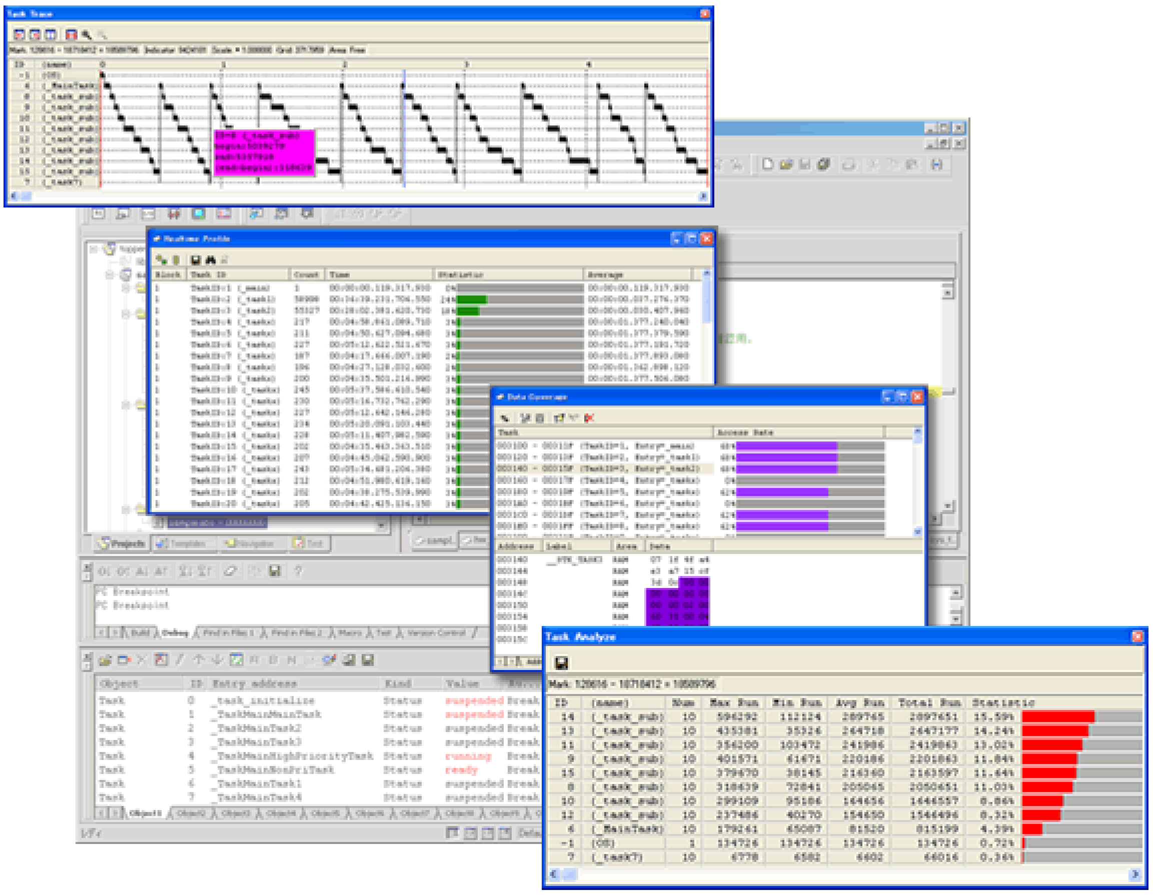


Figure 2.1 Real-time OS aware debugging

2.1 Operating Environment

The operating environment of the real-time OS aware debugging is as follows:

- Host machine

  Windows Vista®, Windows®XP, Windows®2000

- High-performance Embedded Workshop

  V.4.06.00 or later

- **Corresponding Real-Time OS**

Table 2.1 List of Corresponding Real-Time OS

Real-time OS		Microcomputers
Real-time OS for SuperH Family manufactured by Renesas [HI7200/MP]	V.1.00 Release 03 or later	SH2A-DUAL
Real-time OS for SuperH Family manufactured by Renesas [HI7750/4]	V.2.02 Release 04 or later	SH-4, SH-4A
Real-time OS for SuperH Family manufactured by Renesas [HI7700/4]	V.2.03 Release 02 or later	SH4AL-DSP, SH-3, SH3-DSP
Real-time OS for SuperH Family manufactured by Renesas [HI7000/4]	V.2.02 Release 04 or later	SH-2, SH2-DSP, SH-2A, SH-1
Real-time OS for RX Family manufactured by Renesas [RI600/4]	V.1.00 Release 00 or later	RX Family*
Real-time OS for M16C Series, R8C Family manufactured by Renesas [M3T-MR30/4]	V.4.00 Release 00 or later	M16C/60, M16C/50, M16C/30, M16C/20, M16C/10, M16C/Tiny Series
		R8C Family
Real-time OS for H8SX, H8S Family manufactured by Renesas [HI1000/4]	V.1.05 Release 01 or later	H8SX Family
		H8S Family
Smalight OS manufactured by Renesas Northern Japan Semiconductor, Inc.	V3.10 or later	SH-2
		RX Family*
		M16C/60, M16C/30, M16C/Tiny Series
		R8C Family
		H8SX Family
		H8S Family
		H8 Family
ThreadX manufactured by Express Logic, Inc.	G5.1.5.0	SH-2A
NORTi manufactured by MiSPO Co., Ltd.	Version4 Release 1.20	SH-4A, SH-2A
	Version4 Release 1.14	H8SX Family
		H8S Family
osCAN manufactured by Vector Informatik GmbH	Version 3.05	M16C/60, M16C/Tiny Series
		R8C Family
µC/OS-II manufactured by Micrium	V2.86 (AN-Renesas-SH7201)	SH-2A
	V2.89	RX Family*
	V2.83 (AN-1662)	M16C/60 Series
	V2.86 (AN-Renesas-H8S2215R, AN-Renesas-H8S2472, AN-Renesas-H8SX1664)	H8SX Family
		H8S Family
embOS* manufactured by SEGGER Microcontroller	Version 3.80g	SH-2A
µT-Kernel by T-Engine Forum	v1.00.00	M16C/62P Group
TOPPERS/ASP by TOPPERS Project	Release 1.3.1	SH-2A
	Release 1.3.2	M16C/60 Series
TOPPERS/JSP by TOPPERS Project	Release 1.4.3	H8/3048 Group
		H8S/2350 Group

Corresponding Debuggers Table 2.2 List of Corresponding Debuggers

Debugger	Supported Microcomputers	Debugger Version	Support Status
Simulator/debugger	SuperH Family	V.9.08.00 or later	Already supported
	RX Family*	V.1.00.00 or later	
	H8SX, H8S, H8 Family	V.5.07.00 or later	
E1 emulator debugger	RX Family*	—	Planned to be supported
E20 emulator debugger	RX Family*	—	Planned to be supported
E10A-USB emulator debugger	SuperH Family, H8SX Family, H8S Family	V.3.00 Release 00 or later	Already supported
	SH2A-DUAL, SH4A-MULTI	—	Planned to be supported
E8a emulator debugger	M16C Series, R8C Family, H8 Family	V.1.02 Release 00 or later	Already supported
E100 emulator debugger	M16C Series, R8C Family, H8SX Family	V.1.00 Release 00 or later	Already supported
E200F emulator debugger	SH-2A	—	Planned to be supported

- **Corresponding Compilers**
  — C/C++ compiler package for SuperH Family V.9.02 Release 00 or later
  — C/C++ compiler package for RX Family V.1.00 Release 00 or later
  — C/C++ compiler package for M16C Series, R8C Family [M3T-NC30WA] V.5.43 Release 00 or later
  — C/C++ compiler package for H8SX, H8S, H8 Family V.6.02 Release 00 or later

Note: * Supported by High-performance Embedded Workshop V.4.07.00 or later.

3. Quick Start Guide

This section describes how to use the basic functions (refer to the object state, display the task execution history, etc.) of the real-time OS aware debugging of High-performance Embedded Workshop.

Workspaces of High-performance Embedded Workshop using the real-time OS should be prepared in advance. For the method of creating a workspace, refer to the manual or online help of both the real-time OS and High-performance Embedded Workshop.

In the subsequent descriptions, the real-time OS "HI7000/4" and the SuperH Family simulator/debugger which are both manufactured by Renesas Technology Corp. are used. The real-time OS aware debugging is used in a similar manner even when using other real-time OS products or debuggers.

3.1 Downloading a Program using Real-time OS

Launch High-performance Embedded Workshop, open a workspace (a file whose extension is "hws") using HI7000/4, and select [Download Modules] from the [Debug] menu in High-performance Embedded Workshop to download programs to the simulator/debugger.

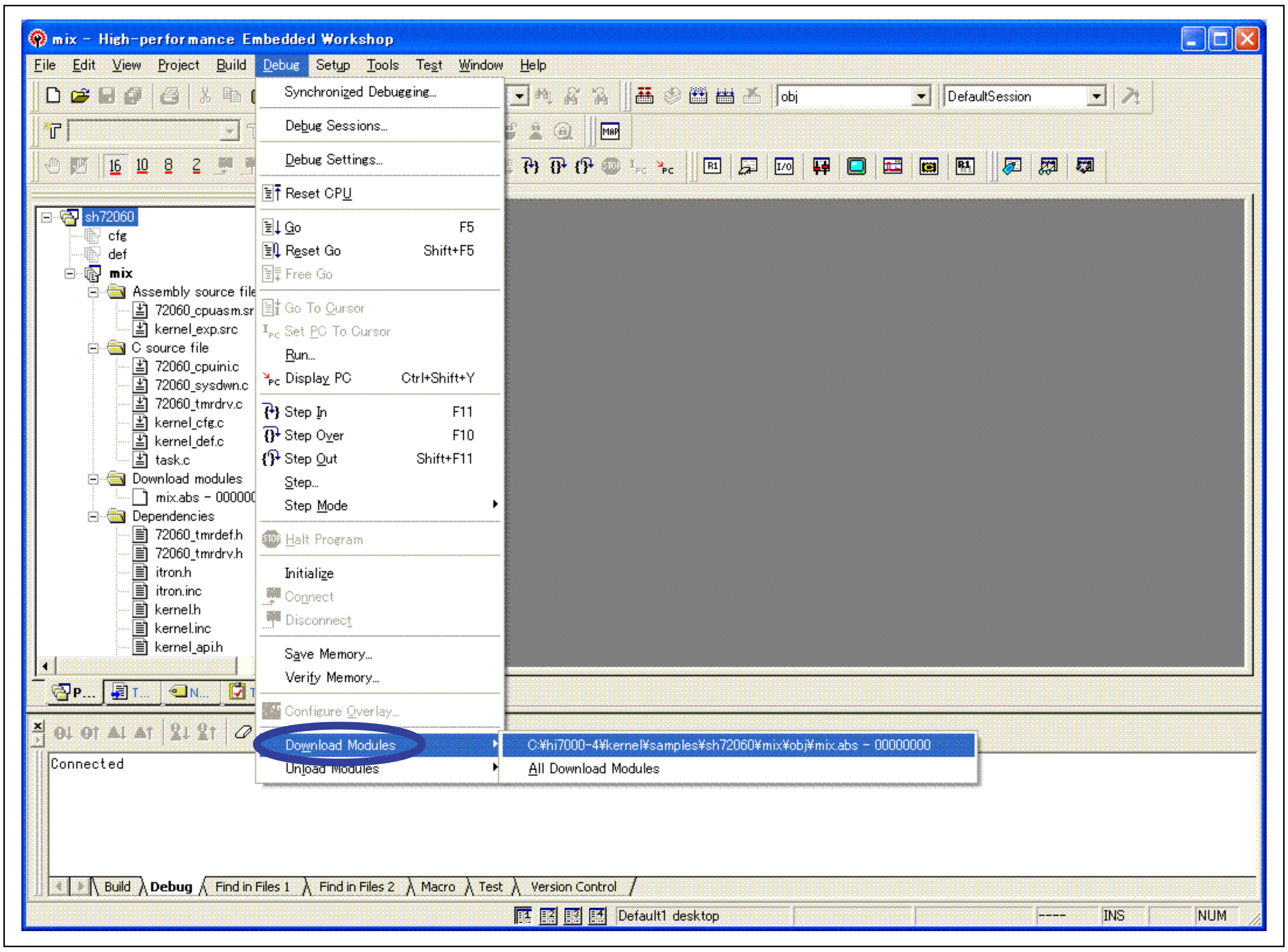


Figure 3.1 Downloading a Program

3.2 Selecting the OS Definition File

When program download is executed, the Select OS Definition File dialog box is automatically displayed. Select the relevant real-time OS and click the [OK] button.

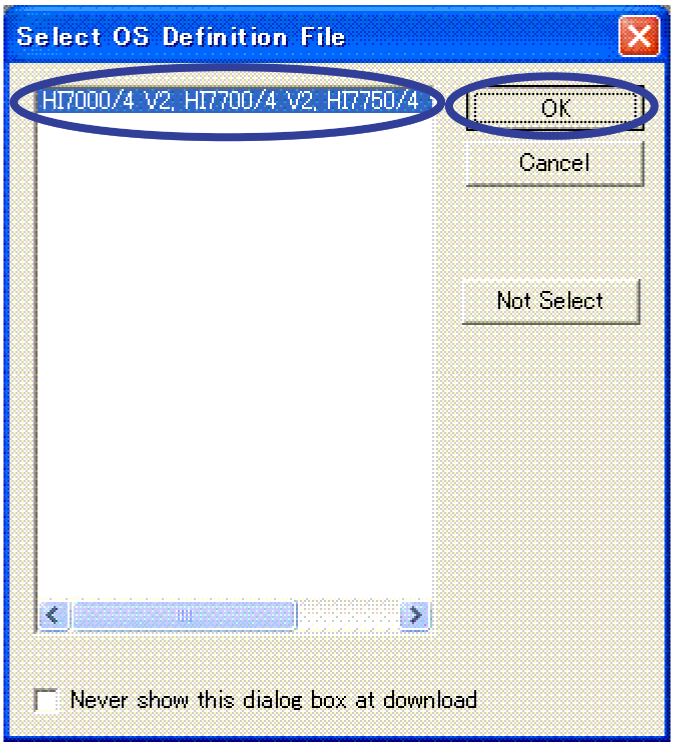
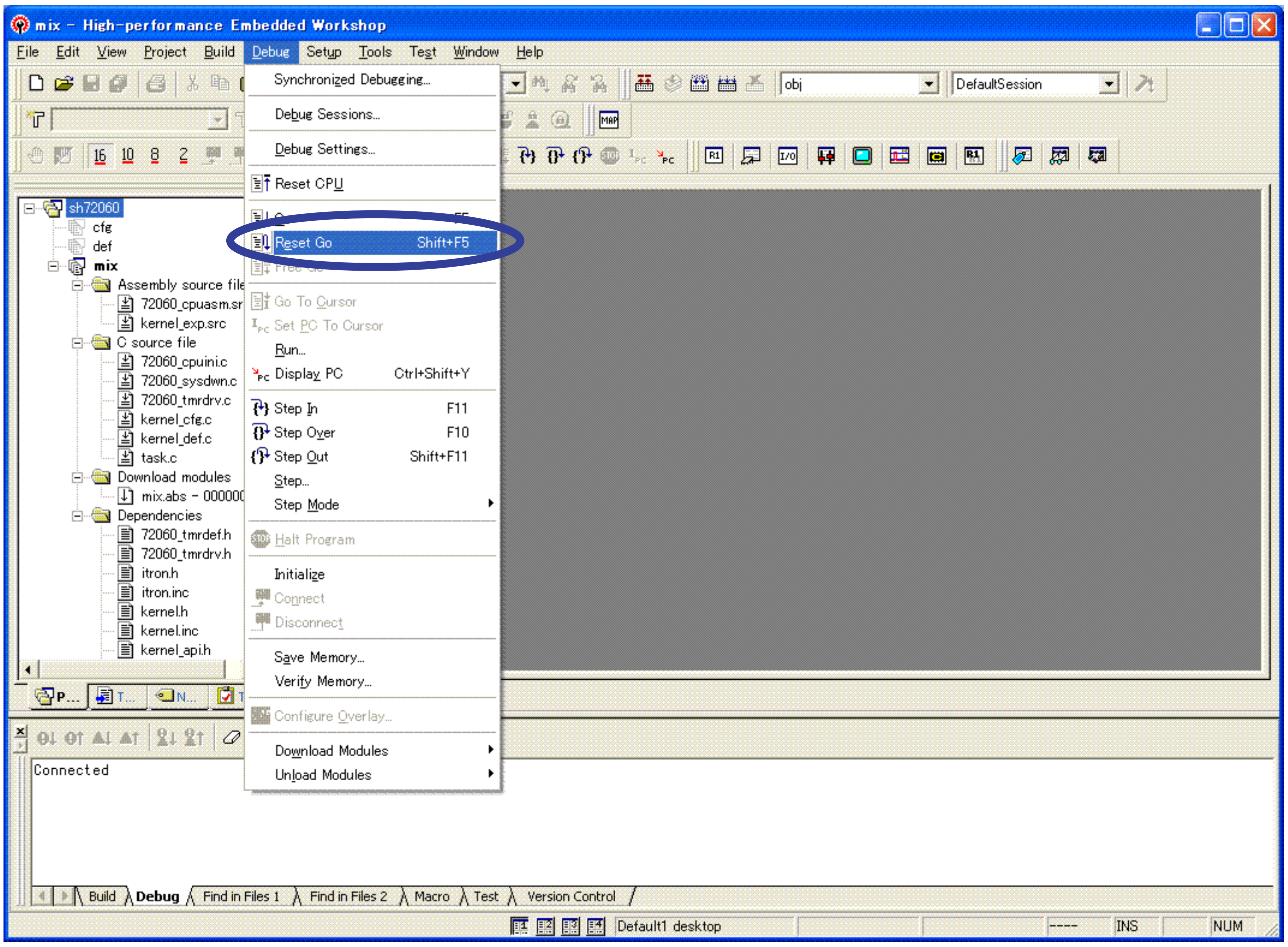

Select OS Definition File
HI7000/4 V2, HI7700/4 V2, HI7750/4
OK
Cancel
Not Select
Never show this dialog box at download

Figure 3.2 Selecting the OS Definition File

3.3 Executing the Program

Select [Reset Go] from the [Debug] menu to execute the program. Then, to complete initialization of the HI7000/4 kernel, wait for a specific period before selecting [Halt Program] from the [Debug] menu to stop the program.



```
Connected
```

Figure 3.3 Executing the Program

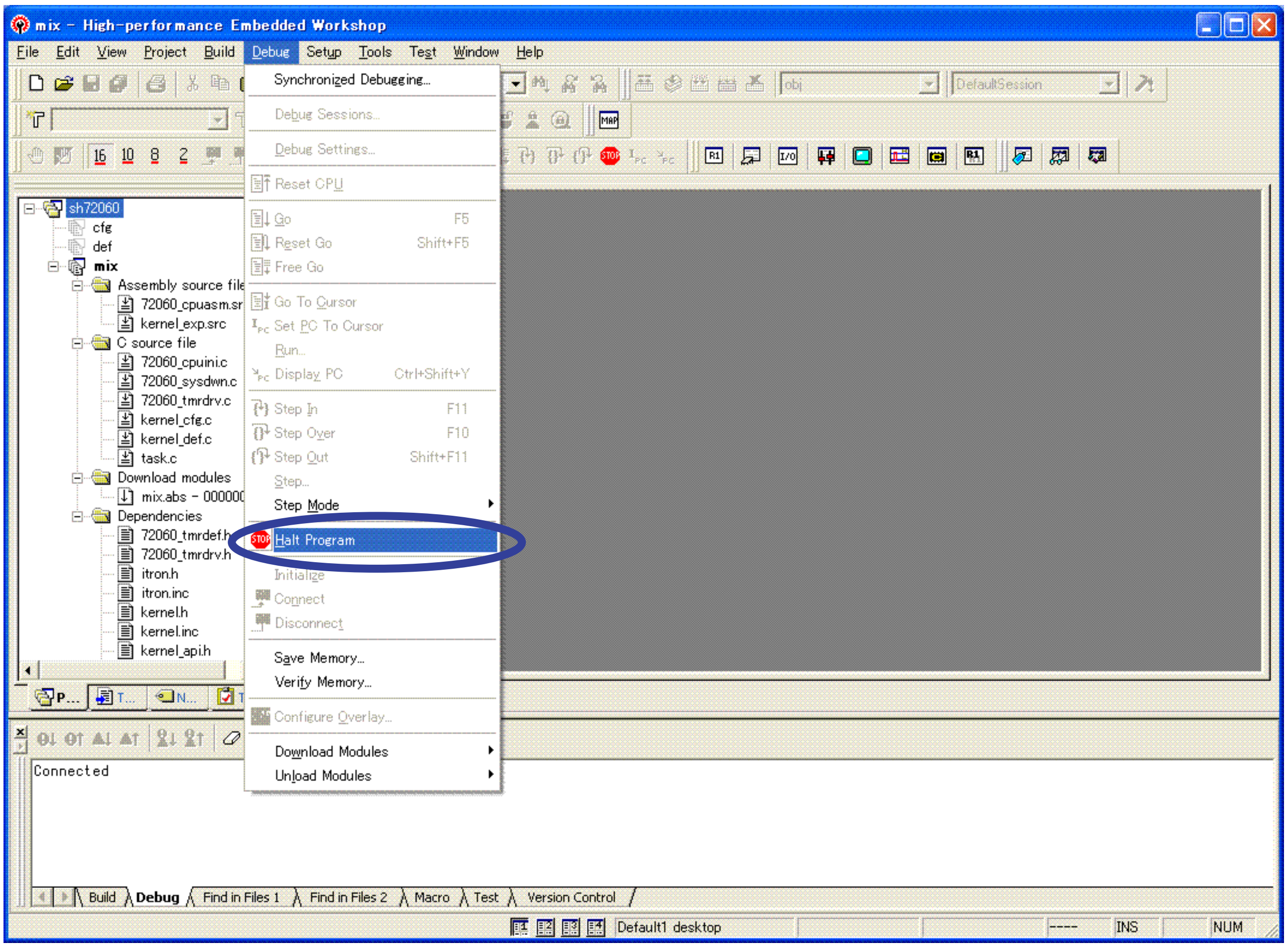
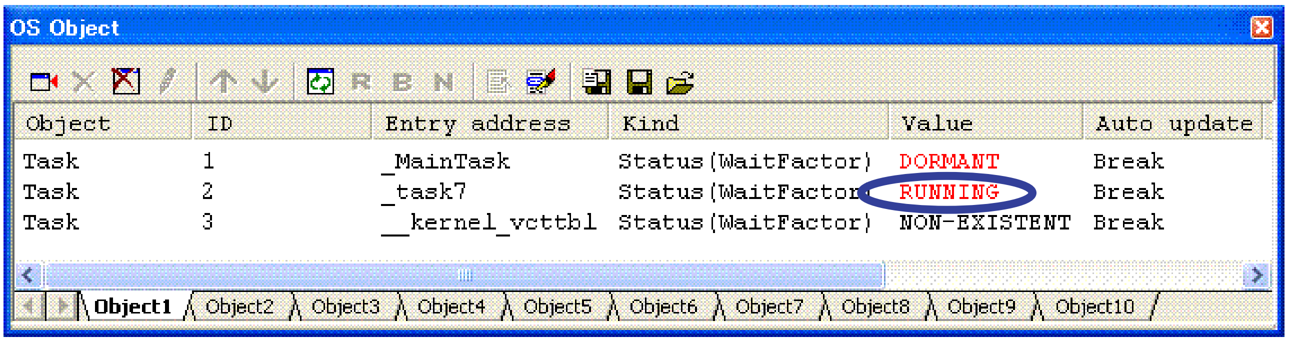


Figure 3.4 Halting the Program

3.4 Referring to the Task State

Select [RTOS]->[OS Object] from the [View] menu to open the OS Object window in which the task state can be referenced. In this example, it is obvious that task ID 2 is in the RUNNING state.



OS Object

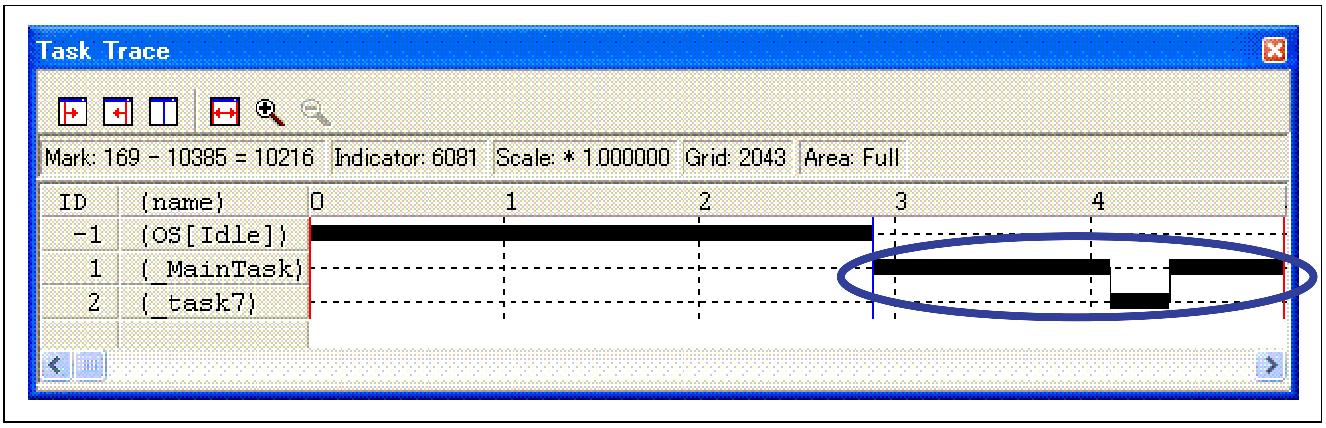
Object	ID	Entry address	Kind	Value	Auto update
Task	1	_MainTask	Status(WaitFactor)	DORMANT	Break
Task	2	_task7	Status(WaitFactor)	RUNNING	Break
Task	3	__kernel_vcttbl	Status(WaitFactor)	NON-EXISTENT	Break

Object1 Object2 Object3 Object4 Object5 Object6 Object7 Object8 Object9 Object10

Figure 3.5 Referring to the Task State

3.5 Displaying the Task Execution History

The Task Trace window displays the task execution history using the trace data of the simulator/debugger or emulator debugger. To implement the trace settings, open the Task Trace window once by selecting [RTOS]->[Task Trace] from the [View] menu. After that, execute the program again. Thus, the task execution history will be displayed in the Task Trace window. In this example, it is obvious that task ID 1 and task ID 2 are operating.



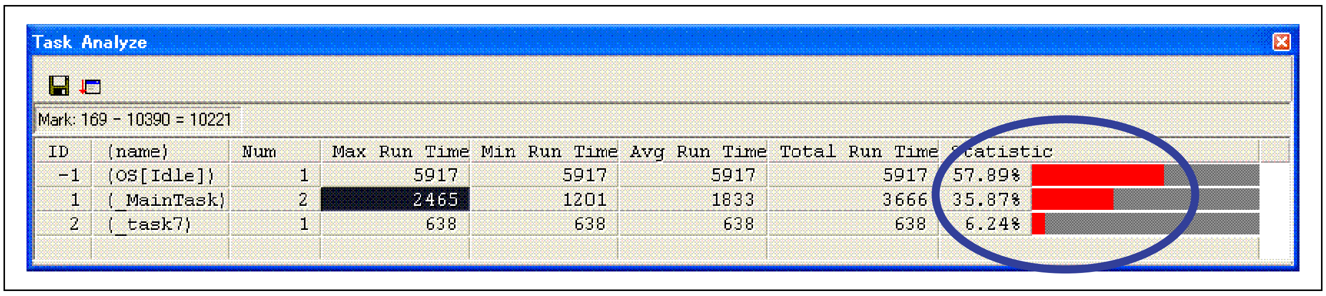
Mark: 169 - 10385 = 10216 | Indicator: 6081 | Scale: * 1.000000 | Grid: 2043 | Area: Full

ID	(name)
-1	(OS[Idle])
1	(_MainTask)
2	(_task7)

Figure 3.6 Displaying the Task Execution History

3.6 Displaying the Task Execution Time

The Task Analyze window opened by selecting [RTOS]->[Task Analyze] from the [View] menu displays the task execution time and CPU occupancy taken up by it using the trace data of the simulator/debugger or emulator debugger.



Mark: 169 - 10390 = 10221

ID	(name)	Num	Max Run Time	Min Run Time	Avg Run Time	Total Run Time	Statistic
-1	(OS[Idle])	1	5917	5917	5917	5917	57.89%
1	(_MainTask)	2	2465	1201	1833	3666	35.87%
2	(_task7)	1	638	638	638	638	6.24%

Figure 3.7 Displaying the Task Execution Time

4. Functions

This section describes each function in detail.

4.1 List of Functions

Table 4.1 lists the functions that can be used by each real-time OS. Table 4.2 lists the functions supported in each debugger.

Table 4.1 Function Relationship (Real-time OS)

Function	HI7200/MP, HI7750/4, HI7700/4, HI7000/4	RI600/4, M3T-MR30/4	HI1000/4	Smalight OS	ThreadX, embOS, TOPPERS/JSP	NORTi, µC/OS-II, µT-Kernel	osCAN, TOPPERS/ASP
OS object	✓	✓	✓	✓	✓	✓	✓
Task trace	✓	✓	✓	✓	✓	✓	✓
Task analyze	✓	✓	✓	✓	✓	✓	✓
Task step	—	✓	✓	✓	—	—	—
Currently executed task display	—	✓	✓	✓	—	✓	✓
Execution history text display	—	✓	✓	✓	—	✓	✓
Real-time profile	—	✓	✓	✓	—	✓	✓
Task specification for each condition	—	✓	✓	✓	—	✓	✓
Stack coverage	—	✓	—	✓	—	—	✓
Memory protection	—	✓	—	✓	—	—	—
Dispatch detection	—	✓	✓	✓	—	—	—

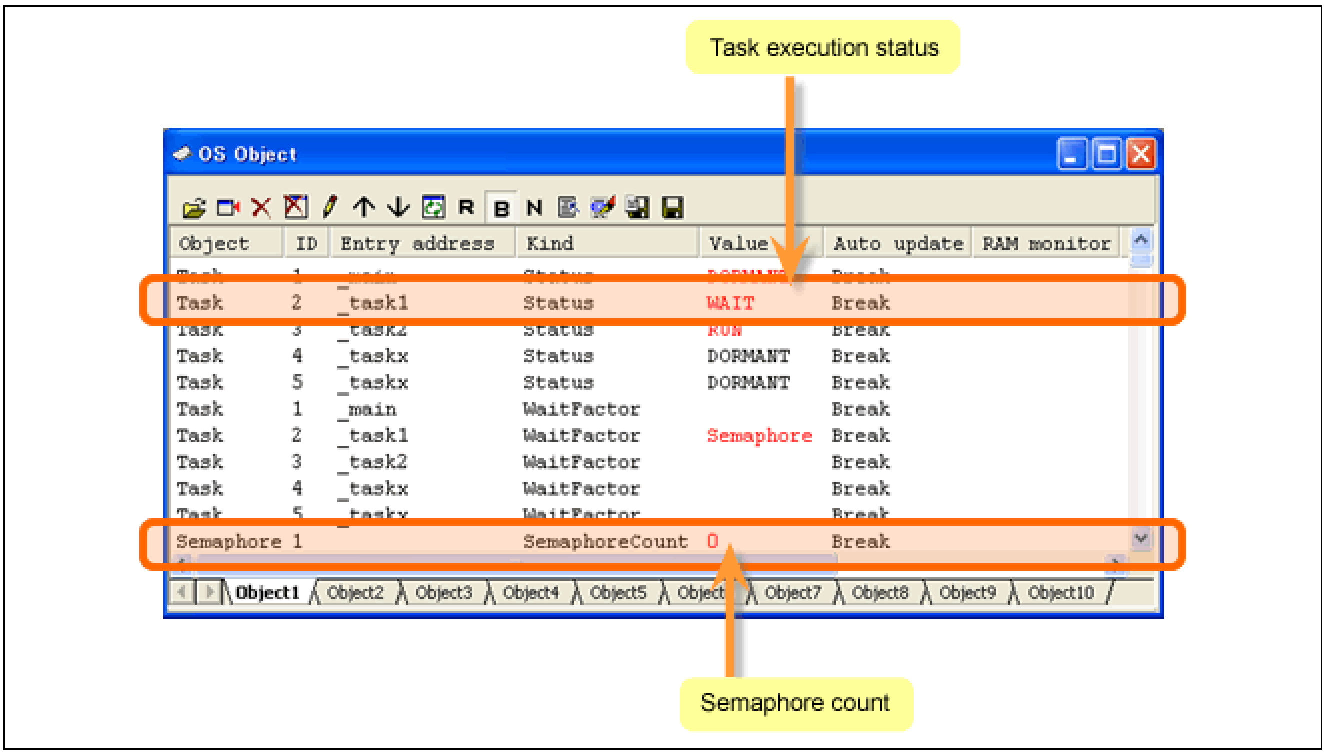
Table 4.2 Function Relationship (Debugger)

Function	Simulator	E1	E20	E10A-USB	E8a	E100	E200F
OS object	✓	✓	✓	✓	✓	✓ （RAM monitoring supported）	✓
Task trace	✓	✓	✓ (Large capacity)	✓ *	—	✓ (Large capacity)	✓
Task analyze	✓	✓	✓ (Large capacity)	✓ *	—	✓ (Large capacity)	✓
Task step	—	✓	✓	—	—	✓	—
Currently executed task display	—	✓	✓	—	—	✓	—
Execution history text display	—	—	—	—	—	✓	—
Real-time profile	—	—	—	—	—	✓	—
Task specification for each condition	—	—	—	—	—	✓	—
Stack coverage	—	—	—	—	—	✓	—
Memory protection	—	—	—	—	—	✓	—
Dispatch detection	—	—	—	—	—	✓	—

Note: * Use a 36-pin or 38-pin cable (14-pin cable cannot be used).
The time is not displayed (except for SH2A-DUAL).

4.2 Functional Descriptions

4.2.1 OS Object Window (Reference of Object State)



Object	ID	Entry address	Kind	Value	Auto update	RAM monitor
Task	1	_main	Status	DORMANT	Break	
Task	2	_task1	Status	WAIT	Break	
Task	3	_task2	Status	RUN	Break	
Task	4	_taskx	Status	DORMANT	Break	
Task	5	_taskx	Status	DORMANT	Break	
Task	1	_main	WaitFactor		Break	
Task	2	_task1	WaitFactor	Semaphore	Break	
Task	3	_task2	WaitFactor		Break	
Task	4	_taskx	WaitFactor		Break	
Task	5	_taskx	WaitFactor		Break	
Semaphore	1		SemaphoreCount	0	Break	

Figure 4.1 OS Object Window

How to Open

- Select [View->RTOS->OS Object].
- Click the [OS Object] toolbar button.

Description

The OS Object window displays the state of OS objects, such as tasks and semaphores.

When this window is opened for the first time, the states of all tasks are displayed automatically. Add display items using [Add objects] in the popup menu as required. The number of display items and the displayed contents differ in each OS.

When the object has an entry address (task, handler, etc.), the [Editor] window can be opened from the popup menu. If the displayed contents are updated, the updated values are displayed in red.

If RAM monitoring is specified for the area of the object values, [In] is displayed in the [RAM monitor] column.

Notes

- Real-time performance is not achieved while this window is opened because memory accesses will occur.
- The window may not be displayed correctly when execution is halted in the midst of program execution or OS kernel execution.

Popup Menu

The usable options are shown below.

Add Object (A)...		Displays the [Select Object Data] dialog box. The added OS object is inserted before the selected line.
Delete Object (D)		Deletes the selected OS object.
Delete All Objects (L)		Deletes all OS objects in the currently displayed sheet.
Edit Object (E)...		Displays the [Select Object Data] dialog box. Another OS object can be selected here.
Up (U)		Moves the selected OS object one line up.
Down (D)		Moves the selected OS object one line down.
Refresh (R)		Forces a manual update of the displayed contents.
Auto Update (U)	Update During Execution (A)	Updates the displayed contents while the user program is being executed and when it has stopped.
	Update Only At Stop (B)	Updates the displayed contents when the user program has stopped.
	No Update (L)	Does not update the displayed contents.
Display Source (W)		Opens the [Editor] window corresponding to the entry address.
Select All (A)		Selects all lines.
Save (S)	Save Displayed Contents (D)...	Saves the displayed contents of all sheets in a file.
	Save Display Items (I)	Saves the display items of all sheets in a file.
Read (L)...		Reads the settings of all sheets from a file.
Display Toolbar		Switches between display and nondisplay of the toolbar.
Customize Toolbar...		Customizes the toolbar buttons.

4.2.2 Task Trace Window (Graphical Display of Task Execution History)

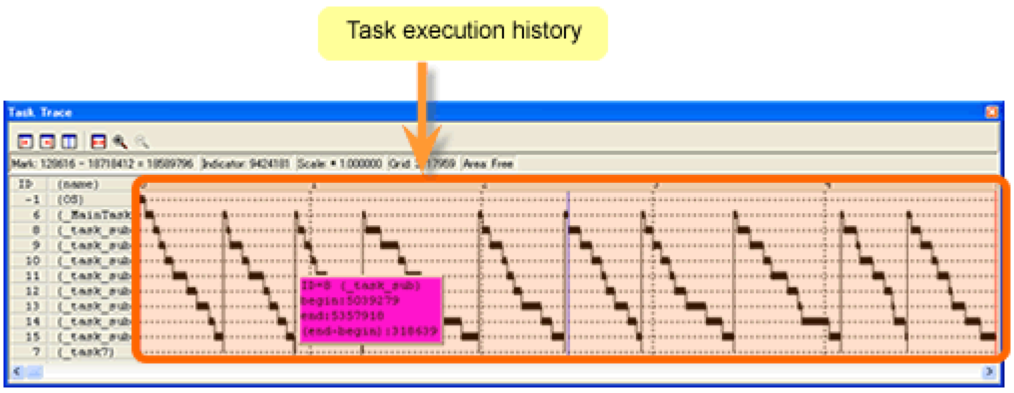


Figure 4.2 Task Trace Window

How to Open

- Select [View->RTOS->Task Trace].
- Click the [Task Trace] toolbar button.

Description

The Task Trace window graphically displays the result of measuring the task execution history of a program using the real-time OS. The items to be displayed are as follows:

Item	Description
ID	Displays the task ID.
(name)	Displays the task entry label name.

Moving the mouse to each information shown in the window opens a popup menu for displaying detailed information.

The status bar displays the following information.

- Time value of starting point marker's location
- Time value of ending point marker's location
- Duration between starting point and ending point markers
- Time value of current position marker's location
- Display magnification
- Duration of grid line interval
- Measurement (trace) range

Grid lines are displayed with the starting point marker as the base point.

Scales are displayed with the time where the starting point marker is located as 0, the left side (forward in terms of time) as negative, and the right side (backward in terms of time) as positive.

The generation cycle and processing time of interrupts can be roughly grasped by grid lines.

The duration of the displayed grid line interval is shown in the "Grid" area of the status bar.

The time values in the Task Trace window all mean the execution elapsed time with the program execution start point assumed to be 0.

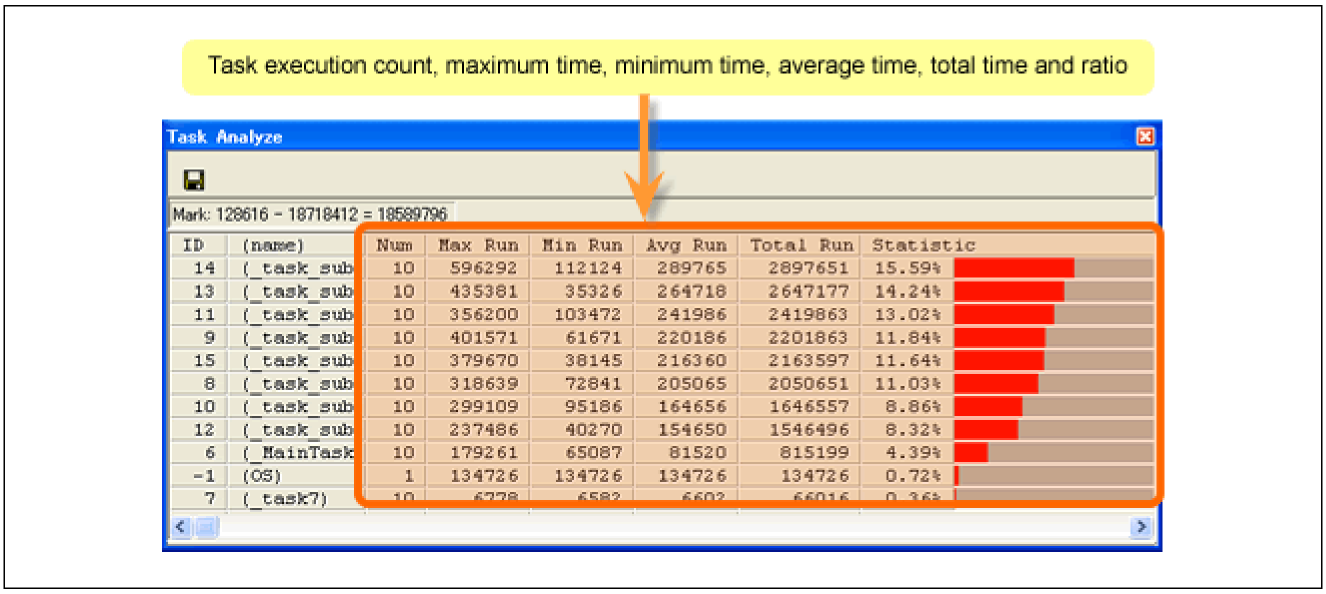
In contrast, the numbers at the upper side of the grid lines (scales) in the Task Trace window are relative values (grid line interval is specified in the Value dialog box) with the starting point marker assumed to be 0, and they have nothing to do with time values (they are to facilitate visualization of the window).

Popup Menu

The usable options are shown below.

Starting Point Marker		Moves the starting point marker to the display area.
Ending Point Marker		Moves the ending point marker to the display area.
Current Position Marker		Moves the current position marker to the display area.
Adjust Display Magnification		Displays the range between the starting point and ending point markers using the entire horizontal width of the window.
Enlarge Display Magnification		Enlarges the display magnification.
Reduce Display Magnification		Reduces the display magnification.
Halt Measure		Halts trace measurement and displays the result.
Re-measure		Re-measures the trace data.
Measurement Range Condition	Full	Stops trace acquisition when the trace memory has become full.
	Free	Continues trace acquisition until program execution is stopped.
Set		Sets the grid line interval and display magnification.
Display Toolbar		Switches between display and nondisplay of the toolbar.
Customize Toolbar...		Customizes the toolbar buttons.

4.2.3 Task Analyze Window (Graphical Display of Task Execution Time)


Task execution count, maximum time, minimum time, average time, total time and ratio
Task Analyze

Mark: 128616 - 18718412 = 18589796

ID	(name)	Num	Max Run	Min Run	Avg Run	Total Run	Statistic
14	(_task_sub	10	596292	112124	289765	2897651	15.59%
13	(_task_sub	10	435381	35326	264718	2647177	14.24%
11	(_task_sub	10	356200	103472	241986	2419863	13.02%
9	(_task_sub	10	401571	61671	220186	2201863	11.84%
15	(_task_sub	10	379670	38145	216360	2163597	11.64%
8	(_task_sub	10	318639	72841	205065	2050651	11.03%
10	(_task_sub	10	299109	95186	164656	1646557	8.86%
12	(_task_sub	10	237486	40270	154650	1546496	8.32%
6	(_MainTask	10	179261	65087	81520	815199	4.39%
-1	(OS)	1	134726	134726	134726	134726	0.72%
7	(_task7)	10	6778	6582	6602	66016	0.36%

Figure 4.3 Task Analyze Window

How to Open

- Select [View->RTOS->Task Analyze].
- Click the [Task Analyze] toolbar button.

Description

The Task Analyze window displays the status of CPU occupation.

The Task Analyze window functions in collaboration with the Task Trace window.

The Task Analyze window displays the result of statistically processing the data measured for the range specified by the starting point and ending point markers of the Task Trace window.

Clicking the maximum execution time and minimum execution time display areas in each line enables searching for the processing history of the maximum execution time and minimum execution time of the task corresponding to the clicked line.

The search result is indicated by the current position marker of the Task Trace window moving to the target location.

Popup Menu

The usable options are shown below.

Save In File (S)...	Saves the displayed contents in a file in the text format or CSV format.
Initialize Display Order (D)	Returns the display order of the tasks to the way they were displayed first. The display order in the beginning is the order in which the tasks were executed.
Display Toolbar	Switches between display and nondisplay of the toolbar.
Customize Toolbar...	Customizes the toolbar buttons.

4.2.4 Task Step (Step Execution Focused on a Specific Task)

- [Debug]->[RTOS Debug]->[Go To Cursor]

  Executes the current task until it reaches the cursor's position.
  The user program is executed from the current PC value and execution is continued until the PC value reaches the address indicated by the current text cursor's position (not the mouse cursor).
  If the task at execution stop differs from the task at execution start, execution will continue until the tasks become the same.

- [Debug]->[RTOS Debug]->[Step In]

  Steps into the current task.
  A single block of the user program is executed and then execution stops. The size of this block is normally a single instruction but the user can specify multiple instructions or a C/C++ source line (Step...).
  When a subroutine is called, execution is halted after entering the subroutine, and the subroutine code is displayed.
  If the task at execution stop differs from the task at execution start, execution will continue until the tasks become the same.
  When step-in execution is performed to a system call, processing may not end correctly. In such a case, perform step-over execution.

- [Debug]->[RTOS Debug]->[Step Over]

  Steps over the current task.
  A single block of the user program is executed and then execution stops. The size of this block is normally a single instruction but the user can specify multiple instructions or a C/C++ source line (Step...).
  When a subroutine is called, the subroutine is not entered, and the user program continues to be executed until the current PC location moves to the next line of the current display.
  If the task at execution stop differs from the task at execution start, execution will continue until the tasks become the same.

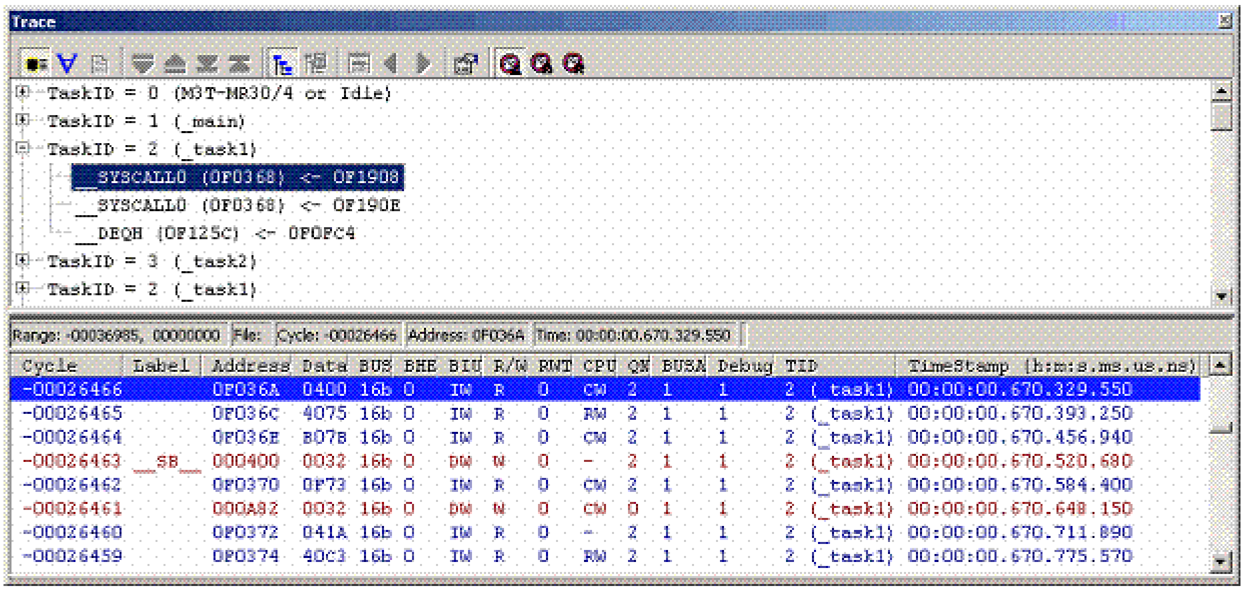
- [Debug]->[RTOS Debug]->[Step Out]

  Steps out of the current task.
  The user program is executed until the end of the current function is reached, and execution is stopped with the PC set to the next line of the calling function.
  If the task at execution stop differs from the task at execution start, execution will continue until the tasks become the same.

4.2.5 Currently Executed Task Display in Status Bar

The task ID of the currently executed task is automatically displayed in the status bar.

4.2.6 Text Display of Task Execution History



```
Trace
TaskID = 0 (M3T-MR30/4 or Idle)
TaskID = 1 (_main)
TaskID = 2 (_task1)
    __SYSCALL0 (0F0368) <- 0F1908
    __SYSCALL0 (0F0368) <- 0F190E
    __DEQH (0F125C) <- 0F0FC4
TaskID = 3 (_task2)
TaskID = 2 (_task1)

Range: -00036985, 00000000 | File: | Cycle: -00026466 | Address: 0F036A | Time: 00:00:00.670.329.550

Cycle      Label   Address Data BUS BHE BIU R/W RWT CPU QN BUSA Debug TID          TimeStamp (h:m:s.ms.us.ns)
-00026466          0F036A  0400 16b 0   IW  R   0   CW  2  1    1     2 (_task1)   00:00:00.670.329.550
-00026465          0F036C  4075 16b 0   IW  R   0   RW  2  1    1     2 (_task1)   00:00:00.670.393.250
-00026464          0F036E  B07B 16b 0   IW  R   0   CW  2  1    1     2 (_task1)   00:00:00.670.456.940
-00026463  __SB__  000400  0032 16b 0   DW  W   0   -   2  1    1     2 (_task1)   00:00:00.670.520.680
-00026462          0F0370  0F73 16b 0   IW  R   0   CW  2  1    1     2 (_task1)   00:00:00.670.584.400
-00026461          000A82  0032 16b 0   DW  W   0   CW  0  1    1     2 (_task1)   00:00:00.670.648.150
-00026460          0F0372  041A 16b 0   IW  R   0   -   2  1    1     2 (_task1)   00:00:00.670.711.890
-00026459          0F0374  40C3 16b 0   IW  R   0   RW  2  1    1     2 (_task1)   00:00:00.670.775.570
```

Figure 4.4 Trace Window

How to Open

- Select [Display Execution History->Display Function Execution History] from the popup menu of the [Trace] window.
  (The upper-stage window is displayed.)

- Select [Analyze Execution History] from the popup menu of the [Trace] window.
  (The analysis result is displayed in the upper-stage window.)

Description

The task execution history can be displayed only when debugging real-time OS programs.

To display the task execution history, the task ID needs to be selected in the [Options] page of the [Set Trace Conditions] dialog box which can be displayed from the [Set Trace...] menu.

The task execution history of the trace result is shown in the upper-stage window. The default is nondisplay.

When [Analyze Execution History] is selected from the popup menu of the upper-stage window, analysis of function execution history is started and the result is displayed.

If the branch destination address is not a function or if there is no corresponding debugging information for that address, an address is displayed (label and address are displayed when a label is set) instead of a function name.

The lower-stage window displays the trace result from the cycle in which the task selected in the upper-stage window was called.

<Display format of task execution history>

(1) The following is displayed when the task ID changes.

TaskID = Task ID number (entry name corresponding to task ID)

Example: TaskID = 1 (_main)

(2) A function called within a task is displayed in the following format.

Function name (function start address) <- Address of function caller

Note

The task execution history cannot be displayed when extracting or deleting conditions for trace are specified.

Upper-stage window:

Displays the execution history for individual tasks.

For the task execution history, the functions called within a task are not displayed in the tree structure. These functions only appear in the display of the execution order of the functions.

Expanded display or collapsed display can be selected for each task. The default is expanded display.

When analysis takes time, it can be canceled from the progress dialog box. When canceled, the task execution history is displayed up to the point where analysis was completed.

When the task entry label is double-clicked in the upper-stage window, the bus information on the point where that task was called is displayed in the lower-stage window.

Lower-stage window:

Displays the trace result from the cycle of the task selected in the upper-stage window.

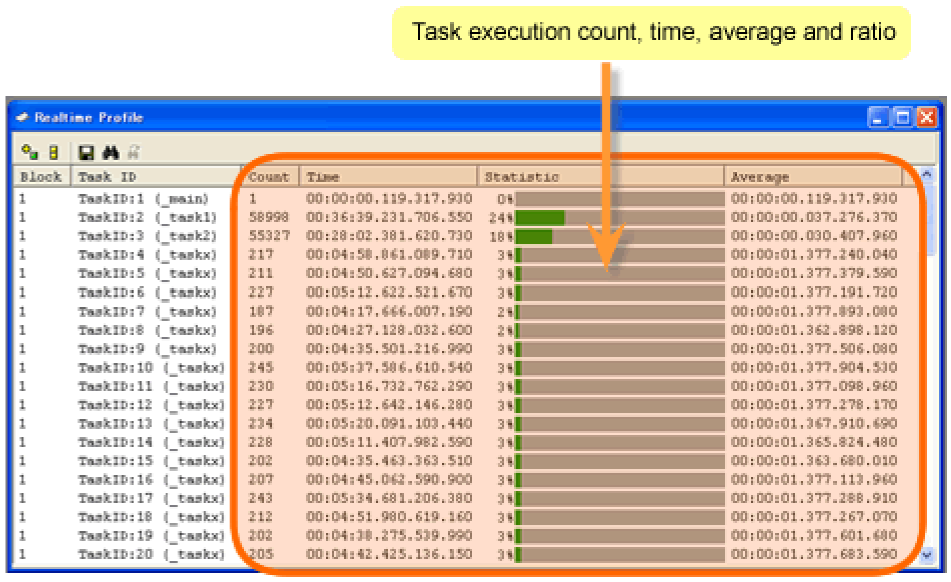
Disassembly display, source display, and mixed display are also possible.

Popup Menu

The options usable in the upper-stage window are shown below.

Analyze Execution History	Analyzes the execution history of functions and displays the analysis result in the upper-stage window. Functions called within a task are not displayed in the tree structure. They only appear in the displayed function execution order.
Execution History Nondisplay	Closes the upper-stage window. The analysis result is cleared.
Function Search...	Displays the [Function Search] dialog box. The function name specified in this dialog box will be searched for.
Function Forward Search	Searches forward from the line in which the function name specified in the [Function Search...] menu was selected.
Function Backward Search	Searches backward from the line in which the function name specified in the [Function Search...] menu was selected.
Jump To List	Displays the trace result from the cycle of the selected function.

4.2.7 Real-time Profile Window (Display of CPU Occupancy for a Non-Stop Long Period)



Block	Task ID	Count	Time	Statistic	Average
1	TaskID:1 (_main)	1	00:00:00.119.317.930	0%	00:00:00.119.317.930
1	TaskID:2 (_task1)	58998	00:36:39.231.706.550	24%	00:00:00.037.276.370
1	TaskID:3 (_task2)	55327	00:28:02.381.620.730	18%	00:00:00.030.407.960
1	TaskID:4 (_taskx)	217	00:04:58.861.089.710	3%	00:00:01.377.240.040
1	TaskID:5 (_taskx)	211	00:04:50.627.094.680	3%	00:00:01.377.379.590
1	TaskID:6 (_taskx)	227	00:05:12.622.521.670	3%	00:00:01.377.191.720
1	TaskID:7 (_taskx)	187	00:04:17.666.007.190	2%	00:00:01.377.893.080
1	TaskID:8 (_taskx)	196	00:04:27.128.032.600	2%	00:00:01.362.898.120
1	TaskID:9 (_taskx)	200	00:04:35.501.216.990	3%	00:00:01.377.506.080
1	TaskID:10 (_taskx)	245	00:05:37.586.610.540	3%	00:00:01.377.904.530
1	TaskID:11 (_taskx)	230	00:05:16.732.762.290	3%	00:00:01.377.098.960
1	TaskID:12 (_taskx)	227	00:05:12.642.146.280	3%	00:00:01.377.278.170
1	TaskID:13 (_taskx)	234	00:05:20.091.103.440	3%	00:00:01.367.910.690
1	TaskID:14 (_taskx)	228	00:05:11.407.982.590	3%	00:00:01.365.824.480
1	TaskID:15 (_taskx)	202	00:04:35.463.363.510	3%	00:00:01.363.680.010
1	TaskID:16 (_taskx)	207	00:04:45.062.590.900	3%	00:00:01.377.113.960
1	TaskID:17 (_taskx)	243	00:05:34.681.206.380	3%	00:00:01.377.288.910
1	TaskID:18 (_taskx)	212	00:04:51.980.619.160	3%	00:00:01.377.267.070
1	TaskID:19 (_taskx)	202	00:04:38.275.539.990	3%	00:00:01.377.601.680
1	TaskID:20 (_taskx)	205	00:04:42.425.136.150	3%	00:00:01.377.683.590

Figure 4.5 Realtime Profile Window

How to Open

- Select [View->Performance->Realtime Profile].
- Click the [Realtime Profile] toolbar button.

Description

The two modes of function mode and task mode are available.

This function can acquire profile information for all functions (or tasks) in a maximum of eight block areas, with 128 Kbytes as one block.

Adjacent address areas can be set as blocks.

Task mode:

E100 emulator automatically collects task information when a load module including an OS has been downloaded.

128-Kbyte blocks are automatically assigned to tasks, starting from the lowest task ID.

This function can usually be used without changing any settings.

If there is a task ID that exceeds the block, the block can be put off by clearing the selected state of the task ID with a low number.

Column contents:

Block	Block number
Task ID	Task ID and task entry label name
Count	Number of times the task was called
Time	Accumulated task execution time The display format of the time stamp is as follows: hour:minute:second.millisecond.microsecond.nanosecond
Statistic	Ratio of the Go-Break execution time to the task's Time
Average	Average execution time for one time of execution

When measurement of Count or Time overflows, "overflow" is displayed. In this case, Average is not shown.

Clicking a column header sorts the displayed contents in the ascending order or descending order of that column.

Note

Up to 8K - 1(= 8191) functions (tasks) can be measured. When the number of functions (tasks) to be measured exceeds 8K - 1(= 8191), that function (task) will not be measured. In this case, that function (task) is grayed.

Popup Menu

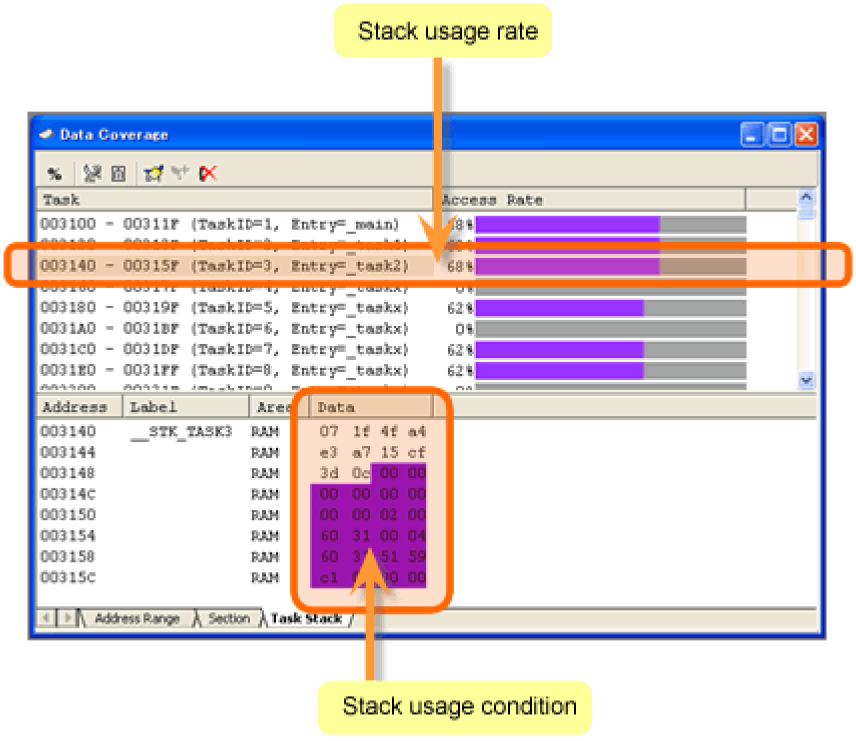
The usable options are shown below.

Set Range (S)...	Displays the [Set Realtime Profile] dialog box. Either function profile or task profile should be set here (they cannot be measured simultaneously).
Clear All Data (L)	Clears all measured results.
Property (P)...	Displays the [Property] dialog box. 10 ns, 20 ns, 40 ns, 80 ns, 160 ns, or 1.6 us can be selected here. This setting will take effect the next time the execution time is measured.
Search (F)...	Searches for the specified function. The line where the function was found is highlighted. In task display, the task ID or task entry label is searched for.
Search Next (N)	Searches for the next function/task.
Save In File (V)...	Saves all measured results in a file in the text format.

4.2.8 Task Specification for Each Condition

The task ID can be specified in the hardware break, trace, and performance analysis functions.
Debugging can be performed while focusing on a specific task by using this specification.
For details, refer to the help file of the E100 emulator.

4.2.9 Stack Coverage (Graphical Display of Task Stack Used Size)



```
Task                                              Access Rate
003100 - 00311F (TaskID=1, Entry=_main)          [illegible]8%
[illegible]
003140 - 00315F (TaskID=3, Entry=_task2)          68%
[illegible]
003180 - 00319F (TaskID=5, Entry=_taskx)          62%
0031A0 - 0031BF (TaskID=6, Entry=_taskx)           0%
0031C0 - 0031DF (TaskID=7, Entry=_taskx)          62%
0031E0 - 0031FF (TaskID=8, Entry=_taskx)          62%
[illegible]

Address  Label        Area  Data
003140   __STK_TASK3  RAM   07 1f 4f a4
003144                RAM   e3 a7 15 cf
003148                RAM   3d 0c 00 00
00314c                RAM   00 00 00 00
003150                RAM   00 00 02 00
003154                RAM   60 31 00 04
003158                RAM   60 3[illegible] 51 59
00315c                RAM   c1 [illegible] [illegible]0 00
```

Figure 4.6 Data Coverage Window

How to Open

- Select [View->Code->Data Coverage].
- Click the [Data Coverage] toolbar button.

Description

This window consists of three sheets.

<Address Range> sheet

Data coverage information is collected for the address range specified by the user, and the result is displayed.

<Section> sheet

Data coverage information is collected for the section specified by the user, and the result is displayed.

<Task Stack> sheet

Data coverage information is collected for the task stack specified by the user, and the result is displayed.

For this sheet, the window is divided into upper and lower windows by the splitter.

Upper-side Window	
[Task]	Task stack for which coverage information is acquired
[Access Rate]	Displays the percentage and graph of the access rate.
Lower-side Window	
[Address]	Address value
[Label]	Label corresponding to the address (displayed only when a label is set)
[Area]	Memory area This column is blank for areas for which coverage information cannot be acquired.
[Data]	Data value Displayed in bytes. Example: 00 00 FF FF Read/write access is indicated by the background color of the data value. Has been accessed: Purple

Popup Menu

The usable options are shown below.

Upper-side window:

Percentage (P)	Calculates and displays the percentage.
Add Range (A)...	Adds a new measurement range.
Edit Range (E)...	Changes the selected measurement range.
Delete Range (D)	Deletes the selected measurement range.

Lower-side window:

Refresh (R)	Forces a manual update of the displayed contents. Update is performed even when auto update is disabled.
Disable Auto Update (K)	When this is selected, the window is not automatically updated at execution stop. Click this when the window does not have to be updated at each step, such as in the case of continuous step execution.
Display Address (A)...	Displays the [Display Address] dialog box. A new start address can be input here.
Byte Count Per Line (D)...	The number of bytes to be displayed in one line can be specified.
Save (V)...	Displays the [Save Coverage Information] dialog box. The location and name of the coverage information file to be saved can be specified here.
Load (L)...	Displays the [Load Coverage Information] dialog box. The location and name of the coverage information file to be loaded can be specified here. The only loadable file extension is ".cdv".
Set Hardware (H)...	Displays the [Allocate Data Coverage Memory] dialog box. The area in which coverage information is collected should be set here.
Clear Coverage Range (C)...	Displays the [Clear Coverage Range] dialog box. The range to be cleared can be specified here.
Clear All Coverage (E)	Clears all data coverage information.

4.2.10 Memory Protection

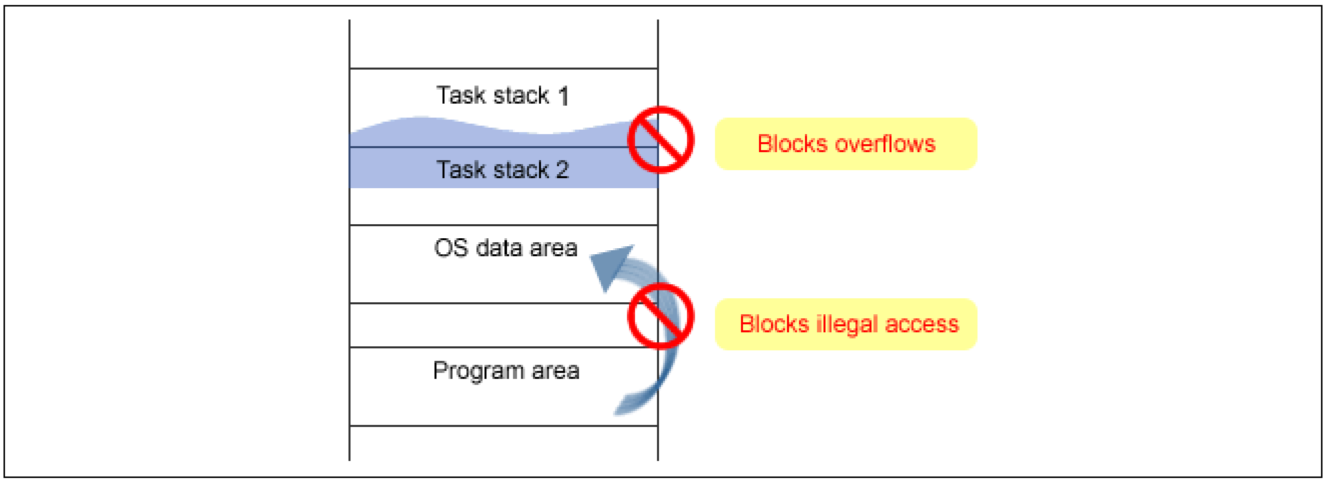


Figure 4.7 Memory Protection

This function detects stack access violation by another task and access violation to the data area of the real-time OS.

Displaying a warning with the balloon of the status bar or setting hardware breakpoints can be done.

For details, refer to the help file of the E100 emulator.

4.2.11 Dispatch Detection

This function detects dispatching of a task.

Displaying a warning with the balloon of the status bar or setting hardware breakpoints can be done.

For details, refer to the help file of the E100 emulator.

5. Notes

This section describes the notes on the real-time OS aware debugging.

Common Notes for Real-time OS

General	1. When a task is created or deleted during program execution, the task ID may not be displayed correctly.
Select OS Definition File dialog box	1. When an OS different from the OS in use is selected from the list in the dialog box, operation cannot be performed correctly. In some cases, the debugger does not respond.
OS Object window	1. Real-time performance is not achieved while this window is opened because memory accesses will occur (when RAM monitoring is not used). 2. This window may not be displayed correctly in cases, such as, before completion of OS kernel initialization, during program execution, and execution stop in the OS kernel. 3. The maximum number of lines that can be displayed in one sheet is 1023. 4. When using RAM monitoring, this window should be updated to the latest information after the settings have been changed. 5. If there is no valid information, this window may not be displayed correctly (dynamically created task, nonexistent task, nonexistent message, etc.).
Task Trace window and Task Analyze window	1. These windows are displayed based on the information of the task that is managed within the OS kernel and currently being executed. Therefore, the timing for task switching or task execution time differs depending on the OS. 2. The time and cycle count are not displayed for some debuggers. 3. When these windows are used, other trace windows or event functions cannot be used in some cases. In turn, when other trace windows or event functions are used, these windows cannot be used in some cases. 4. When these windows are used, real-time performance of the program may be deteriorated. In this case, change the settings in the trace window of each debugger to perform real-time trace or give priority to the CPU, in order to achieve real-time performance. However, note that when real-time performance is achieved, part of the trace information may be lost.
Task execution history, task execution time, and real-time profile	1. These items are displayed based on the information of the task that is managed within the OS kernel and currently being executed. Therefore, the timing for task switching or task execution time differs depending on the OS.
Memory protection	1. The OS areas in which illegal access is to be detected differ depending on the OS.
Toolchain	1. When the build options of the toolchain (compiler, assembler, etc.) have been changed from those at real-time OS delivery, the real-time OS aware debugging may not operate correctly.
Emulator debugger	1. When memory access during program execution is inhibited by the emulator debugger setting, the task ID of windows are not displayed correctly in some cases.
E100 emulator debugger	1. In step execution that was started by selecting [Debug]->[RTOS Debug], step execution may not halt depending on the user program.
SH-2A microcomputer	1. After a CPU reset, if execution stops before OS kernel initialization was completed, the debugger may not respond.

Notes for Individual Real-time OS

HI7200/MP	1. The profile ID displayed in Profile in the OS Object window stands for the ID of each task. However note that profile ID 0 indicates the kernel idling state while the maximum profile ID indicates the entire system.
Smalight OS	1. Among the task IDs displayed in each window, 255 means that the OS kernel is idling.
ThreadX	1. The task ID displayed in the debugger is the debugger internal ID and is not the same as the thread ID of the ThreadX. 2. The maximum number of threads that can be displayed in the OS Object window is 255.
NORTi	1. Among the IDs displayed in the Task Trace window and Task Analyze window, adding 1 to the maximum task ID means that the OS kernel is idling.
osCAN	1. The task ID displayed in the debugger is the debugger internal ID and is not the same as the task ID of the osCAN. 2. When using task stack coverage, click the WithStackCheck check box of the OIL-configurator.
μC/OS-II	1. The task ID displayed in the debugger is the debugger internal ID and is not the same as the task ID of the μC/OS-II. In addition, the task entry address displayed in the debugger represents the task name. 2. When the contents of the configuration file (os_cfg.h) are changed from the initial values, the real-time OS aware debugging may not operate correctly. 3. The maximum number of tasks that can be displayed in the OS Object window is 255. 4. "Ready" shown in Task State in the OS Object window stands for "task is executable", "task is not created", or "task is delayed".
embOS	1. The task ID displayed in the debugger is the debugger internal ID and is not the same as the task ID of the embOS. In addition, the task entry address displayed in the debugger represents the task name. 2. The maximum number of tasks that can be displayed in the OS Object window is 255. 3. The task name is not displayed when the XR libraries are used.
μT-Kernel	1. Do not change the contents of the system configuration definition file (config¥config.h) from the initial values. 2. The maximum task ID that can be displayed in the OS Object window is 32.

6. FAQ

No.	Question	Answer
1	The time is not displayed in the Task Trace window and Task Analyze window.	When using the E10A-USB emulator, the time cannot be displayed (except for SH2A-DUAL).
2	The Task Trace window and Task Analyze window are not displayed.	The Task Trace window and Task Analyze window cannot be displayed when using E8a emulator or when using a 14-pin cable with the E10A-USB emulator. When using the E10A-USB emulator, a 36-pin or 38-pin cable should be used.

7. Website and Support

Renesas Technology Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb.16.10	—	First edition issued