RENESAS

# Debugging Extension

User's Manual

HS6400IWIN1SE

Renesas Microcomputer Development Environment System

User's Manual

Rev.1.00
2003.8.5

Renesas Technology
www.renesas.com

## Cautions

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

The product described in this publication is based on the ITRON specifications and was developed under the guidance of Dr. Ken Sakamura of The University of Tokyo.

1. µITRON is an acronym of the “Micro Industrial TRON” and TRON is an acronym of “The Realtime Operating system Nucleus”.
2. Microsoft® Windows® 98 operating system, Microsoft® Windows® Millennium Edition (Windows® ME) operating system, Microsoft® Windows NT® operating system, Microsoft® Windows® 2000 operating system, and Microsoft® Windows® XP operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.
3. SuperH™ is a trademark of Renesas Technology Corp.
4. All other product names are trademarks or registered trademarks of the respective holders.

# Preface

The Debugging Extension (hereafter called the DX) is software for adding the multitasking debugging functions of the Realtime Operating System (hereafter called the RTOS) to the High-performance Embedded Workshop (hereafter called the HEW).

This manual describes how the DX is used.

Please read this manual and the related manuals listed below before use so that you fully understand the DX before using it.

This user’s manual contains the following four sections and appendixes:

| | |
|---|---|
| Section 1 | Introduction to the DX |
| Section 2 | How to operate the DX (based on an example) |
| Section 3 | The functions of the DX |
| Section 4 | Precautions and restrictions |
| Appendix | Troubleshooting |

Be sure to read section 4, Precautions and Restrictions, before using the DX.

Please refer to the online help file for details of the DX functions.

The following is a list of the related manuals:

- High-performance Embedded Workshop (HEW) User's Manual
- The user’s manual for the RTOS you are using
- The user’s manual for the compiler you are using
- The hardware manual and programming manual for the microcomputer you are using

Symbols used in this manual have the following meanings:

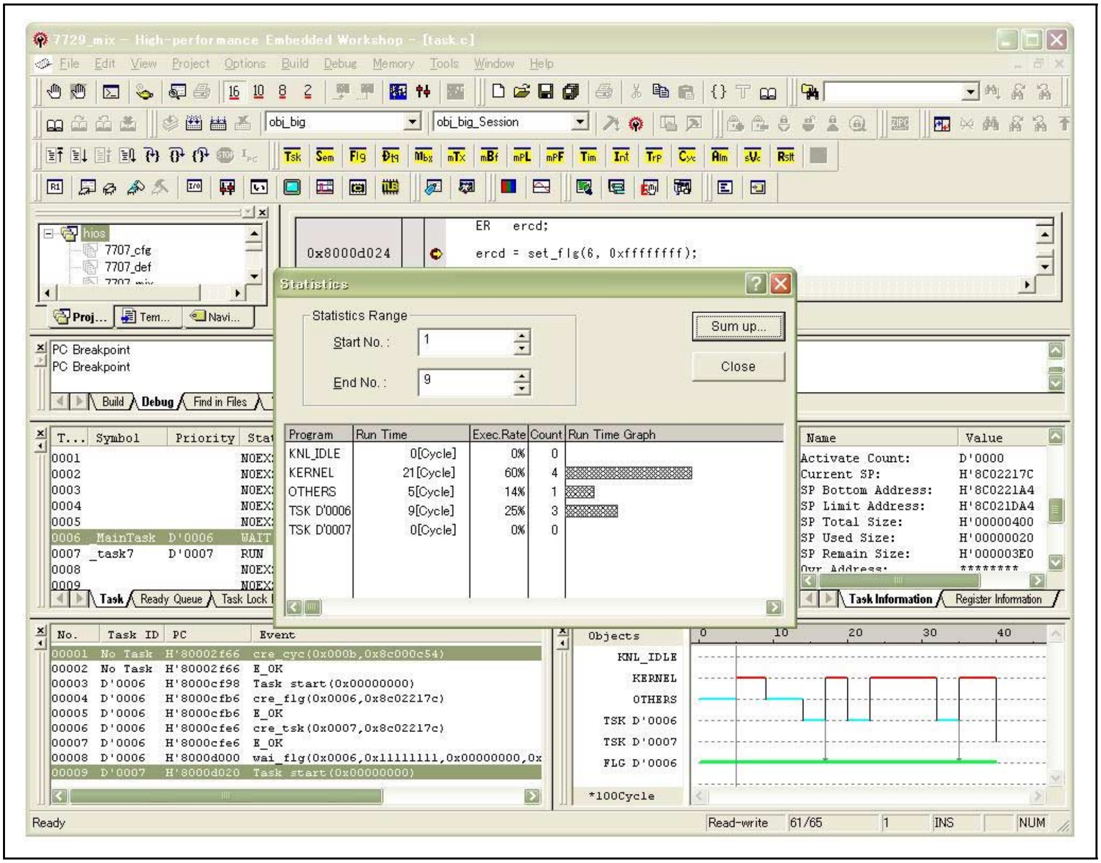
| | |
|---|---|
| [Menu -> Menu Option]: | The menu name is on the left of the '->', and the item to be selected from the menu is on the right of the '->'. (Example: [File -> New...]). |
| H' and D': | The prefix H' is attached for hexadecimal integers. The prefix D' is attached for decimal integers. If no prefix is attached, a decimal integer is assumed. |

# Contents

# Section 1 Overview

## 1.1 Outline

The DX is software for debugging applications which are created for the RTOS. It is installed in the HEW and RTOS systems.



```
            ER    ercd;
0x8000d024        ercd = set_flg(6, 0xffffffff);
```

```
PC Breakpoint
PC Breakpoint
```

Start No. : 1
End No. : 9

| Program | Run Time | Exec.Rate | Count | Run Time Graph |
|---|---|---|---|---|
| KNL_IDLE | 0[Cycle] | 0% | 0 | |
| KERNEL | 21[Cycle] | 60% | 4 | ▨ |
| OTHERS | 5[Cycle] | 14% | 1 | ▨ |
| TSK D'0006 | 9[Cycle] | 25% | 3 | ▨ |
| TSK D'0007 | 0[Cycle] | 0% | 0 | |

```
T...  Symbol      Priority  Stat
0001                        NOEX:
0002                        NOEX:
0003                        NOEX:
0004                        NOEX:
0005                        NOEX:
0006  _MainTask   D'0006    WAIT
0007  _task7      D'0007    RUN
0008                        NOEX:
0009                        NOEX:
```

```
Name                    Value
Activate Count:         D'0000
Current SP:             H'8C02217C
SP Bottom Address:      H'8C0221A4
SP Limit Address:       H'8C021DA4
SP Total Size:          H'00000400
SP Used Size:           H'00000020
SP Remain Size:         H'000003E0
Ovr Address:            ********
```

```
No.    Task ID  PC          Event
00001  No Task  H'80002f66  cre_cyc(0x000b,0x8c000c54)
00002  No Task  H'80002f66  E_OK
00003  D'0006   H'8000cf98  Task start(0x00000000)
00004  D'0006   H'8000cfb6  cre_flg(0x0006,0x8c02217c)
00005  D'0006   H'8000cfb6  E_OK
00006  D'0006   H'8000cfe6  cre_tsk(0x0007,0x8c02217c)
00007  D'0006   H'8000cfe6  E_OK
00008  D'0006   H'8000d000  wai_flg(0x0006,0x11111111,0x00000000,0x
00009  D'0007   H'8000d020  Task start(0x00000000)
```

**Figure 1.1 DX Overview**

## 1.2 Features

- Graphical user interface
  The status of objects such as tasks can be referred to and modified through windows and dialog boxes, and multitasking applications can be debugged in the HEW environment.

- Display of tracing information for service calls
  The history of service calls in RTOS systems can be displayed graphically. The user can also select the items to be displayed in this history.

## 1.3 Preparations for Using the DX

The followings must be done before using the DX for debugging.

**Install the DX:** Follow the instructions by the installer to install the DX.

**Configure the RTOS:** Configure the RTOS system you are using to match the DX. For an example of configuration, refer to section 2, Tutorial.

**Initiate the HEW:** Initiate the HEW so that multitasking debugging is enabled after a program has been built. For examples of the use of the DX, refer to section 2, Tutorial. For the functions of the DX, refer to section 3, Functions, and to the online help file.

# Section 2 Tutorial

Examples of the operation of the DX using the HI7000/4 series as the RTOS are explained in this section. For details, refer to the online help file.

The following environment is assumed.

- RTOS: HI7700/4
- Microcomputer: SH7729
- Debugger: SuperH™ RISC engine simulator/debugger

## 2.1 Configuring the Tutorial

The sample provided by the HI7700/4 is used. For details on the configuration, refer to section 5, Configuration, in the HI7000/4 Series User's Manual.

### 2.1.1 Configurator

The sample HCF file (hiuser\sh7729\7729.hcf) provided by the HI7700/4 is used to create the configuration file. The following configurator settings must be made.

- Check CFG_ACTION.
- Check CFG_TRACE.
- Set CFG_TRCTYPE to 'emulator trace'. (This setting allows RTOS tracing by the DX even in the simulator.)
- Set CFG_MAXTSKID to seven or more.
- Set CFG_STSTKID to five or less
- Set CFG_MAXTSKPRI to seven or more.
- Set CFG_MAXFLGID to six or more.
- Install the following service calls:
  cre_tsk, ext_tsk, cre_flg, wai_flg, and set_flg

### 2.1.2 Build

The whole linkage method of HI7700/4 is used in this example. Use the HEW to open the sample hios.hws, and select 7729_mix as the project name for the build. Select [Options->SuperH RISC engine Standard Toolchain…] and then [User] from [Gbr relative logic operation] in the [Optimize] category on the [C/C++] page. Select [Build->Build] to start a build.

## 2.2 Contents of the Sample Program

The sample source program is hiuser\tutorial\task.c, which includes two tasks, MainTask and Task7.

When the system is initiated, MainTask is automatically executed because creation and initiation of MainTask is specified by the configurator with the following conditions:

- Task ID: 6
- Initial priority: 6
- Start address: MainTask()
- Initial state: READY (the TA_ACT attribute is specified)

TASK7 is dynamically created by MainTask with the following conditions:

- Task ID: 7
- Initial priority level: 7
- Start address: Task7()
- Initial state: READY (TA_ACT attribute is specified)

The priority of MainTask is higher.

MainTask creates an event flag with ID6 and Task7, and uses the service call wai_flg to wait for the event flag with ID6. Then, the event flag with ID6 is deleted with the service call del_flg, and MainTask is exited by the service call ext_tsk.

Task7 uses the service call set_flg to set the event flag with ID6, then enters an endless loop with a while(1); statement.

Note that this sample program is created for the user to learn the operation of the kernel and the DX, and its contents in terms of processing have no meaning.

## 2.3 Executing the Sample Program

(1) Start up the simulator/debugger. Select [Options->Debug Settings...] in the HEW to display the [Debug Settings] dialog box. Then select 7729_mix from the list on the left, [SH3-DSP Simulator] from the [Target] list box on the [Target] page, and hiuser\obj_big\7729_mix.abs of HI7000/4 by the [Add…] button, respectively.

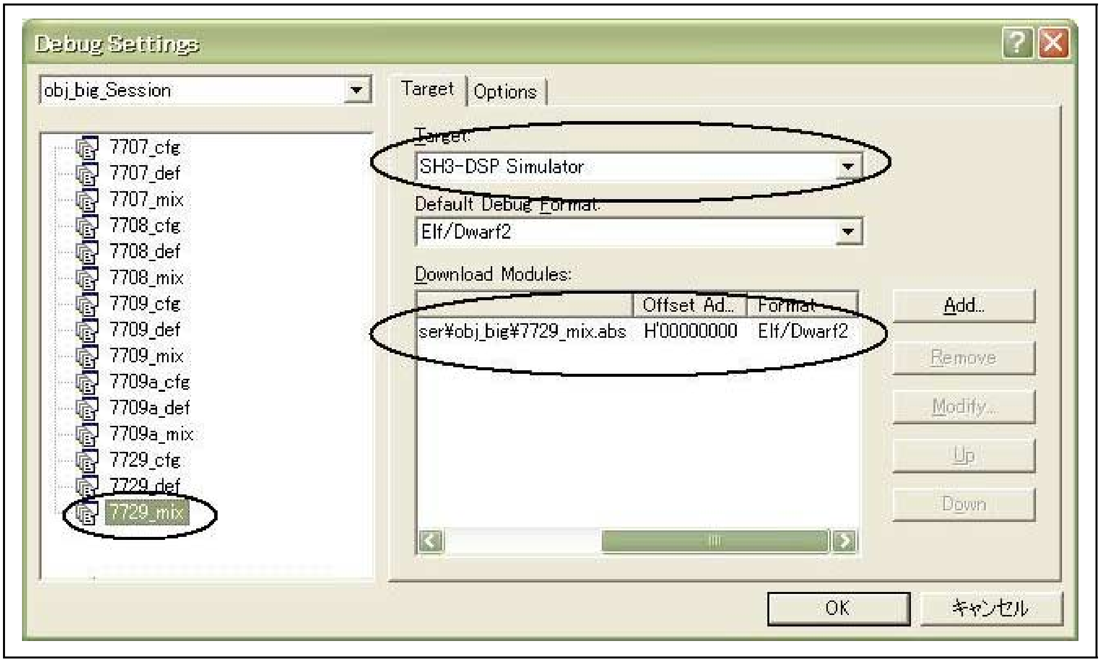


Figure 2.1 [Debug Settings] Dialog Box

(2) Start up the DX. Select [RTOS HI7000/4_4DX ECX] from [Project -> Components…]. Then press [Load] and [OK].

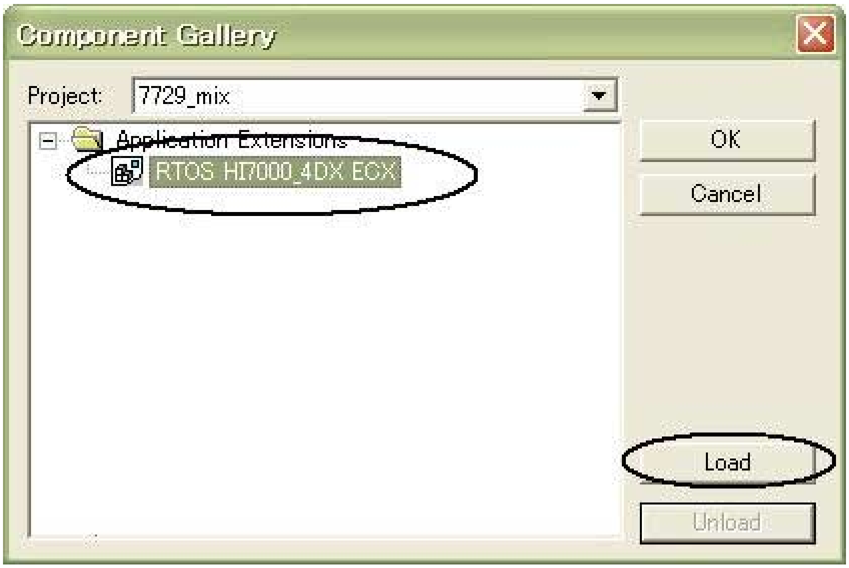
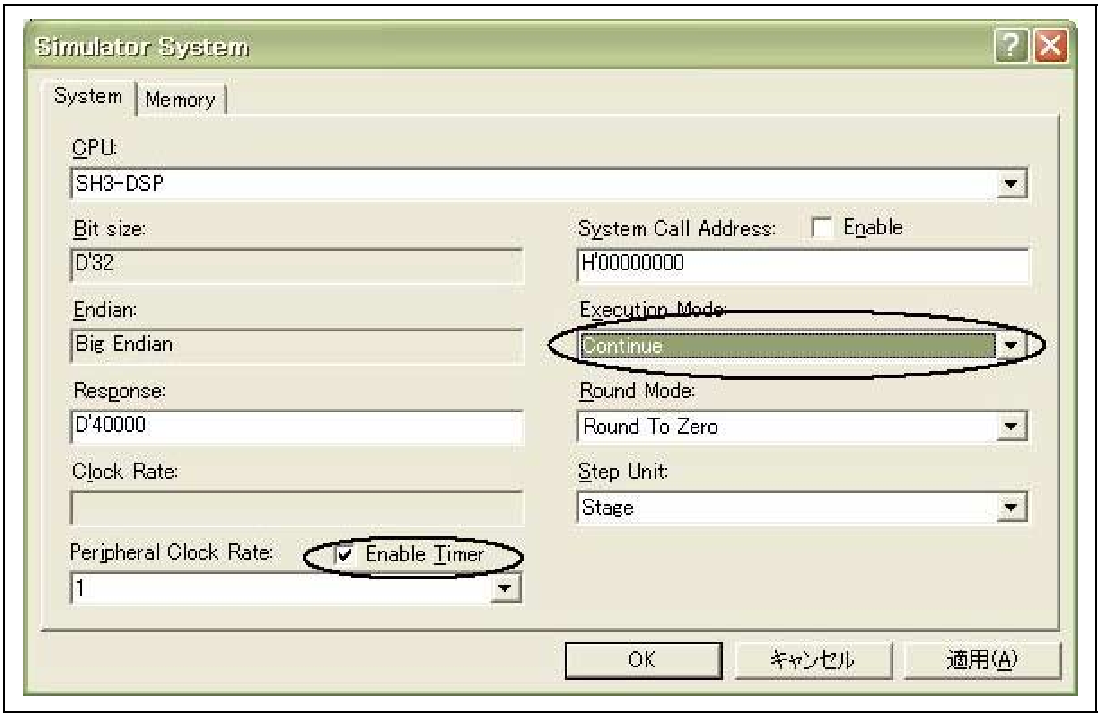


**Figure 2.2 [Component Gallery] Dialog Box**

(3) Set up the simulator/debugger. Check [Enable Timer] in the [Simulator System] dialog box displayed by selecting [Options -> Simulator -> System…], and select [Continue] from the [Execution Mode] list box.


Simulator System

System | Memory

CPU: SH3-DSP

Bit size: D'32

System Call Address: H'00000000 ☐ Enable

Endian: Big Endian

Execution Mode: Continue

Response: D'40000

Round Mode: Round To Zero

Clock Rate:

Step Unit: Stage

Peripheral Clock Rate: 1 ☑ Enable Timer

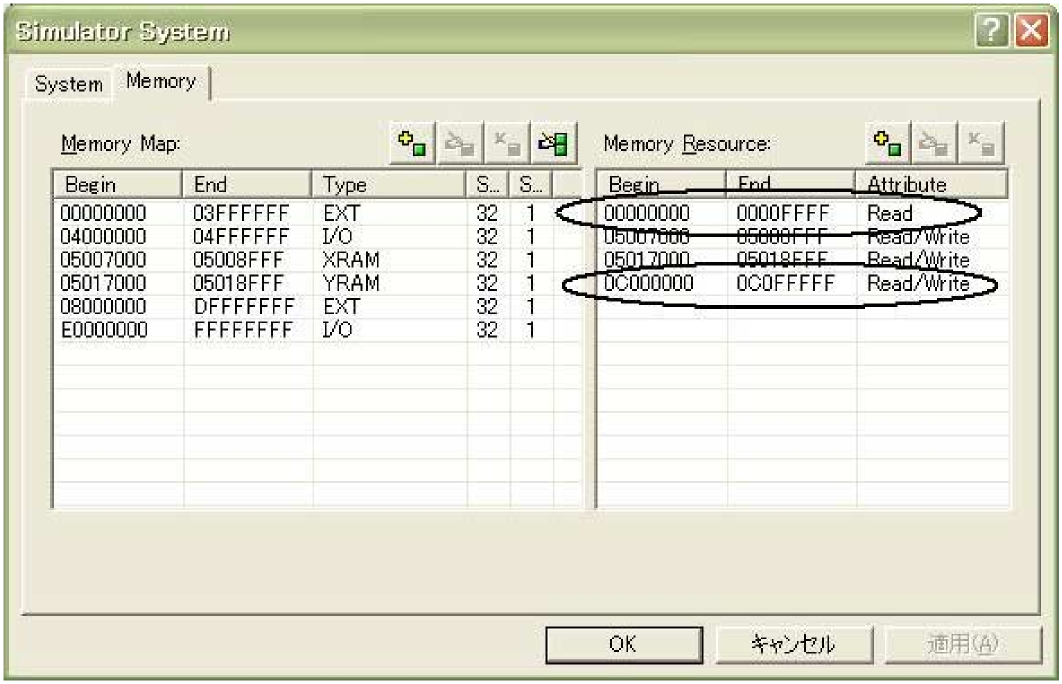
OK | キャンセル | 適用(A)

**Figure 2.3 [Simulator System] Dialog Box ([System] Page)**

(4) The memory resources listed in table 2.1 must be ensured in the [Memory] page of the [Simulator System] dialog box.

**Table 2.1 Memory Resources for the Simulator/Debugger**

| Start Address | End Address | Attribute |
|---|---|---|
| H’00000000 | H’0000FFFF | Read |
| H’0C000000 | H’0C0FFFFF | Read/Write |



Memory Map:

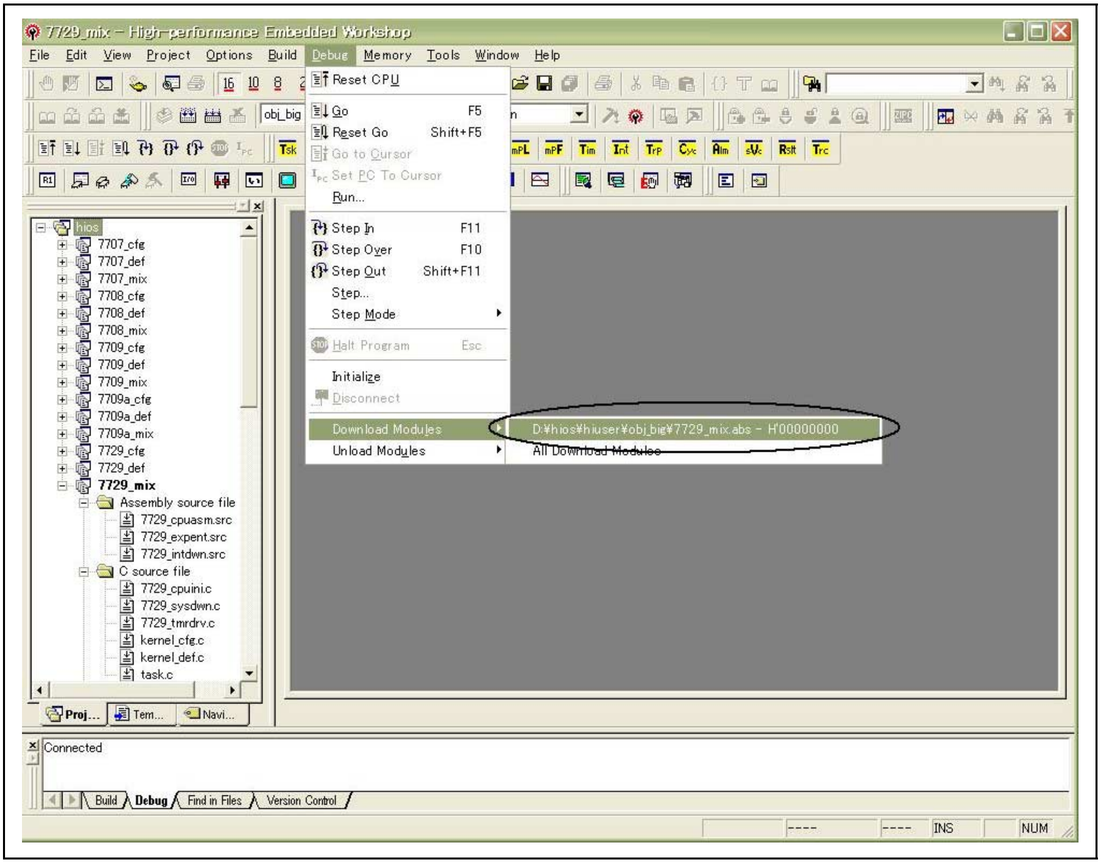
| Begin | End | Type | S... | S... |
|---|---|---|---|---|
| 00000000 | 03FFFFFF | EXT | 32 | 1 |
| 04000000 | 04FFFFFF | I/O | 32 | 1 |
| 05007000 | 05008FFF | XRAM | 32 | 1 |
| 05017000 | 05018FFF | YRAM | 32 | 1 |
| 08000000 | DFFFFFFF | EXT | 32 | 1 |
| E0000000 | FFFFFFFF | I/O | 32 | 1 |

Memory Resource:

| Begin | End | Attribute |
|---|---|---|
| 00000000 | 0000FFFF | Read |
| 05007000 | 05008FFF | Read/Write |
| 05017000 | 05018FFF | Read/Write |
| 0C000000 | 0C0FFFFF | Read/Write |

**Figure 2.4 [Simulator System] Dialog Box ([Memory] Page)**

(5) Download the load module. Select hiuser\obj_big\7729_mix.abs from [Debug->Download Modules].



Connected

**Figure 2.5 Downloading the Load Module**

(6) Make the settings for RTOS tracing. Select [View->RTOS->Trace]. Click the right-hand mouse button on the [RTOS Trace] window and then select [Set Trace Options…] from the popup menu.

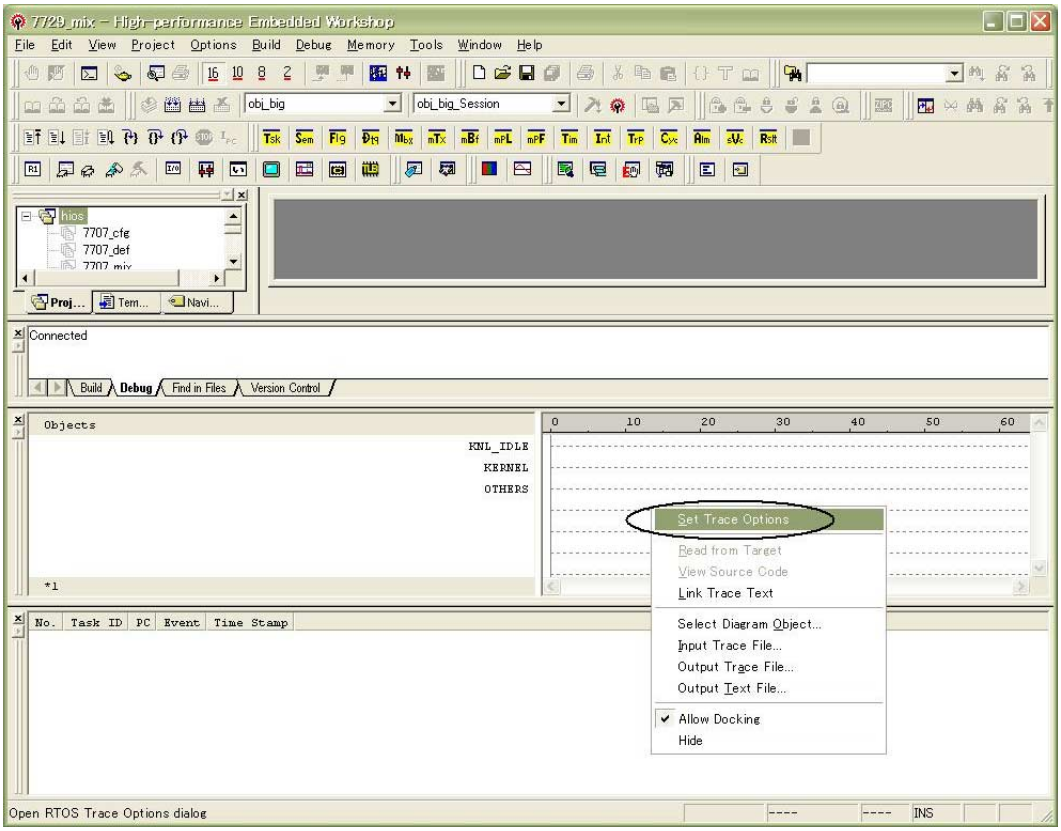
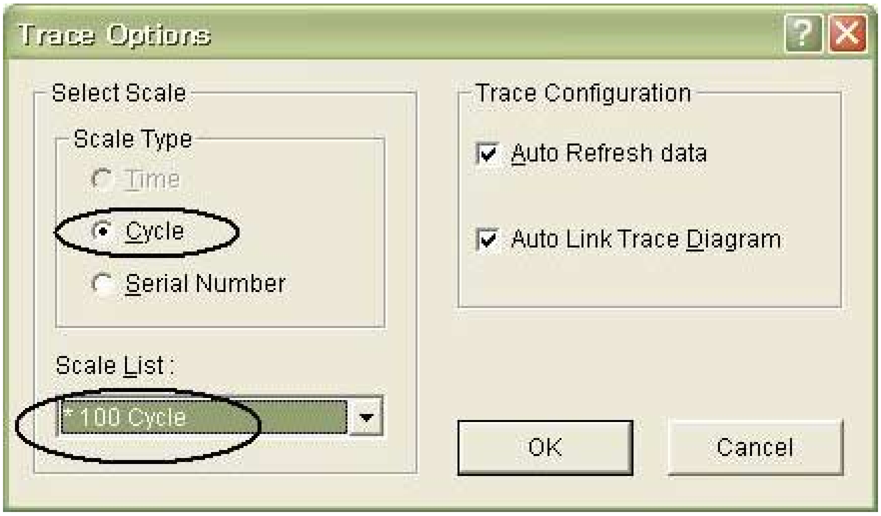


**Figure 2.6 Setting RTOS Tracing**

(7) Select [Cycle] from [Scale Type] and [*100 Cycle] from [Scale List], respectively.


Trace Options

Select Scale

Scale Type

Time

[✓ Cycle]

Serial Number

Scale List :

[✓ *100 Cycle]

Trace Configuration

✓ Auto Refresh data

✓ Auto Link Trace Diagram

**Figure 2.7 [Trace Options] Dialog Box**

(8) Select [Debug->Reset Go] to execute the program.

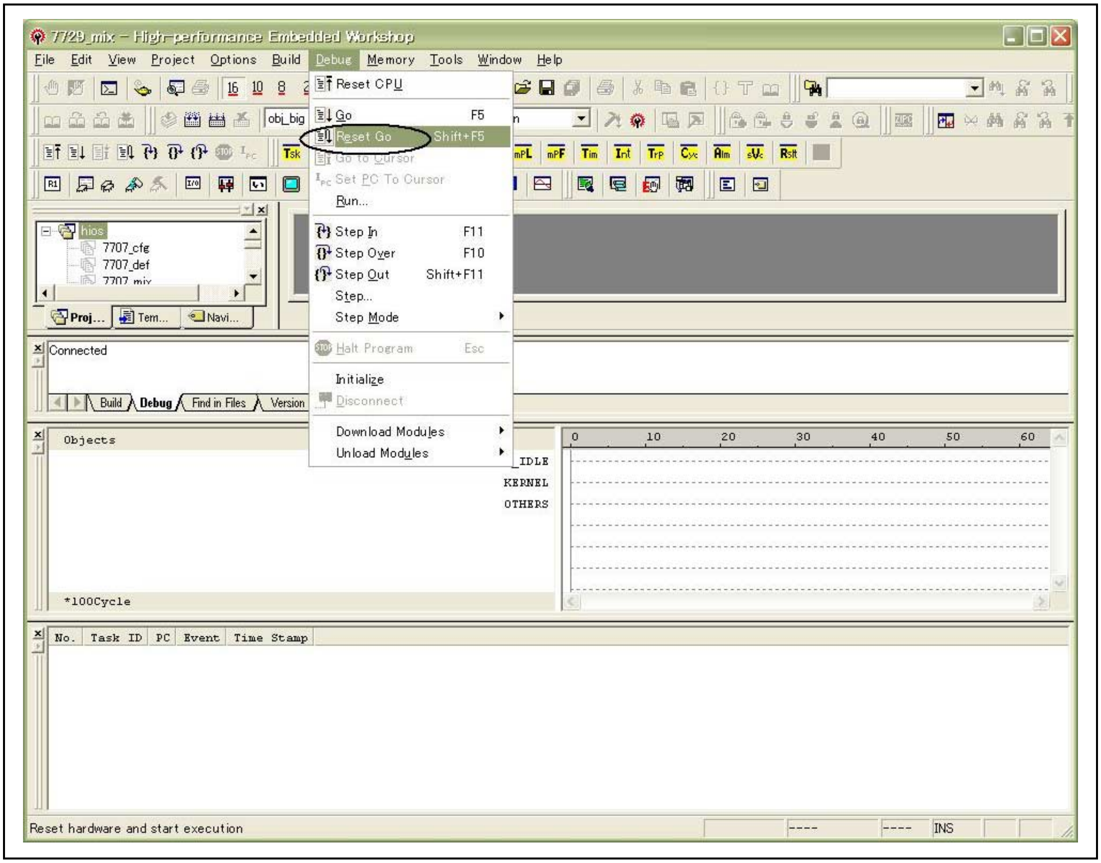


Figure 2.8 Executing the Program

(9) Select [Debug->Halt Program] to stop the program execution.

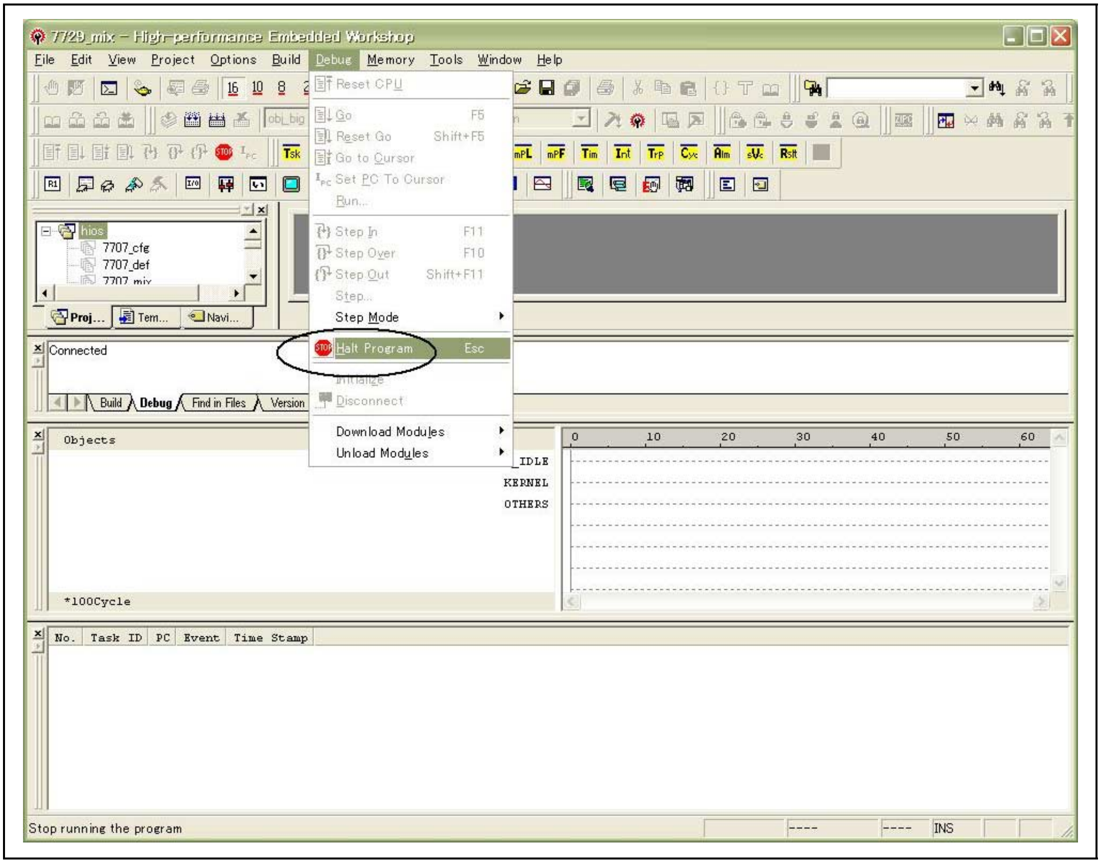
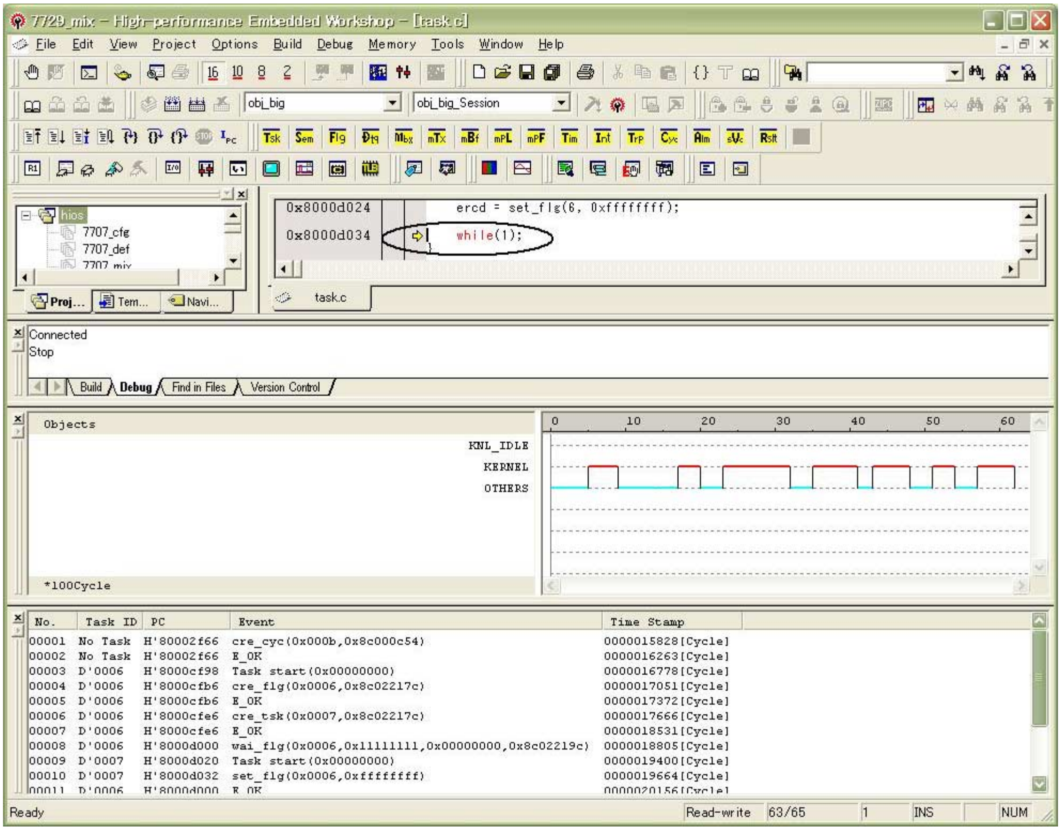


**Figure 2.9 Stopping the Program Execution**

(10) Execution stops within the while(1); statement of Task7 as shown in figure 2.10.



```
0x8000d024        ercd = set_flg(6, 0xffffffff);
0x8000d034        while(1);
```

```
Connected
Stop
```

```
No.    Task ID  PC          Event                                              Time Stamp
00001  No Task  H'80002f66  cre_cyc(0x000b,0x8c000c54)                         0000015828[Cycle]
00002  No Task  H'80002f66  E_OK                                               0000016263[Cycle]
00003  D'0006   H'8000cf98  Task start(0x00000000)                             0000016778[Cycle]
00004  D'0006   H'8000cfb6  cre_flg(0x0006,0x8c02217c)                         0000017051[Cycle]
00005  D'0006   H'8000cfb6  E_OK                                               0000017372[Cycle]
00006  D'0006   H'8000cfe6  cre_tsk(0x0007,0x8c02217c)                         0000017666[Cycle]
00007  D'0006   H'8000cfe6  E_OK                                               0000018531[Cycle]
00008  D'0006   H'8000d000  wai_flg(0x0006,0x11111111,0x00000000,0x8c02219c)   0000018805[Cycle]
00009  D'0007   H'8000d020  Task start(0x00000000)                             0000019400[Cycle]
00010  D'0007   H'8000d032  set_flg(0x0006,0xffffffff)                         0000019664[Cycle]
00011  D'0006   H'8000d000  E_OK                                               0000020156[Cycle]
```

**Figure 2.10 Source Window with the Execution Stopped**

(11) You can check how the program was executed by using the service-call trace function. To display MainTask (task ID 6) and Task7 (task ID 7) in such a way that the diagram is easy to see, click the right-hand mouse button on the [RTOS Trace] window and select [Select Diagram Object...] from the pop-up menu. Add task IDs 6 and 7, and close the dialog box.

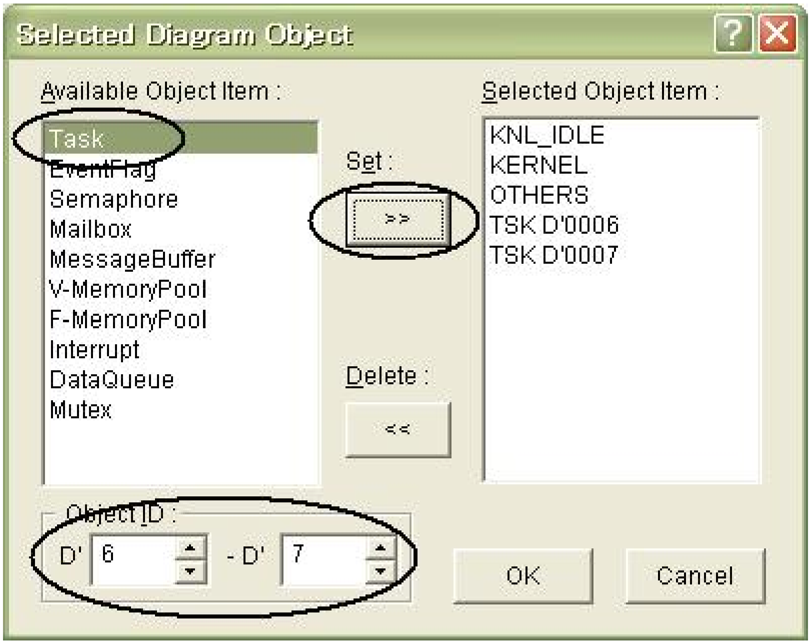
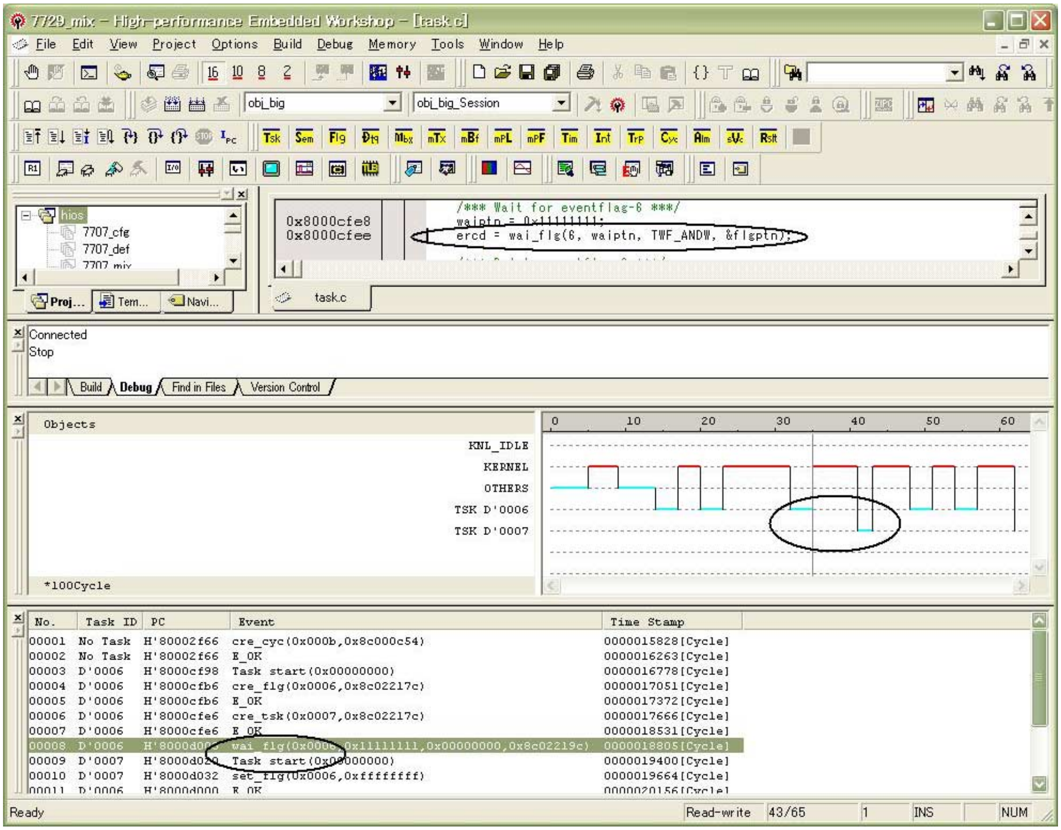


**Figure 2.11 [Select Diagram Object] Window**

(12) Task IDs 6 and 7 are displayed in the [RTOS Trace Diagram] window. With a glance at the window, you can see that the execution of Task7 starts after MainTask has issued wai_flg. Double-clicking the row where the service call is displayed in the [RTOS Trace Text] window opens the corresponding source window.



```
0x8000cfe8      /*** Wait for eventflag-6 ***/
                waiptn = 0x11111111;
0x8000cfee      ercd = wai_flg(6, waiptn, TWF_ANDW, &flgptn);
```

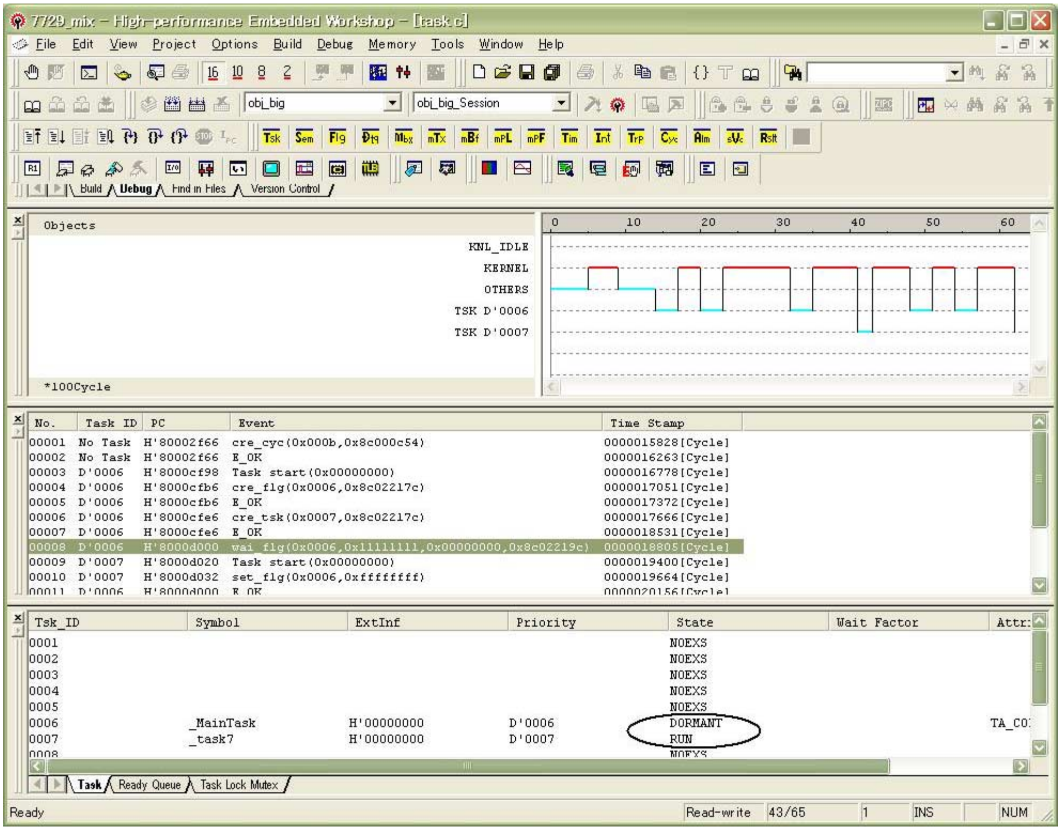
Connected
Stop

```
No.    Task ID  PC          Event                                                Time Stamp
00001  No Task  H'80002f66  cre_cyc(0x000b,0x8c000c54)                           0000015828[Cycle]
00002  No Task  H'80002f66  E_OK                                                 0000016263[Cycle]
00003  D'0006   H'8000cf98  Task start(0x00000000)                               0000016778[Cycle]
00004  D'0006   H'8000cfb6  cre_flg(0x0006,0x8c02217c)                           0000017051[Cycle]
00005  D'0006   H'8000cfb6  E_OK                                                 0000017372[Cycle]
00006  D'0006   H'8000cfe6  cre_tsk(0x0007,0x8c02217c)                           0000017666[Cycle]
00007  D'0006   H'8000cfe6  E_OK                                                 0000018531[Cycle]
00008  D'0006   H'8000d00c  wai_flg(0x0006,0x11111111,0x00000000,0x8c02219c)     0000018805[Cycle]
00009  D'0007   H'8000d020  Task start(0x00000000)                               0000019400[Cycle]
00010  D'0007   H'8000d032  set_flg(0x0006,0xffffffff)                           0000019664[Cycle]
00011  D'0006   H'8000d000  E_OK                                                 0000020156[Cycle]
```

Ready

**Figure 2.12 [RTOS Trace] Window**

(13) You can check the states of MainTask and Task7 in the [Task] window. Select [View ->RTOS->Task] to open the [Task] window. The MainTask is in the DORMANT state, while Task7 is in the RUN state.

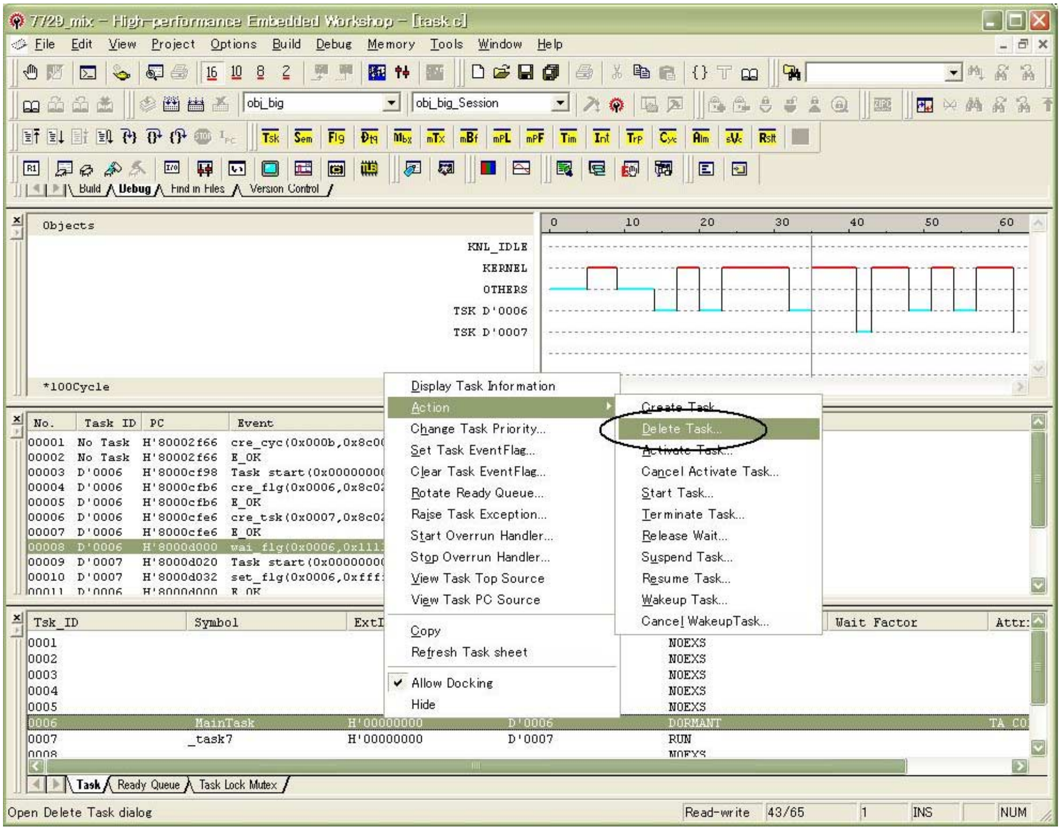

7729_mix - High-performance Embedded Workshop - [task.c]
File Edit View Project Options Build Debug Memory Tools Window Help
obj_big
obj_big_Session
Build Debug Find in Files Version Control
Objects
KNL_IDLE
KERNEL
OTHERS
TSK D'0006
TSK D'0007
*100Cycle
0 10 20 30 40 50 60
Task Ready Queue Task Lock Mutex
Ready
Read-write 43/65 1 INS NUM

```
No.    Task ID  PC          Event                                                 Time Stamp
00001  No Task  H'80002f66  cre_cyc(0x000b,0x8c000c54)                            0000015828[Cycle]
00002  No Task  H'80002f66  E_OK                                                  0000016263[Cycle]
00003  D'0006   H'8000cf98  Task start(0x00000000)                                0000016778[Cycle]
00004  D'0006   H'8000cfb6  cre_flg(0x0006,0x8c02217c)                            0000017051[Cycle]
00005  D'0006   H'8000cfb6  E_OK                                                  0000017372[Cycle]
00006  D'0006   H'8000cfe6  cre_tsk(0x0007,0x8c02217c)                            0000017666[Cycle]
00007  D'0006   H'8000cfe6  E_OK                                                  0000018531[Cycle]
00008  D'0006   H'8000d000  wai_flg(0x0006,0x11111111,0x00000000,0x8c02219c)      0000018805[Cycle]
00009  D'0007   H'8000d020  Task start(0x00000000)                                0000019400[Cycle]
00010  D'0007   H'8000d032  set_flg(0x0006,0xffffffff)                            0000019664[Cycle]
00011  D'0006   H'8000d000  E_OK                                                  0000020156[Cycle]
```

```
Tsk_ID   Symbol       ExtInf       Priority   State     Wait Factor   Attr:
0001                                          NOEXS
0002                                          NOEXS
0003                                          NOEXS
0004                                          NOEXS
0005                                          NOEXS
0006     _MainTask    H'00000000   D'0006     DORMANT                 TA_CO
0007     _task7       H'00000000   D'0007     RUN
0008                                          NOEXS
```

**Figure 2.13 [Task] Window**

(14) Next, execute the user program when the priority of MainTask is lower than Task7. First, MainTask and Task7 must be deleted to return them to their initial state. To delete MainTask, select (highlight) MainTask in the [Task] window, click the right-hand mouse button to launch the pop-up menu, then select [Action->Delete Task...] from this pop-up menu.

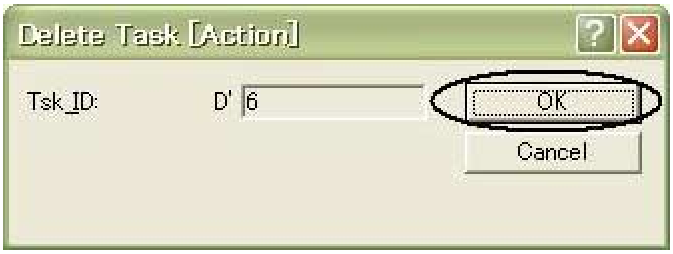


```
No.    Task ID  PC           Event
00001  No Task  H'80002f66   cre_cyc(0x000b,0x8c0
00002  No Task  H'80002f66   E_OK
00003  D'0006   H'8000cf98   Task start(0x0000000
00004  D'0006   H'8000cfb6   cre_flg(0x0006,0x8c0
00005  D'0006   H'8000cfb6   E_OK
00006  D'0006   H'8000cfe6   cre_tsk(0x0007,0x8c0
00007  D'0006   H'8000cfe6   E_OK
00008  D'0006   H'8000d000   wai_flg(0x0006,0x111
00009  D'0007   H'8000d020   Task start(0x0000000
00010  D'0007   H'8000d032   set_flg(0x0006,0xfff
00011  D'0006   H'8000d000   E_OK
```

```
Tsk_ID  Symbol     ExtI                               Wait Factor   Attr
0001                                          NOEXS
0002                                          NOEXS
0003                                          NOEXS
0004                                          NOEXS
0005                                          NOEXS
0006    MainTask   H'00000000   D'0006        DORMANT               TA_CO
0007    _task7     H'00000000   D'0007        RUN
0008                                          NOEXS
```

**Figure 2.14 [Task] Window Pop-Up Menu**

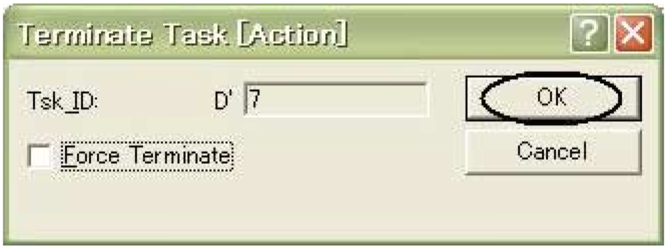
(15) Press the [OK] button in the [Delete Task] dialog box.



Tsk_ID: D' 6

**Figure 2.15 [Delete Task] Dialog Box**

(16) Since Task7 is still in the RUN state, it must be forcibly terminated before deletion. To terminate Task7, select (highlight) Task7 in the [Task] window, click the right-hand mouse button to launch the pop-up menu, then select [Action->Terminate Task...]. This opens the [Terminate Task] dialog box, so press the [OK] button. Task7 can then be deleted in the same way as MainTask.

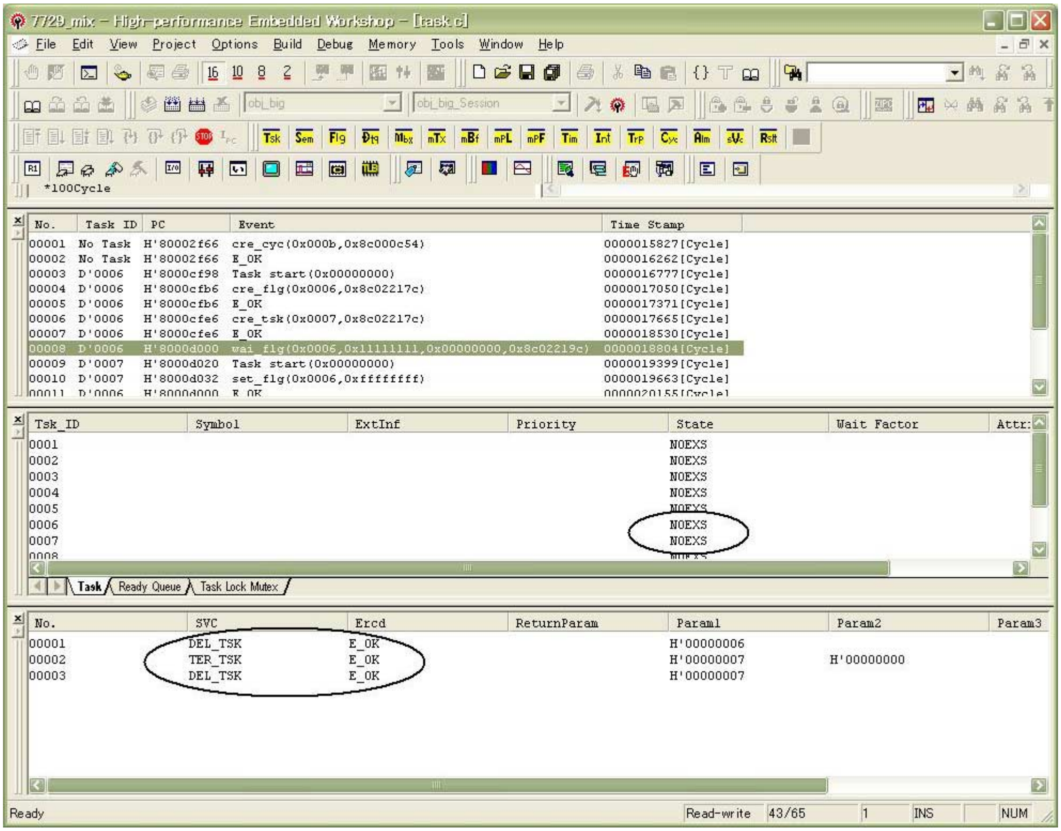


Tsk_ID: D' 7

☐ Force Terminate

**Figure 2.16 [Terminate Task] Dialog Box**

(17) You can check the result of the manipulation of kernel objects, such as the termination and deletion of a task, by selecting [View->RTOS->Action Result]. However, even if the [Action Result] window is opened, the results will not be visible here. The DX sends requests for the execution of service calls to the kernel to realize manipulation of kernel objects. Thus a task will be actually deleted after execution by the kernel. To obtain the results in this case, resume the execution of the target by selecting [Debug->Go]. The results will be automatically displayed. Then select [Refresh task sheet] from the pop-up menu of the [Task] window. You will see that MainTask (task ID 6) and Task7 (task ID 7) are both in the NOEXS state.



7729_mix - High-performance Embedded Workshop - [task.c]
File Edit View Project Options Build Debug Memory Tools Window Help
obj_big
obj_big_Session
*100Cycle
Task Ready Queue Task Lock Mutex
Ready
Read-write 43/65 1 INS NUM

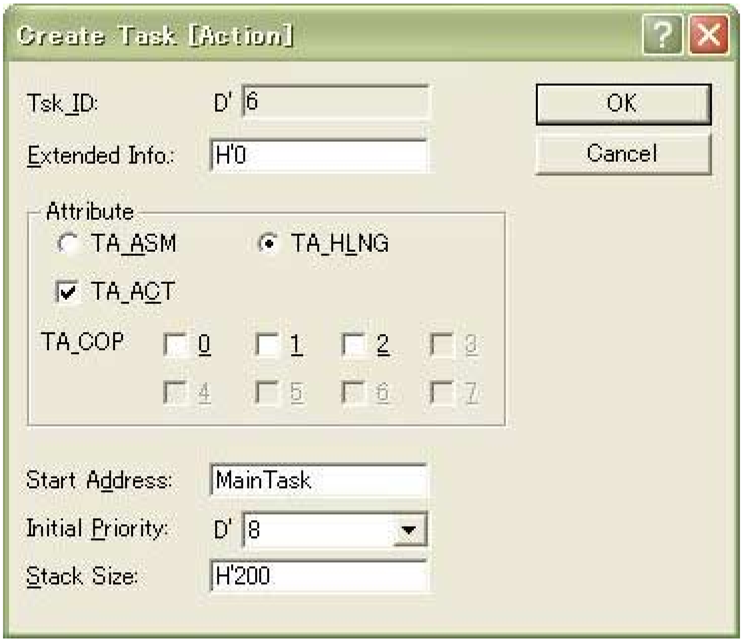
```
No.   Task ID  PC         Event                                               Time Stamp
00001 No Task  H'80002f66 cre_cyc(0x000b,0x8c000c54)                          0000015827[Cycle]
00002 No Task  H'80002f66 E_OK                                                0000016262[Cycle]
00003 D'0006   H'8000cf98 Task start(0x00000000)                              0000016777[Cycle]
00004 D'0006   H'8000cfb6 cre_flg(0x0006,0x8c02217c)                          0000017050[Cycle]
00005 D'0006   H'8000cfb6 E_OK                                                0000017371[Cycle]
00006 D'0006   H'8000cfe6 cre_tsk(0x0007,0x8c02217c)                          0000017665[Cycle]
00007 D'0006   H'8000cfe6 E_OK                                                0000018530[Cycle]
00008 D'0006   H'8000d000 wai_flg(0x0006,0x11111111,0x00000000,0x8c02219c)    0000018804[Cycle]
00009 D'0007   H'8000d020 Task start(0x00000000)                              0000019399[Cycle]
00010 D'0007   H'8000d032 set_flg(0x0006,0xffffffff)                          0000019663[Cycle]
00011 D'0006   H'8000d000 E_OK                                                0000020155[Cycle]
```

```
Tsk_ID   Symbol   ExtInf   Priority   State   Wait Factor   Attr:
0001                                  NOEXS
0002                                  NOEXS
0003                                  NOEXS
0004                                  NOEXS
0005                                  NOEXS
0006                                  NOEXS
0007                                  NOEXS
0008
```

```
No.     SVC       Ercd   ReturnParam   Param1       Param2       Param3
00001   DEL_TSK   E_OK                 H'00000006
00002   TER_TSK   E_OK                 H'00000007   H'00000000
00003   DEL_TSK   E_OK                 H'00000007
```

**Figure 2.17 Result of the Manipulation of Objects**

(18) Select [Debug->Halt Program] to stop the program execution. To create and initiate MainTask with a priority of 8 that is lower than Task7, select (highlight) task ID 6 in the [Task] window, and then select [Action->Create Task...] from the pop-up menu. This opens the [Create Task] dialog box. Make the entries as shown in figure 2.18.


Create Task [Action]

Tsk_ID: D' 6

Extended Info.: H'0

Attribute
- TA_ASM [✓ TA_HLNG]
- [✓ TA_ACT]

TA_COP: 0, 1, 2, 3, 4, 5, 6, 7 (none checked)

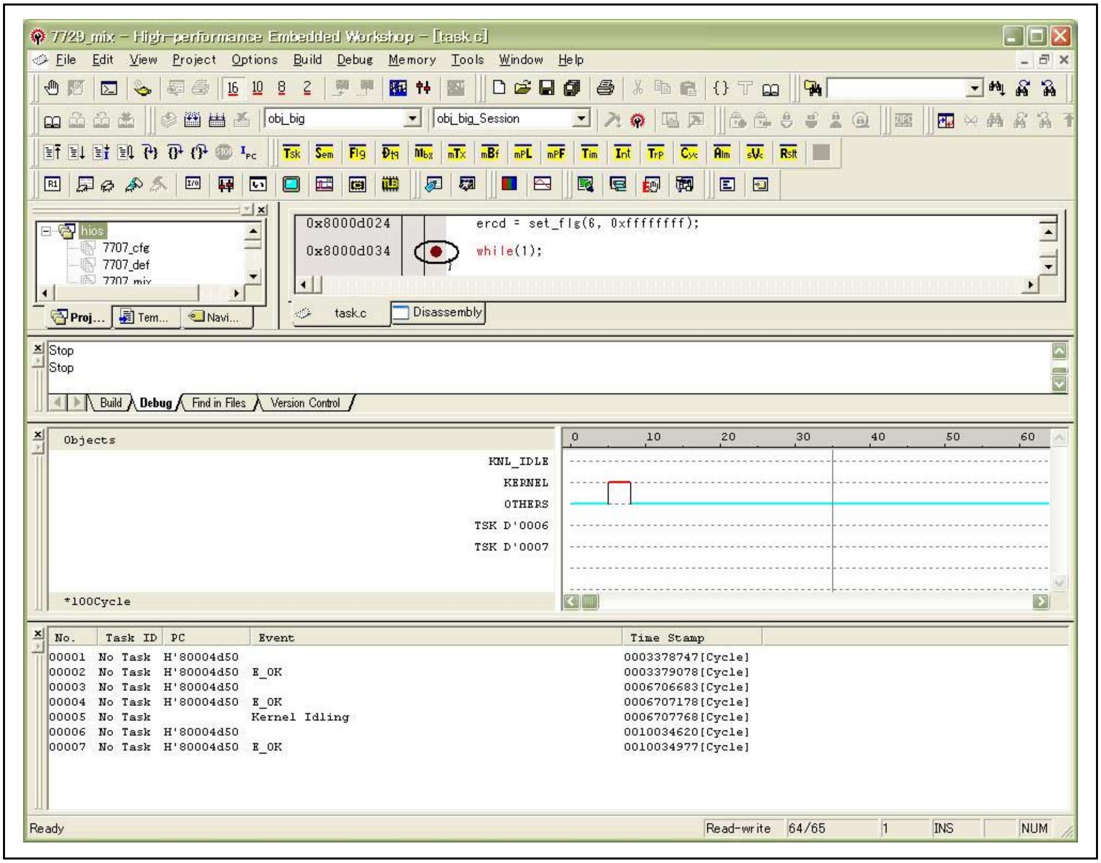
Start Address: MainTask

Initial Priority: D' 8

Stack Size: H'200

**Figure 2.18 [Create Task] Dialog Box**

(19) Selecting [Hide] from the pop-up menu of the [Task] window or [Action Result] window will close this window. Before doing so, a breakpoint has to be set so that execution of the program will stop at the while(1); statement of Task7. To set a breakpoint, double-click the part circled in figure 2.19.


7729_mix - High-performance Embedded Workshop - [task.c]
File Edit View Project Options Build Debug Memory Tools Window Help
obj_big
obj_big_Session
hios
7707_cfg
7707_def
7707_mix
Proj... Tem... Navi...
task.c Disassembly
Build Debug Find in Files Version Control
Ready Read-write 64/65 1 INS NUM

```
0x8000d024        ercd = set_flg(6, 0xffffffff);
0x8000d034   ●    while(1);
```

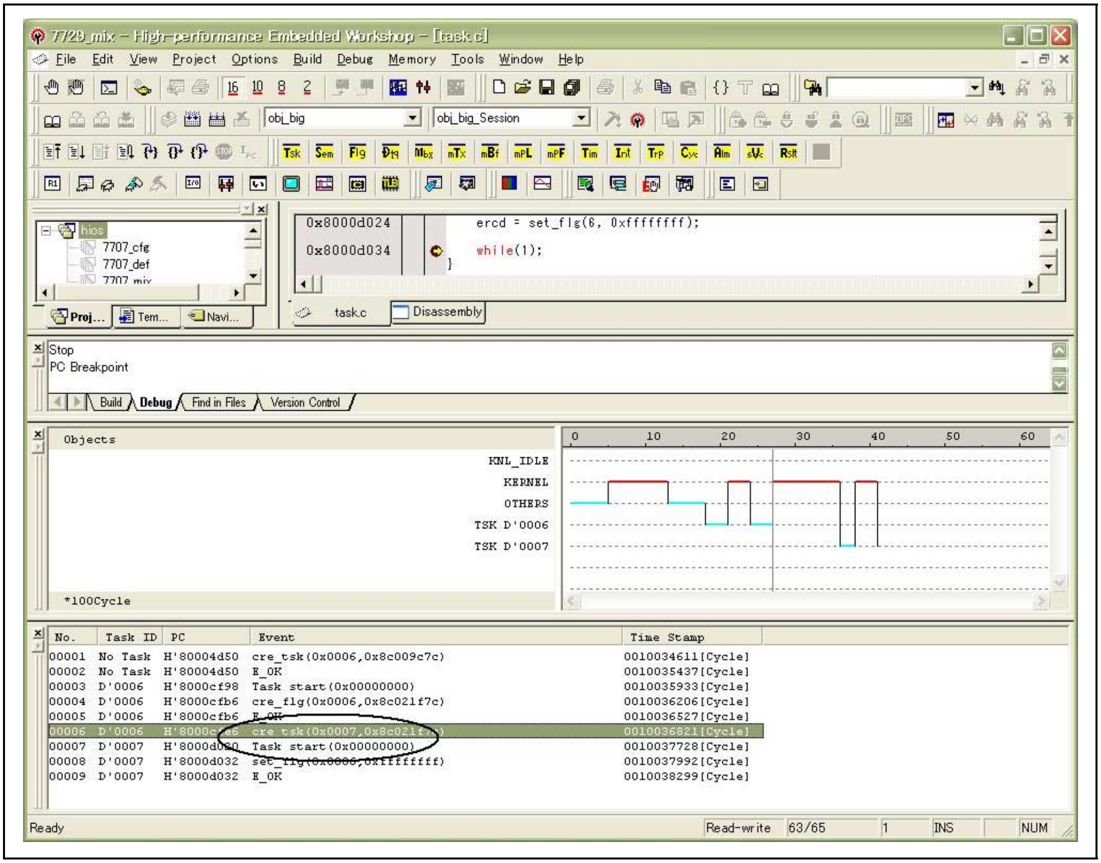
```
Stop
Stop
```

```
Objects
KNL_IDLE
KERNEL
OTHERS
TSK D'0006
TSK D'0007
*100Cycle
```

```
No.    Task ID  PC          Event            Time Stamp
00001  No Task  H'80004d50                   0003378747[Cycle]
00002  No Task  H'80004d50  E_OK             0003379078[Cycle]
00003  No Task  H'80004d50                   0006706683[Cycle]
00004  No Task  H'80004d50  E_OK             0006707178[Cycle]
00005  No Task              Kernel Idling    0006707768[Cycle]
00006  No Task  H'80004d50                   0010034620[Cycle]
00007  No Task  H'80004d50  E_OK             0010034977[Cycle]
```

**Figure 2.19 Setting a Breakpoint**

(20) When the execution of the target is resumed, MainTask is initiated, and execution stops at the breakpoint which was set at the while(1); statement of Task7. Check the execution by updating the information displayed for the service-call trace. Since the execution is switched to Task7 after the cre_tsk service call by MainTask, MainTask has not yet issued the wai_flg service call.



```
0x8000d024        ercd = set_flg(6, 0xffffffff);
0x8000d034        while(1);
                }
```

Stop
PC Breakpoint

```
No.    Task ID  PC          Event                         Time Stamp
00001  No Task  H'80004d50  cre_tsk(0x0006,0x8c009c7c)    0010034611[Cycle]
00002  No Task  H'80004d50  E_OK                          0010035437[Cycle]
00003  D'0006   H'8000cf98  Task start(0x00000000)        0010035933[Cycle]
00004  D'0006   H'8000cfb6  cre_flg(0x0006,0x8c021f7c)    0010036206[Cycle]
00005  D'0006   H'8000cfb6  E_OK                          0010036527[Cycle]
00006  D'0006   H'8000cfe6  cre_tsk(0x0007,0x8c021f7c)    0010036821[Cycle]
00007  D'0007   H'8000d020  Task start(0x00000000)        0010037728[Cycle]
00008  D'0007   H'8000d032  set_flg(0x0006,0xffffffff)    0010037992[Cycle]
00009  D'0007   H'8000d032  E_OK                          0010038299[Cycle]
```

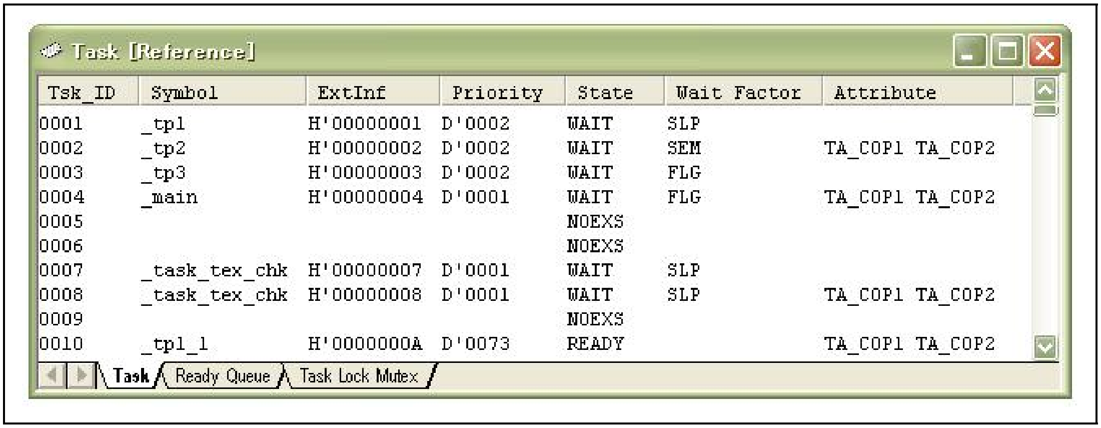
**Figure 2.20 [RTOS Trace Diagram] Window and [RTOS Trace Text] Window**

# Section 3 Functions

This section is an outline of the functions of the DX. For details on these functions, refer to the online help file.

## 3.1 Referencing the States of Objects

Selecting an object (such as a task) from the [View->RTOS] menu will display a window which shows the state of each object type, as shown in figure 3.1.



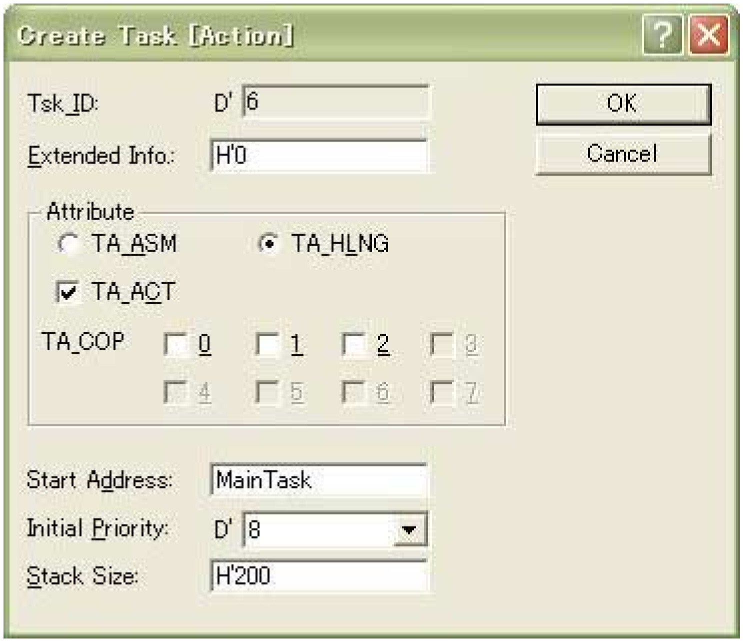
| Tsk_ID | Symbol | ExtInf | Priority | State | Wait Factor | Attribute |
|---|---|---|---|---|---|---|
| 0001 | _tp1 | H'00000001 | D'0002 | WAIT | SLP | |
| 0002 | _tp2 | H'00000002 | D'0002 | WAIT | SEM | TA_COP1 TA_COP2 |
| 0003 | _tp3 | H'00000003 | D'0002 | WAIT | FLG | |
| 0004 | _main | H'00000004 | D'0001 | WAIT | FLG | TA_COP1 TA_COP2 |
| 0005 | | | | NOEXS | | |
| 0006 | | | | NOEXS | | |
| 0007 | _task_tex_chk | H'00000007 | D'0001 | WAIT | SLP | |
| 0008 | _task_tex_chk | H'00000008 | D'0001 | WAIT | SLP | TA_COP1 TA_COP2 |
| 0009 | | | | NOEXS | | |
| 0010 | _tp1_1 | H'0000000A | D'0073 | READY | | TA_COP1 TA_COP2 |

**Figure 3.1 Example Display of Task State**

## 3.2 Manipulating the Objects

From the window for each object type, the user can request the manipulation of the object to the kernel. To use this function, the RTOS you are using needs to be configured so as to allow the DX to manipulate the objects.

The manipulation of each object can be requested through the dialog box that can be opened from the pop-up menu of each object window, as shown in figure 3.2.


Create Task [Action]

Tsk_ID: D' 6

Extended Info.: H'0

Attribute

TA_ASM [✓ TA_HLNG]

[✓ TA_ACT]

TA_COP 0 1 2 3 4 5 6 7

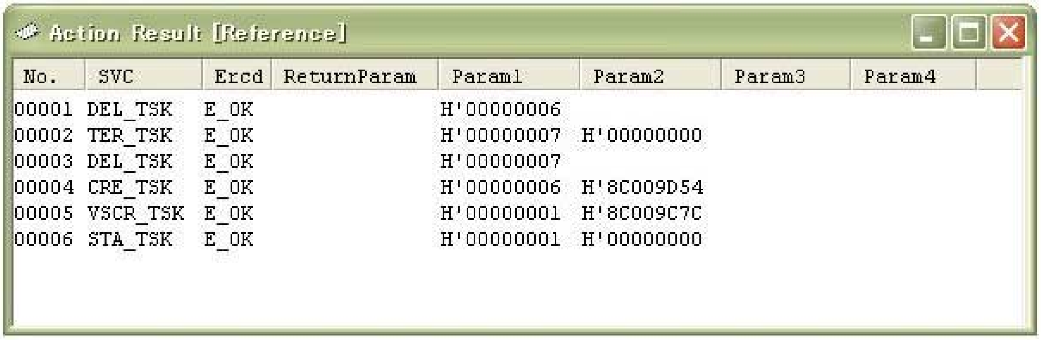
Start Address: MainTask

Initial Priority: D' 8

Stack Size: H'200

**Figure 3.2 Example of a Request for Manipulation of an Object**

The request for manipulation of the object is sent to the target, and the corresponding service call is issued. The DX will display the results in the [Action Result] window, as shown in figure 3.3.



| No. | SVC | Ercd | ReturnParam | Param1 | Param2 | Param3 | Param4 |
|---|---|---|---|---|---|---|---|
| 00001 | DEL_TSK | E_OK | | H'00000006 | | | |
| 00002 | TER_TSK | E_OK | | H'00000007 | H'00000000 | | |
| 00003 | DEL_TSK | E_OK | | H'00000007 | | | |
| 00004 | CRE_TSK | E_OK | | H'00000006 | H'8C009D54 | | |
| 00005 | VSCR_TSK | E_OK | | H'00000001 | H'8C009C7C | | |
| 00006 | STA_TSK | E_OK | | H'00000001 | H'00000000 | | |

**Figure 3.3 Result of Manipulation of an Object**

The processing of the request for manipulation of an object starts when the target is executed. A request can be made for the manipulation of the object after the target has been stopped, but execution will not start until the next execution of the target.

The request for manipulation of the object is queued in the target memory, and the release of the request from the queue is indicated by the display of the result in the [Action Result] window. Up to four requests can be queued.

If manipulation of an object is requested after the target has been stopped, the result is not displayed in the [Action Result] window until the next execution of the target. Accordingly, up to four requests for manipulation of an object can be made at the same time while the target is stopped.

The display in the [Action Result] window is updated with the following timing:

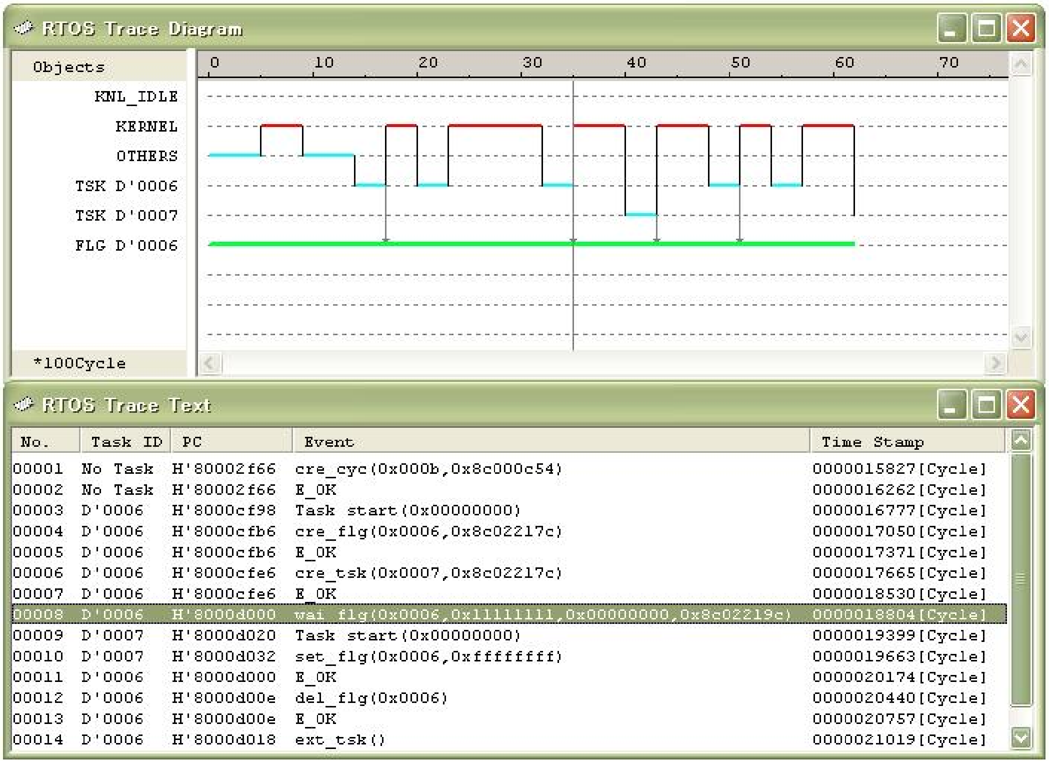
- During execution of the target (cyclic)
- [Refresh] is selected from the pop-up menu displayed by clicking the right-hand mouse button in the [Action Result] window
- User system break (stopped)

## 3.3 Displaying Service Call Trace

When [View->RTOS->Trace] is selected, the history of service calls is displayed in diagrammatic and textual forms, as shown in figure 3.4. To use this function, the RTOS you are using needs to be configured so as to enable display of service call tracing by the DX.

The location of the data that forms the history of service call tracing can be selected as the tool (simulator/debugger or emulator) or the target memory. Depending on the configurator in use, ‘Emulator’ may be displayed instead of ‘Tool’. Note that this configurator can also be used with the simulator/debugger.

The acquired information can be saved in a file, and this file can be read and displayed.



| No. | Task ID | PC | Event | Time Stamp |
|---|---|---|---|---|
| 00001 | No Task | H'80002f66 | cre_cyc(0x000b,0x8c000c54) | 0000015827[Cycle] |
| 00002 | No Task | H'80002f66 | E_OK | 0000016262[Cycle] |
| 00003 | D'0006 | H'8000cf98 | Task start(0x00000000) | 0000016777[Cycle] |
| 00004 | D'0006 | H'8000cfb6 | cre_flg(0x0006,0x8c02217c) | 0000017050[Cycle] |
| 00005 | D'0006 | H'8000cfb6 | E_OK | 0000017371[Cycle] |
| 00006 | D'0006 | H'8000cfe6 | cre_tsk(0x0007,0x8c02217c) | 0000017665[Cycle] |
| 00007 | D'0006 | H'8000cfe6 | E_OK | 0000018530[Cycle] |
| 00008 | D'0006 | H'8000d000 | wai_flg(0x0006,0x11111111,0x00000000,0x8c02219c) | 0000018804[Cycle] |
| 00009 | D'0007 | H'8000d020 | Task start(0x00000000) | 0000019399[Cycle] |
| 00010 | D'0007 | H'8000d032 | set_flg(0x0006,0xffffffff) | 0000019663[Cycle] |
| 00011 | D'0006 | H'8000d000 | E_OK | 0000020174[Cycle] |
| 00012 | D'0006 | H'8000d00e | del_flg(0x0006) | 0000020440[Cycle] |
| 00013 | D'0006 | H'8000d00e | E_OK | 0000020757[Cycle] |
| 00014 | D'0006 | H'8000d018 | ext_tsk() | 0000021019[Cycle] |

**Figure 3.4 [RTOS Trace Diagram] Window and [RTOS Trace Text] Window**

## 3.4 List of Functions

### 3.4.1 Menu Items

After the DX has been installed, the items listed in table 3.1, which allow accesses to the DX functions, are added to the [View] menu of the HEW.

**Table 3.1 Items to be Added to the HEW's [View] Menu**

| Pull-Down Menu | Submenu | Function |
|---|---|---|
| RTOS | Task | Opens the [Task] window |
| | Semaphore | Opens the [Semaphore] window |
| | EventFlag | Opens the [EventFlag] window |
| | DataQueue | Opens the [DataQueue] window |
| | Mailbox | Opens the [Mailbox] window |
| | Mutex | Opens the [Mutex] window |
| | MessageBuffer | Opens the [MessageBuffer] window |
| | V-MemoryPool | Opens the [V-MemoryPool] window |
| | F-MemoryPool | Opens the [F-MemoryPool] window |
| | Timer | Opens the [Timer] window |
| | Interrupt | Opens the [Interrupt] window |
| | Trap | Opens the [Trap] window |
| | Cyclic | Opens the [Cyclic] window |
| | Alarm | Opens the [Alarm] window |
| | Extended SVC | Opens the [Extended SVC] window |
| | Action Result | Opens the [Action Result] window |
| | Trace | Opens the [Trace] window |

Selecting [View->CPU->Status] displays the DX state in the [Status] window common in HEW.

Note that some functions are not supported depending on the RTOS in use.

### 3.4.2 Windows and Dialog Boxes

Table 3.2 is a list of windows and dialog boxes for the DX. For further details on the windows and dialog boxes, refer to the online help file. The online help file can be launched by selecting [Help ->RTOS Help].

**Table 3.2 Windows and Dialog Boxes**

| Classification | Window Name | Functions |
|---|---|---|
| Tasks | [Task] window | • Displays the states of all tasks<br>• Displays the ready queue<br>• Displays locked mutexes for all tasks |
| | [Task Detail Information] window | Displays detailed information on tasks |
| | [Create Task] dialog box* | Creates tasks |
| | [Delete Task] dialog box* | Deletes tasks |
| | [Activate Task] dialog box* | Initiates tasks |
| | [Cancel Activate Task] dialog box* | Cancels request for task initiation |
| | [Start Task] dialog box* | Initiates task (with initiation code specified) |
| | [Terminate Task] dialog box* | Forcibly terminates tasks |
| | [Release Wait] dialog box* | Forces tasks out of the wait state |
| | [Suspend Task] dialog box* | Moves tasks into SUSPENDED state |
| | [Resume Task] dialog box* | Resumes the execution of tasks in the SUSPENDED state |
| | [Wakeup Task] dialog box* | Wakes tasks up |
| | [Cancel Wakeup Task] dialog box* | Cancels wakeup requests |
| | [Change Task Priority] dialog box* | Changes task priority levels |
| | [Set Task EventFlag] dialog box* | Sets task-dependent event flags |
| | [Clear Task EventFlag] dialog box* | Clears task-dependent event flags |
| | [Rotate Ready Queue] dialog box* | Rotates the ready queue |
| | [Raise Task Exception Routine] dialog box* | Requests task exception processing |
| | [Start Overrun Handler] dialog box* | Starts operation of the overrun handler |
| | [Stop Overrun Handler] dialog box* | Stops the overrun handler |

Note: Cannot be used when the RTOS in use was configured without enabling the manipulation of objects.

**Table 3.2 Windows and Dialog Boxes (cont)**

| Classification | Window Name | Functions |
|---|---|---|
| Semaphores | [Semaphore] window | • Displays states of all semaphores<br>• Displays the wait task queue for each semaphore |
| | [Increment Semaphore Count] dialog box* | Increments semaphore count |
| | [Decrement Semaphore Count] dialog box* | Decrements semaphore count |
| Event flags | [EventFlag] window | • Displays the states for all event flags<br>• Displays the wait task queue for each event flag |
| | [Set EventFlag] dialog box* | Sets an event flag |
| | [Clear EventFlag] dialog box* | Clears an event flag |
| Data queue | [DataQueue] window | • Displays the states of all data queues<br>• Displays the receive data acquired in all data queues |
| | [Send Data to DataQueue] dialog box* | Sends data to data queue |
| | [DataQueue Wait Task Queue] window | Displays the wait task queue for each data queue |
| Mailboxes | [Mailbox] window | • Displays the states of all mailboxes<br>• Displays the wait task queue for each mailbox<br>• Displays the received message queues for all mailboxes |
| | [Send Message to Mailbox] dialog box* | Sends messages to mailboxes |
| | [Receive Message from Mailbox] dialog box* | Receive messages from mailboxes |
| Mutex | [Mutex] window | • Displays the states of all mutexes<br>• Displays the wait task queue for each mutex |

Note: Cannot be used when the RTOS in use was configured without enabling the manipulation of objects.

**Table 3.2 Windows and Dialog Boxes (cont)**

| Classification | Window Name | Functions |
|---|---|---|
| Message buffer | [MessageBuffer] window | • Displays the states of all message buffers<br>• Displays the wait task queue for each message buffer<br>• Displays the received message queue for each message buffer |
| | [Send Message to MessageBuffer] dialog box* | Sends messages to message buffers |
| Variable-size memory pool | [Variable-size MemoryPool] window | • Displays the states of all variable-size memory pools<br>• Displays the wait task queue for each variable-size memory pool |
| | [Get Variable-size MemoryPool] dialog box* | Acquires variable-size memory blocks |
| | [Release Variable-size MemoryPool] dialog box* | Releases variable-size memory blocks |
| Fixed-size memory pool | [Fixed-size MemoryPool] window | • Displays the states of all fixed-size memory pools<br>• Displays the wait task queue for each fixed-size memory pool |
| | [Get Fixed-size MemoryPool] dialog box* | Acquires fixed-size memory blocks |
| | [Release Fixed-size MemoryPool] dialog box* | Releases fixed-size memory blocks |
| Timer | [Timer] window | • Displays the system clock setting<br>• Displays the wait queue for the timer |
| | [Set Time] dialog box* | Modifies the system clock setting |
| Interrupt handler, exception processing routine | [Interrupt Handler] window | Displays the states of all interrupt handlers and exception processing routines |
| Trap exception processing routine | [Trap Routine] window | Displays the states of all trap exception processing routines |

Note: Cannot be used when the RTOS in use was configured without enabling the manipulation of objects.

**Table 3.2 Windows and Dialog Boxes (cont)**

| Classification | Window Name | Functions |
|---|---|---|
| Cyclic handler | [Cyclic Handler] window | Displays the states of all cyclic handlers |
| | [Start Cyclic Handler] dialog box* | Starts up cyclic handlers |
| | [Stop Cyclic Handler] dialog box* | Stops cyclic handlers |
| Alarm handler | [Alarm Handler] window | Displays the state of all alarm handlers |
| | [Start Alarm Handler] dialog box* | Starts up alarm handlers |
| | [Stop Alarm Handler] dialog box* | Stops alarm handlers |
| Extended SVC routine | [Extended SVC Routine] window | Displays the states of all extended SVC routines |
| Result | [Action Result] window* | Displays the results of object manipulation |
| Trace | [RTOS Trace Diagram] window | • Displays the object item list<br>• Displays trace information (diagrammatic form) |
| | [RTOS Trace Text] window | • Displays the object item list<br>• Displays trace information (textual form) |
| | [RTOS Trace Options] dialog box | Makes various settings for trace |
| | [Select Diagram Object] dialog box | Defines the target object for trace display in the diagrammatic form |
| | [Select Text Object] dialog box | Defines trace information for the kernel resource |
| | [Load RTOS Trace Information File] dialog box | Inputs the trace information file |
| | [RTOS Trace Statistics] dialog box | Displays overall view of program execution |
| | [RTOS Trace Find] window | Searches for event information |
| | [Save As] dialog box | Outputs trace information to a file |

Note: Cannot be used when the RTOS in use was configured without enabling the manipulation of objects.

# Section 4 Precautions and Restrictions

## 4.1 Realtime Operation of the User System and Cyclic Handler

The DX functions are realized by referencing or updating data in the memory of the user system. If the following functions are used during the execution of the user program, memory access will occur, and operation will not be in realtime.

- Opening or updating a window
- Clicking the [OK] button of a dialog box to manipulate an object

If the RTOS in use was configured with the manipulation of objects enabled, the throughput of the user system falls a little because a cyclic handler for the DX with a certain cycle is automatically installed in the system.

## 4.2 Consistency of Displayed Contents

Reference to an object's state is by directly reading from data in the memory of the user system. Therefore, if such data is read during the execution of an RTOS kernel routine, the information displayed may not be correct. Also, if an object's state is referred to before the initialization of the kernel has been completed, the information displayed may not be correct.

# Appendix A Troubleshooting

## A.1 Displaying [Action Result] Window Results

If an object manipulation request is made through a dialog box and the user program is executed but the display is not updated, this could be because the cyclic handler for the DX is not operating correctly.

The following is a list of causes and the measures to be taken, in such cases.

1. No timer interrupt occurs.
    A. Timer interrupts do not occur.
    Correct the timer driver so that timer interrupts will occur.
    B. Timer interrupts do occur but are not accepted.
    In the following cases, timer interrupts are not accepted.
        a. Too many interrupts occur that have an interrupt level higher than the timer interrupt
        b. The timer interrupt level has been masked for a long time.
        For interrupts that have a level equal to or higher than the timer interrupt level, keep the masking time as short as possible.
2. The system goes down.
    When the system goes down, timer interrupts may not be accepted.
    Take care so that the system will not go down.

**Debugging Extension User's Manual**

| | |
|---|---|
| Publication Date: | Rev.1.00, August 5, 2003 |
| Published by: | Sales Strategic Planning Div. |
| | Renesas Technology Corp. |
| Edited by: | Technical Documentation & Information Department |
| | Renesas Kodaira Semiconductor Co., Ltd. |

©2003 Renesas Technology Corp. All rights reserved. Printed in Japan.

Debugging Extension
User’s Manual

Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan

REJ10B0029-0100H