

F²MC-16L/16LX/16/16H/16F
μITRON2.01 SPECIFICATIONS COMPLIANT
SOFTUNE REALOS/907
ANALYZER MANUAL

F²MC-16L/16LX/16/16H/16F
μITRON2.01 SPECIFICATIONS COMPLIANT
SOFTUNE REALOS/907
ANALYZER MANUAL

PREFACE

■ Objectives

The Softune REALOS/907 analyzer (referred to as REALOS Analyzer in this manual) is an analysis tool. It is used to debug a system that uses Softune REALOS/907, which conforms to the ITRON2.01 specifications, on the Fujitsu F²MC-16 Family of 16-bit microprocessors. It runs under Softune Workbench, which is part of the Fujitsu SoftuneV3 integrated development environment.

This manual provides information needed to set up and use the REALOS Analyzer.

This manual assumes that the following operating environment is used.

○ [Host computer]

- IBM PC/AT compatible machine (Fujitsu FMV Series) running Windows 95/98/NT4.0
 - Operating environment: 150 MHz or higher Pentium processor (200 MHz or higher recommended), 48M bytes of memory for Windows 95/98/NT4.0 (64M bytes recommended), hard disk space of 40M bytes or more

○ [Real-time OS]

- Softune REALOS/907

○ [Required development tools]

- Softune Workbench for F²MC-16 Family

■ Trademarks

TRON is an abbreviation of The Realtime Operating system Nucleus.

ITRON is an abbreviation of Industrial TRON.

ITRON is an abbreviation of Micro Industrial TRON.

Softune is a trademark of Fujitsu Limited.

REALOS (REALtime Operating System) is a registered trademark of Fujitsu Limited.

F²MC is an abbreviation of Fujitsu Flexible Microcontroller and is a registered trademark of Fujitsu Limited.

Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation in the United States and other countries.

Other system names and product names in this manual are the trademarks of their respective companies or organizations. The symbols TM and ® are sometimes omitted in the text.

■ Intended Reader

This manual is intended for engineers developing various types of products using Softune Workbench and Softune REALOS/907 and explains how they can use the REALOS Analyzer to debug applications. Be sure to read this manual completely.

■ Organization of This Manual

This manual consists of four chapters and an appendix.

CHAPTER 1 OVERVIEW

This chapter provides precautionary information relating to application analysis using the Softune REALOS/907 analyzer and outlines the Softune REALOS/907 analyzer.

CHAPTER 2 TASK ANALYSIS MODULE

This chapter outlines the task analysis module of the Softune REALOS/907 analyzer and describes the installation procedure in an application. Be sure to read this chapter when using the task trace function.

CHAPTER 3 BASIC OPERATION

This chapter describes basic operations for analyzing applications with the Softune REALOS/907 analyzer.

CHAPTER 4 WINDOW

This chapter describes the information displayed in each Softune REALOS/907 analyzer window.

APPENDIX

The appendix describes the limitations on and configuration of the Softune REALOS/907 analyzer. It also lists error messages and provides sample programs.

1. The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.
2. The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended to be incorporated in devices for actual use. Also, FUJITSU is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.
3. The contents of this document may not be reproduced or copied without the permission of FUJITSU LIMITED.
4. FUJITSU semiconductor devices are intended for use in standard applications (computers, office automation and other office equipments, industrial, communications, and measurement equipments, personal or household devices, etc.).

CAUTION:

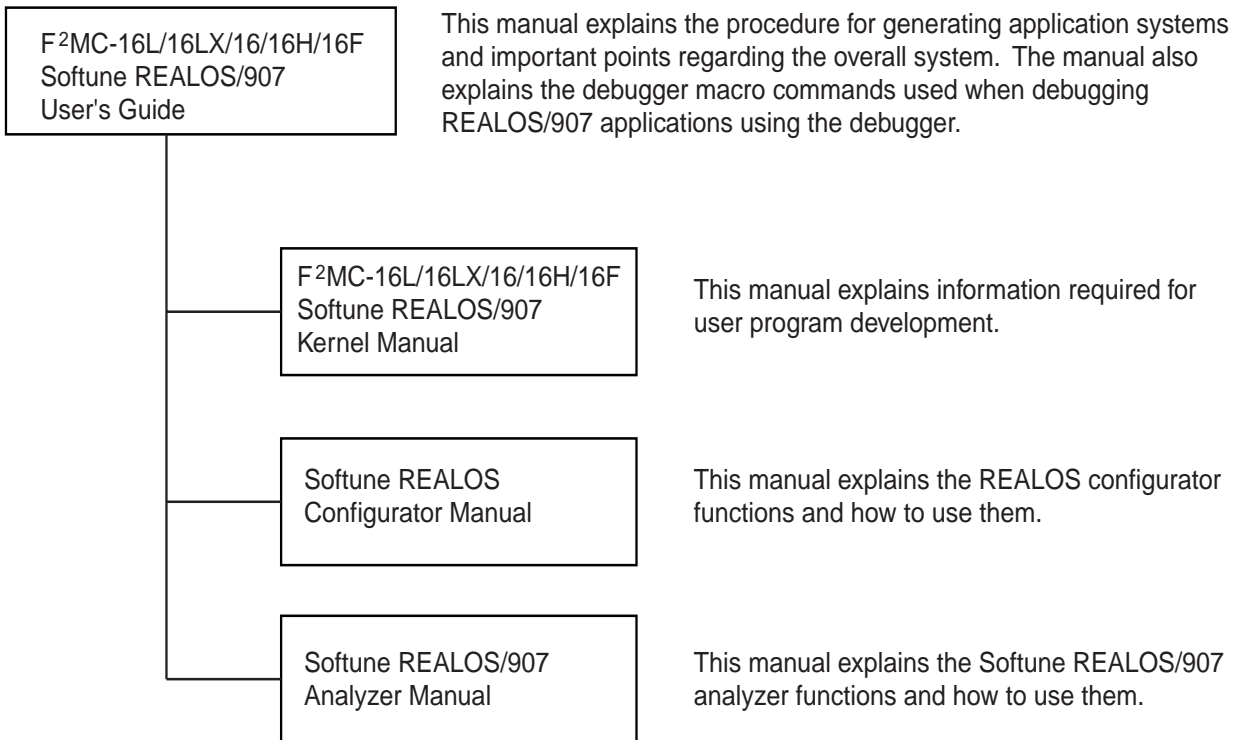
Customers considering the use of our products in special applications where failure or abnormal operation may directly affect human lives or cause physical injury or property damage, or where extremely high levels of reliability are demanded (such as aerospace systems, atomic energy controls, sea floor repeaters, vehicle operating controls, medical devices for life support, etc.) are requested to consult with FUJITSU sales representatives before such use. The company will not be responsible for damages arising from such use without prior approval.

5. Any semiconductor devices have inherently a certain rate of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
6. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Control Law of Japan, the prior authorization by Japanese government should be required for export of those products from Japan.

Manual Set

■ Softune REALOS/907 Manual Set

The Softune REALOS/907 manual set consists of the following four volumes. A first-time user of Softune REALOS/907 should read the User's Guide before reading the other manuals.



READING THIS MANUAL

■ Page Layout

The contents of each section are summarized underneath the title. Reading these summaries will give you a quick overview of the REALOS Analyzer.

Also, higher-level section headings are given in lower-level sections so that you always know to which section the text you are currently reading belongs.

■ Product Names

Product names in this manual are abbreviated as follows:

Microsoft® Windows® 95 operating system: Windows 95

Microsoft® Windows® 98 operating system: Windows 98

Microsoft® Windows NT® Workstation operating system Version 4.0: Windows NT 4.0

CONTENTS

CHAPTER 1 OVERVIEW	1
1.1 Precautionary Information about Use	2
1.2 REALOS Analyzer Features	4
1.3 Function Overview	5
1.3.1 REALOS Project Window	6
1.3.2 Softune Workbench Operator Command	7
1.3.3 Object Display	8
1.3.4 Stack Utilization Analysis	10
1.3.5 Task Trace	12
1.3.6 Monitoring	14
CHAPTER 2 TASK ANALYSIS MODULE	15
2.1 Task Analysis Module Overview	16
2.2 Customizing the Task Analysis Module	18
2.3 Task Analysis Module Installation	20
2.4 Initializing the Task Analysis Module	21
CHAPTER 3 BASIC OPERATION	23
3.1 Starting REALOS Analyzer	24
3.2 Terminating REALOS Analyzer	25
3.3 Collecting REALOS Data	26
3.4 Menus	27
3.5 Toolbars	30
3.6 Status Bar	32
3.7 REALOS Project Window	34
3.8 Commands	35
3.9 Object Display	36
3.10 Stack	38
3.11 Task Trace	39
3.12 File Handling	43
3.13 Monitoring	44
CHAPTER 4 WINDOWS	45
4.1 REALOS Project Window	46
4.1.1 Information Displayed in the REALOS Project Window	47
4.1.2 REALOS Project Window Popup Menus	50
4.2 Object Window	52
4.2.1 Information Displayed in the Object Window	53
4.2.2 Object Window Popup Menus	65
4.3 Stack Utilization List	67
4.3.1 Stack Initialization	68
4.3.2 Information Displayed in the Stack Utilization List	69
4.3.3 Stack Utilization List View Setting	71
4.3.4 Stack Utilization Check	72

4.3.5	Stack Utilization Graph	73
4.4	Task Trace Figure Window	74
4.4.1	Information Displayed in the Task Trace Figure Window	75
4.4.2	Task Trace Figure Window Toolbar	80
4.4.3	Task Trace Figure Window Status Bar	84
4.4.4	Task Trace Figure Window Popup Menus	85
4.4.5	Printing a Task Trace Figure	86
4.4.6	Running Time Graph	87
4.4.7	Task Trace Figure Information Dialog Box	90
4.5	Task Trace Tree Window	92
4.5.1	Information Displayed in a Task Trace Tree Window	93
4.5.2	Task Trace Tree Window Toolbar	95
4.5.3	Displaying a Graph in the Task Trace Tree Window	96
4.6	Object Trace Window	98
4.6.1	Information Displayed in the Object Trace Window	99
4.7	Monitoring	101
4.7.1	Information Displayed in the Task Status Monitor Window	102
4.7.2	Information Displayed in the Stack Monitor Window	105
4.8	Setup	107
4.8.1	Mode	108
4.8.2	Select Task, Object	114
4.8.3	Task Trace	116
4.9	Help	117
4.9.1	Help Topics	118
4.9.2	About fra907se	119

APPENDIX	121
APPENDIX A Restrictions	122
APPENDIX B Configuration	124
APPENDIX C Error Messages	125
APPENDIX D Sample Programs	127

INDEX.....	131
-------------------	------------

FIGURES

Figure 1.3-1	Example of the REALOS Project Window	6
Figure 1.3-2	Example of Object Display Windows	9
Figure 1.3-3	Example of Displaying Detail Stack Information	10
Figure 1.3-4	Example of Task Trace Results	13
Figure 1.3-5	Example of Task Status Monitor and Stack Monitor Display	14
Figure 2.1-1	Outline of the Task Analysis Module Operation	17
Figure 2.4-1	Definition of the Initialize Handler (Softune Workbench Screen)	21
Figure 2.4-2	Example of Addition of the Initialization Routine of the Task Analysis Module (Editor Screen)	22
Figure 3.1-1	Example of Starting REALOS Analyzer	24
Figure 3.3-1	Example of Collecting Data Dialog Box	26
Figure 3.5-1	Example of Main Toolbar	30
Figure 3.5-2	Example of Window Toolbar	30
Figure 3.5-3	Example of Setup Toolbar	31
Figure 3.6-1	Example of Status Bar	33
Figure 3.9-1	Example of Sorting by Priority	36
Figure 3.9-2	Example of Displaying the Free Block History	37
Figure 3.9-3	Example of Displaying the Jump Function (Task Waiting for Message[***] Task List Window)	37
Figure 3.11-1	Names of Split Windows in a Task Trace Figure	40
Figure 3.11-2	Example of Scrolling the Detail Window by Clicking the Highlighted Area	41
Figure 4.1-1	REALOS Project Window	47
Figure 4.1-2	Concept of Task Dispatch Break	51
Figure 4.2-1	Object Windows	53
Figure 4.2-2	Queue List Window (From Left to Right: Ready Queues, Timer Queues, Alarm Queues)	58
Figure 4.2-3	Toolbar in the Queue List Window	58
Figure 4.3-1	Stack Utilization List	69
Figure 4.3-2	Stack Utilization List View Setting	71
Figure 4.3-3	Results of Checking for Free Stack Space (Check with 0x50 Bytes Specified)	72
Figure 4.3-4	Graph Showing Maximum Available Stack Space	73
Figure 4.4-1	Example of Time Display in the Time Display Window	75
Figure 4.4-2	Task Trace Figure Window Toolbar	80
Figure 4.4-3	Example of Sequentially Sorting Dispatches	81
Figure 4.4-4	Example of Task Trace Figure Window Status Bar	84
Figure 4.4-5	Displaying the Running Time Graph	87
Figure 4.4-6	Trace Figure Information Dialog Box	90
Figure 4.4-7	Displaying the Related Event List	91

Figure 4.5-1	Example of the Task Trace Tree Window	93
Figure 4.5-2	Task Trace Tree Window Toolbar	95
Figure 4.6-1	Example of an Object Trace Window	99
Figure 4.7-1	Task Status Monitor Window	102
Figure 4.7-2	Stack Monitor Window	106
Figure 4.8-1	Select Task, Object Dialog Box	114
Figure 4.8-2	Monitoring and Object Trace Setup	115
Figure 4.8-3	Overview of Trace Data Buffering	116
Figure 4.9-1	About fra907se Dialog Box	119
Figure A-1	Contents of Trace Buffer	123
Figure B-1	Configuration of Softune REALOS Analyzer Files	124

TABLES

Table 4.1-1	Items Displayed in the REALOS Project Window	47
Table 4.1-2	REALOS Project Window Popup Menus	50
Table 4.2-1	Task List Window	54
Table 4.2-2	Task Statuses	54
Table 4.2-3	Semaphore List Window	55
Table 4.2-4	Event Flag List Window	55
Table 4.2-5	Mailbox List Window	55
Table 4.2-6	MemoryPool List Window	56
Table 4.2-7	Cyclic Handler List Window	56
Table 4.2-8	Alarm Handler List Window	57
Table 4.2-9	Display Items in Tree Form	57
Table 4.2-10	Ready Queue Display Items	58
Table 4.2-11	Timer Queue Display Items	59
Table 4.2-12	Alarm Queue Display Items	59
Table 4.2-13	Icons in the Task List Window	59
Table 4.2-14	Icons in the Semaphore List Window	61
Table 4.2-15	Icons in the Event Flag List Window	61
Table 4.2-16	Icons in the Mailbox List Window	62
Table 4.2-17	Icons in the MemoryPool List Window	62
Table 4.2-18	Icons in the Cyclic Handler List Window	63
Table 4.2-19	Icons in the Alarm Handler List Window	64
Table 4.2-20	Popup Menus of an Object Window	65
Table 4.3-1	Items Displayed in the Stack Utilization List	69
Table 4.4-1	Icons Indicating Event Other Than Issuing of a System Call	75
Table 4.4-2	Icons Indicating Event Other Than Issuing of a System Call	78
Table 4.4-3	Line Types for Indicating Task Status	78
Table 4.4-4	Display Items for Each Type of Traced Event	82
Table 4.4-5	Information List Items	88
Table 4.7-1	Icons	102
Table 4.7-2	Task Statuses and Stack Pointer Value	105
Table 4.8-1	Task List Window	109
Table 4.8-2	Semaphore List Window	110
Table 4.8-3	Event Flag List Window	110
Table 4.8-4	Mailbox List Window	111
Table 4.8-5	MemoryPool List Window	111

Table 4.8-6	Cyclic Handler List window	112
Table 4.8-7	Alarm Handler List Window	112
Table 4.8-8	Queue Window	113
Table A-1	Processing Time and Size of Task Analysis Module	122

CHAPTER 1 OVERVIEW

This chapter provides precautionary information about use of the Softune REALOS/907 Analyzer (REALOS Analyzer) and outlines its functions.

- 1.1 Precautionary Information about Use
- 1.2 REALOS Analyzer Features
- 1.3 Function Overview

1.1 Precautionary Information about Use

This section provides precautionary information about use of the Softune REALOS/907 Analyzer.

■ Notes on Development Environments

○ **Softune REALOS/907 is required.**

The Softune REALOS/907 Analyzer is a tool that analyzes application programs using Softune REALOS (referred to as application programs hereafter).

○ **Fujitsu integrated development environment Softune Workbench is required.**

The REALOS Analyzer cannot be activated independently and must be activated from Softune Workbench.

○ **The REALOS Analyzer operates linked to a debugger.**

The REALOS Analyzer operates linked to a Softune Workbench debugger. The analyzer cannot be used by another debugger such as Third party.

If processing is stopped during data collection, check whether an error has occurred in the debugger.

■ Notes on Use of the Emulator Debugger

○ **When external memory is used**

The REALOS Analyzer accesses the REALOS/907 data area (including the task stack area) and the data area of the task analysis module immediately after a target file is loaded or a reset is executed. If the REALOS Analyzer is unable to access the area, it will not operate correctly.

Therefore, if the above data areas are allocated in external memory, the external memory must be made accessible (with any method) before the target file is loaded. As an example, a command procedure file describing processing that enables access to external memory can be executed.

○ **When a monitoring function is used**

The monitoring function is required only under the conditions listed below. If it is used otherwise, an error occurs on the emulator debugger side. This is because monitoring is implemented by a function that reads memory while the emulator debugger is operating.

- Condition for using the monitoring function

- When the REALOS/907 data area (including the task stack area) is allocated in external memory

-> The area must be allocated in emulation memory.

- When the REALOS/907 data area (including the task stack area) is allocated in internal memory

-> The internal memory area must be mirrored.

■ Note on Use of the Monitor Debugger

○ Monitoring function

The monitoring function cannot be used because the monitor debugger cannot read memory while debugging is being executed.

■ Notes on Module Configuration

○ Task trace function requires the task analysis module.

For the task analysis module, see Chapter 2 "Task Analysis Module".

○ Do not install the task analysis module in a product.

Be sure to replace the task analysis module with the OS object file (dbgfk.obj) provided by Softune REALOS before delivering the analyzer as a product.

○ The execution time is delayed.

The execution time is delayed if the task analysis module is installed.

■ Terms

This manual uses the following definitions:

○ Task trace

Trace of events such as task dispatch and system call issuance

○ Trace buffer

Buffer that stores task trace information

○ Task analysis module

Program for implementing a task trace

○ Monitoring

Function that obtains information at predetermined intervals

○ Idle loop

OS routine that is executed if no tasks are being executed

○ Idle task

Endless-loop task that does not enter the wait state. Provided by the user.

1.2 REALOS Analyzer Features

The REALOS Analyzer has the following features.

■ REALOS Analyzer Features

○ Task trace figure

The task trace figure of the REALOS Analyzer enables tasks to be checked, with time stamps, to see if they are executed at the expected times in the expected sequence.

○ Object list display

The REALOS Analyzer displays a list of objects, which can be used to check the status of each object. The history of REALOS management information changes can also be checked.

○ Display of graphs

While the task trace function of the REALOS Analyzer is being executed, each object and task can be checked to see if each is being used effectively.

○ Used stack analysis

The stack utilization of each task of the REALOS Analyzer can be checked to see if it is within the estimated range.

○ Monitoring function

This function enables the status of each task and the status of stack utilization to be checked during execution.

○ Operation by three types of debuggers

The REALOS Analyzer can use a simulator, emulator, and monitor debugger to perform analysis provided the tool is a Softune Workbench debugger.

1.3 Function Overview

This section describes the functions of the Softune REALOS/907 analyzer.

■ Notes on Function Overview

This section explains the overview of the functions of the Softune REALOS/907 analyzer.

For the basic operation of each function, see Chapter 3. For the displayed information, see Chapter 4.

1.3.1 REALOS Project Window

1.3.2 Softune Workbench Operator Command

1.3.3 Object Display

1.3.4 Stack Utilization Analysis

1.3.5 Task Trace

1.3.6 Monitoring

1.3.1 REALOS Project Window

The REALOS project window displays REALOS management information hierarchically.

■ REALOS Project Window

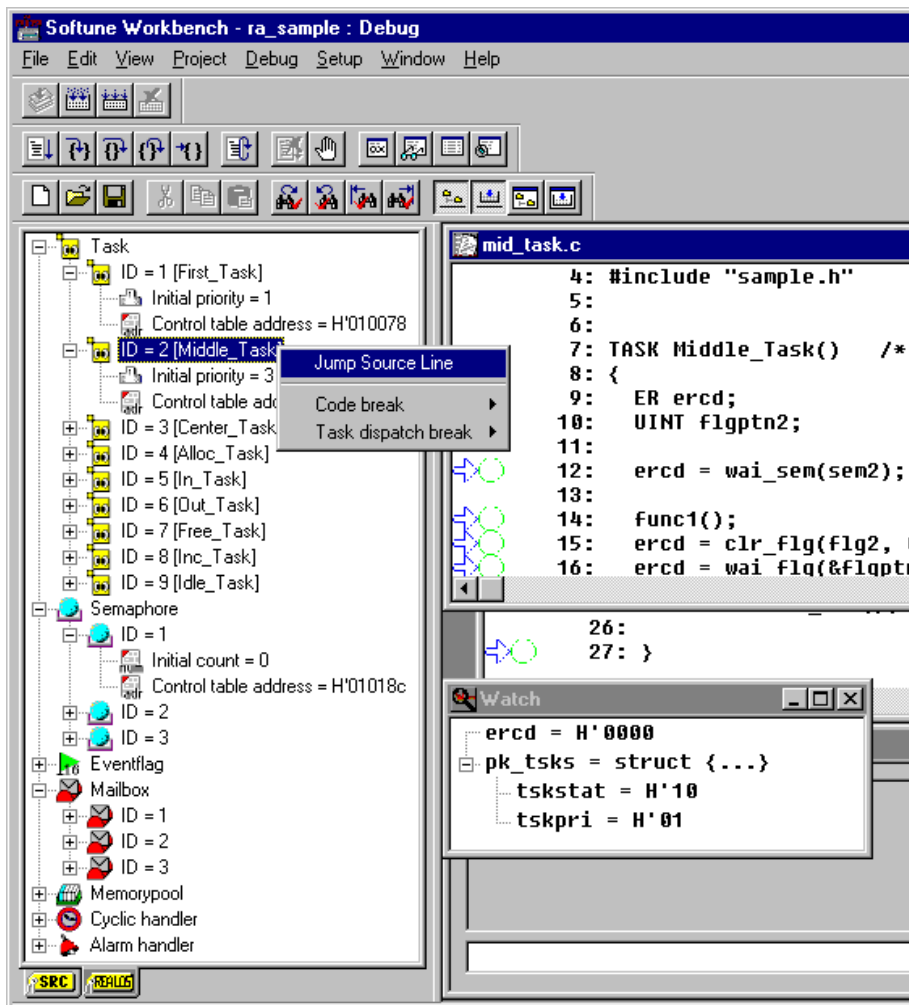
The REALOS project window is added to the Softune Workbench project window when Softune Workbench starts a debugger.

This window displays hierarchically the initial information of each object set by the configurator.

From this window, source line jumps and breaks can be set, and the data size of each object can be checked.

Figure 1.3-1 shows an example of the REALOS project window.

Figure 1.3-1 Example of the REALOS Project Window



1.3.2 Softune Workbench Operator Command

Softune Workbench debugging can be executed, aborted, and reset from the REALOS Analyzer.

■ Softune Workbench Operator Command

Softune Workbench debugging can be executed, aborted, and reset from the REALOS Analyzer.

Execute the processing from the [Command] menu or toolbar.

1.3.3 Object Display

When a debugger stops, the current status of an application-defined REALOS object is displayed. A window is provided for each object type.

■ Object Display

When a debugger stops, the current status of an application-defined REALOS object is displayed. The following object display windows are provided:

- Task List window
- Semaphore List window
- Event Flag List window
- Mailbox List window
- MemoryPool List window
- Cyclic Handler List window
- Alarm Handler List window
- Queue List window (ready queue, timer queue, alarm handler queue)

Each object display window displays list-format information and queued tasks hierarchically.

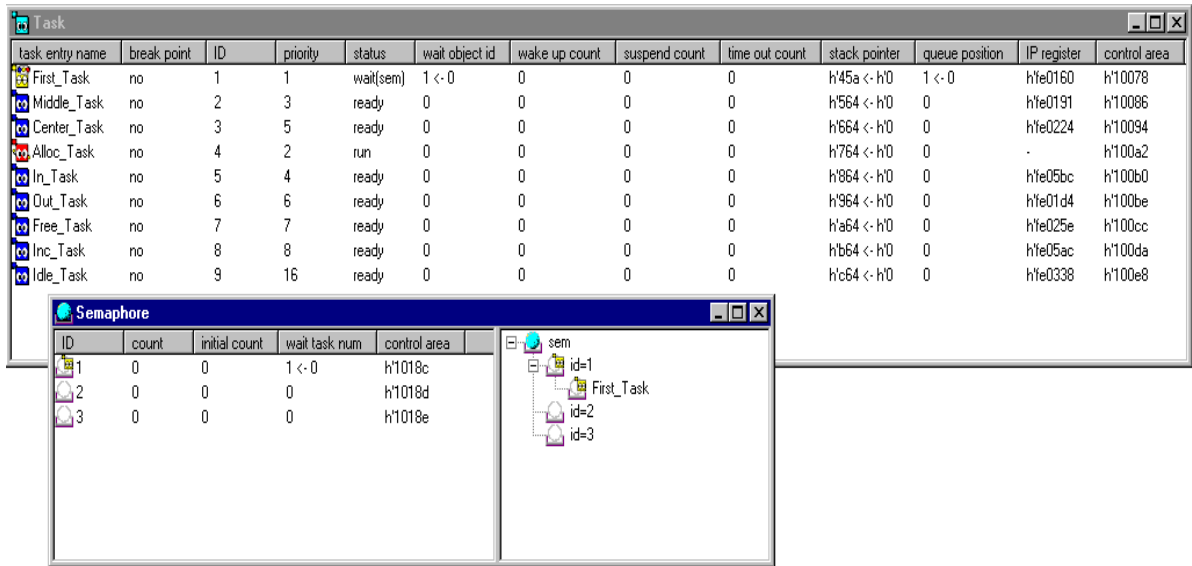
■ Standard Functions of Object Display Windows

Each object display window has the following standard functions:

- Sorting of information
- Display of history of changes for each information item
- Jump to a related window or item

Figure 1.3-2 shows an example of object display windows.

Figure 1.3-2 Example of Object Display Windows



Note:

If the PC is in the kernel, the information displayed by the REALOS Analyzer is not assured because of ongoing kernel processing.

1.3.4 Stack Utilization Analysis

This function analyzes the maximum amount of stack space used on the stack allocated to a task. When an application system is not initialized yet, the stack area is filled with a specific pattern. This function displays maximum use by analyzing the change of the pattern when the application stops.

■ Stack Utilization Analysis

When [Initialize] on the [Stack] menu is clicked, this function fills the stack area with a specific pattern before an application system is initialized. (If "Display this dialog Next Time" is selected, this function automatically fills the area with the pattern after each reset.)

This function analyzes the pattern and displays the result when [Used Stack] on the [Stack] menu is clicked.

■ Detail Stack Information Display

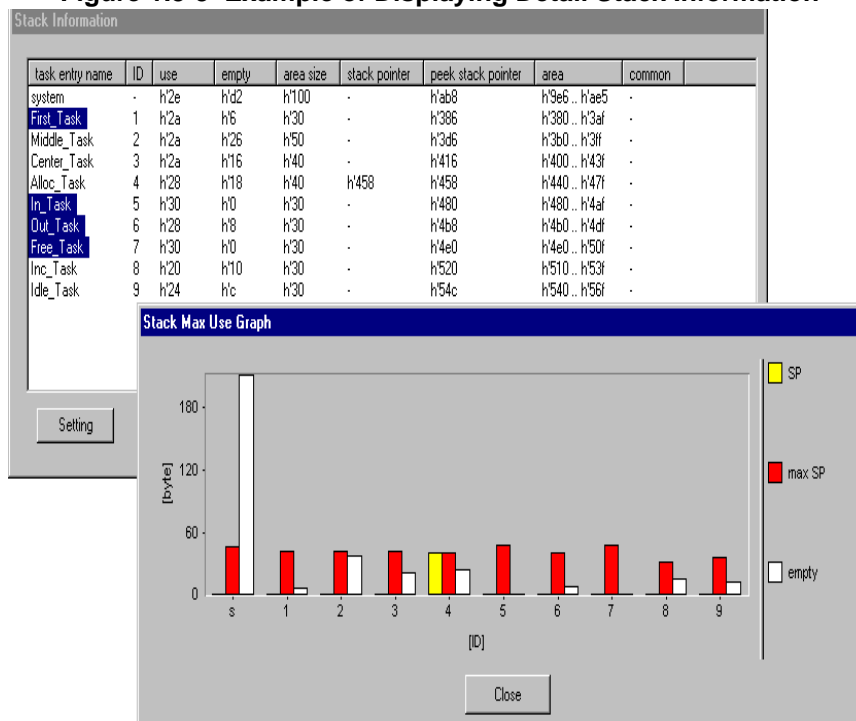
When [Used Stack] is clicked, this function displays the results of stack utilization analysis.

The results, displayed in list format, can be sorted by item. The following functions can be executed for the results:

- Checking the remaining stack area
- Display in graph format

Figure 1.3-3 shows an example of displaying detail stack information.

Figure 1.3-3 Example of Displaying Detail Stack Information



Note:

If the PC is in the kernel, the information displayed by the REALOS Analyzer is not assured because of ongoing kernel processing.

1.3.5 Task Trace

This function uses icons to display the results of tracing task dispatch occurrence and system call issuance in trace figures and trace tree windows.

■ Task Trace

When a Softune Workbench debugger stops, the Task Analysis Module (R_D_dbgA.obj) traces the timing of task dispatching that has occurred and system calls that have been issued since the last time the debugger stopped (task trace).

The REALOS Analyzer collects the results of the task trace, indicates the events in the form of icons (dispatch occurrence and system call issuance), and displays the results in a task trace figure and a hierarchical task trace tree.

■ Display of Analysis Results in Graphs

This function displays analysis information for task traces in graphs. The following results of analysis are displayed in graphs:

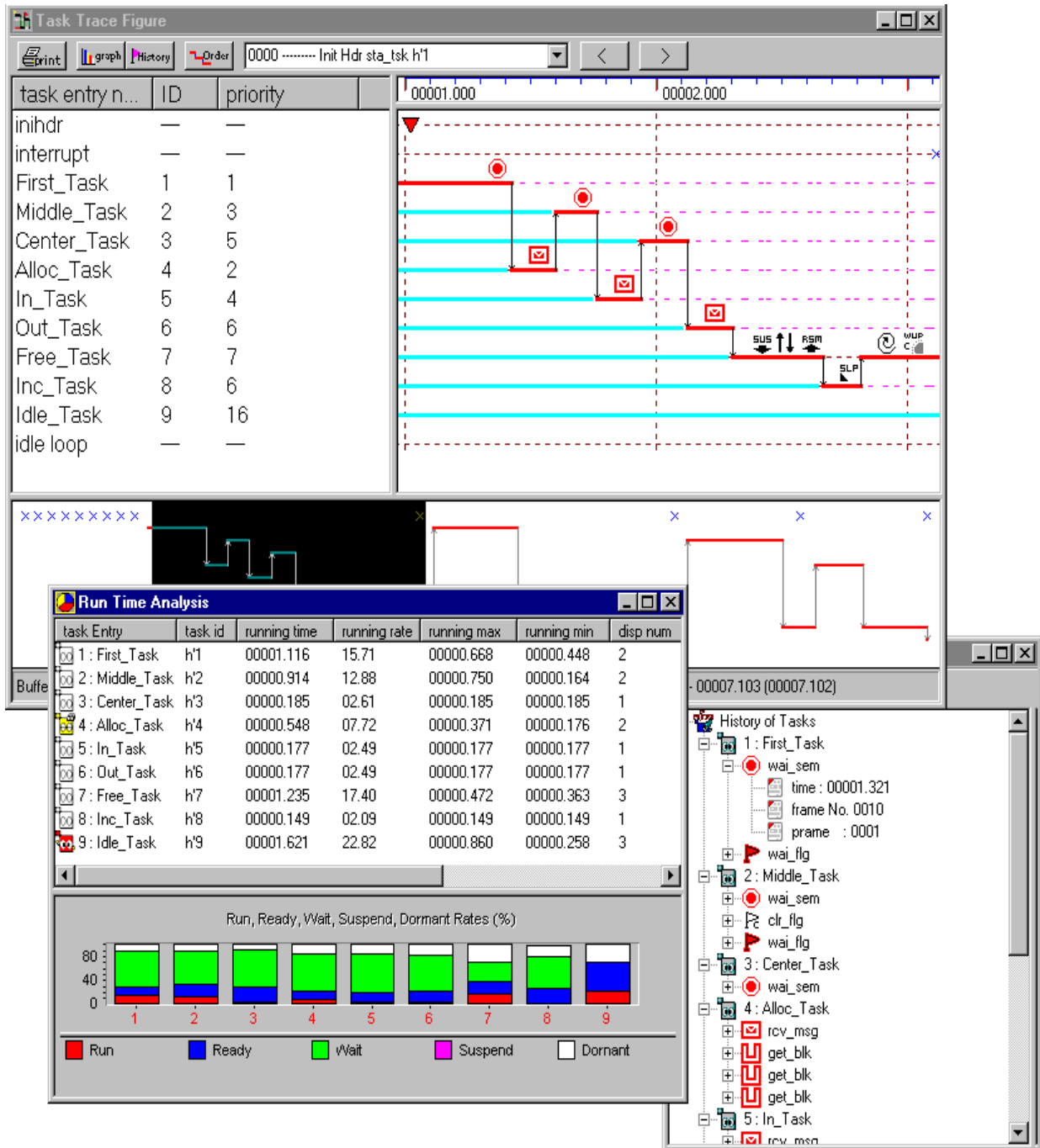
- Ratio of running time/running-enabled time/wait time/dormant time of a task
- Maximum continuous running time of a task
- Minimum continuous running time of a task
- Task dispatch count
- System call issuance count of each task
- Issuance count of each system call type

■ Object Trace

In synchronization with an ordinary task trace, the status transit history of each object can be checked.

Figure 1.3-4 shows an example of the results of a task trace.

Figure 1.3-4 Example of Task Trace Results



Note:

If the PC is in the kernel, the information displayed by the REALOS Analyzer is not assured because of ongoing kernel processing.

Some graphs cannot be displayed depending on the setting of [Mode] and [Task Trace] on the [Setup] menu.

1.3.6 Monitoring

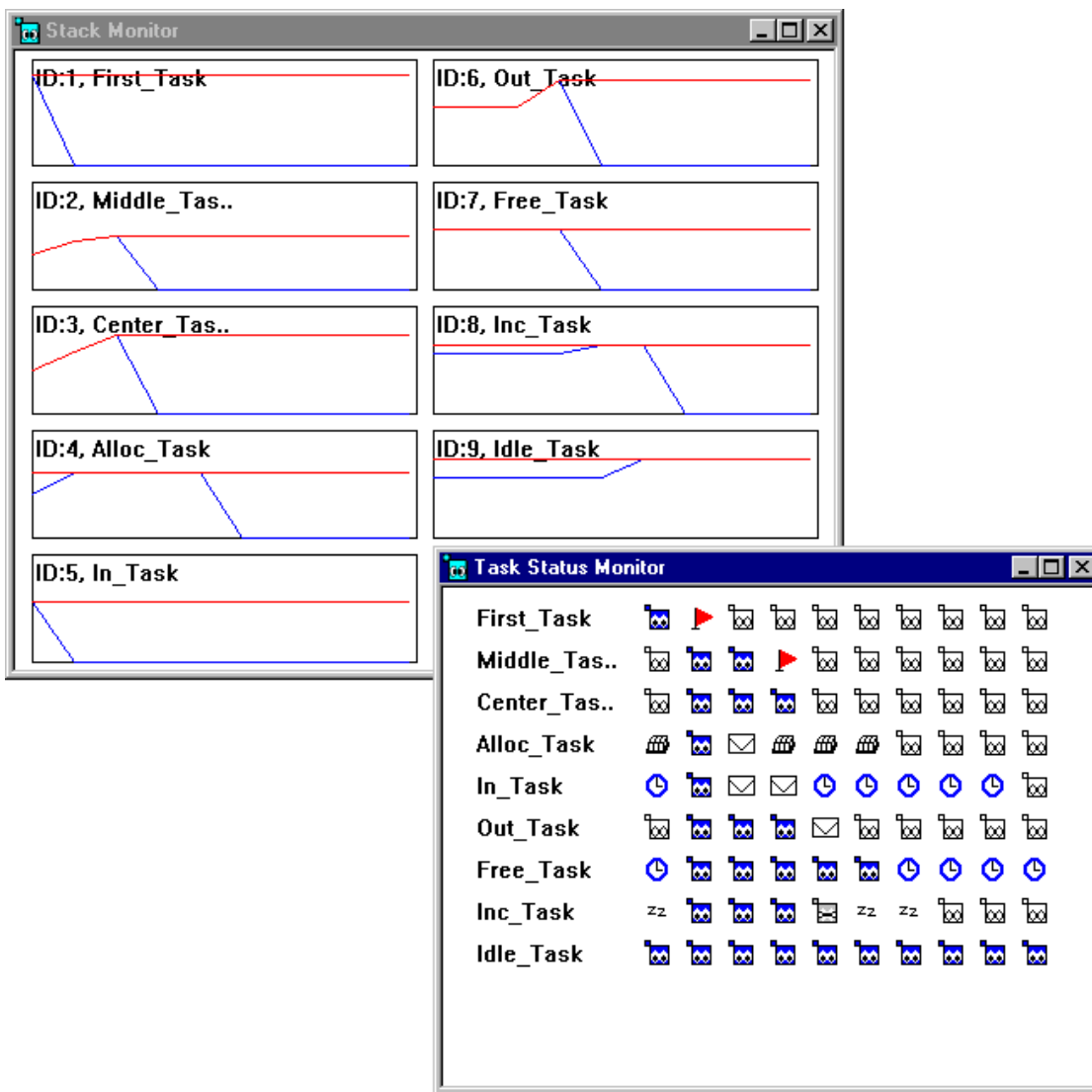
This function monitors task statuses and stack utilization during execution of a Softune Workbench debugger.

■ Monitoring

This function monitors task statuses and stack pointers at predetermined intervals during execution.

Figure 1.3-5 shows an example of task status monitor and stack monitor display.

Figure 1.3-5 Example of Task Status Monitor and Stack Monitor Display



Note:

The monitoring function cannot be used by the monitor debugger.

CHAPTER 2 TASK ANALYSIS MODULE

This chapter outlines the task analysis module and describes the customization, installation, and initialization methods necessary for using the task trace function of the REALOS Analyzer.

Be sure to read this chapter before using the task trace function.

- 2.1 Task Analysis Module Overview
- 2.2 Customizing the Task Analysis Module
- 2.3 Task Analysis Module Installation
- 2.4 Initializing the Task Analysis Module

2.1 Task Analysis Module Overview

Before the task trace function can be implemented with the REALOS Analyzer, the Task Analysis Module dedicated to the REALOS Analyzer must be linked.

■ Task Analysis Module Overview

The REALOS/907 analyzer can implement the task trace function with the Task Analysis Module installed in an application. If the task trace function is not used, this module does not have to be installed.

The task analysis module contains the following files:

- R_D_dbgA.obj
- R_D_trcA.asm

By changing the contents of R_D_trcA.asm, you can customize the size of the trace buffer used for storing trace results and customize timer processing for time measurement.

The object-format file R_D_dbgA.obj enables the REALOS Analyzer to implement the task trace function.

To construct an application program by using Softune REALOS/907, you must link one of the following three files:

OS object file: dbgfk.obj

This file is provided with Softune REALOS/907 for installing products that support only the Softune Workbench object display function during debugging.

Debug module: r_d_dbg.obj

This file is provided with the F2MC-16 Family Softune Workbench for debugging that supports the object display function, system call issuance, and dispatch breaks.

Task analysis module: R_D_dbgA.obj, R_D_trcA.asm

These files are provided with the Softune REALOS/907 analyzer. (They are stored in \\install-folder\sample\907\ra_sample.)

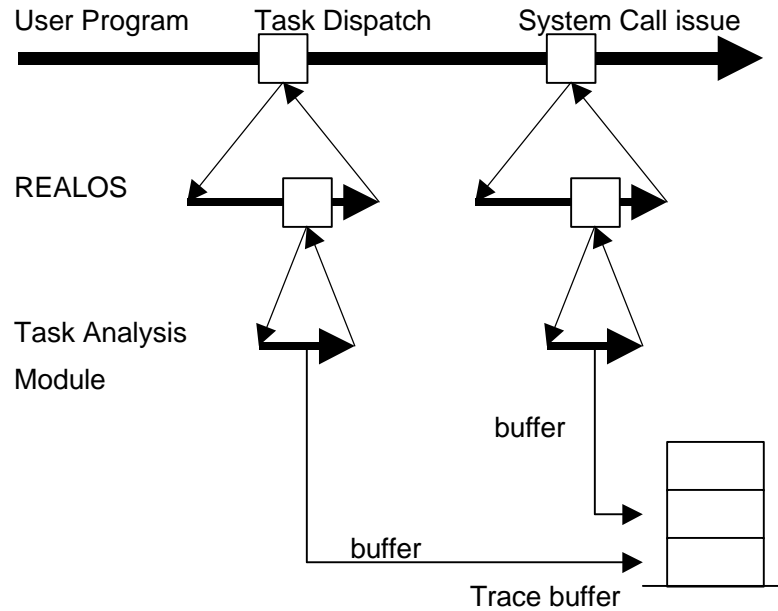
A file in an install folder is automatically linked if the REALOS Analyzer is selected as the debug setting of the configurator. For more information, see the "Configurator Manual."

■ Overview of Task Analysis Module Operation

The task analysis module is a program that adds processing that is applied when a REALOS kernel program dispatches a task or issues a system call. Thus, this module implements task tracing.

When task tracing is implemented with a time stamp, the REALOS system clock is used. Figure 2.1-1 outlines the operation of the task analysis module.

Figure 2.1-1 Outline of the Task Analysis Module Operation



The task analysis module includes a debug module function (`r_d_dbg.obj`) that is provided by the Softune Workbench. Therefore, if the task analysis module is linked to the analyzer, the analyzer implements functions such as the issuing of system calls by Softune Workbench.

■ Task Analysis Module Customization

The following task analysis module items can be customized:

- Resource timer (reload timer) to be used for the system clock
- Buffer size (expandable to handle up to 2048 events)

Note:

The task analysis module is not designed for installation in a product. When delivering the analyzer as a product, be sure to remove this module.

Excluding the task trace function, functions such as the object list function and monitoring function can operate even though the task analysis module is not installed.

2.2 Customizing the Task Analysis Module

This section describes the procedure for customizing the task analysis module.

The timer processing and trace buffer size can be customized.

To do so, change the contents of the following file:

- R_D_trcA.asm
-

■ Customizing the Task Analysis Module

The provided task analysis module is designed for systems with the following specifications:

- Microcomputer: MB90550A Series
- System clock timer: INT25 16-bit reload timer 0 (reload value: 1000)

If this module is used with another system, the module must be customized.

To customize the module, change parts of the R_D_trcA.asm file.

○ Timer

The time measurement timer uses the Softune REALOS/907 system clock.

Be sure to install the system clock.

Use the 16-bit reload timer in the microcontroller for the system clock. The reload timer channel and reload timer value can be customized.

- Changing the reload timer channel
 - To change the reload timer channel used for the system clock, change the following symbol definition values in the R_D_trcA.asm file. The default values are the addresses of the TMR (16-bit timer register) and TMCSR (control status register). These values are effective when MB90550A reload timer channel 0 is used.

```
RELOAD_TIMER_ADDR      .equ  H'5c
RELOAD_TIMER_CTRL     .equ  H'5a
```

For the 16-bit timer register and control status register, see the "MB90XX Hardware Manual."

- Changing the reload value of the reload timer
 - Change the definition value of R_D_RELOAD_TIMER in the R_D_TrcA.asm file to match the reload timer value used for the REALOS/907 system clock. D'1000 is defined as the default value of R_D_RELOAD_TIMER as shown below (2 ms per system clock pulse at 16 MHz).

```
R_D_RELOAD_TIMER      .equ  D'1000
```

○ Trace buffer size

The trace buffer size is defined by TRC_DATA_NUM in the R_D_trcA.asm file. A value from 0

2.2 Customizing the Task Analysis Module

to 2048 can be specified for TRC_DATA_NUM.

The default value for the trace buffer is the size for 50 steps.

```
TRC_DATA_NUM          .equ      50
```

The trace buffer size can also be set with the debug setting of the configurator.

Note:

The task debug module in the \\install-folder\sample\907\ra_sample folder provided with the REALOS Analyzer is designed for the MB90550A Series. To use other chips and timers, you must customize R_D_trcA.asm.

This section explains how to install the task analysis module.

2.3 Task Analysis Module Installation

This section explains how to install the task analysis module.

■ Installation

To build an application with Softune Workbench, select [REALOS Analyzer Use] as the debug setting of the configurator.

The task trace buffer size can be set with the configurator.

Generally, link the provided task analysis module. (File stored in \\install-folder\sample\907\ra_sample)

For more information, see the "Configurator Manual."

■ Installation (When Softune Workbench Is Not Used)

If Softune Workbench is not being used, link the task analysis module (R_D_dbgA.obj, R_D_trcA.asm) with debug information.

■ Allocation Section

The task analysis module is allocated in the R97_CODE and R97_DATA sections.

Be sure to link the module with debug information.

Note:

The trace buffer is allocated in the DBGDAT section.

2.4 Initializing the Task Analysis Module

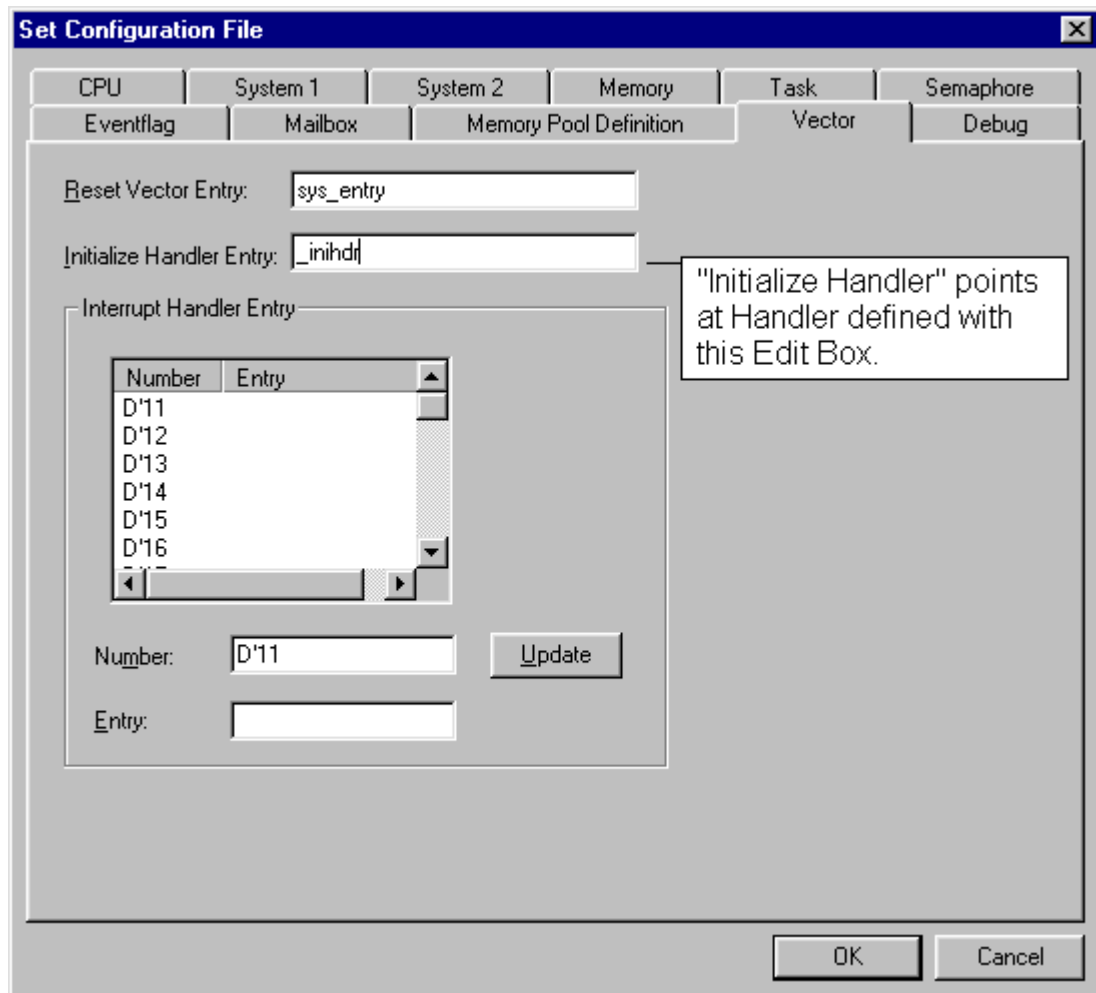
The task analysis module must be initialized before it can be used.

To initialize the module, add a call to the task analysis module initialization routine (r97_d_init) in the initialize handler. Call the initialization routine before issuing the system call.

■ Initialize Handler

The initialize handler is defined in [Configuration File Setting] of Softune Workbench as shown in Figure 2.4-1.

Figure 2.4-1 Definition of the Initialize Handler (Softune Workbench Screen)



■ Adding the Task Analysis Module Initialization Routine

As shown in Figure 2.4-2, add processing to call the initialization routine (r97_d_init) to the initialize handler.

Be sure to call the initialization routine before making a system call.

Example: The initialize handler (_inihdr) is in the Init.asm file.

Figure 2.4-2 Example of Addition of the Initialization Routine of the Task Analysis Module (Editor Screen)

```

216 ;; ^ ^ .IMPORT ^_sw_object ^
217 ;; ^ ^ .IMPORT ^r97_d_init ^
218
219 .EXPORT ^_inihdr ^
220 inihdr: ^ mov ^rp, #0 ^ ;Set register bank No.
221 callp ^r97_d_init ^
222 ;; ^ ^ callp ^_uart_init ^
223 ;; ^ ^ callp ^_sclk_init ^
224 ;; ^ ^ mv ^a, #XSOFID ^
225 ;; ^ ^ scall ^sig_sem ^
226 ;; ^ ^ mv ^a, #FLG_I ^
227 ;; ^ ^ scall ^set_flg ^
228 ;; ^ ^ mv ^a, #FLG_R ^
229 ;; ^ ^ scall ^set_flg ^
    
```

call does the initialization routine "r97_d_init" of "Task Analysys Module" with "InitializeHandler".

222: 1

CHAPTER 3 BASIC OPERATION

This chapter explains basic operation of each REALOS Analyzer function used from Softune Workbench.

- 3.1 Starting REALOS Analyzer
- 3.2 Terminating REALOS Analyzer
- 3.3 Collecting REALOS Data
- 3.4 Menus
- 3.5 Toolbars
- 3.6 Status Bar
- 3.7 REALOS Project Window
- 3.8 Commands
- 3.9 Object Display
- 3.10 Stack
- 3.11 Task Trace
- 3.12 File Handling
- 3.13 Monitoring

3.1 Starting REALOS Analyzer

Start the REALOS Analyzer by selecting [Project] - [REALOS Analyzer] - [Start] from Softune Workbench.

■ Starting the REALOS Analyzer

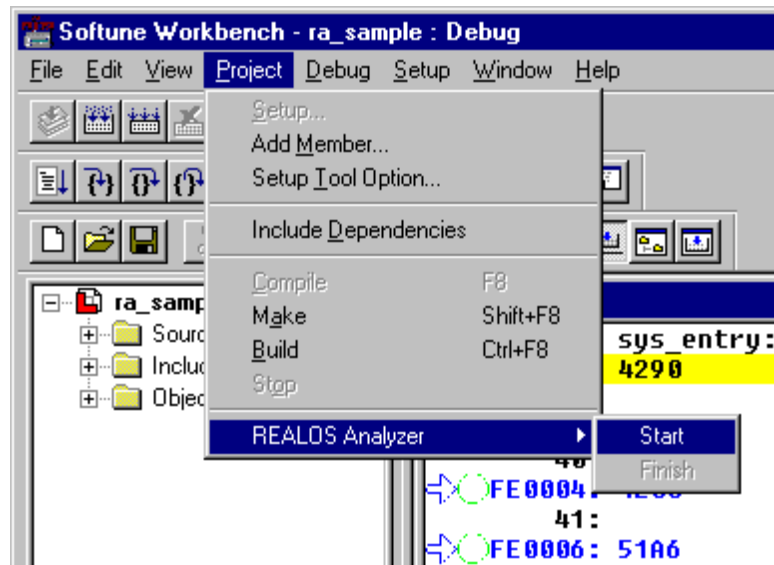
The REALOS Analyzer is started from Softune Workbench. To start the REALOS Analyzer, select [Project] - [REALOS Analyzer] - [Start] from the Softune Workbench menu.

If debugging is not started by Softune Workbench, the REALOS Analyzer cannot be started.

Conversely, after debugging has been started, the REALOS Analyzer can be started anytime.

Figure 3.1-1 shows an example of starting the REALOS Analyzer.

Figure 3.1-1 Example of Starting REALOS Analyzer



Note:

Even though fran907s.exe is executed in \bin of the directory where the REALOS Analyzer is installed, the REALOS Analyzer does not operate normally. The REALOS Analyzer must always be started from Softune Workbench.

3.2 Terminating REALOS Analyzer

Terminate the REALOS Analyzer with either of the following methods:

- Selecting [REALOS Analyzer] - [Exit] from the [Project] menu of Softune Workbench
 - Selecting [File] - [Exit] from REALOS Analyzer
-

■ Terminating the REALOS Analyzer

You can terminate the REALOS Analyzer from either Softune Workbench or REALOS:

- Terminating the REALOS Analyzer from Softune Workbench
 - Select [REALOS Analyzer] - [Exit] from the [Project] menu of Softune Workbench.
- Terminating the REALOS Analyzer from REALOS
 - Select [Exit] from the [File] menu of the REALOS Analyzer.

When debugging terminates, the REALOS Analyzer terminates automatically.

3.3 Collecting REALOS Data

The REALOS Analyzer collects REALOS management information from Softune Workbench. The data collection dialog box is displayed when REALOS data is being collected by the REALOS Analyzer.

■ Collecting REALOS Data

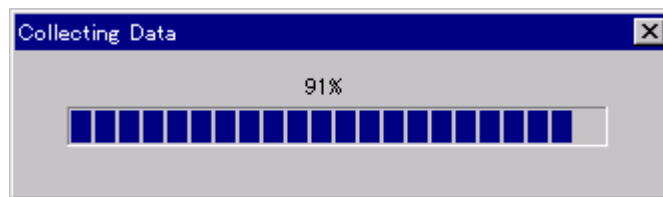
The REALOS Analyzer collects REALOS management information from Softune Workbench while Softune Workbench is in the debugging state.

■ Collecting Data Dialog Box

The Collecting Data dialog box is displayed in the REALOS Analyzer during communication. The Collecting Data dialog box displays a progress bar indicating the progress of processing. The amount of processing completed is displayed as a percentage.

Figure 3.3-1 shows an example of the Collecting Data dialog box.

Figure 3.3-1 Example of Collecting Data Dialog Box



■ Communication Time

The communication time is slower in the order of simulator debugger, monitor debugger, and emulator debugger. It is also slower as the number of objects being used by the application and trace buffer size increase.

In the emulator and monitor debuggers, the memory access time influences the communication time.

For this reason, communication with an ICE or board should be set as a LAN connection or RS connection at a fast baud rate.

Note:

Even though the Collecting Data dialog box is closed manually, communication itself continues.

3.4 Menus

This section explains REALOS Analyzer menus.

■ [File] Menu

- **[Open]**
Opens a data file that has already been created.
- **[Save]**
Saves task trace data to a file.
- **[Exit]**
Terminates the application.

■ [Command] Menu

- **[Debugger-Go]**
Executes the debugger from Softune Workbench.
- **[Debugger-Abort]**
Stops the debugger from Softune Workbench.
- **[Debugger-Reset]**
Resets the MCU of the debugger from Softune Workbench.
- **[Update]**
Updates REALOS data.

■ [Object] Menu

- **[Task]**
Lists all the task information being used by the application program.
- **[Semaphore]**
Lists hierarchically all the semaphore information being used by the application program and the queuing state of a task waiting for a semaphore.
- **[Eventflag]**
Lists hierarchically all the event flag information being used by the application program and the queuing state of a task waiting for an event flag.

CHAPTER 3 BASIC OPERATION

○ [Mailbox]

Lists hierarchically all the mailbox information being used by the application program and the message queuing state of a task waiting for a message.

○ [Memorypool]

Lists hierarchically all the memorypool information being used by the application program and the memory block queuing state of a task waiting for a memory block.

○ [Cyclic Handler]

Lists all the cyclically activated handler information being used by the application program.

○ [Alarm Handler]

Lists all the alarm handler information being used by the application program.

○ [Queue]

Lists hierarchically information for any of the following queues:

- Ready queue
- Timer queue
- Alarm queue

■ [Stack] Menu

○ [Initialize]

Embeds the initial pattern for analyzing stack utilization (at reset).

○ [Used Stack]

Lists stack utilization levels.

■ [Trace] Menu

○ [Task Trace Figure]

Displays the task trace results in a transition diagram.

○ [Task Trace Tree]

Displays the task trace results hierarchically (tree).

○ [Object Trace]

Displays the object trace in a transition diagram according to the results of the task trace.

■ [Monitor] Menu

○ [Task Status]

Monitors the task status.

- **[Stack Monitor]**
Monitors the stack pointer.

■ **[Setup] Menu**

- **[Mode]**
Selects simple or detail mode.
(Initial setting: Simple mode)
- **[Select Task, Object]**
Selects the task object to be analyzed and displayed.
(Initial setting: All task objects)
- **[Task Trace]**
Selects ring buffer mode or buffer full mode of task trace.
(Initial setting: Ring buffer mode)

■ **[Window] Menu**

Window display function of an ordinary Windows application

■ **[Help] Menu**

- **[Help Topics]**
Opens the help file.
- **[About fran907s]**
Displays information about the program, and the version and copyright information

3.5 Toolbars

This section explains the REALOS Analyzer toolbars.

■ Toolbar

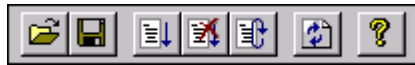
The main window has the following three toolbars:

- Main toolbar
- Window toolbar
- Setup toolbar

■ Main Toolbar

Figure 3.5-1 shows an example of the main toolbar.

Figure 3.5-1 Example of Main Toolbar



The main toolbar includes the following functions:

(From the left to right)

- File Open
- File Save
- Debugger-Go
- Debugger-Abort
- Debugger-Reset
- Update
- Help Topics

■ Window Toolbar

Figure 3.5-2 shows an example of the window toolbar.

Figure 3.5-2 Example of Window Toolbar



The window toolbar includes the following functions:

(From left to right)

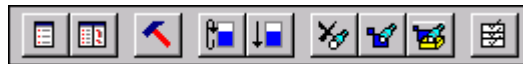
- Open Task list Window
- Open Semaphore list Window
- Open Eventflag list Window
- Open Mailbox list Window

- Open Memorypool list Window
- Open Cyclic Handler list Window
- Open Alarm Handler list Window
- Open Queue list Window
- Open Stack Information
- Open Task Trace Figure
- Open Task Trace Tree
- Execute Task Status Monitor
- Execute Stack Monitor

■ Setup Toolbar

Figure 3.5-3 shows an example of the setup toolbar.

Figure 3.5-3 Example of Setup Toolbar



The setup toolbar includes the following functions:

{From left to right}

- Set Simple Mode
- Set Detail Mode
- Select Task and Object
- Initialize Stack
- Set Ring Buffer Mode
- Set Full Buffer Mode

3.6 Status Bar

The following four types of information are displayed on the status bar of the REALOS Analyzer:

- **Debugger execution information**
 - **PC information**
 - **Initialization information**
 - **Module information**
-

■ Status Bar

The following four types of information are displayed on the status bar of the main window:

- Debugger execution information
- PC information
- Initialization information
- Module information

■ Explanation of Information

The following explains debugger execution information, PC information, initialization information, and module information in this order, which is the order, from left to right, of information on the status bar.

○ Debugger execution information

During debugger execution, "Execute" is displayed.

○ PC information

Information indicating whether the current PC is in the REALOS kernel is displayed.

- Inside kernel: The current PC is inside the REALOS kernel.
- Outside kernel: The current PC is outside the REALOS kernel.

When the current PC is in the REALOS kernel, the information displayed by the REALOS Analyzer is not guaranteed because the kernel is performing processing.

○ Initialization information

Information indicating whether REALOS initialization has been completed is displayed.

- Uninitialized: REALOS initialization has not been completed.
- Initialized: REALOS initialization has been completed.

If REALOS initialization is not complete, time is not displayed in the REALOS Analyzer task trace information because the system clock has not been initialized. Time is only displayed in the information generated after REALOS initialization has been completed.

○ Module information

Information indicating whether the task analysis module is built into the module being debugged is displayed.

- Disable trace: A task analysis module dedicated to the REALOS Analyzer is not built into the module being debugged. The task trace function cannot be used.
- Enable trace: A task analysis module dedicated to the REALOS Analyzer is built into the module being debugged.

Figure 3.6-1 shows an example of the status bar.

Figure 3.6-1 Example of Status Bar



3.8 Commands

The REALOS Analyzer uses the following commands to control the Softune Workbench debugger:

- **Debugger-Go/DebuggerAbort/Debugger-Reset**
 - **Update**
-

■ Types of Commands

The commands that control Softune Workbench debugging are explained below.

○ **[Debugger-Go]**

Executes the Softune Workbench debugger.

○ **[Debugger-Abort]**

Stops the Softune Workbench debugger.

○ **[Debugger-Reset]**

Resets the MCU of the Softune Workbench debugger.

○ **[Update]**

Information collected the previous time the Softune Workbench debugger stopped is not updated because no information is collected when the debugger executes a step. Use this function to update information.

When break or abort is executed, the REALOS Analyzer automatically collects information from Softune Workbench and reflects it in the window.

3.9 Object Display

Besides information display, the following functions for REALOS Analyzer object display are provided:

- Sort function
- History function
- Jump function

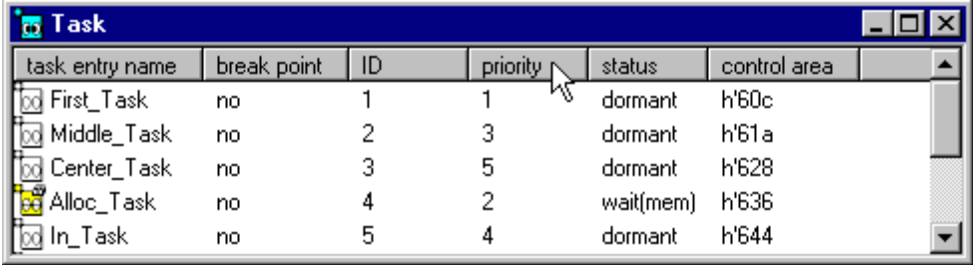
■ Sort Function

Clicking the heading of a column sorts the items in the column.

Clicking the column heading again restores the previous display order.

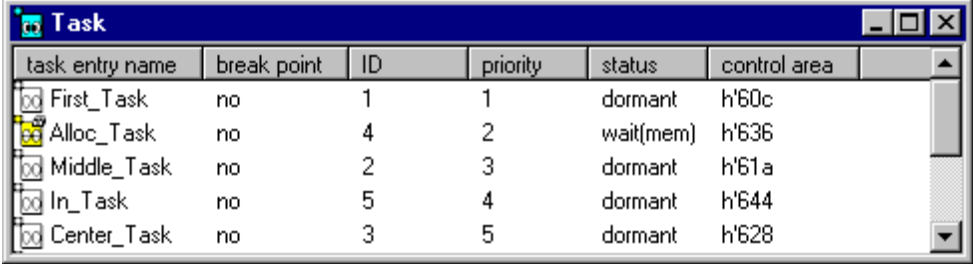
Figure 3.9-1 shows an example of sorting by priority.

Figure 3.9-1 Example of Sorting by Priority



task entry name	break point	ID	priority	status	control area
First_Task	no	1	1	dormant	h'60c
Middle_Task	no	2	3	dormant	h'61a
Center_Task	no	3	5	dormant	h'628
Alloc_Task	no	4	2	wait(mem)	h'636
In_Task	no	5	4	dormant	h'644

▼ Sorting



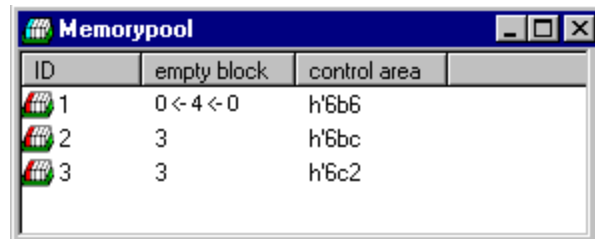
task entry name	break point	ID	priority	status	control area
First_Task	no	1	1	dormant	h'60c
Alloc_Task	no	4	2	wait(mem)	h'636
Middle_Task	no	2	3	dormant	h'61a
In_Task	no	5	4	dormant	h'644
Center_Task	no	3	5	dormant	h'628

■ History Function

The history function displays the history of changes for a data item. The display order is left to right, from the most recent to the least recent. The history is limited to 255 characters. If there have been no changes, no history is displayed.

Figure 3.9-2 shows an example of displaying the history when the number of free blocks in the memorypool has changed from 0 to 4 to 0.

Figure 3.9-2 Example of Displaying the Free Block History



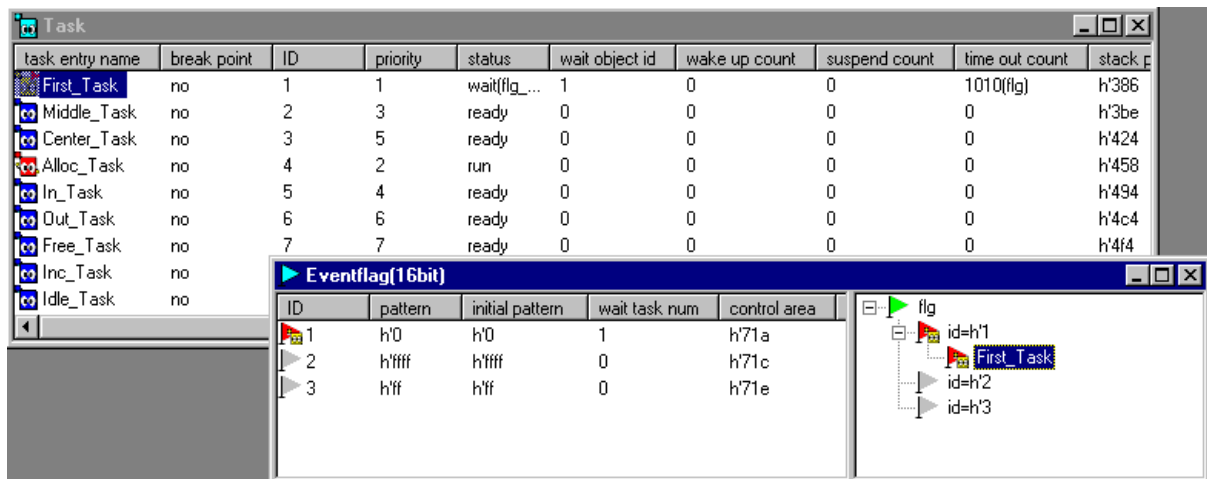
ID	empty block	control area
1	0 <- 4 <- 0	h'6b6
2	3	h'6bc
3	3	h'6c2

■ Jump Function

When a related object is double-clicked, the REALOS Analyzer jumps to the item in a related window.

Suppose, for example, that a task waiting for a flag exists in the Eventflag list window. Double-clicking the task highlights the line with the selected task ID and opens the Task list window.

Figure 3.9-3 Example of Displaying the Jump Function (Task Waiting for Message[***] Task List Window)



task entry name	break point	ID	priority	status	wait object id	wake up count	suspend count	time out count	stack p
First_Task	no	1	1	wait(flag...	1	0	0	1010(flag)	h'386
Middle_Task	no	2	3	ready	0	0	0	0	h'3be
Center_Task	no	3	5	ready	0	0	0	0	h'424
Alloc_Task	no	4	2	run	0	0	0	0	h'458
In_Task	no	5	4	ready	0	0	0	0	h'494
Out_Task	no	6	6	ready	0	0	0	0	h'4c4
Free_Task	no	7	7	ready	0	0	0	0	h'4f4
Inc_Task	no								
Idle_Task	no								

ID	pattern	initial pattern	wait task num	control area
1	h'0	h'0	1	h'71a
2	h'fff	h'fff	0	h'71c
3	h'ff	h'ff	0	h'71e

■ Differences in Display Modes

The REALOS Analyzer provides two modes: simple and detail. Which mode to select depends on the amount of the data that the REALOS Analyzer exchanges with Softune Workbench.

When simple mode is selected for object display, fewer items in the list are displayed and the window displaying waiting tasks is not displayed.

■ Update Timing

Object display information is automatically updated when the debugger stops because of a break or execution stop (excluding step execution).

When a step is executed, information must be updated by clicking [Update] from the [Command] menu because updating is not automatic.

3.10 Stack

Collect stack utilization information with the following procedure:

You can check each stack area by clicking [Initialize] from the [Stack] menu before executing the application. You can also check stack utilization by clicking [Used Stack] from the [Stack] menu after executing the application.

■ Procedure for Analyzing the Stack Utilization

Analyze stack utilization in the following order:

○ [Stack] - [Initialize]

This operation pads memory in the stack area with the pattern specified by the REALOS Analyzer. The operation can be executed according to the following timing:

- At reset or when, before r97_entry is executed, the debugger is inactive and REALOS has not been initialized .

○ [Stack] - [Used Stack]

When this operation is executed, the REALOS Analyzer allocates memory in the stack area and searches for the boundary on which the previously padded pattern was rewritten. Execution occurs on the following timing:

- Debugger is inactive after execution

■ Basic Functions of Stack Information List window

The Stack Information List window has the following functions:

- Setup
- Check
- Graph display

○ Setup

The setup function sets check items.

The setup function can also set (select) a check for amount of unused area and a check for simultaneous use of the shared stack. The set information is used at check.

○ Check

The check function checks the following two functions:

- Function that searches for the tasks where the amount of unused area is few
 - The setup function determines the amount of unused area to be checked.
- Function that searches for tasks simultaneously using a shared stack

○ Graph

The graph function displays stack utilization in a graph and the current stack pointer.

3.11 Task Trace

When debugging execution stops, the REALOS Analyzer can display the results of a task trace in a format of task trace figure or task trace tree, or object trace.

The REALOS Analyzer can also analyze task trace data from various aspects and create a graph.

■ Methods for Displaying Task Trace Data

When debugging execution stops, the task analysis module can use the following three methods to display accumulated task trace data:

- Task trace figure
- Task trace tree
- Object trace

■ Update Timing

Task trace data is automatically updated when the debugger stops because of a break or execution stop (excluding step execution). Each window is also updated automatically at the same time.

■ Setting Trace Buffer Mode

In a task trace, ring buffer mode or buffer full mode can be selected as the trace buffer mode.

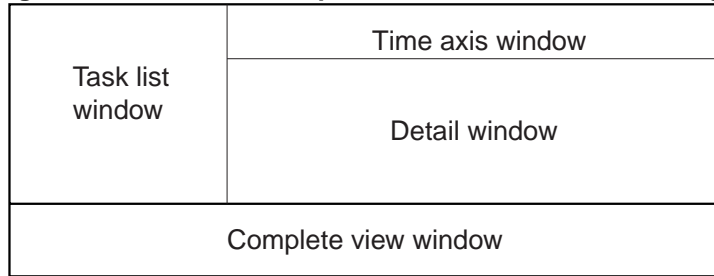
- Ring buffer mode
 - In ring buffer mode, only the most recent data is stored up to the size of the buffer is stored.
- Buffer full mode
 - In buffer full mode, sampling stops when the buffer becomes full. However, although the buffer becomes full, debugger execution does not stop. In this mode, the executable and wait states are also indicated with lines.

■ Task trace figure

The task trace figure is a state transition diagram plotted horizontally against time. In the task trace figure, each task is positioned vertically. This diagram makes it easier for the user to understand state changes because timing is handled intuitively.

Figure 3.11-1 shows the names of the split windows in a task trace figure.

Figure 3.11-1 Names of Split Windows in a Task Trace Figure



The contents of each split window are explained below.

○ **Task list window**

Entry name, ID, and priority are listed from left to right.

The following are listed vertically:

- Initialization
- Interrupt processing
- Task x number of definitions (in order by ID)
- Idle group

If you select a task by clicking [Setup]- [Select Task Object], the items are displayed in the following order:

- Initialization
- Interrupt processing
- Task x number of selected tasks (in order by ID)
- ID group

If there are too many items to be displayed, a scroll bar is displayed to the right of the task list. When you use the scroll bar, the task transition diagram displayed in the detail window moves in unison.

○ **Time axis window**

The time axis window displays time.

The unit of time is one system clock.

○ **Detail window**

Main window for the task transition diagram. This window displays events such as the issuance of system calls as icons and the execution of a task as a solid line.

○ **Complete view window**

The complete view window displays a complete view of the task transition diagram. The range currently displayed in the detail window is highlighted. The icons representing events are not displayed in this window.

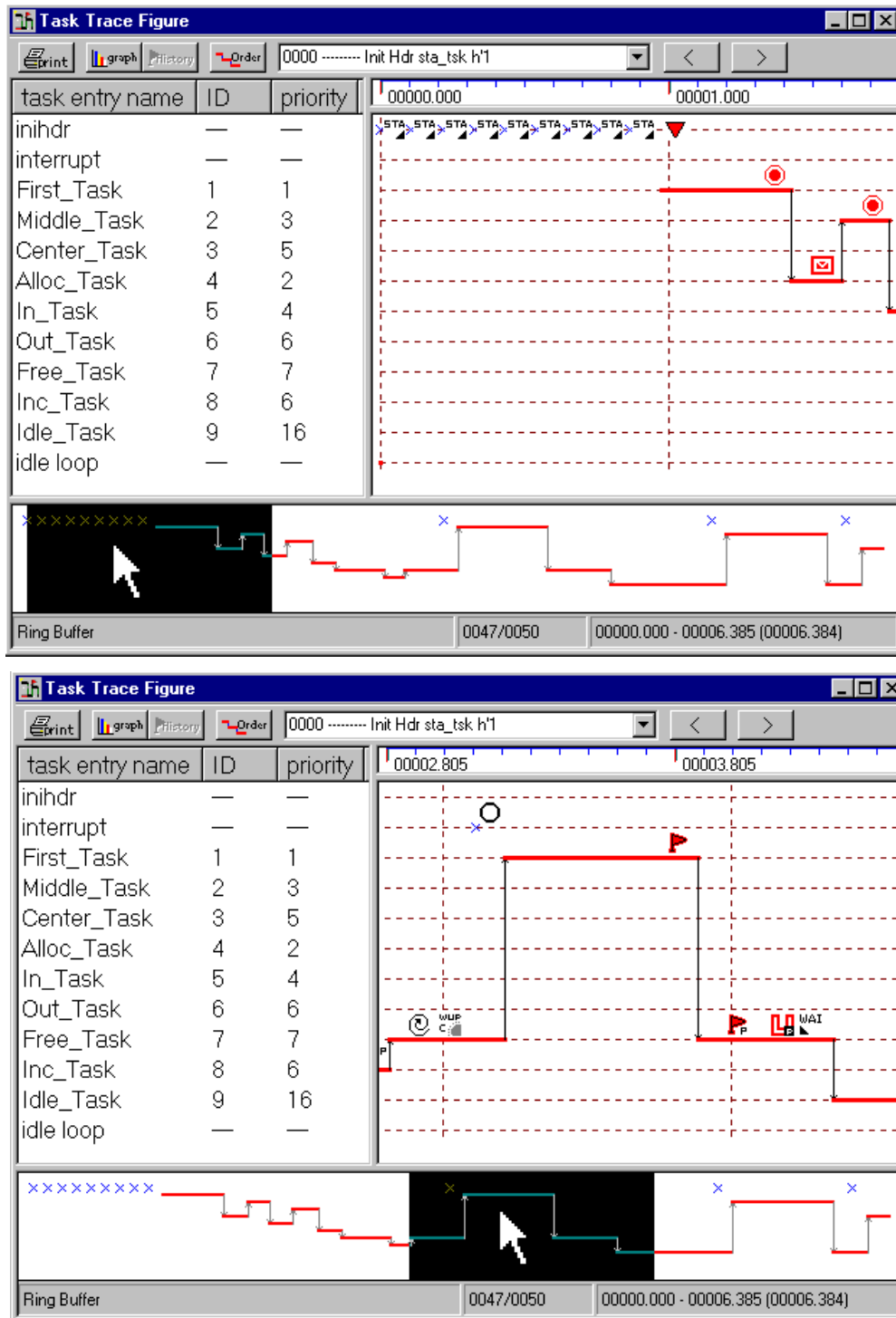
■ **Basic Operation of the Task Trace Figure**

○ **Scrolling**

The task trace figure contains a detail window and a complete view window. In the complete

view window, the area displayed as the detail window is highlighted. To scroll the information in the detail window, click and drag the highlighted area or click the drag destination. Figure 3.11-2 is an example of scrolling the detail window by clicking the highlighted area. You can also use the left and right scroll buttons on the toolbar to scroll the detail window 100 points at a time.

Figure 3.11-2 Example of Scrolling the Detail Window by Clicking the Highlighted Area



○ Scaling function

The scaling function enlarges the detail window size at a rate of "twice" at a time up to "ten

CHAPTER 3 BASIC OPERATION

times". It can also reduce the size at a rate of "1/2" at a time down to "1/10". Use the popup menu to enlarge and reduce the detail window size.

○ Item sort function

As with the Object display window, you can sort the items in the Task Trace Figure window by column (entry name, ID, priority) by clicking a task list window column where each task is listed vertically.

○ Icon information display

Clicking an icon in the detail window displays information for the icon in the dialog box and the combo box on the toolbar. Use the dialog box to retrieve related trace data.

○ Jump to a selected frame

The REALOS Analyzer can jump to the frame selected in the combo box on the toolbar.

○ Print function

The detail window of the Task Trace Figure window can be printed. Print it by clicking the [print] button on the toolbar of the Task Trace Figure window.

○ Displaying an execution time graph

The execution time and dispatch count for each task can be displayed in a list form or graph form.

■ Task trace tree

The task trace tree displays the following three events hierarchically, allowing statistical analysis from different points of view:

- System call issuance for each task
- Issuing task for each system call
- System call issuance sequenced by time

You can switch the display by clicking a button on the toolbar of the Task Trace Tree window. Clicking the [Graph] button displays a graph.

■ Object trace

An object trace displays the state transitions of each object. The state changes can be verified with respect to timing because the object trace displays state transitions in conjunction with the task trace figure. (Beforehand, set [Task Trace] to [Full Buffer] and [Mode] to [Detail] on the [Setup] menu.)

You can also display an object trace by clicking [Object Trace] on the [Trace] menu. In addition, you can display an object trace by clicking a button on the toolbar in the Task Trace Figure window.

To see the state transitions of an object, select the object you want to display from the window on the left in which objects are hierarchically displayed.

Note:

If task trace execution stops in the kernel, time information for the last or first frame may be incorrect.

3.12 File Handling

If the task trace figure will be redisplayed, the REALOS Analyzer can save the data in a file and open a previously saved file.

■ REALOS Analyzer Data File

Data file that the REALOS Analyzer uses to display the task trace figure. The extension for the file is (*.ran).

■ Opening a Previously Saved File

When active, the REALOS Analyzer can open a previously saved REALOS Analyzer data file and display the task trace figure contained in it.

■ Saving Data

When debugging stops and the task trace figure opens, the REALOS Analyzer can save data for drawing the task trace figure. In this case, if the execution time graph is already open, the REALOS Analyzer can also save the data in CSV format.

3.13 Monitoring

Monitoring is executed to sample data when debugging is executed.

■ Executing Monitoring

Opening the task status monitor or stack monitor executes monitoring. If the debugger has stopped, monitoring is started when debugging is executed.

■ Monitoring Setup

In the monitoring setup, a monitoring interval and monitoring task can be selected from [Set Monitor/Object Trace] of [Select Task, Object] on the [Setup] menu.

Note:

If monitoring is executed before REALOS initialization is complete, sampling occurs before REALOS has been initialized. This may cause the display to be incorrect. Be sure to execute monitoring after REALOS initialization has been completed.

The first selected tasks are ID=1 to ID=10. If the number of the tasks defined in the application is less than 10, all the tasks are selected.

CHAPTER 4 WINDOWS

This chapter explains the windows used by the REALOS Analyzer.

- 4.1 REALOS Project Window
- 4.2 Object Window
- 4.3 Stack Information
- 4.4 Task Trace Figure Window
- 4.5 Task Trace Tree Window
- 4.6 Object Trace Window
- 4.7 Monitoring
- 4.8 Setup
- 4.9 Help

4.1 REALOS Project Window

A REALOS project window is a Softune Workbench project window in which the initial information managed by REALOS is displayed.

■ 4.1 Notes on REALOS Project Window

This section explains the information displayed in and the popup menu of the REALOS Project Window.

4.1.1 Information Displayed in the REALOS Project Window

4.1.2 REALOS Project Window Popup Menus

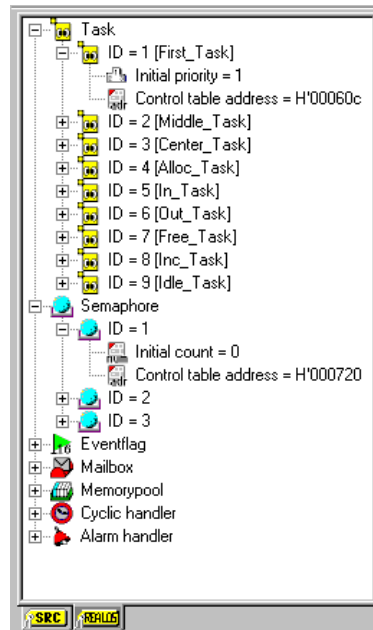
4.1.1 Information Displayed in the REALOS Project Window

The REALOS project window displays the initial information for objects managed by REALOS, such as tasks and semaphores. The objects are sorted by ID.

■ REALOS Project Window

Figure 4.1-1 shows an example of the REALOS project window.

Figure 4.1-1 REALOS Project Window



■ Information Displayed in the REALOS Project Window

Table 4.1-1 lists the items displayed in the REALOS project window.

Table 4.1-1 Items Displayed in the REALOS Project Window

Root Icon	Object	Icon	Item	Description
	Task		ID	ID and entry name set by the configurator
			Priority of activation	Activation priority set by the configurator
			Control table address	Start address of the task control area maintained by the kernel

Table 4.1-1 Items Displayed in the REALOS Project Window (Continued)
























Root Icon	Object	Icon	Item	Description
	Semaphore		ID	ID set by the configurator
			Initial value	The initial count of semaphores
			Control table address	Start address of the semaphore control area maintained by the kernel
	Eventflag		ID	ID set by the configurator
			Initial pattern	Initial patterns of event flags
			Control table address	Start address of the event flag control area maintained by the kernel
	Mailbox		ID	ID set by the configurator
			Control table address	Start address of the mailbox control area maintained by the kernel
	Memorypool		ID	ID set by the configurator
			Block start address	Address at which the first block is allocated
			Block count	Number of blocks
			Block size	Block unit size.
			Control table address	Start address of the memorypool control area maintained by the kernel

Table 4.1-1 Items Displayed in the REALOS Project Window (Continued)

Root Icon	Object	Icon	Item	Description
	Cyclic Handler		ID	Handler number set by the configurator
			Control table address	Start address of the cyclic handler control area maintained by the kernel
	Alarm Handler		ID	Handler number set by the configurator
			Control table address	Start address of the alarm handler control area maintained by the kernel

Note:

The REALOS project window is registered only after it is activated by the debugger.

4.1.2 REALOS Project Window Popup Menus

You can display the popup menus of the REALOS project window by doing the following:

- Right-clicking an object type
- Right-clicking an object

■ REALOS Project Window Popup Menus

Table 4.1-2 lists the popup menus of the REALOS project window.

Table 4.1-2 REALOS Project Window Popup Menus


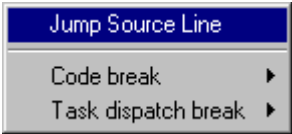
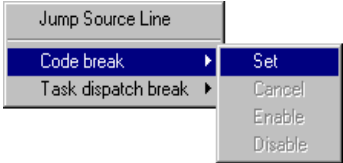
Popup menu	Object right-clicked	Item name	Function to be executed
	Object type	Defined Number	The number of definitions for the selected object is displayed.
		Used Memory	The amount of OS data for the selected object is displayed.
	Task	Jump source line	The start position of the source line of the selected task is displayed.
	Task	Break point	Popup menu
		Set	Sets a break point at the beginning of the source.
		Cancel	Removes the break point from the beginning of the source.
		Enable	Enables the break point at the beginning of the source.
		Disable	Disables the break point at the beginning of the source.

Table 4.1-2 REALOS Project Window Popup Menus (Continued)

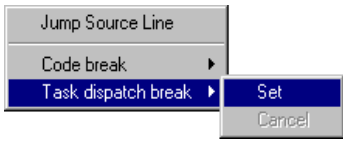
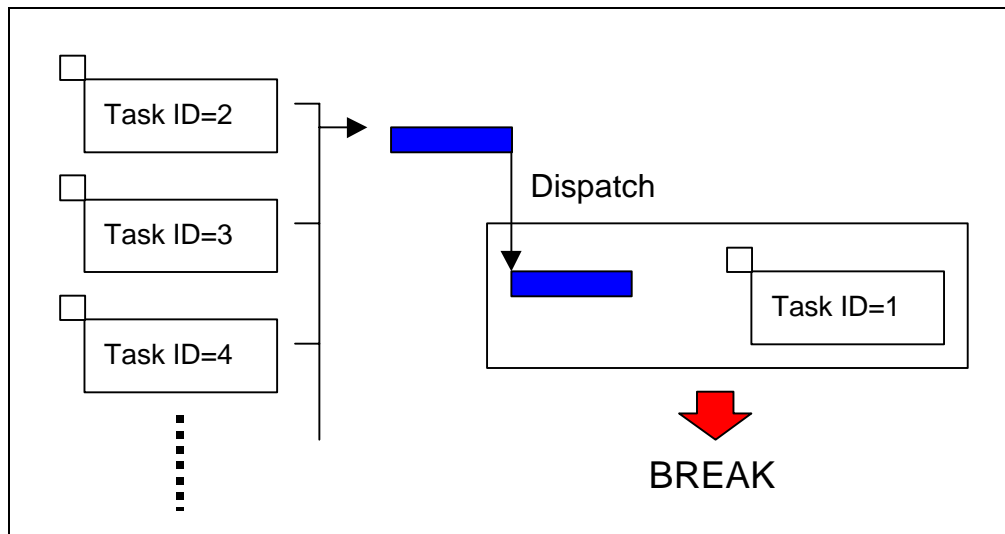
Popup menu	Object right-clicked	Item name	Function to be executed
	Task	Dispatch break	Popup menu
		Set	Sets a dispatch break.
		Cancel	Removes the dispatch break.

Figure 4.1-2 shows how the dispatch function can be used with the REALOS project window.

Figure 4.1-2 Concept of Task Dispatch Break



If a dispatch break is set for task 1, a break occurs when another task is dispatched to task 1.

4.2 Object Window

The object window displays information about objects managed by REALOS. Object windows specific to each object are provided.

You can choose whether the display area of an object window is arranged in list form or in tree form (only list form is available for tasks, cyclic handlers, and alarm handlers).

■ 4.2 Notes on Object Window

This section explains the Information displayed in and the popup menu of the Object Window.

4.2.1 Information Displayed in the Object Window

4.2.2 Object Window Popup Menus

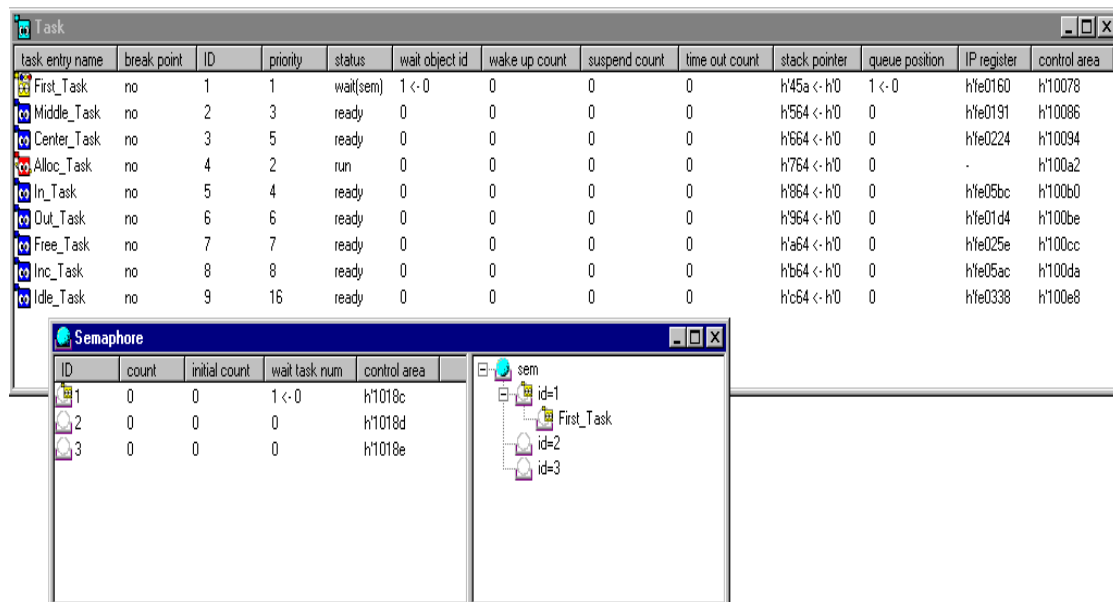
4.2.1 Information Displayed in the Object Window

An object window for the REALOS Analyzer displays information about objects managed by REALOS. A pane in list form displays information about objects, and a pane in tree form displays queues for the object.

■ Object Window

Figure 4.2-1 shows examples of object windows.

Figure 4.2-1 Object Windows



■ Functions Common to All Object Windows

The REALOS Analyzer includes a window that displays information about objects managed by REALOS.

A pane in list form displays information about objects.

A pane in tree form displays queues for the object.

The following three functions are common to all object windows:

- Sort function
- History function
- Jump function

For information about these functions, see Section 3.9, "Object Display".

CHAPTER 4 WINDOWS

■ List Form of Object Window

Tables 4.2-1 to 4.2-8 list the items displayed in the list form of an object window.

○ Task List window

Table 4.2-1 Task List Window

Column name	Description
Task entry name	Task entry name
Break point	State of the break point at the task start address
ID	Task ID
Priority	Task priority
Status	Task status (See Table 4-2.2 for the possibilities.)
Wait object id	ID of an object placed in the wait state
Wake up count	Time left before a task wakes up (in detail mode)
Suspend count	Suspend count (in detail mode)
Time out count	Time remaining before time-out (in detail mode) The flag pattern is displayed in the event flag wait state.
Stack pointer	Stack pointer value (in detail mode)
Queue position	Queued object sequence number (in detail mode)
IP register	IP register value (in detail mode)
Control area	Start address of the task control area

Table 4.2-2 Task Statuses

Representation	Status
run	Run state
ready	Ready for run state
dormant	Dormant state
suspend	Suspended state
wait(slp)*	Wait state (slp_tsk)
wait(tim)*	Wait state (wai_tsk)
wait(sem)*	Semaphore wait state
wait(flag)*	1-bit event flag wait state
wait(flag_clr)*	1-bit event flag wait state (with clear)
wait(flag_or)*	16-bit event flag wait state (OR wait)
wait(flag_and)*	16-bit event flag wait state (AND wait)

Table 4.2-2 Task Statuses (Continued)

Representation	Status
wait(flag_cor)*	16-bit event flag wait state (OR wait with clear)
wait(flag_cand)*	16-bit event flag wait state (AND wait with clear)
wait(flag_mbx)*	Message wait state
wait(flag_mpl)*	Memory block acquisition wait state

*: sus-wait(xxx) Double wait state

○ Semaphore List window

Table 4.2-3 Semaphore List Window

Column name	Description
ID	Semaphore ID
count	Current count value
initial count	Initial count value (in detail mode)
wait task num	Number of tasks placed in the wait state (in detail mode)
control area	Start address of the semaphore control area

○ Event Flag List window

Table 4.2-4 Event Flag List Window

Column name	Description
ID	Event flag ID
pattern	Current flag pattern
initial pattern	Initial flag pattern
wait task num	Number of tasks placed in the wait state (in detail mode)
control area	Start address of the event flag control area

○ Mailbox List window

Table 4.2-5 Mailbox List Window

Column name	Description
ID	Mailbox ID

Table 4.2-5 Mailbox List Window (Continued)

Column name	Description
wait num	Number of tasks and messages placed in the wait state (in detail mode)
control area	Start address of the mailbox control area

○ **MemoryPool List window**

Table 4.2-6 MemoryPool List Window

Column name	Description
ID	memorypool ID
Empty block	Number of empty blocks
block num	Maximum number of blocks (in detail mode)
block size	Block size (in detail mode)
wait num	IDs or block addresses of tasks placed in the wait state
control area	Start address of the memorypool control area

○ **Cyclic Handler List window**

Table 4.2-7 Cyclic Handler List Window

Column name	Description
handler No.	Handler number
activation	Activated state of the handler
time	Time remaining before handler activation
interval	Time interval the handler activation
handler address	Start address of the handler
control area	Start address of the cyclic handler control area

- Alarm Handler List window

Table 4.2-8 Alarm Handler List Window

Column name	Description
handler No.	Handler number
time	Time remaining before handler activation
handler address	Start address of the handler
control area	Start address of the alarm handler control area

- Tree Form of Object Window

The following items are displayed in the tree form of an object window.

- Display in tree form

Semaphores, event flags, mailboxes, and memorypools are displayed in a tree structure, as shown in Table 4.2-9. For mailboxes and memorypools, however, an object-specific item may be displayed as an additional LEVEL 2 item.

Table 4.2-9 Display Items in Tree Form

LEVEL0	LEVEL1	LEVEL2
Object name	Object ID number	Task entry name in the wait state (common)
		Message (only for mailboxes)
		Memory block address (only for memorypools)

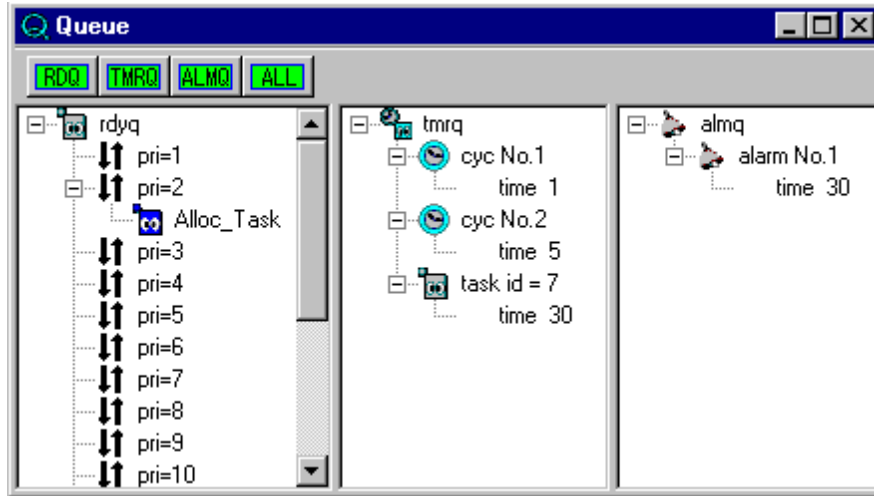
Note:

LEVEL indicates a level in the tree structure.

- Queue List Window

A Queue List window displays ready queues, timer queues, and alarm queues in tree form. Figure 4.2-2 shows an example of the Queue List window.

Figure 4.2-2 Queue List Window (From Left to Right: Ready Queues, Timer Queues, Alarm Queues)

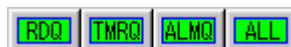


○ **Toolbar in the Queue List window**

The Queue List window has the toolbar shown in Figure 4.2-3. The buttons on the toolbar function as follows:

- [RDQ] button
 - Displays only ready queues.
- [TMRQ] button
 - Displays only timer queues.
- [ALMQ] button
 - Displays only alarm queues.
- [ALL] button
 - Displays ready, timer, and alarm queues.

Figure 4.2-3 Toolbar in the Queue List Window



○ **Ready queue display items**

Ready queues are displayed arranged in three hierarchical levels, as shown in Table 4.2-10. When you click a task entry name that is in the ready state, you jump to a task object window.

Table 4.2-10 Ready Queue Display Items

LEVEL0	LEVEL1	LEVEL2
Ready queue "rdyq" (Root level of tree display)	Priority	Task entry name in the ready state

- **Timer queue display items**

Timer queues are displayed arranged in three hierarchical levels, as shown in Table 4.2-11.

Table 4.2-11 Timer Queue Display Items

LEVEL0	LEVEL1	LEVEL2
Timer queue "tmrq" (Root level of tree display)	Task ID	Time remaining
	Cyclic handler number	

- **Alarm queue display items**

Alarm queues are displayed arranged in three hierarchical levels, as shown in Table 4.2-12.

Table 4.2-12 Alarm Queue Display Items

LEVEL0	LEVEL1	LEVEL2
Alarm queue "almq" (Root level of tree display)	Alarm handler number	Time remaining

- **Icons in Object Windows**

Tables 4.2-13 to 4.2-19 list the icons used in object windows.

- **Task List window**

Table 4.2-13 Icons in the Task List Window














Icon	Description
	Initial state
	Dormant state
	Ready state
	Run state
	Suspend state

Table 4.2-13 Icons in the Task List Window (Continued)

Icon	Description
	Event flag wait state
	Message wait state
	Memorypool wait state
	Semaphore wait state
	Sleep state
	Timer wait state
	Suspend - event flag wait state
	Suspend - message wait state
	Suspend - memorypool wait state
	Suspend - semaphore wait state
	Suspend - sleep state
	Suspend - timer wait state

○ Semaphore List window

Table 4.2-14 Icons in the Semaphore List Window

Icon	Description
	Initial state
	Count = 0
	Count = 1
	Count > 1
	At least one task is queued.
	Queued task

○ Event Flag List window

Table 4.2-15 Icons in the Event Flag List Window











Icon	Description
	Initial state
	There are no queued tasks.
	At least one task is queued.

Table 4.2-15 Icons in the Event Flag List Window (Continued)

Icon	Description
	Queued task

○ Mailbox List window

Table 4.2-16 Icons in the Mailbox List Window

Icon	Description
	Initial state
	There are no messages.
	There is at least one message.
	At least one task is queued.
	Queued task
	Queued message

○ MemoryPool List window

Table 4.2-17 Icons in the MemoryPool List Window







Icon	Description
	Initial state

Table 4.2-17 Icons in the MemoryPool List Window (Continued)

Icon	Description
	Empty memory blocks are available.
	No free memory blocks.
	At least one task is queued.
	Queued task
	Queued block

○ Cyclic Handler List window

Table 4.2-18 Icons in the Cyclic Handler List Window











Icon	Description
	Initial state
	Not defined
	Defined (activation state: off)
	Left time of less than 15%
	Left time of 15% to 30%

Table 4.2-18 Icons in the Cyclic Handler List Window (Continued)

Icon	Description
	Left time of 30% to 50%
	Left time of more than 50%

○ Alarm Handler List window

Table 4.2-19 Icons in the Alarm Handler List Window

Icon	Description
	Initial state
	Defined to be off
	Defined to be on

4.2.2 Object Window Popup Menus

When you right-click in the display area of a list form object window, a popup menu is displayed (only for tasks, cyclic handlers, and alarm handlers).

■ Task List Popup Menu

The popup menu for tasks provides the following functions:

- Jump to the source line
- Break point operation
- Dispatch break operation

■ Cyclic Handler and Alarm Handler Popup Menu

The popup menu for cyclic handlers and alarm handlers provides the following function:

- Jump to the source line

■ Popup Menu Functions

The functions of the popup menus for an object window are the same as those provided by the popup menus for the REALOS object window (see Section 4.1.2, "Popup Menus of REALOS Project Window"). Table 4.2-20 lists the popup menu functions.

Table 4.2-20 Popup Menus of an Object Window

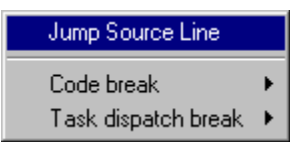
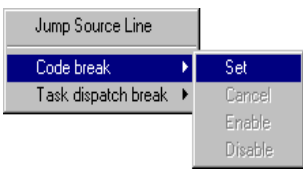
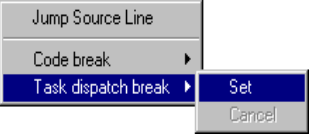
Popup menu	Item right-clicked	Item name	Function to be executed
	Task Cyclic handler Alarm handler	Jump source line	The start position of the source line of the selected task appears.
	Task	Break point	Popup menu
		Set	Sets a break point at the beginning of the source.
		Cancel	Removes the break point from the beginning of the source.
		Enable	Enables the break point at the beginning of the source.
		Disable	Disables the break point at the beginning of the source.

Table 4.2-20 Popup Menus of an Object Window (Continued)

Popup menu	Item right-clicked	Item name	Function to be executed
	Task	Dispatch break	Popup menu
		Set	Sets a dispatch break.
		Cancel	Removes the dispatch break.

Note:

In Windows 98, the function of jumping to the source line does not bring Softune Workbench to the foreground.

4.3 Stack Utilization List

The stack utilization list is a window for displaying how tasks use stacks and for displaying stack area information.

■ 4.3 Notes on Stack Utilization List

This section explains the stack initialization, and the information displayed in and the settings of the Stack Utilization List.

4.3.1 Stack Initialization

4.3.2 Information Displayed in the Stack Utilization List

4.3.3 Stack Utilization List View Setting

4.3.4 Stack Utilization Check

4.3.5 Stack Utilization Graph

4.3.1 Stack Initialization

Execute stack initialization.

After a reset or before REALOS initialization (before the execution of r97_entry), the stack areas must be initialized with the specified pattern.

■ Stack Initialization

Execute stack initialization at the following point:

- After a reset, but before REALOS initialization (before the execution of r97_entry)

Outside this period, initialization cannot be executed and display of the associated control display is grayed out. If you want to analyze stack utilization, execute stack initialization each time the application program is reset.

Stack initialization is essential processing for the analysis of stack utilization.

4.3.2 Information Displayed in the Stack Utilization List

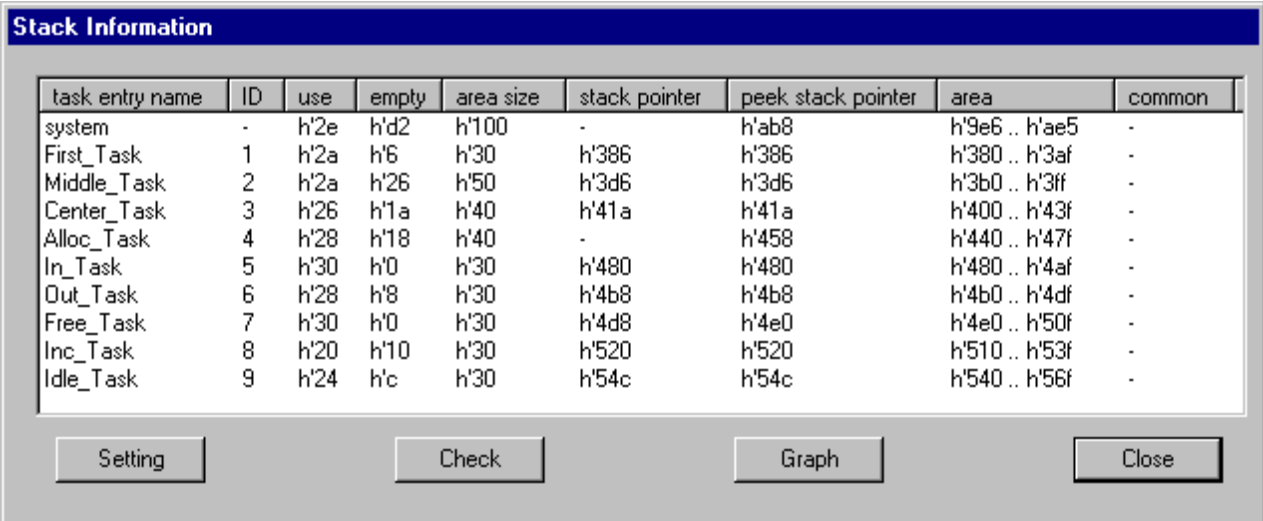
The stack utilization list displays the following information:

- Task entry name
- Task ID
- Stack pointer
- Peak stack pointer
- Used
- Free
- Area
- Area size
- Common stack

■ Example of Stack Utilization List

Figure 4.3-1 shows an example of a stack utilization list.

Figure 4.3-1 Stack Utilization List



task entry name	ID	use	empty	area size	stack pointer	peek stack pointer	area	common
system	-	h'2e	h'd2	h'100	-	h'ab8	h'9e6 .. h'ae5	-
First_Task	1	h'2a	h'6	h'30	h'386	h'386	h'380 .. h'3af	-
Middle_Task	2	h'2a	h'26	h'50	h'3d6	h'3d6	h'3b0 .. h'3ff	-
Center_Task	3	h'26	h'1a	h'40	h'41a	h'41a	h'400 .. h'43f	-
Alloc_Task	4	h'28	h'18	h'40	-	h'458	h'440 .. h'47f	-
In_Task	5	h'30	h'0	h'30	h'480	h'480	h'480 .. h'4af	-
Out_Task	6	h'28	h'8	h'30	h'4b8	h'4b8	h'4b0 .. h'4df	-
Free_Task	7	h'30	h'0	h'30	h'4d8	h'4e0	h'4e0 .. h'50f	-
Inc_Task	8	h'20	h'10	h'30	h'520	h'520	h'510 .. h'53f	-
Idle_Task	9	h'24	h'c	h'30	h'54c	h'54c	h'540 .. h'56f	-

■ Items Displayed in the Stack Utilization List

Table 4.3-1 lists the items displayed in the stack utilization list.

Table 4.3-1 Items Displayed in the Stack Utilization List

Item name	Description
task entry name	The entry name of a task that uses a stack
task ID	ID of a task that uses a stack
stack pointer	Current stack pointer value

Table 4.3-1 Items Displayed in the Stack Utilization List (Continued)

Item name	Description
peak stack pointer	After filling, the greatest stack pointer value
used	The amount of stack used (in bytes)
empty	The amount of free stack (in bytes)
area	The lower-limit and upper-limit addresses of a stack area
area size	The size of the stack area is displayed.
common stack	When a common stack is used, the IDs of the tasks that share the stack are displayed.

Note:

Use the values for "used" after the stack area has been initialized. If initialization has not been executed, the display may indicate that no space is available.

If internal execution of the REALOS kernel stops, the stack pointer value may be unreliable.

The items "stack pointer", "peak stack pointer", "used", and "empty" are displayed only when you select a task with [Setup] - [Select Task, Object].

4.3.3 Stack Utilization List View Setting

When you click the [Set] button in the Stack Utilization List window, the Stack Utilization List View Setting dialog opens. The following can be selected:

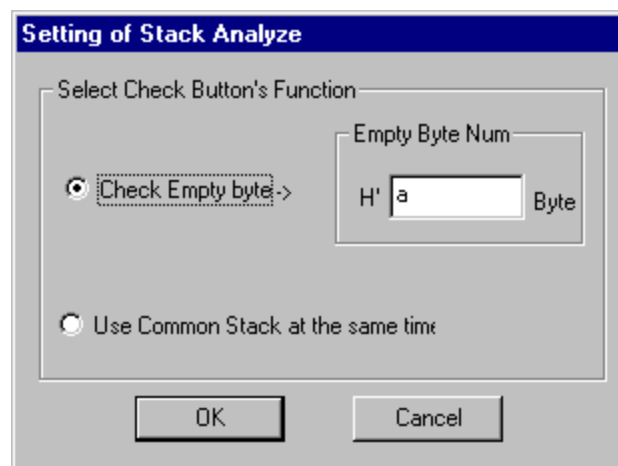
- Check Empty byte
- Use Common Stack at the same time

A check function selected in this window is executed as described in Section 4.3.4, "Stack Utilization Check."

■ Stack Utilization List View Setting

Figure 4.3-2 shows an example of the setting dialog box.

Figure 4.3-2 Stack Utilization List View Setting



○ Check Empty byte

Specify the amount of stack space you expect to be free. A task for which the amount of free stack space is less than the specified stack amount is highlighted in the list.

You must enter a value for number of free bytes to perform this check. Use a hexadecimal value.

○ Use Common Stack at the same time

This check is made to see whether there is concurrent use of a common stack during the break state. If concurrent use is found, the stack is highlighted in the list.

4.3.4 Stack Utilization Check

When you click the [Check] button in the Stack Utilization List window, either of the following check functions is executed:

- Check Empty byte
- Use Common Stack at the same time

The selection of a check function is made as described in Section 4.3.3, "Stack Utilization List View Setting."

■ Executing a Stack Utilization Check Function

Click the [Check] button in the Stack Utilization List window (see Section 4.3.2, "Information Displayed in the Stack Utilization List").

Task entries that meet a certain condition are highlighted in the stack usage list.

If you want to remove the highlighting, click the [Check] button again.

Figure 4.3-3 shows an example of execution of a check for free stack space.

Figure 4.3-3 Results of Checking for Free Stack Space (Check with 0x50 Bytes Specified)

Stack Information								
task entry name	ID	use	empty	area size	stack pointer	peek stack pointer	area	common
system	-	h'2e	h'd2	h'100	-	h'ab8	h'9e6 .. h'ae5	-
First_Task	1	h'2a	h'6	h'30	h'386	h'386	h'380 .. h'3af	-
Middle_Task	2	h'2a	h'26	h'50	h'3d6	h'3d6	h'3b0 .. h'3ff	-
Center_Task	3	h'26	h'1a	h'40	h'41a	h'41a	h'400 .. h'43f	-
Alloc_Task	4	h'28	h'18	h'40	-	h'458	h'440 .. h'47f	-
In_Task	5	h'30	h'0	h'30	h'480	h'480	h'480 .. h'4af	-
Out_Task	6	h'28	h'8	h'30	h'4b8	h'4b8	h'4b0 .. h'4df	-
Free_Task	7	h'30	h'0	h'30	h'4d8	h'4e0	h'4e0 .. h'50f	-
Inc_Task	8	h'20	h'10	h'30	h'520	h'520	h'510 .. h'53f	-
Idle_Task	9	h'24	h'c	h'30	h'54c	h'54c	h'540 .. h'56f	-

Setting Check Graph Close

4.3.5 Stack Utilization Graph

When you click the [Graph] button in the Stack Utilization List window, a stack utilization graph is displayed.

The vertical axis of the bar graph represents stack area size. The horizontal axis represents the task ID. Task ID S, which is displayed in the first position, means a system stack.

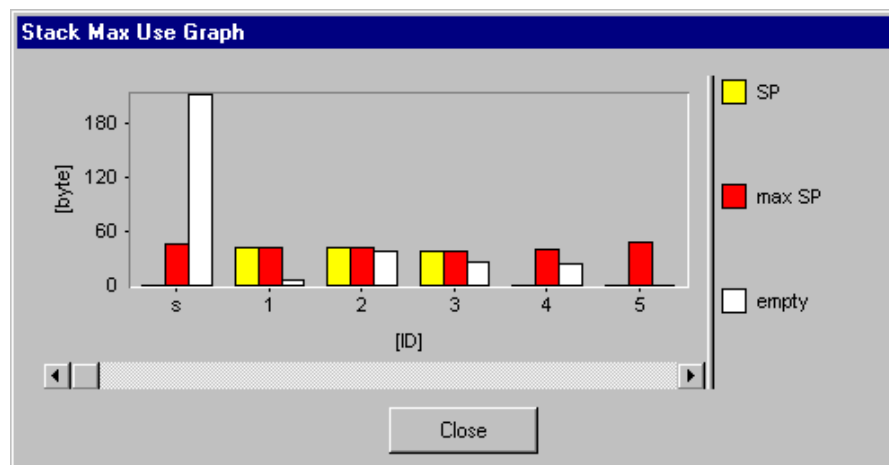
■ Stack Utilization Graph

In the graph window, three bars, with different colors, are displayed for each task.

- Yellow bar
 - Indicates the current stack pointer value ("SP" in the graph).
- Red bar
 - Indicates the peak stack pointer value after the stack area has been filled ("max SP" in the graph).
- White bar
 - Indicates the amount of unused area ("no used" in the graph).

Figure 4.3-4 shows an example of graph display.

Figure 4.3-4 Graph Showing Maximum Available Stack Space



■ When Task Not Selected

If a task is not selected with [Setup] - [Select Task, Object], the bar graph for the task indicates that its stack area is completely free.

Note:

If the stack area has not been filled, the stack utilization graph may consist entirely of red bars.

4.4 Task Trace Figure Window

The Task Trace Figure window displays the results of a task trace in a trace figure. The tasks are listed on the vertical axis. The horizontal axis represents time.

■ 4.4 Notes on Task Trace Figure Window

This section explains the information displayed in and the functions of the Task Trace Figure Window.

4.4.1 Information Displayed in the Task Trace Figure Window

4.4.2 Task Trace Figure Window Toolbar

4.4.3 Task Trace Figure Window Status Bar

4.4.4 Task Trace Figure Window Popup Menus

4.4.5 Printing a Task Trace Figure

4.4.6 Running Time Graph

4.4.7 Task Trace Figure Information Dialog Box

4.4.1 Information Displayed in the Task Trace Figure Window

Four kinds of information are displayed in the Task Trace Figure window:

- **Time at which an event occurs:** Displayed in the time display window.
- **Event:** An icon associated with the event that has occurred is displayed.
- **Task status:** A specific line type designates a specific task status.
- **Dispatch:** Indicated by an arrow.

■ Information Displayed in the Task Trace Figure Window

Four kinds of information are displayed in the Task Trace Figure window:

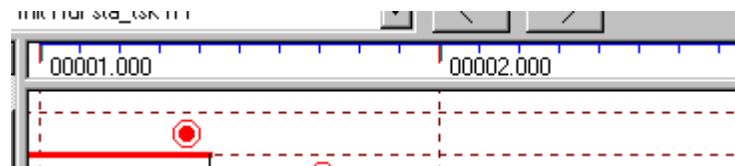
- Time at which an event occurs: Displayed in the time display window.
- Event: An icon associated with the event that has occurred is displayed.
- Task status: A specific line type designates a specific task status.
- Dispatch: Indicated by an arrow.

■ Time at Which an Event Occurs

The time at which an event occurs is displayed in the time display window.

Figure 4.4-1 shows an example of time display in the time display window.

Figure 4.4-1 Example of Time Display in the Time Display Window



■ Task Trace Figure Icons

There are two major categories of event indicator icons:

- Icons indicating that a system call was issued
- Icons indicating an event such as a time-out occurred

Table 4.4-1 lists the icons related to issuing of system calls.

Table 4.4-1 Icons Indicating Event Other Than Issuing of a System Call

Icon	System call name
STA ▲	sta_tsk
EXT ▲	ext_tsk

Table 4.4-1 Icons Indicating Event Other Than Issuing of a System Call (Continued)
















Icon	System call name
	ter_tsk
	chg_pri
	rot_rdq
	tsk_sts
	sus_tsk
	rem_tsk
	frsm_tsk
	slp_tsk
	wai_tsk
	wup_tsk
	can_wup
	sig_sem
	wai_sem
	preq_sem
	sem_sts

Table 4.4-1 Icons Indicating Event Other Than Issuing of a System Call (Continued)















Icon	System call name
	set_flg
	clr_flg
	wai_flg
	cwai_flg
	pol_flg
	cpol_flg
	flg_sts
	snd_msg
	rcv_msg
	prcv_msg
	mbx_sts
	get_blk
	pget_blk
	rel_blk
	mpl_sts

Table 4.4-1 Icons Indicating Event Other Than Issuing of a System Call (Continued)








Icon	System call name
	set_tim
	act_cyc
	cyh_sts
	alh_sts
	get_ver

Table 4.4-2 lists icons that an event other than issuing of a system call.

Table 4.4-2 Icons Indicating Event Other Than Issuing of a System Call

Icon	Event indicated by icon	Display location
	Time-out	Task
	First event after completion of initialization	Initialization process

■ Line Types for Indicating Task Status

Table 4.4-3 lists the line types used to indicate the status of a specific task.

Table 4.4-3 Line Types for Indicating Task Status

Line	Color	Status
 (Solid line)	Red	Run state
None	None	Dormant state
 (Solid line)	Light blue	Ready for run state
 (Dash-dot line)	Light blue	Wait state

Table 4.4-3 Line Types for Indicating Task Status (Continued)

Line	Color	Status
— . . — . . — (Dash-two-dot line)	Light blue	Suspended state
— . . — . . — (Dash-two-dot line)	Red	Double wait state

■ Dispatch Indication

Dispatching is indicated by a black solid line with an arrow that is perpendicular to the time axis.

The end of the solid line indicates the dispatching source of a task and the location pointed to by the arrow indicates its destination.

■ System Clock Reset Indication

The REALOS Analyzer displays time based on the system clock.

If the system clock time is changed during task tracing by `set_tim` or other operation, the change affects the time display.

If `set_tim` is executed to change the system clock time, a blue solid line, perpendicular to the time axis, is drawn from the point of `set_tim` execution to indicate that the time has been changed.

If, however, the system clock time is changed by an action other than `set_tim`, the solid line indicating task execution and the line indicating task status may overlap or may be too long.

Note:

Of the system calls supported by REALOS/907, the `get_tid` task control function, the `ret_int`, `ret_wup`, `chg_ilv`, and `ilv_sts` interrupt control functions, and the `get_tim`, `def_cyc`, `def_alm`, and `ret_tmr` time control functions are not within the scope of task tracing. These system calls, therefore, are not displayed in a Task Trace Figure provided by the REALOS Analyzer.

If the trace buffer mode is a ring buffer, only the line indicating run state is displayed.

4.4.2 Task Trace Figure Window Toolbar

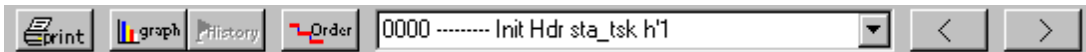
The Task Trace Figure window toolbar includes the following six functions:

- Print
 - Display running time graph window
 - Display Object Trace window
 - Sort dispatches sequentially
 - Display task trace information list (combo box)
 - Scroll
-

■ Task Trace Figure Window Toolbar

Figure 4.4-2 shows the Task Trace Figure window toolbar.

Figure 4.4-2 Task Trace Figure Window Toolbar



The following functions are assigned to the toolbar buttons from left to right:

- Print
- Display running time graph window
- Display Object Trace window
- Sort dispatches sequentially
- Display task trace information list (combo box)
- Scroll (to the left and the right)

■ Print

Prints the contents of the detail window of a Task Trace Figure.

See Section 4.4.5, "Printing a Task Trace Figure," for more information.

■ Display Running Time Graph Window

Analyzes the running time of the tasks obtained from the task trace data, and opens the running time graph window to display the results of analysis in graph form.

See Section 4.4.6, "Running Time Graph," for more information.

■ Display Object Trace Window

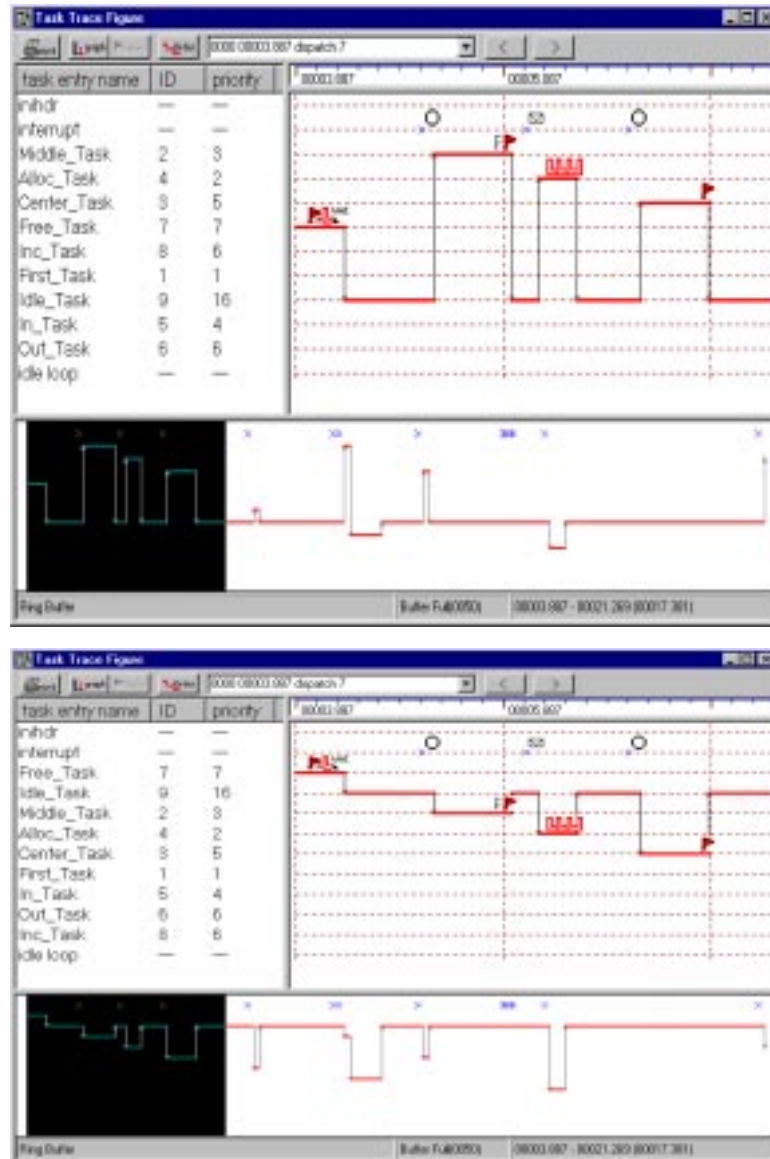
Analyzes the data about objects obtained from the task trace data, and creates and displays a trace figure of the objects. The operation is the same as selecting [Object Trace] from the [Trace] menu. A mode setting is required to open an Object Trace window. See Section 4.6, "Object Trace Window," for more information.

■ Sort Dispatches Sequentially

This function sorts tasks in the order in which they were dispatched.

When there are many tasks and dispatching between them is frequent, it is difficult to check task-to-task transitions with the detail window. Sorting the tasks sequenced according to time dispatched places the more recently processed tasks at the top of the list, making it easier to understand the circumstances of task-to-task transitions. Figure 4.4-3 shows an example of such a sequential sort.

Figure 4.4-3 Example of Sequentially Sorting Dispatches



■ Display Task Trace Information List (Combo Box)

Task trace data is displayed in the combo box on the Task Trace Figure window toolbar. The combo box contains the task trace data displayed in the following format:

event-number_time_event-type_task-ID_system-call_parameter

- **event-number**
 0 to (number of traced events - 1) (2047 maximum)
 0 is the oldest trace data. As this number is incremented, more recent event data is displayed.
- **time**
 It is not a time relative to the beginning of the displayed trace.
 This is the time indicated by the timer.
- **event-type**
 Event type is the type of an event, identified with an event number, that has occurred.
 There are the following event types:
 - task: A system call that has been issued from a task
 - inihdr: A system call that has been issued from an initialization process
 - interrupt: A system call that has been issued from an interrupt process
 - timeout: Time-out event
 - dispatch: Dispatch event
- **task-id**
 Task ID identifies the task associated with an event that has occurred. Task ID is displayed only for a system call that has been issued from a task and for time-out and dispatch events.
- **system-call**
 If the event is issuing of a system call, the issued system call name is displayed in this position.
- **parameter**
 An argument for the system call is displayed in this position.
 However, display of parameters is not enabled for all system calls.

Table 4.4-1 defines what display items are enabled for the type of traced event.

Table 4.4-4 Display Items for Each Type of Traced Event

	Time	Task ID	System call	Parameter
task	○	○	○	△
inihdr	×	×	○	△

Table 4.4-4 Display Items for Each Type of Traced Event (Continued)

	Time	Task ID	System call	Parameter
interrupt	○	×	○	△
timeout	○	○	×	×
dispatch	○	○	×	×

○: Enabled ×: Disabled △: Partially enabled

Note:

Parameter display is enabled for issuance of a system call. However, if there are no arguments, such as an `ext_tsk` parameter, parameter display is disabled.

If a system call returns an error code without terminating normally, it is not traced. For example, `prcv_msg` is traced only when a message has been acquired.

4.4.3 Task Trace Figure Window Status Bar

The following three items are displayed on the Task Trace Figure window status bar:

- Mode of trace buffer
 - Buffer size and data size
 - Trace time
-

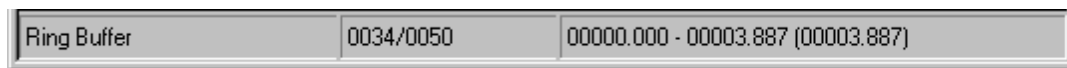
■ Task Trace Figure Window Status Bar

The following three items are displayed on the Task Trace Figure window status bar from left to right:

- Mode of trace buffer
- Buffer size and data size
- Trace time

Figure 4.4-4 shows an example of the Task Trace Figure window status bar.

Figure 4.4-4 Example of Task Trace Figure Window Status Bar



■ Mode of Trace Buffer

Displays the trace mode set with [Setup] - [Task Trace]. Either ring buffer or full buffer is displayed.

■ Buffer Size and Data Size

The trace buffer size and trace data size are displayed in the following format:

trace data size / trace buffer size

■ Trace Time

The time for tracing event number 0 and the time for tracing the last event are displayed in the following format:

first event time - last event time (total trace time)

4.4.4 Task Trace Figure Window Popup Menus

The Task Trace Figure window popup menus provide the following functions:

- [Enlarge/Reduce]
 - [Search]
-

■ Task Trace Figure Window Popup Menus

The Task Trace Figure window popup menus provide the following functions:

- [Enlarge/Reduce]
- [Search]

■ [Enlarge/Reduce]

Enlarges or reduces the time scale of a trace figure.

■ [Search]

Searches for data related to:

- semaphore
- event flag
- mailbox
- memorypool
- time

and displays all related events in a dialog box.

4.4.5 Printing a Task Trace Figure

A currently displayed Task Trace Figure can be printed.

■ Printing a Task Trace Figure

The print size of a Task Trace Figure is the same as the displayed diagram.

Note:

Because most task diagrams are longer horizontally, using landscape mode to print it is recommended.

4.4.6 Running Time Graph

You can display the results of analyzing task running time in a format that includes:

- Running time (%)
- Longest running time
- Shortest running time
- Dispatching count
- Time per status (%)

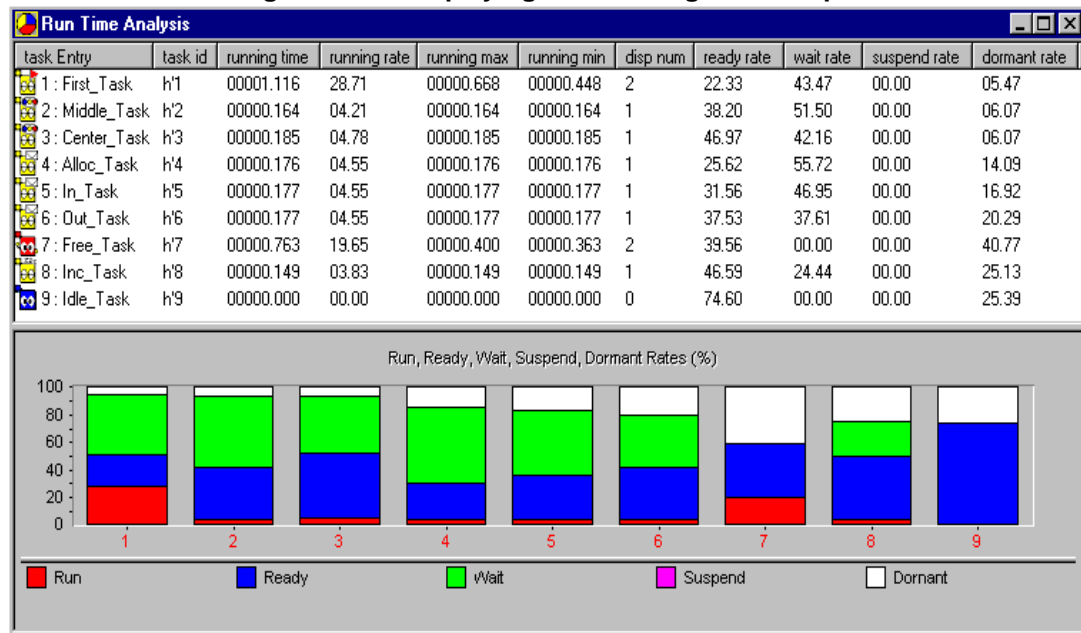
■ Running Time Graph

When you select display of the running time graph from the Task Trace Figure window toolbar, a running time graph is displayed. The window for the graph consists of an upper and a lower half. The following information is displayed in the window:

- Information list in the upper half
- Graph representation in the lower half

Figure 4.4-5 shows a display example.

Figure 4.4-5 Displaying the Running Time Graph



■ Information List

Information about running time is displayed in list form.

Table 4.4-5 lists the items contained in the information list.

Table 4.4-5 Information List Items

Item	Description	Graph	Required setting
entry name	Task entry name	✕	None.
task ID	Task ID	✕	None.
running time	The total time that the task ran	✕	None.
running rate	Total running time/total trace time	○	None.
running max	Longest continuous run time	○	None.
running min	Shortest continuous run time	○	None.
disp num	Number of dispatches to the task	○	None.
ready rate	Total of time that the task was ready to run	○	Buffer full
wait rate	Total time that the task was waiting	○	Buffer full
suspend rate	Total time that the task was suspended	○	Buffer full
dormant rate	Total time that the task was dormant	✕	Buffer full

■ **Graph**

Displays an item in the information list in graph form.

To display an item in the information list in graph form, click the column for the item. See Table 4.4-5 for the items that can be selected.

The basic graph displayed differs depending on the trace buffer setting.

- Ring buffer mode: Running rate graph
- Buffer full mode: Status time rates graph (The displayed graph includes all applicable statuses.)

To return to the basic graph after checking a different type of graph, click [running time].

■ Viewing the Graph

The values for the selected item are indicated by the vertical axis.

- Rate value: %
- Time value: Time of 1 system clock (When the time for 1 system clock is set at 1 ms, the value is in milliseconds)
- Count value: Number of times

Task entry numbers are given on the horizontal axis.

If, for example, a normal task ID is 1, the horizontal axis indicates 1 for the task.

Because a sort or the selection of a task alters the number sequence on the horizontal axis, a number does not always match the ID.

A task entry number appearing on the horizontal axis of the graph precedes the task name in the information list. After a sort or the selection of a task, check the task entry numbers.

Note:

Sometimes, the total of the rates for statuses does not equal 100%. This is an error generated by the calculation process. One double wait state is counted as one suspended state.

4.4.7 Task Trace Figure Information Dialog Box

The Task Trace Figure information dialog box opens when an icon indicating an event in a Task Trace Figure is clicked. The dialog box displays information about the event associated with the clicked icon.

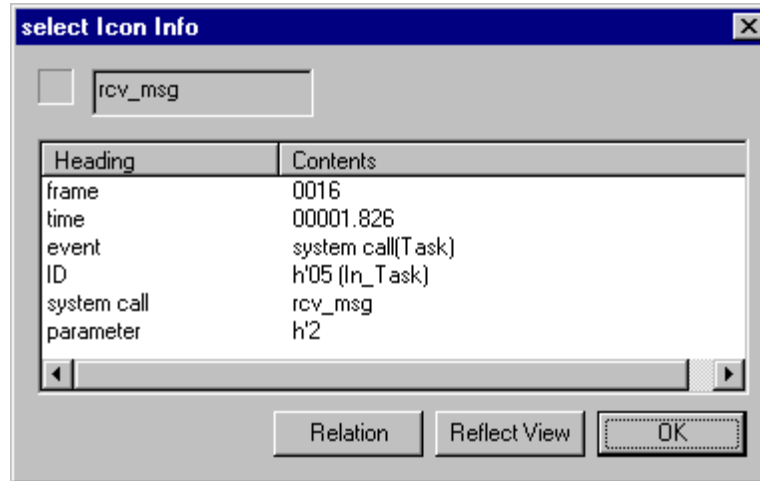
■ Task Trace Figure Information Dialog Box

When you click an icon indicating an event in a Task Trace Figure, the Task Trace Figure information dialog box opens.

The dialog box displays information about the event associated with the clicked icon.

Figure 4.4-6 shows an example of the dialog box.

Figure 4.4-6 Trace Figure Information Dialog Box



■ Information Dialog Box

The information dialog box displays the following information:

- **Icon indicating the event type**

Displays the icon selected by clicking.

- **System call name**

Displays the system call name indicated by the selected icon if the event type is issuance of a system call.

- **Event information list**

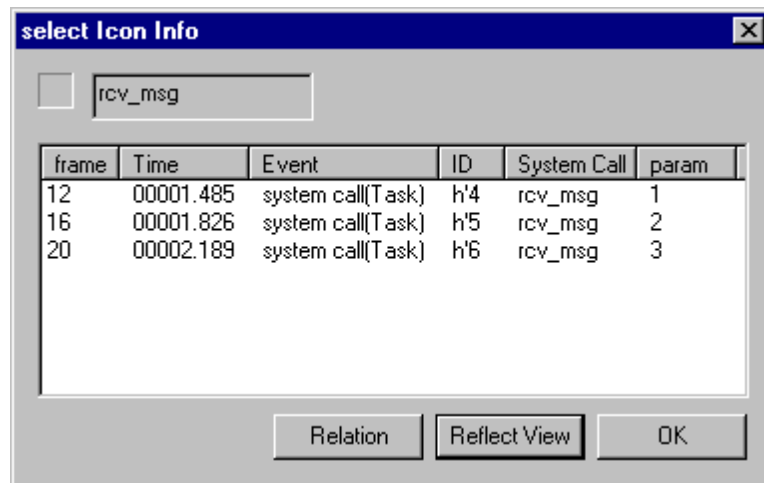
Displays event information in list form.

■ Buttons in the Information Dialog Box

When you click a button in the information dialog box, the appropriate function is implemented as indicated below.

- **[Related]**
Displays the related events in list form.
Figure 4.4-7 shows a display example.

Figure 4.4-7 Displaying the Related Event List



- **[Reflect in the Trace Figure]**
Only related events are displayed in the Task Trace Figure.
- **[Close]**
Closes the information dialog box.

■ Related Events

Related events refer to events whose data is obtained by manipulating the same type of object for the selected event.

If the selected event is obtained without manipulating an object, the events whose data is obtained without manipulating an object are listed as related events.

Example) ext_tsk when sta_tsk is selected

4.5 Task Trace Tree Window

A Task Trace Tree window displays the results of tracing of tasks in tree form.

■ 4.5 Notes on Task Trace Tree Window

This section explains the information displayed in and the functions of the Task Trace Tree Window.

4.5.1 Information Displayed in a Task Trace Tree Window

4.5.2 Task Trace Tree Window Toolbar

4.5.3 Displaying a Graph in the Task Trace Tree Window

4.5.1 Information Displayed in a Task Trace Tree Window

A Task Trace Tree window displays the results of tracing tasks. The information is arranged hierarchically according to how the trace data is sorted. The following types of sorts are provided:

- Sorting issued system calls by task
- Sorting by system call the tasks from which a system call has been issued
- Sorting the system calls sequentially by time issued

■ Information Displayed in a Task Trace Tree Window

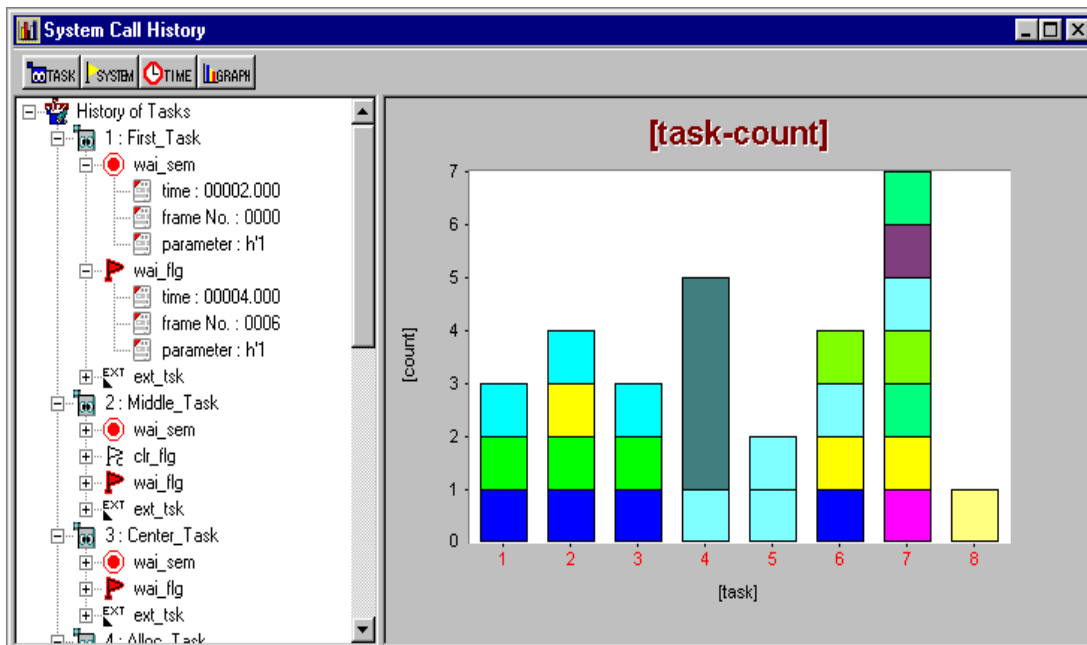
A Task Trace Tree window displays the results of tracing tasks. The information is arranged hierarchically according to how the trace data is sorted. The following types of sorts are provided:

- Sorting issued system calls by task
- Sorting by system call the tasks from which a system call has been issued
- Sorting the system calls sequentially by time issued

The window also displays the sorted trace data in graph form.

Figure 4.5-1 shows an example of the Task Trace Tree window.

Figure 4.5-1 Example of the Task Trace Tree Window



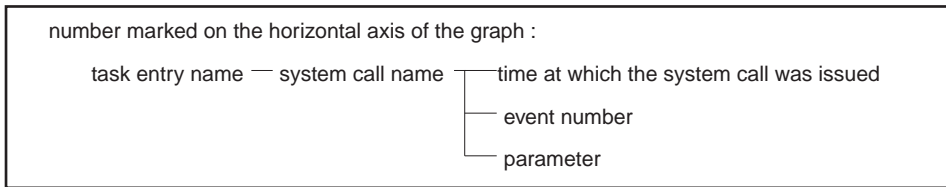
■ System Calls Sorted by Task

Issued system calls are sorted by task and displayed.

The tasks are listed in ID order.

CHAPTER 4 WINDOWS

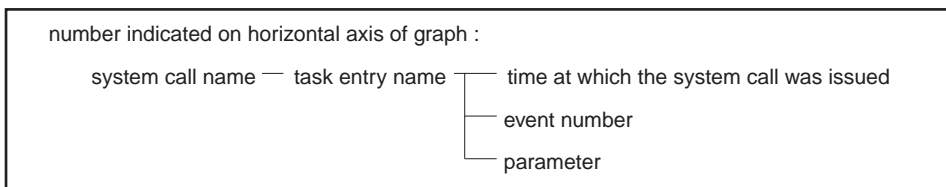
The trace data per task is arranged hierarchically as shown below.



■ Tasks from Which System Calls Have Been Issued

The tasks from which system calls have been issued are sorted by system call and displayed.

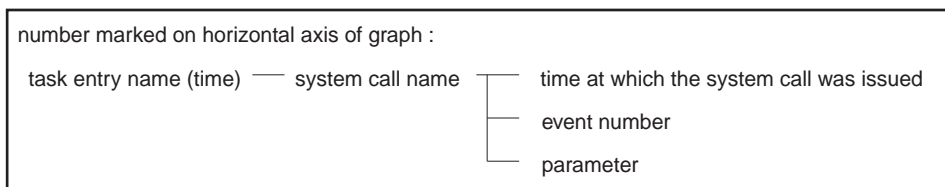
The trace data per system call is arranged hierarchically as shown below.



■ System Calls Sorted by Time Issued

Issued system calls are sorted according to time issued and displayed.

The trace data per task is arranged hierarchically as shown below.



■ Changing the Sorting Method

To change from one type of sort to another type, use the buttons on the Task Trace Tree window toolbar.

For information about the toolbar buttons, see Section 4.5.2, " Task Trace Tree Window Toolbar."

■ Graph Display

By using the [Graph] button on the Task Trace Tree window toolbar, you can display the sorted trace data in graph form.

For the information about the [Graph] button, see Section 4.5.2, " Task Trace Tree Window Toolbar."

For the information about the graph, see Section 4.5.3, " Displaying a Graph in the Task Trace Tree Window."

4.5.2 Task Trace Tree Window Toolbar

Using the Task Trace Tree window toolbar, you can change how trace data to be displayed is sorted and view the sorted trace data in graph form.

■ Task Trace Tree Window Toolbar

Using the Task Trace Tree window toolbar, you can change how trace data to be displayed is sorted and view the sorted trace data in graph form.

Figure 4.5-2 shows the Task Trace Tree window toolbar.

Figure 4.5-2 Task Trace Tree Window Toolbar



The following functions are assigned to the toolbar buttons from left to right:

- System calls sorted by task
- Tasks from which system calls have been issued
- System calls sequenced by time issued
- Display of graph

■ System Calls Sorted by Task

When selected, the leftmost task button makes displays issued system calls sorted by task.

For information about how the system calls sorted by task are displayed, see Section 4.5.1, "Information Displayed in the Task Trace Tree Window."

■ Tasks from Which System Calls Have Been Issued

When selected, the second button displays the tasks from which System Calls Have Been issued. Sorting is by system call.

For information about how tasks sorted by system call are displayed, see Section 4.5.1, "Information Displayed in the Task Trace Tree Window."

■ System Calls Sequenced by Time Issued

When selected, the third button displays system calls sorted in sequence by time issued.

For information about how the system calls sorted by time issued are displayed, see Section 4.5.1, "Information Displayed in the Task Trace Tree Window."

■ Display of Graph

When selected, the fourth button displays in graph form on the right side of the window the hierarchical trace data currently displayed on the left. Clicking this button when a graph is already displayed ends the display of the graph.

For the information about displaying the graph, see Section 4.5.3, "Displaying a Graph in the Task Trace Tree Window."

4.5.3 Displaying a Graph in the Task Trace Tree Window

In the Task Trace Tree window, the hierarchical trace data currently selected is displayed in graph form on the right side of the window.

■ Displaying a Graph in the Task Trace Tree Window

In the Task Trace Tree window, the hierarchical trace data currently selected is displayed in graph form on the right side of the window.

■ System Calls Sorted by Task

The graph displays issued system calls sorted by task.

The vertical and horizontal axes indicate the following:

- Vertical axis: Indicates the number of times system calls have been issued from a task.
 - The bar graph represents the number of all system calls that have been issued from the task. Colors are used to differentiate among the system calls.
- Horizontal axis: Sequential numbers indicate the names of task entries for which trace data is currently displayed in a hierarchical form.

For information about how system calls sorted by task are displayed in tree form, see Section 4.5.1, "Information Displayed in the Task Trace Tree Window."

■ Tasks from Which System Calls Have Been Issued

The graph displays the tasks from which system calls have been issued sorted by system call.

The vertical and horizontal axes indicate the following:

- Vertical axis: Indicates the number of times the system call was called.
 - The bar graph represents the number of times the system call was called from a task. Colors are used to differentiate tasks.
- Horizontal axis: Sequential numbers indicate names of system calls for which trace data is currently displayed in a hierarchical form.

For information about how the tasks sorted by system call are displayed in tree form, see Section 4.5.1, "Information Displayed in the Task Trace Tree Window."

■ System Calls Sorted by Time Issued

The graph displays system calls sorted according to the time they were issued.

The vertical and horizontal axes indicate the following:

- Vertical axis: Indicates the number of times the system call was called.
 - The bar graph represents the number of times a system call was called from a task. Colors are used to differentiate among the system calls.
- Horizontal axis: 10 equal divisions of the task trace time.

For example, if the total task trace time is 10,000, one division is 1,000. The bar displayed at position 1 indicates the number of system calls issued from the first event to 1,000.

4.5 Task Trace Tree Window

For information about how system calls sorted by time issued are displayed in tree form, see Section 4.5.1, "Information Displayed in the Task Trace Tree Window."

4.6 Object Trace Window

An Object Trace window displays the history of status transitions of an object based on the results of tracing the tasks.

■ 4.6 Notes on Object Trace Window

This section explains the information displayed in the Object Trace Window.

4.6.1 Information Displayed in the Object Trace Window

4.6.1 Information Displayed in the Object Trace Window

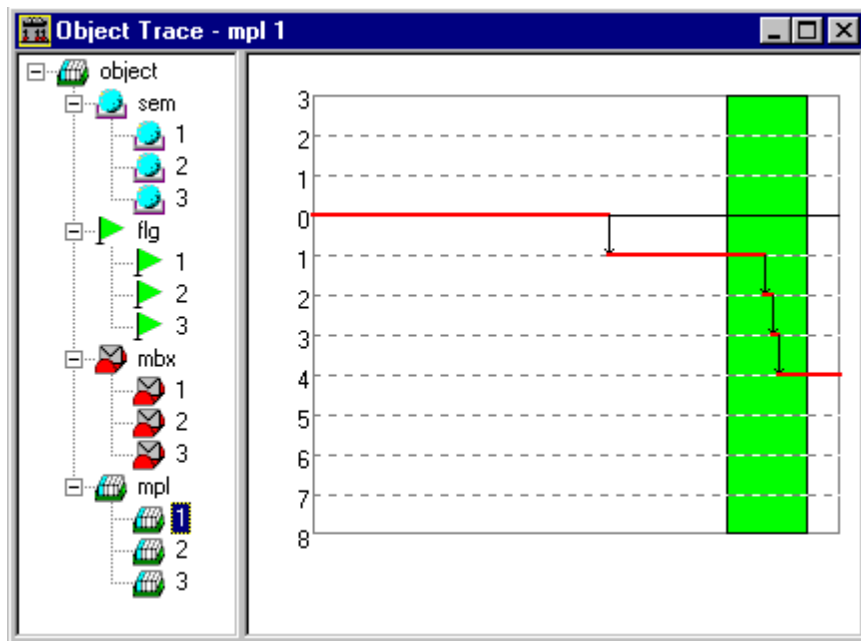
An Object Trace window displays a trace figure of an object used by an application.

■ Information Displayed in the Object Trace Window

An Object Trace window displays a trace figure of an object used by an application.

Figure 4.6-1 shows an example of an Object Trace window.

Figure 4.6-1 Example of an Object Trace Window



This window consists of two panes. In the left pane, the objects used by the application are displayed in hierarchical form. When you double-click an object in the left pane, the trace figure for the selected object is displayed in the right pane.

■ Vertical Axis of the Object Trace Window

The vertical axis of the Object Trace window indicates the following:

- The scale from 0 upwards indicates the amount of resources for the object. For a memorypool, the upper limit is the number of blocks that has been set.
- Position 0 indicates that nothing is linked to the object queue.
- The scale from 0 downwards indicates the number of task queues.

■ Horizontal Axis of the Object Trace Window

The horizontal axis of the Object Trace window indicates time.

The current size set for the window represents the total task trace time.

In this trace figure, the point currently displayed in conjunction with a Task Trace Figure is colored green.

Note:

The Object Trace display window can be opened only if [Full Buffer] has been selected for [Setup] - [Task Trace] and [Detail] selected for [Setup] - [Mode].

If the kernel has stopped, a phase mismatch is possible for the object display.

4.7 Monitoring

The provided monitoring facilities are a task status monitor and a stack monitor. Use these monitors to monitor the status and the stacks of tasks selected during execution.

■ 4.7 Notes on Monitoring

This section explains the information displayed in the Task Status Monitor window and the Stack Monitor window.

4.7.1 Information Displayed in the Task Status Monitor Window

4.7.2 Information Displayed in the Stack Monitor Window

4.7.1 Information Displayed in the Task Status Monitor Window

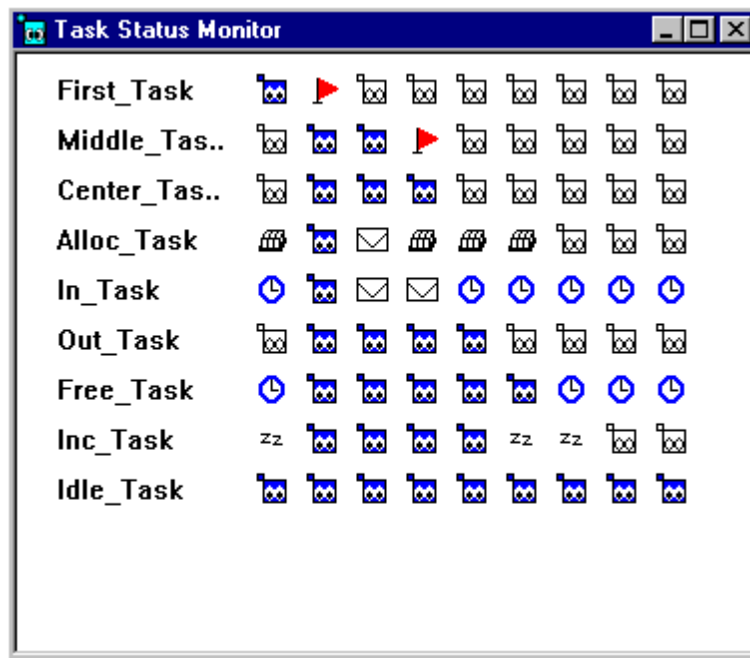
The Task Status Monitor window monitors the status of the selected tasks during the execution of the debugger, and monitoring results are periodically updated and displayed. Tasks are listed vertically, and their update histories is displayed horizontally.

■ Information Displayed in the Task Status Monitor Window

The Task Status Monitor window monitors the status of the tasks during execution of the Softune Workbench debugger.

The selected tasks are listed vertically, and the passage of time is represented horizontally. Ten buffers are used to stored monitoring data, with the more recent buffers on the right. The initial setting for tasks is 10.

Figure 4.7-1 Task Status Monitor Window



■ Icons

Table 4.7-1 lists the icons used in the Task Status Monitor window.

Table 4.7-1 Icons



Icon	color	Status indicated by icon
	Light blue	Initial state

Table 4.7-1 Icons (Continued)

Icon	color	Status indicated by icon
	White	Dormant state
	Blue	Ready state
	Red	Run state
	Gray	Suspend state
	Yellow	Semaphore wait state
	Red	Event flag wait state
	White	Memorypool wait state
	White	Message wait state
	White	Time-out wait state
zz	White	Sleep state
	Gray	Suspend - semaphore wait state
	Gray	Suspend - event flag wait state
	Gray	Suspend - memorypool wait state
	Gray	Suspend - message wait state
	Gray	Suspend - time-out wait state
	Gray	Suspend - sleep state

CHAPTER 4 WINDOWS

Table 4.7-1 Icons (Continued)

Icon	color	Status indicated by icon
	Black	A data acquisition error occurred

Note:

Up to 10 tasks can be selected for monitoring by the task status monitor.

A task in the run state is displayed as a task in the ready for run state.

The monitoring function executes memory access during MCU execution. If a free bus cannot be acquired during MCU execution, data acquisition fails and the icon indicating a data acquisition error is displayed.

4.7.2 Information Displayed in the Stack Monitor Window

The Stack Monitor window monitors the stack pointer of the selected tasks during execution of the debugger, and the monitoring results are periodically updated and displayed. Stack size (%) is displayed vertically, and the update history horizontally.

■ Information Displayed in the Stack Monitor Window

The Stack Monitor window monitors the stack pointer of a task at regular intervals during execution of the Softune Workbench debugger.

Stack size (%) is displayed vertically, and the update history horizontally.

The more recent data is displayed toward the right.

Ten buffers are used to stored monitoring data, with the more recent buffers on the right. The initial setting for tasks is 10.

Stack pointer data for the run and dormant states is not reliable. Table 4.7-2 lists the stack pointer states, which vary according to the task status.

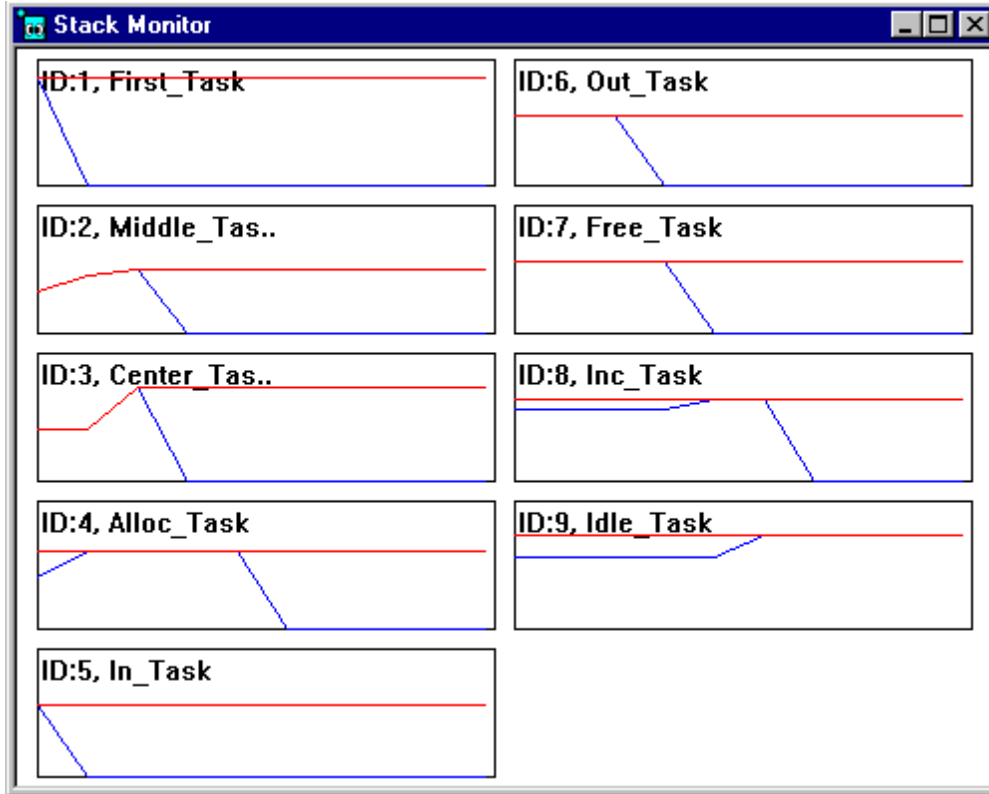
Table 4.7-2 Task Statuses and Stack Pointer Value

Status	Stack pointer value
Run	Invalid value (The previous stack pointer state is inherited.)
Read for run	Valid value
Dormant	Stack pointer when stack is not in use
Suspended	Valid value
Wait	Valid value

Figure 4.7-2 shows a display example.

- A blue line indicates a stack pointer.
- A red line indicates the maximum stack pointer value monitored.

Figure 4.7-2 Stack Monitor Window



Note:

Up to 10 tasks can be selected for monitoring by the stack monitor.

A stack pointer in the run state inherits its previous state.

A stack pointer in the dormant state is regarded as the stack point of a stack that is not being used.

4.8 Setup

This section explains the setup required for executing the REALOS Analyzer.

■ Notes on Setup

This section explains the setup.

4.8.1 Mode

4.8.2 Select Task, Object

4.8.3 Task Trace

4.8.1 Mode

Simple mode and detail mode are available.

■ Simple Mode and Detail Mode

○ Simple mode

- Only the information for some items is acquired.
- Queue information is not acquired.
- Communication time is less than for detail mode.

○ Detail mode

- All items are displayed in a list.
- Queue information can be acquired.

■ Differences Between Simple Mode and Detail Mode

Tables 4.8-1 to 4.8-8 list the differences between simple mode and detail mode for each type of object window.

○ Task List window

Table 4.8-1 Task List Window

Item name	Simple mode	Detail mode
Task entry name	○	○
Break point	○	○
ID	○	○
Priority	○	○
Status	○	○
Wait object ID	×	○
Wake-up count	×	○
Suspend count	×	○
Time-out count	×	○
Stack pointer	×	○
Queue position	×	○
IP register	×	○
Control area	○	○

○ :Displayed × :Not displayed

○ Semaphore List window

Table 4.8-2 Semaphore List Window

Item name	Simple mode	Detail mode
ID	○	○
Count	○	○
Initial count	×	○
Wait task num	×	○
Control area	○	○

○ :Displayed × :Not displayed

○ Event Flag List window

Table 4.8-3 Event Flag List Window

Item name	Simple mode	Detail mode
ID	○	○
Pattern	○	○
Initial pattern	○	○
Wait task num	×	○
Control area	○	○

○ :Displayed × :Not displayed

○ Mailbox List Window

Table 4.8-4 Mailbox List Window

Item name	Simple mode	Detail mode
ID	○	○
Wait num	×	○
Control area	○	○

○:Displayed × :Not displayed

○ MemoryPool List window

Table 4.8-5 MemoryPool List Window

Item name	Simple mode	Detail mode
ID	○	○
Empty block	○	○
Block num	×	○
Block size	×	○
Wait num	×	○
Control area	○	○

○:Displayed × :Not displayed

○ Cyclic Handler List window

Table 4.8-6 Cyclic Handler List window

Item name	Simple mode	Detail mode
ID	<input type="radio"/>	<input type="radio"/>
Activation	<input type="radio"/>	<input type="radio"/>
Time	<input type="radio"/>	<input type="radio"/>
Interval	<input type="radio"/>	<input type="radio"/>
Handler address	<input type="radio"/>	<input type="radio"/>
Control area	<input type="radio"/>	<input type="radio"/>

○ :Displayed × :Not displayed

○ Alarm Handler List window

Table 4.8-7 Alarm Handler List Window

Item name	Simple mode	Detail mode
Handler number	<input type="radio"/>	<input type="radio"/>
Time	<input type="radio"/>	<input type="radio"/>
Handler address	<input type="radio"/>	<input type="radio"/>
Control area	<input type="radio"/>	<input type="radio"/>

○ :Displayed × :Not displayed

- Ready queue, timer queue, and alarm queue window display

Table 4.8-8 Queue Window

Simple mode	Detail mode
×	○

○:Displayed × :Not displayed

4.8.2 Select Task, Object

Open the Select Task, Object dialog box to select the tasks and objects to be displayed in the windows.

■ Select Task, Object Dialog Box

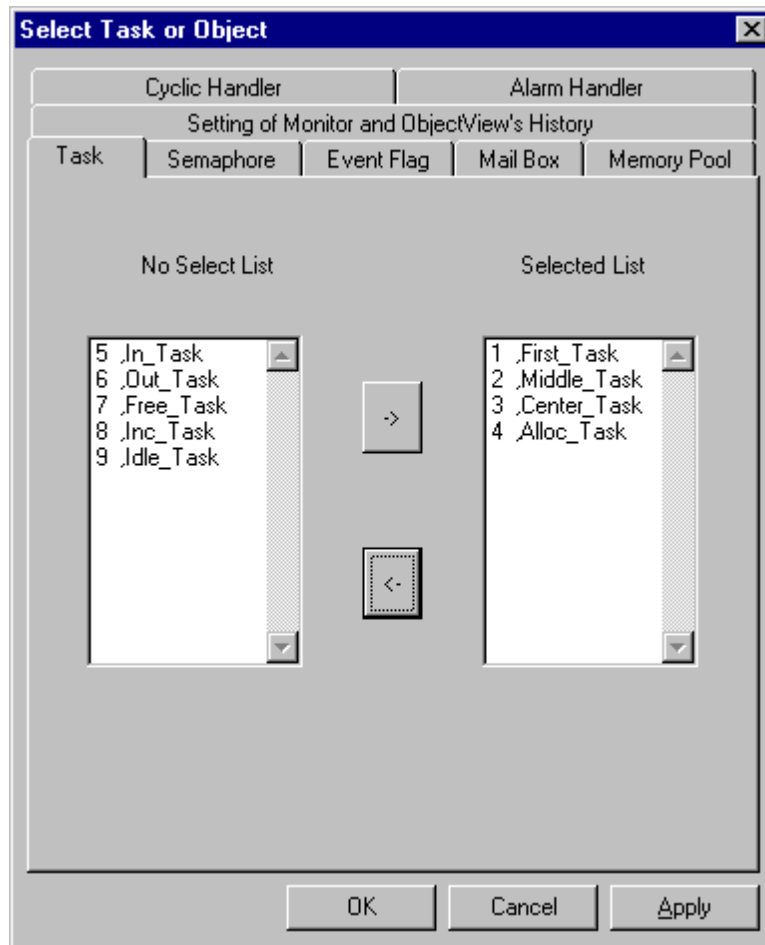
Using the tab dialog box, select tasks and objects for which you want to obtain trace data.

Move tasks or objects for which you want to obtain trace data to [selected list] and tasks or objects for which you do not want to obtain trace data to [unselected list].

After selecting a task or object, move it using the arrow buttons provided in the center of the dialog box window.

When selection is complete, press the [OK] or [Apply] button. The setting is updated.

Figure 4.8-1 Select Task, Object Dialog Box



■ Monitoring and Object Trace Setup Dialog Box

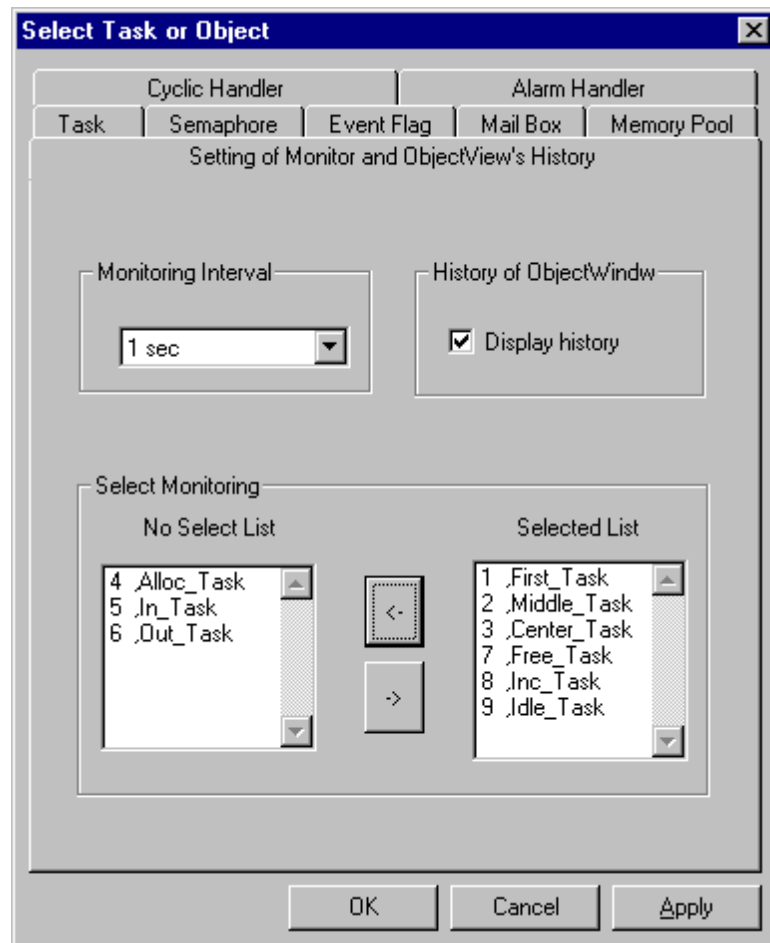
Using this dialog box, set up a monitoring function and the object view history display.

- **Monitoring setup**
 - Select the tasks to be monitored by the task status monitor and the stack monitor.
 - The initial setting is 10 tasks.
 - Set a monitoring interval:
 - 1 sec, 2 sec, 3 sec, 4 sec, 5 sec, 10 sec, 20 sec
- **Object view history display**

Choose whether the object view history is to be displayed.

As the initial setting, the [Display the history] check box is checked.

Figure 4.8-2 Monitoring and Object Trace Setup



Note:

Up to 10 tasks can be selected and set for monitoring. Task selection and setup are the same for the task status monitor and the stack monitor.

4.8.3 Task Trace

Two buffer modes, ring buffer mode and full buffer mode, are available for trace data buffering.

■ **Trace Buffer**

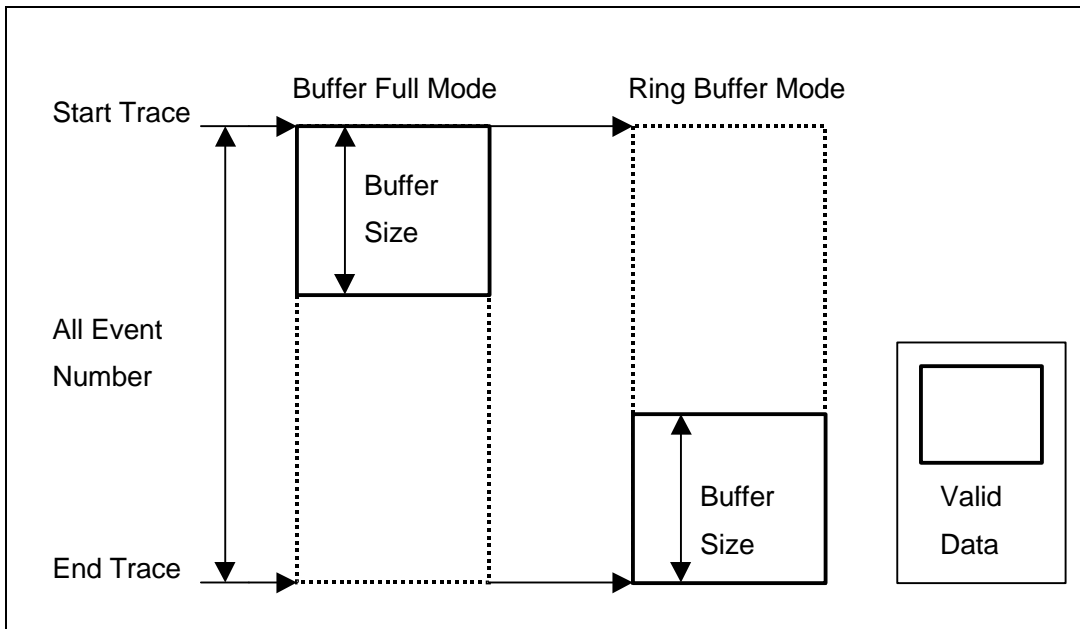
Two buffer modes, ring buffer mode and full buffer mode, are available for trace data buffering.

The primary difference between these modes is as follows:

- Ring buffer mode: The latest information is always acquired.
- Full buffer mode: Buffering terminates when the buffer is full of data.

Figure 4.8-3 provides an overview of trace data buffering.

Figure 4.8-3 Overview of Trace Data Buffering



■ **Object Trace and Status Line Display**

The display of object histories and status lines is enabled only in full buffer mode.

In addition, the object trace display requires detail mode.

4.9 Help

This section explains the functions that are executed to use the [Help] menu.

■ 4.9 Notes on Help

This section explains the functions that are executed in the Help menu and the displayed information.

4.9.1 Help Topics

4.9.2 About fra907se

4.9.1 Help Topics

The Help Topics command is described below.

■ **Help Topics**

Activates online help.

4.9.2 About fra907se

Version information is described below.

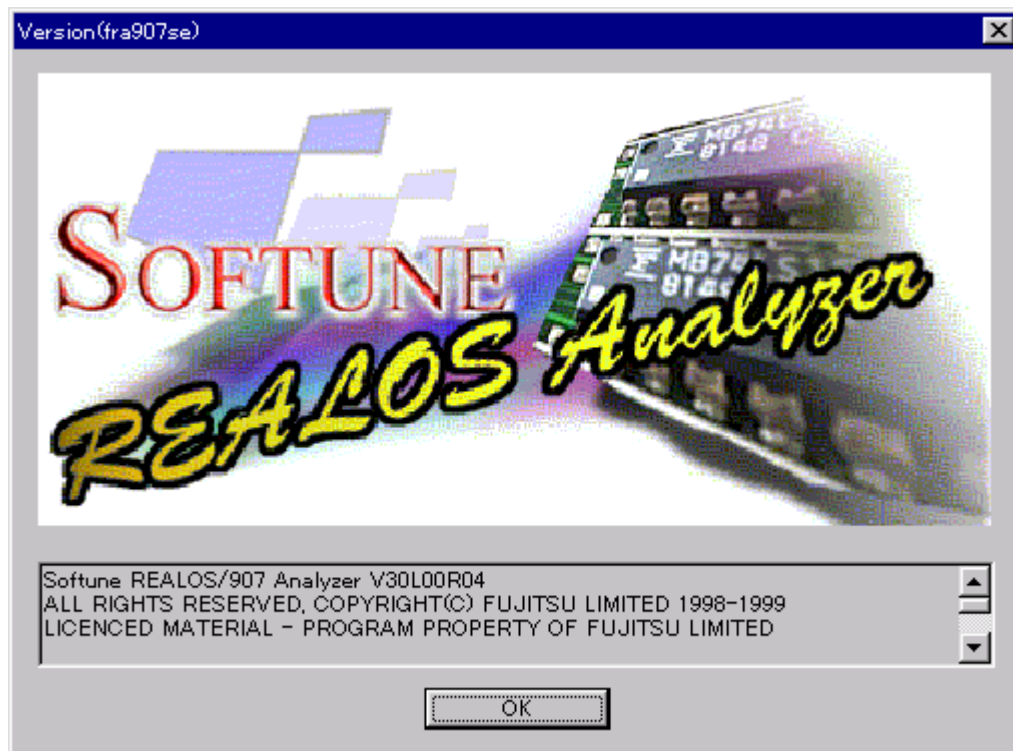
■ About fra907se

Figure 4.9-1 shows an example of the About fra907se dialog box.

The following are displayed as version information:

- Softtune REALOS/907 Analyzer version
- License and copyright
- REALOS version
- μ ITRON version

Figure 4.9-1 About fra907se Dialog Box



APPENDIX

The appendixes explain the restrictions on each REALOS Analyzer function, the structure of REALOS Analyzer files, error messages, and the sample programs supplied with the REALOS Analyzer.

APPENDIX A Restrictions

APPENDIX B Configuration

APPENDIX C Error Messages

APPENDIX D Sample Programs

APPENDIX A Restrictions

This appendix explains the restrictions on the Softune REALOS/907 analyzer.

■ Restrictions

If Softune Workbench is debugging a program with the monitor debugger, the monitoring function cannot be used.

In Windows 98, the source line jump function cannot display Softune Workbench in the foreground.

The REALOS Analyzer reads and writes to memory during Softune Workbench processing. For this reason, drawing with Softune Workbench when the REALOS Analyzer is used may be slower than drawing with Softune Workbench when the REALOS Analyzer is not used.

■ Processing Time and Size of Task Analysis Module

When the task analysis module is built into the REALOS Analyzer, processing time and size increase as shown in the following table.

Table A-1 Processing Time and Size of Task Analysis Module

Size	
Code	about 1100 bytes
Data	about 740 bytes (including 50 events in the trace buffer; 1 event=14 bytes)
Processing time (measured with the FMC-16LX simulator)	
System call issuance (no dispatching)	about 500 cycles
Dispatch to a task by issuing of a system call	about 1060 cycles
Conversion to the idle loop by issuing of a system call	about 1000 cycles
Dispatch from the idle loop to a task by issuing of a system call	about 1250 cycles
Dispatch from one task to another because of a time-out	about 960 cycles
Dispatch from the idle loop to a task because of a time-out	about 960 cycles
Execution of idle loop (when a system clock is generated)	about 160 cycles

Note: Depending on the conditions, the above data may be off a little.

■ Contents of Trace Buffer

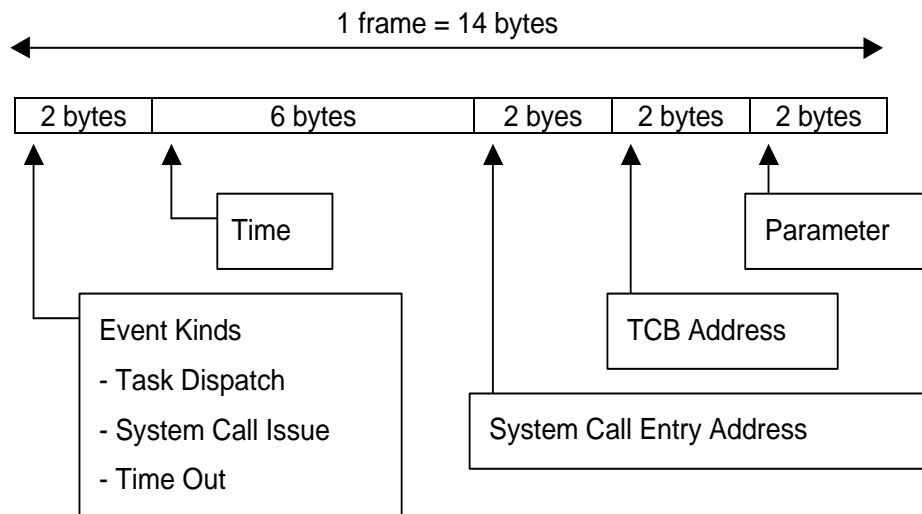
The trace buffer is a ring buffer. In this of structure, each time any of the following events occurs, one event is stored in the trace buffer. (Starting with the oldest data, data in the trace buffer is overwritten with new data.)

- Task dispatch (including dispatch to the idle loop)
- Issuing of a system call

- Time-out

One event in the trace buffer corresponds to one frame. One frame consists of 14 bytes (for example, 100 steps require 1,400 bytes).

Figure A-1 Contents of Trace Buffer



■ Time Measurement got Task Trace Figure

Since the emulator and monitor debuggers measure time based on the system clock used by REALOS, they can continue time measurement for a time that is less than the system clock. In time measurement by the simulator debugger, however, one system clock becomes the minimum time unit.

For example, when the system clock timer interrupt is set to 1,000 cycles, the timer value 1.000 also becomes 1,000 cycles. In the simulator debugger, values after the decimal point are ignored.

To use Softune REALOS/907 system clocks, build the system calls of the time management function into the REALOS Analyzer.

■ Number of Frames That Can Be Displayed in a Task Trace Figure

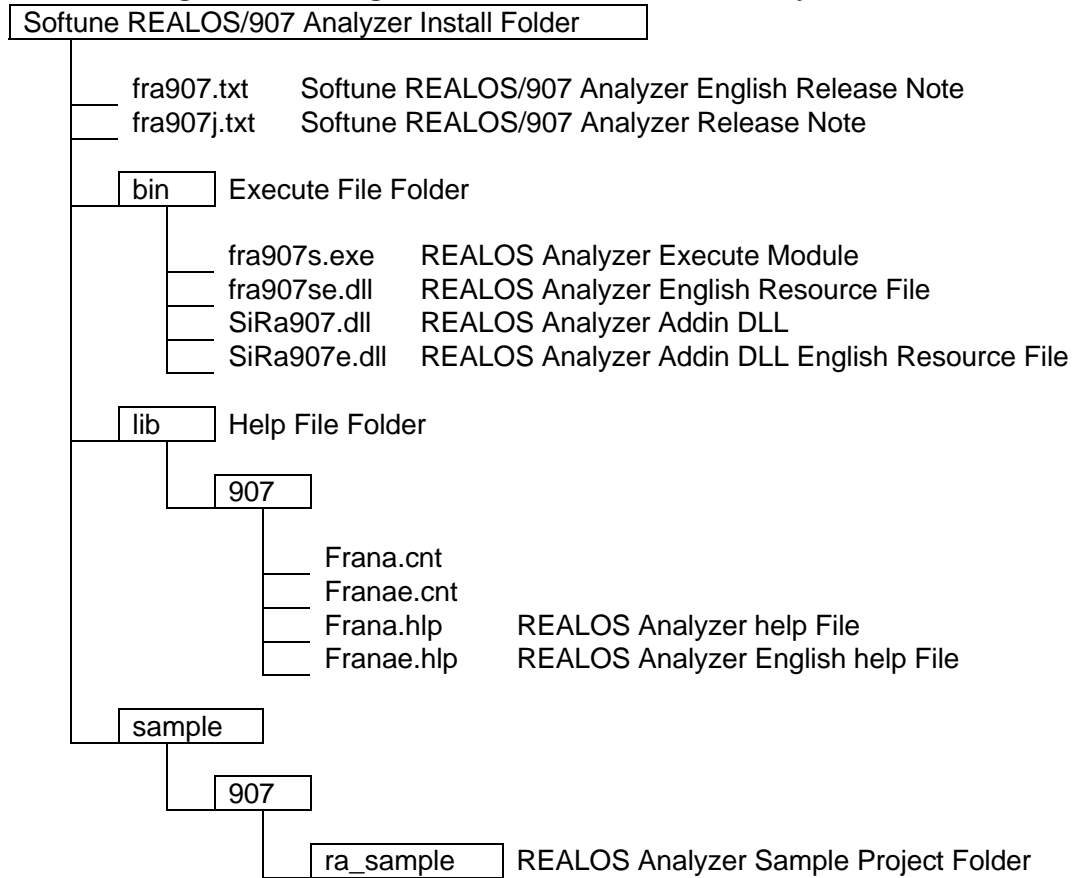
Since up to 2,048 frames can be displayed in the Task Trace Figure, set the trace buffer to 2,048 or less.

APPENDIX B Configuration

This appendix explains the configuration of Softune REALOS/907 analyzer files.

■ Configuration of Files

Figure B-1 Configuration of Softune REALOS Analyzer Files



APPENDIX C Error Messages

This appendix explains the Softune REALOS/907 analyzer error messages.

■ Memory File Errors (E46xxM)

E4601M	DATA FILE CANNOT BE OPENED
[Explanation]	The data file cannot be opened.
[Operator response]	The specified data file does not exist. If the file exists in another folder, browse to it from the file dialog box or check the state of the host machine.
E4602M	DATA FILE CANNOT BE ACCESSED
[Explanation]	An error occurred during access to the data file.
[Operator response]	Check the status of the host machine disk.
E4603	HELP FILE CANNOT BE ACCESSED
[Explanation]	The help file (frana.hlp) does not exist in the specified folder.
[Operator response]	When the specified folder is installed correctly, the help file is usually under \lib\907 in that folder. If the help file (frana.hlp) is not under \lib\907, the folder may have not been installed correctly. Reinstall the folder.
E460M	INVALID FILE FORMAT
[Explanation]	Correct data could not be read from the specified file.
[Operator response]	The format of the specified file is different. Specify the correct file.
E4610M	INSUFFICIENT MEMORY
[Explanation]	There is not enough free memory area on the host machine to execute commands.
[Operator response]	Quit unnecessary applications or drivers. Alternatively, increase memory and restart the REALOS Analyzer.

■ Memory File Errors (E45xxM)

E4510M	MEMORY ACCESS ERROR OR NO SYMBOL
[Explanation]	Memory read/write failed or no symbol exists.
[Operator response]	Check whether the memory area to which the REALOS data area and task analysis module data area were allocated can be read and written to. Also check whether the memory area was compiled and linked together with debug information.

■ Fatal Errors (F95xxMM)

F9501M	INSTALLATION INFORMATION CANNOT BE OBTAINED
--------	---

APPENDIX C Error Messages

[Explanation] Softune REALOS/907 analyzer installation information is invalid.

[Operator response] Possible causes are as follows:

1. Softune REALOS/907 analyzer installation information was damaged.
2. The Softune REALOS/907 analyzer is not installed correctly.
3. The installed file was moved.
4. For 3, return the file to its original location. In other cases, reinstall the analyzer.

F9502M COMMUNICATION ERROR

[Explanation] An internal error occurred during communication with Softune Workbench.

[Operator response] If this error occurs during data collection, collect the data again by clicking [Update] on the [Command] menu.

If this error occurs frequently, call your Fujitsu sales representative.

APPENDIX D Sample Programs

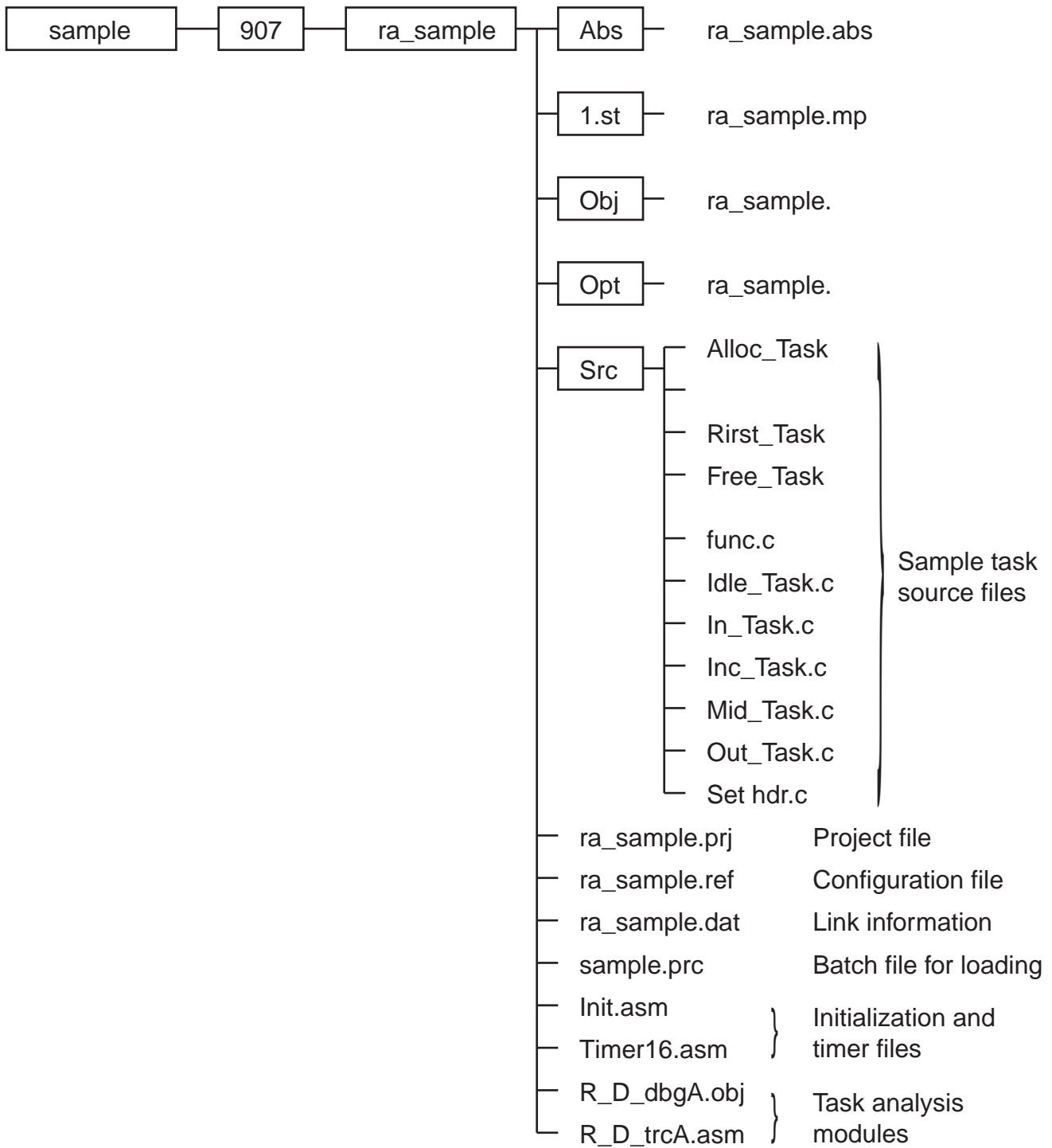
This appendix explains the sample programs supplied with the REALOS Analyzer.

■ Configuration of Sample Program Files

Sample programs are stored in the sample\907\ra_sample folder.

The ra_sample configuration is as follows:

APPENDIX D Sample Programs



■ Sample Program System Configuration

The number of objects used by sample programs and the operating environment are as follows:

○ Number of objects

- Number of tasks: 9
- Number of semaphores: 3

- Number of event flags: 3
- Number of mailboxes: 3
- Number of memory pools: 1
- Number of cyclically activated handlers: 2
- Number of alarm handlers: 1

○ **Task Names**

- First_Task
- Middle_Task
- Center_Task
- Alloc_Task
- In_Task
- Out_Task
- Free_Task
- Inc_Task
- Idle_Task

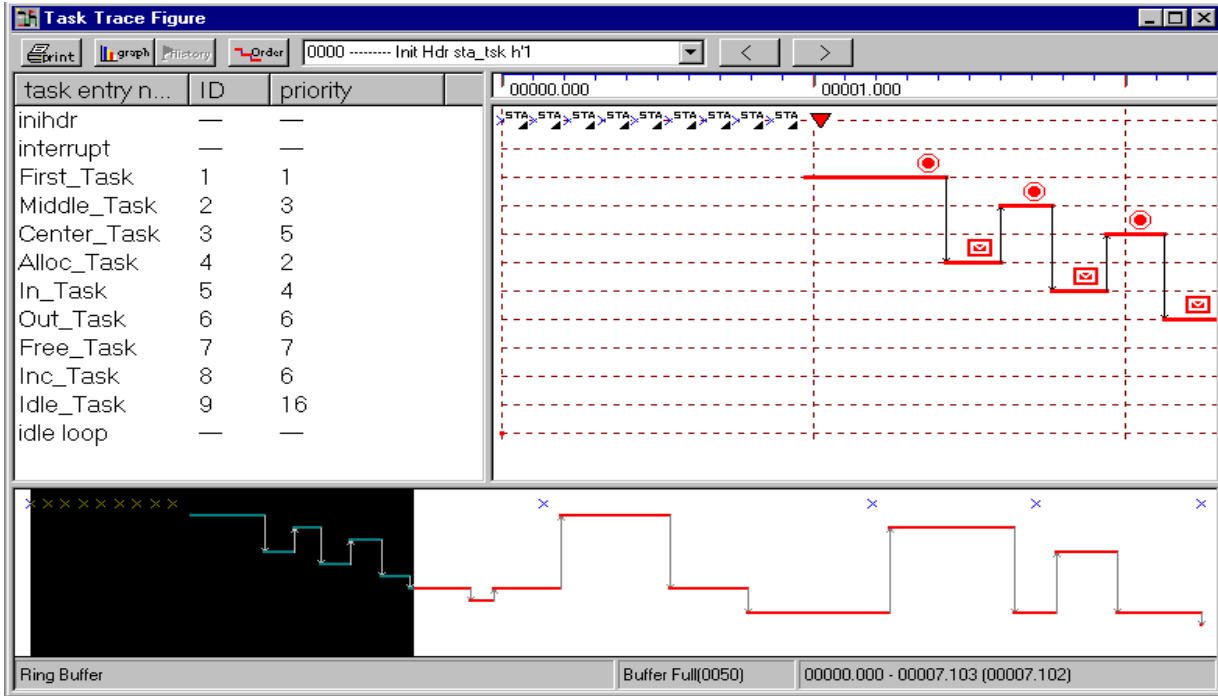
○ **Operating Environment**

- CPU: MB90550A Series
- Timer: INT 25 16-bit reload timer (reload value: 1000)
- ICE: MB2141A
- Support board: MB2176-01

■ **Example of Sample Program Execution**

The figure below is an example of displaying the Task Trace Figure when sample programs are executed for `ext_tsk()` in `Free_Task`.

APPENDIX D Sample Programs



INDEX

**The index follows on the next page.
This is listed in alphabetic order.**

Index

A	
about fra907se	119
alarm handler popup menu	65
ALL button.....	58
allocation section.....	20
ALMQ button	58
analysis result in graph	12
B	
buffer full mode	39
buffer size and data size	84
C	
check function for stack utilization.....	72
collecting data dialog box.....	26
collecting REALOS data.....	26
combo box.....	81
command menu	27
command, type of.....	35
communication time	26
complete view window	40
customization for task analysis module.....	17, 18
cyclic handler and alarm handler popup menu.....	65
D	
data file for REALOS Analyzer.....	43
data size	84
data, saving.....	43
debugger execution information.....	32
detail mode.....	108
detail stack information display	10
detail window.....	40
development environment.....	2
dialog box for task trace figure information	90
dispatch indication.....	79
dispatch, sorting sequentially	81
display mode, difference in	37
E	
emulator debugger	2
enlarge	85
event occurrence time	75
F	
feature for REALOS Analyzer.....	4
file menu	27
G	
graph.....	88
graph display.....	94
graph for running time.....	87
graph for stack utilization	73
graph in task trace tree window	96
graph, viewing.....	89
H	
help menu	29
help topic.....	118
history function.....	36
I	
icon	102
icon for task trace figure.....	75
icon in object window	59
idle loop.....	3
idle task.....	3
information dialog box.....	90
information dialog box, button in.....	90
information list.....	87
initialization information.....	32
initialization routine for task analysis module.....	21
initialize handler	21
J	
jump function.....	37
L	
list form of object window.....	54
M	
main toolbar	30
module configuration.....	3
module information	32
monitor debugger.....	3
monitor menu	28
monitoring	3, 14

monitoring and object trace setup dialog box 114
 monitoring setup 44
 monitoring, executing 44

N

note on development environment 2
 note on module configuration 3
 note on use of emulator debugger 2
 note on use of monitor debugger 3

O

object display 8
 object display window, standard function of 8
 object menu 27
 object selection dialog box 114
 object trace 12, 42
 object trace and status line display 116
 object trace setup dialog box 114
 object trace window, displaying 80
 object trace window, horizontal axis of 99
 object trace window, information displayed in 99
 object trace window, vertical axis of 99
 object window 53
 object window, function common to 53
 object window, icon in 59
 object window, list form of 54
 object window, tree form of 57
 operator command for softune workbench 7
 overview of task analysis module operation 16

P

PC information 32
 popup menu 65
 popup menu for cyclic handler and alarm handler 65
 popup menu for REALOS project window 50
 popup menu for task list 65
 popup menu for task trace figure window 85
 previously saved file, opening 43
 printing 80
 printing task trace figure 86

Q

queue list window 57

R

RDQ button 58
 REALOS Analyzer data file 43
 REALOS Analyzer feature 4

REALOS Analyzer, starting 24
 REALOS Analyzer, terminating 25
 REALOS data, collecting 26
 REALOS project window 6, 47
 REALOS project window popup menu 50
 REALOS project window, adding 34
 REALOS project window, basic function of 34
 REALOS project window, information displayed in 47
 reduce 85
 related event 91
 ring buffer mode 39
 running time graph 87
 running time graph window, displaying 80

S

search 85
 select task, object dialog box 114
 setup menu 29
 setup toolbar 31
 simple mode and detail mode 108
 softune workbench operator command 7
 sort function 36
 stack 68
 stack menu 28
 stack monitor window, information displayed in 105
 stack utilization analysis 10
 stack utilization check function, executing 72
 stack utilization graph 73
 stack utilization list window, basic function of 38
 stack utilization list, example of 69
 stack utilization list, item displayed in 69
 stack utilization, procedure for analyzing 38
 starting REALOS Analyzer 24
 status bar 32
 status bar for task trace figure window 84
 status line display 116
 system call being issued, task from which 94, 96
 system call sorted by task 93, 96
 system call sorted by time issued 94, 96
 system clock reset indication 79

T

task analysis module 3
 task analysis module customization 17, 18
 task analysis module initialization routine, adding 21
 task analysis module operation 16
 task analysis module overview 16
 task list popup menu 65

INDEX

task list window	40	task trace tree window, information displayed in ...	93
task not being selected	73	terminating REALOS Analyzer.....	25
task status monitor window, information displayed in 102		time axis window.....	40
task status, line type for indicating	78	TMRQ button	58
task trace.....	3, 12	toolbar	30
task trace data.....	39	toolbar for task trace figure window	80
task trace figure.....	39	toolbar for task trace tree window	95
task trace figure icon	75	trace buffer.....	3, 39, 116
task trace figure information dialog box.....	90	trace buffer, mode of.....	84
task trace figure window popup menu.....	85	trace menu.....	28
task trace figure window status bar.....	84	trace time	84
task trace figure window toolbar.....	80	tree form of object window	57
task trace figure window, information displayed in .	75		
task trace figure, basic operation of	40	U	
task trace figure, printing	86	update timing	37, 39
task trace information list (combo box).....	81		
task trace tree	42	W	
task trace tree window toolbar	95	window menu.....	29
task trace tree window, graph in	96	window toolbar	30

CM42-00326-1E

FUJITSU SEMICONDUCTOR • CONTROLLER MANUAL

F²MC-16L/16LX/16/16H/16F

μ ITRON2.01 SPECIFICATIONS COMPLIANT

SOFTUNE REALOS/907

ANALYZER MANUAL

July 1999 the first edition

Published **FUJITSU LIMITED** Electronic Devices

Edited Technical Communication Dept.

FUJITSU



* C M 4 2 - 0 0 3 2 6 - 1 E *

FUJITSU SEMICONDUCTOR F²MC-16L/16LX/16/16H/16F μ ITRON2.01 SPECIFICATIONS COMPLIANT SOFTUNE REALOS/907 ANALYZER MANUAL