HITACHI

SOFTWARE MANUAL

OPERATION

S10V CPMS DEBUGGER For Windows®

S10V

Programmable Controller

SVE-3-126（B）

SOFTWARE MANUAL

OPERATION

S10V CPMS DEBUGGER For Windows®

S10V

Programmable Controller

First Edition, July 2005, SVE-3-126(A) (out of print)
Second Edition, November 2012, SVE-3-126(B)

The contents of this publication may be revised without prior notice.

Printed in Japan.

BI<IC> (FL-MW2007)

SAFETY PRECAUTIONS

- Read this manual thoroughly and follow all the safety precautions and instructions given in this manual before operations such as system configuration and program creation.
- Keep this manual handy so that you can refer to it any time you want.
- If you have any question concerning any part of this manual, contact your nearest Hitachi branch office or service engineer.
- Hitachi will not be responsible for any accident or failure resulting from your operation in any manner not described in this manual.
- Hitachi will not be responsible for any accident or failure resulting from modification of software provided by Hitachi.
- Hitachi will not be responsible for reliability of software not provided by Hitachi.
- Make it a rule to back up every file. Any trouble on the file unit, power failure during file access or incorrect operation may destroy some of the files you have stored. To prevent data destruction and loss, make file backup a routine task.
- Furnish protective circuits externally and make a system design in a way that ensures safety in system operations and provides adequate safeguards to prevent personal injury and death and serious property damage even if the product should become faulty or malfunction or if an employed program is defective.
- If an emergency stop circuit, interlock circuit, or similar circuit is to be formulated, it must be positioned external to the programmable controller. If you do not observe this precaution, equipment damage or accident may occur when this programmable controller becomes defective.
- Before changing the program, generating a forced output, or performing the RUN, STOP, or like procedure during an operation, thoroughly verify the safety because the use of an incorrect procedure may cause equipment damage or other accident.
- This manual contains information on potential hazards that is intended as a guide for safe use of this product. The potential hazards listed in the manual are divided into four hazard levels of danger, warning, caution, and notice, according to the level of their severity. The following are definitions of the safety labels containing the corresponding signal words DANGER, WARNING, CAUTION, and NOTICE.

⚠ DANGER : This safety label identifies precautions that, if not heeded, will result in death or serious injury.

⚠ WARNING : Identifies precautions that, if not heeded, could result in death or serious injury.

⚠ CAUTION : Identifies precautions that, if not heeded, could result in minor or moderate injury.

NOTICE : This safety label without a safety alert symbol identifies precautions that, if not heeded, could result in property damage or loss not related to personal injury.

Failure to observe any of the ⚠CAUTION and *NOTICE* statements used in this manual could also lead to a serious consequence, depending on the situation in which this product is used. Therefore, be sure to observe all of those statements without fail.

The following are definitions of the phrases "serious injury," "minor or moderate injury," and "property damage or loss not related to personal injury" used in the above definitions of the safety labels.

Serious injury: Is an injury that requires hospitalization for medical treatment, has aftereffects, and/or requires long-term follow-up care. Examples of serious injuries are as follows: vision loss, burn (caused by dry heat or extreme cold), electric-shock injury, broken bone, poisoning, etc.

Minor or moderate injury: Is an injury that does not require either hospitalization for medical treatment or long-term follow-up care. Examples of minor or moderate injuries are as follows: burn, electric-shock injury, etc.

Property damage or loss not related to personal injury: Is a damage to or loss of personal property. Examples of property damages or losses not related to personal injury are as follows: damage to this product or other equipment or their breakdown, loss of useful data, etc.

The safety precautions stated in this manual are based on the general rules of safety applicable to this product. These safety precautions are a necessary complement to the various safety measures included in this product. Although they have been planned carefully, the safety precautions posted on this product and in the manual do not cover every possible hazard. Common sense and caution must be used when operating this product. For safe operation and maintenance of this product, establish your own safety rules and regulations according to your unique needs. A variety of industry standards are available to establish such safety rules and regulations.

1. ⚠ Hazard Warning Statements

The following are the hazard warning statements contained in this manual.

1.1 NOTICE Statement

(chapter 1, page 1-4)

NOTICE

- This product allows rewriting programs and internal-register values while the PCs is running. However, you should notice that careless rewriting may lead to a serious accident, such as breakdown of the equipment. If any of such items needs to be rewritten, be sure to check the condition of the equipment first. You may rewrite it only when you are sure that the equipment has no problem.
- When loading task data into a CMU module, CPMS Debugger writes the data to internal flash memory of the CMU. Note that writing different items of data concurrently to the same address of internal flash memory of the CMU may damage the data. Therefore, do not send data to the CMU module concurrently from CPMS Debugger and another writing tool (e.g., HI-FLOW system, RPDP, NX/Tools-S10V system).

(chapter 2, page 2-2)

NOTICE

Before installing the CPMS debugger, be sure to exit all the currently open Windows® programs. Do not forget to exit anti-virus software and other memory-resident programs. If you install the CPMS debugger without exiting such programs, an error may occur during installation. If such an error occurs, first uninstall the CPMS debugger as directed in “2.2 Uninstalling,” exit all the Windows® programs, and then install the CPMS debugger again.

(chapter 2, page 2-4)

NOTICE

- If Windows® opens a window during the uninstall process to display the question “Remove Shared File?”, click the No button to retain shared files.
- When you want to reinstall the CPMS Debugger System, be sure to perform an uninstall and then perform an install.

This Page Intentionally Left Blank

This manual provides information on the following program products:

<Program products>
S-7895-07, S10V CPMS DEBUGGER SYSTEM, 01-03
S-7895-62, S10V CPMS DEBUGGER SYSTEM, 01-00

Revision record

Revision No.	Revision record (revision details and reason for revision)	Month, Year	Remarks
A	First edition	July 2005	
B	Windows® 7 (32-bit) operating system is newly supported.	November 2012	

In addition to the above changes, all the unclear descriptions and typographical errors found are also corrected without prior notice.

PREFACE

Thank you for purchasing Hitachi's S10V CPMS Debugger System.

The CPMS Debugger system runs on a personal computer. This system registers, initiates, and deletes the tasks that operate on programmable controllers (PCs), monitors the bit status at each memory address, and displays the status of PCs.

This document describes how to operate the S10V CPMS Debugger System. It covers the following versions of the ladder chart system:

P.P. model	System name and version	Supported OS
S-7895-07	S10V CPMS Debugger System for Windows®, 01-01	Windows® 2000/XP
S-7895-62	S10V CPMS Debugger System for Windows®, 01-00	Windows® 7 (32-bit)

<Related manuals>

- S10V SOFTWARE MANUAL CPMS GENERAL DESCRIPTION AND MACRO SPECIFICATIONS (Manual number SVE-3-201)
- S10V SOFTWARE MANUAL OPERATION RPDP/S10V For Windows® (Manual number SVE-3-133)

<Trademarks>

- Microsoft® Windows® operating system, Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system, Microsoft® Windows® 7 (32-bit) operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.
- Ethernet® is a registered trademarks of Xerox Corp.

<Definitions of Terms>

PCs: An abbreviation of Programmable Controllers.
This is a general term for PLC such as the S10V, S10α and S10mini series.

<Note for storage capacity calculations>

- Memory capacities and requirements, file sizes and storage requirements, etc. must be calculated according to the formula 2^n. The following examples show the results of such calculations by 2^n (to the right of the equals signs).
  1 KB (kilobyte) = 1024 bytes
  1 MB (megabyte) = 1,048,576 bytes
  1 GB (gigabyte) = 1,073,741,824 bytes
- As for disk capacities, they must be calculated using the formula 10^n. Listed below are the results of calculating the above example capacities using 10^n in place of 2^n.
  1 KB (kilobyte) = 1000 bytes
  1 MB (megabyte) = 1000^2 bytes
  1 GB (gigabyte) = 1000^3 bytes

CONTENTS

FIGURES

TABLES

1 BEFORE USE

This manual is intended for those who perform programming on Windows® personal computers.

1.1 System Overview

The S10V CPMS Debugger System for Windows® (hereinafter called “CPMS Debugger”) enables the user to register, initiate, and monitor tasks designed for the S10V programmable controllers through operations equivalent to those for general Windows® applications.

1.2 Required Hardware and Software

The following hardware and software are required for the use of the CPMS debugger:

- Personal computer (main unit) containing a Pentium 300 MHz or faster CPU, or a 1 GHz or faster CPU (when Windows® 7 (32-bit version) is used)
- Display having a resolution of 800 × 600 dots (SVGA) or higher
- Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system or Microsoft® Windows® 7 (32-bit) operating system
- At least 64 MB of RAM (when Windows® 2000 is used)
- At least 128 MB of RAM (when Windows® XP is used)
- At least 1 GB of RAM (when Windows® 7 (32-bit) is used)
- At least 10 MB of free hard disk space
- S10V series ladder processor unit (LPU) and computer mode unit (CMU)
- S10V series power supply and backboard
- Cable for connecting the personal computer to the CMU or ET.NET module (LQE720) (10BASE-T or 100BASE-T twisted pair cross cable with RJ-45 modular connectors)
- RI/O stations, power supplies, backboards, cards, and wiring cables, as required

1.3 Precautions on Combined Use of CPMS Debugger and RPDP/S10V

When CPMS Debugger is combined with RPDP/S10V, CPMS Debugger must be used only to reference the status of tasks created using RPDP/S10V or abort said tasks. To create and modify programs, RPDP/S10V must be used.
A task registered by CPMS Debugger does not conform to the development environment of RPDP/S10V. Therefore, if RPDP/S10V executes the svrpl command, any task registered by the CPMS debugger will be invalidated.
Carefully note that RPDP/S10V cannot manage the tasks newly registered by CPMS Debugger.

1.4 Precautions on Using NX/HOST-S10V

When the conventional NX/HOST-S10 (S10mini series) is used, a user task created and registered in a programmable controller (PCs) by CPMS Debugger is not deleted when a NX/HOST-S10 system file is sent to the PCs, unless the areas used to store the system file and user task overlap.
Conversely, when the NX/HOST-S10V (S10V series) is used, a user task created and registered in a PCs by CPMS Debugger is deleted when a NX/HOST-S10V system file is sent to the PCs, even if the areas used to store the system file and user task do not overlap.

NOTICE

- This product allows rewriting programs and internal-register values while the PCs is running.　However, you should notice that careless rewriting may lead to a serious accident, such as breakdown of the equipment.　If any of such items needs to be rewritten, be sure to check the condition of the equipment first.　You may rewrite it only when you are sure that the equipment has no problem.
- When loading task data into a CMU module, CPMS Debugger writes the data to internal flash memory of the CMU.　Note that writing different items of data concurrently to the same address of internal flash memory of the CMU may damage the data.　Therefore, do not send data to the CMU module concurrently from CPMS Debugger and another writing tool (e.g., HI-FLOW system, RPDP, NX/Tools-S10V system).

Users of this product must have adequate knowledge of the Windows® environment and user interface.　This system conforms to the Windows® standard.　This manual is prepared for users who are familiar with the basic Windows® operating procedures.

- If you use a personal computer with the suspend feature, disable the feature. If the suspend feature comes into operation during execution of this system, the system may malfunction.
- An inadequate free memory space available in RAM may cause an application error.　In the event of such an error, check the amount of free memory space available.　If it is found inadequate, add more RAM.

2 SYSTEM INSTALLATION

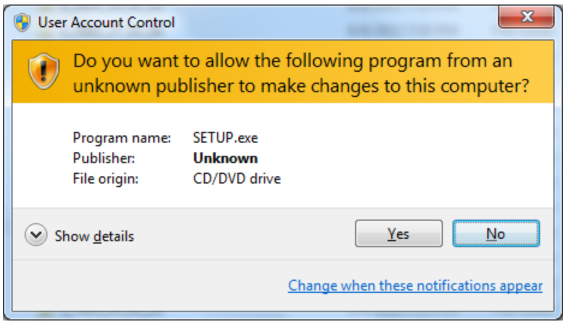
2.1 Installing

To install the CPMS DEBUGGER, you must execute the setup program that is stored in the CPMS DEBUGGER DISK1 folder on the CD.
Double-click “setup. exe” that is stored in the DISK1 folder on the S10V CPU link system CD.
Since no window opens upon completion of installation, attach a shortcut to the desktop as needed.
Click the Start button and choose [(All) Programs] – [Hitachi S10V] – [S10V CPMS DEBUGGER SYSTEM] – [S10V CPMS DEBUGGER SYSTEM] from the [Start] menu on the Windows® screen. Click and hold the right mouse button on the [S10V CPMS DEBUGGER SYSTEM] and move the pointer to the desktop. Then, choose [Copy Here] from the pop-up menu.

NOTICE
Before installing the CPMS debugger, be sure to exit all the currently open Windows® programs. Do not forget to exit anti-virus software and other memory-resident programs. If you install the CPMS debugger without exiting such programs, an error may occur during installation. If such an error occurs, first uninstall the CPMS debugger as directed in “2.2 Uninstalling,” exit all the Windows® programs, and then install the CPMS debugger again.

<Notes on installing in Windows® 7 (32-bit)>

Installing the CPMS Debugger System in Windows® 7 (32-bit) operating system requires prior logging onto the operating system with an appropriate Administrator account, which is the Administrator account first created in the initial condition of your personal computer. When you have so logged on, you can then double-click "setup.exe" that is stored in the DISK 1 folder on the CPMS Debugger System CD. When "setup.exe" is started, the dialog box as shown below will appear. Click the Yes button to continue the execution of the setup program.



User Account Control

Do you want to allow the following program from an unknown publisher to make changes to this computer?

Program name: SETUP.exe
Publisher: **Unknown**
File origin: CD/DVD drive

The CPMS Debugger System cannot be installed on a per-user basis. To install the CPMS Debugger System successfully, the user must first log onto the operating system with an appropriate Administrator account, which is the Administrator account first created in the initial condition of your personal computer.
The CPMS Debugger System may not be installed properly in any of the following cases: 1) administrator permission is acquired by using User Account Control(*) with a standard user account and 2) logon is made with an Administrator account that has been created using User Account Control with a standard user account.
If you make a logon with a user account that is different from the one you have used for the installation of the CPMS Debugger System, the installed program may be missing from the program menu displayed. In this case, you should perform the following series of steps: 1) make a logon again with the Administrator account first created in the initial condition of your personal computer; 2) uninstall the installed program; and 3) install the program again.
When you want to create a new account, be sure to make a logon with an Administrator account. Do not use User Account Control at that time.

(*) User Account Control is a Microsoft Windows feature that temporarily grants administrative rights to standard user accounts.

A message reporting a read-only file detected may be displayed during the reinstallation of the CPMS Debugger System. In this case, click the Yes button to set off overwriting.

2.2 Uninstalling

The existing CPMS Debugger System needs to be uninstalled when, for instance, you want to upgrade it. The procedure required for uninstalling it is as follows:

(1) Uninstalling from Windows® 2000

Click on Start button on your Windows desktop and choose [Settings] – [Control Panel]. When the Control Panel opens, double-click on [Add/Remove Programs]. Then, choose "S10V CPMS DEBUGGER SYSTEM" in the [Change or Remove Programs] tab and click the Change/Remove button. When the [Confirm File Deletion] dialog box appears, click the Yes button.

(2) Uninstalling from Windows® XP

Click on Start button on your Windows desktop and choose ([Settings] –)[Control Panel]. When the Control Panel opens, double-click on [Add/Remove Programs]. Then, choose "S10V CPMS DEBUGGER SYSTEM" in the [Change or Remove Programs] tab and click the Change/Remove button. When the [Confirm File Deletion] dialog box appears, click the Yes button.

(3) Uninstalling from Windows® 7 (32-bit)

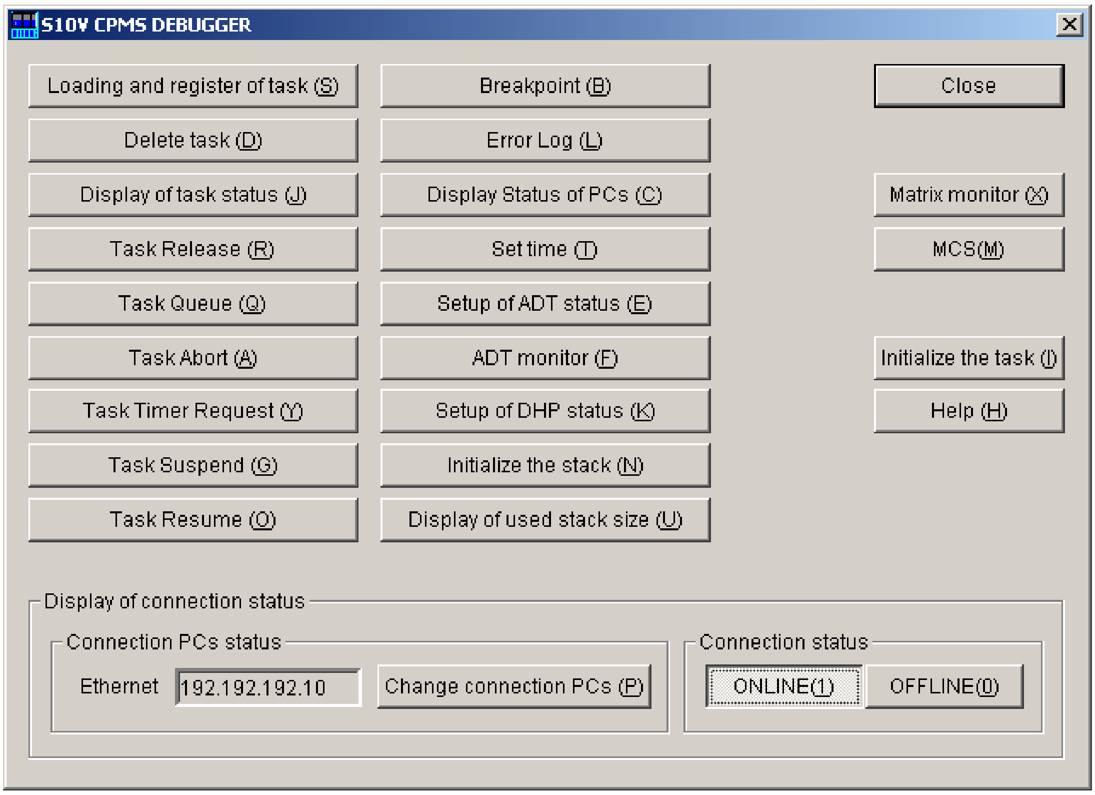
Click on Start button on your Windows desktop and choose [Control Panel]. When the Control Panel opens, click [Programs and features]. Then, select "S10V CPMS DEBUGGER SYSTEM" and click Uninstall/Change button. When the [Confirm File Deletion] dialog box appears, click the Yes button.

NOTICE

- If Windows® opens a window during the uninstall process to display the question "Remove Shared File?", click the No button to retain shared files.
- When you want to reinstall the CPMS Debugger System, be sure to perform an uninstall and then perform an install.

2.3 Starting the System

To start CPMS Debugger, click the Start button on the Windows® screen, select [(All) Programs], select [Hitachi S10V], and then double-click [S10V CPMS DEBUGGER SYSTEM]. After CPMS Debugger starts up, the window below appears. To use a CPMS Debugger function, click the button corresponding to the desired function.



Ethernet 192.192.192.10

Figure 2-1 [CPMS DEBUGGER] Window

(1) [Ethernet] box

This box shows the IP address of the connected programmable controller (PCs).

(2) Change connection PCs button

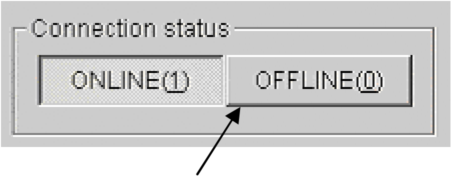
Clicking this button opens the [Communication type] window, which is used to specify the type of communication between the personal computer and a PCs.

When the type of communication is set, the connection status of the relevant PCs becomes “ONLINE”.

(3) Connection status indicators/buttons

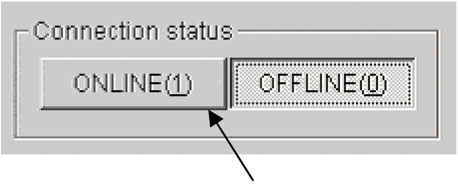
These indicators/buttons indicate and switch the connection status of the currently connected PCs. To switch the connection status, click the ONLINE or OFFLINE button.

<When connection status is "ONLINE">



To change the status to "offline," click the OFFLINE button.

<When connection status is "OFFLINE">



To change the status to "online," click the ONLINE button.

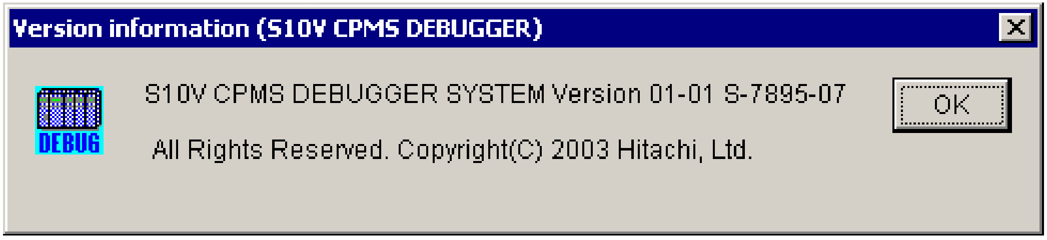
(4) Help button

Clicking this button or pressing the [F1] key on the keyboard displays online help for CPMS Debugger.

(5) Version information

Right-clicking on the icon at the upper-left corner or on the title bar of the [CPMS DEBUGGER] window opens the [Version information] window as shown below. To close this window, click the OK button.

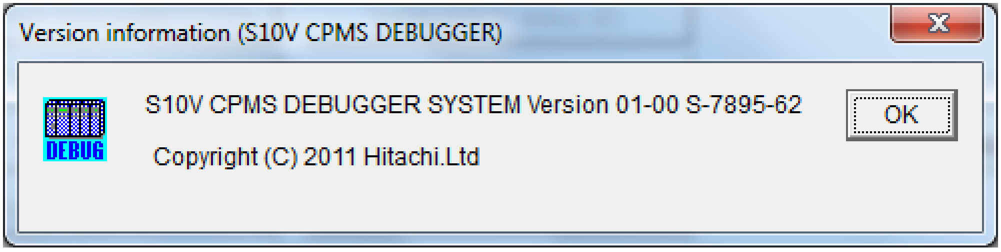
(P.P. model: S-7895-07)



S10V CPMS DEBUGGER SYSTEM Version 01-01 S-7895-07

All Rights Reserved. Copyright(C) 2003 Hitachi, Ltd.

(P.P. model: S-7895-62)



S10V CPMS DEBUGGER SYSTEM Version 01-00 S-7895-62

Copyright (C) 2011 Hitachi.Ltd

2.4 Ending the System

To exit CPMS Debugger, click the ☒ button at the upper-right corner or the Close button of the [CPMS DEBUGGER] window (shown in Figure 2-1).

This Page Intentionally Left Blank

3 COMMANDS

3.1 System of Commands

CPMS Debugger has a system of commands as shown below.
Sections 3.2 and later summarize the commands. For details, see online help.

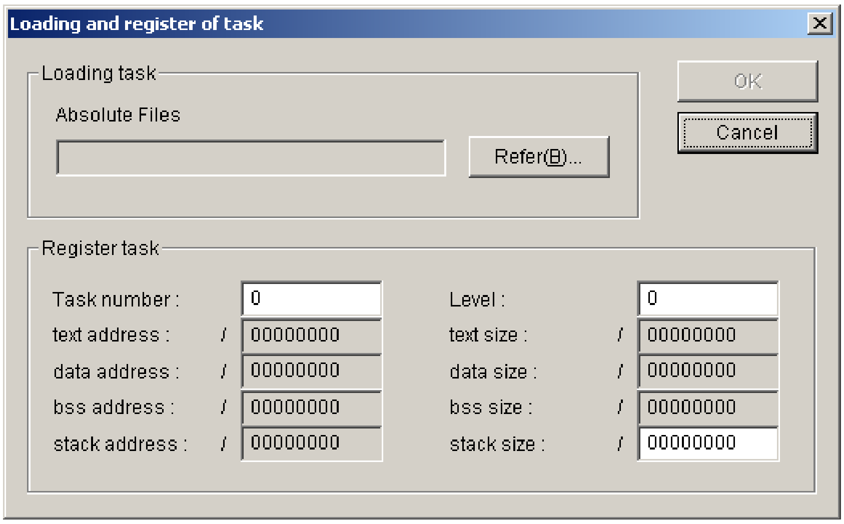
Commands

- Loading and register of task
- Delete task
- Display of task status
- Task Release
- Task Queue
- Task Abort
- Task Timer Request
- Task Suspend
- Task Resume
- Break point
- Error Log
- Display Status of PCs
- Set time
- Setup of ADT status
- ADT monitor
- Setup of DHP status
- Initialize the stack
- Display of used stack size
- Matrix monitor
- MCS
- Change connection PCs
- Initialize the task

3.2 Task Loading and Register

Function: The “Loading and register of task” command opens the window to load and register a user-created task into the currently connected programmable controller (PCs).

Procedure: Click the Loading and register of task button in the [CPMS DEBUGGER] window.



Task number :		0	Level :		0
text address :	/	00000000	text size :	/	00000000
data address :	/	00000000	data size :	/	00000000
bss address :	/	00000000	bss size :	/	00000000
stack address :	/	00000000	stack size :	/	00000000

Figure 3-1 [Loading and register of task] Window

(1) [Absolute Files] box

This box shows the task file to be loaded into the PCs. To specify the task file, click the Refer button.

(2) Refer button

Clicking this button opens the window as shown below. Specify the task file to be loaded into the PCs in this window.

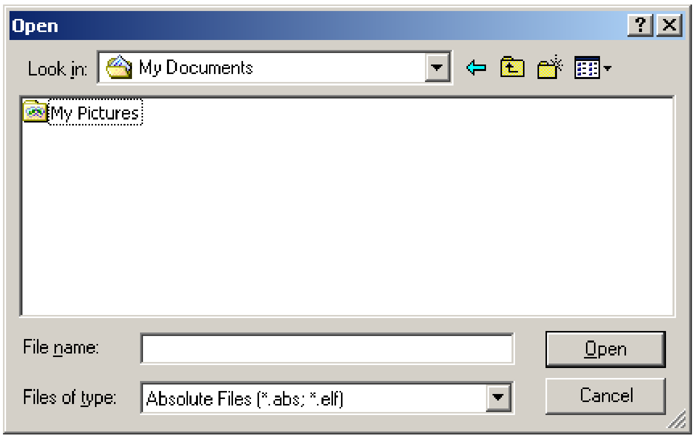


Figure 3-2 [Open] Window

(3) [Task number] box

This box is used to specify the task number of the task to be registered. For a user task, specify a task number from 1 to 224. For a system task, specify a task number from 225 to 229.

(4) [Level] box

This box is used to specify the initial execution level of the task to be registered. For a user task, specify a level from 4 to 27. For a system task, specify a level from 0 to 31.

(5) [text address] box

This box shows the start address of the area in which to load the text section of the specified task file.

(6) [text size] box

This box shows the text section size (in bytes) of the specified task file.

(7) [data address] box

This box shows the start address of the area in which to load the data section of the specified task file.

(8) [data size] box

This box shows the data section size (in bytes) of the specified task file.

(9) [bss address] box

This box shows the start address of the bss section to be used by the specified task file.

(10) [bss size] box

This box shows the size of the bss section (in bytes) to be used by the specified task file.

(11) [stack address] box

This box shows the start address of the stack section to be used by the specified task file.

(12) [stack size] box

This box shows the size of the stack section (in bytes) to be used by the specified task file. For the stack size, specify a multiple of 0x1000 from 0x00001000 to 0x00800000. If the specified value is not a multiple of 0x1000, the value will be rounded to the nearest multiple of 0x1000.

(13) OK button

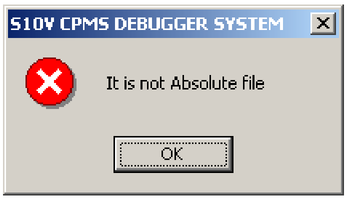
Clicking this button loads and registers the task according to user specifications. When loading and registration have been successfully carried out, the [CPMS DEBUGGER] window will return.

(14) Cancel button

Clicking this button cancels task loading and registration, and returns to the [CPMS DEBUGGER] window.

<Notes>

CPMS Debugger only supports executable and linking format (ELF) task files. Note that specifying a non-ELF task file will display the following error message:



S10V CPMS DEBUGGER SYSTEM

It is not Absolute file

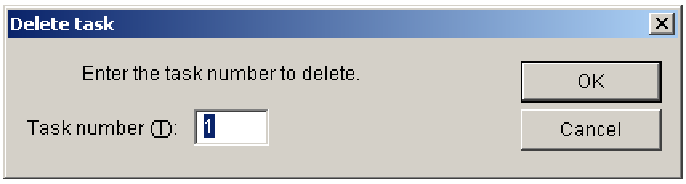
Figure 3-3 Error Message Box

The start address of the task file to be loaded must always be aligned on a page boundary. If the start address (of the text section) of the task file to be loaded is not aligned on a page boundary, the task file cannot be loaded, and the OK button is shaded.

3.3 Delete Task

Function: The “Delete task” command opens the window to delete a task.

Procedure: Click the Delete task button in the [CPMS DEBUGGER] window.



Delete task

Enter the task number to delete.

Task number (I): 1

Figure 3-4 [Delete task] Window

(1) [Task number] box

This box is used to specify the task number of the task to be deleted.

(2) OK button

Clicking this button deletes the task of the specified task number. After the task is deleted, the [CPMS DEBUGGER] window appears.

(3) Cancel button

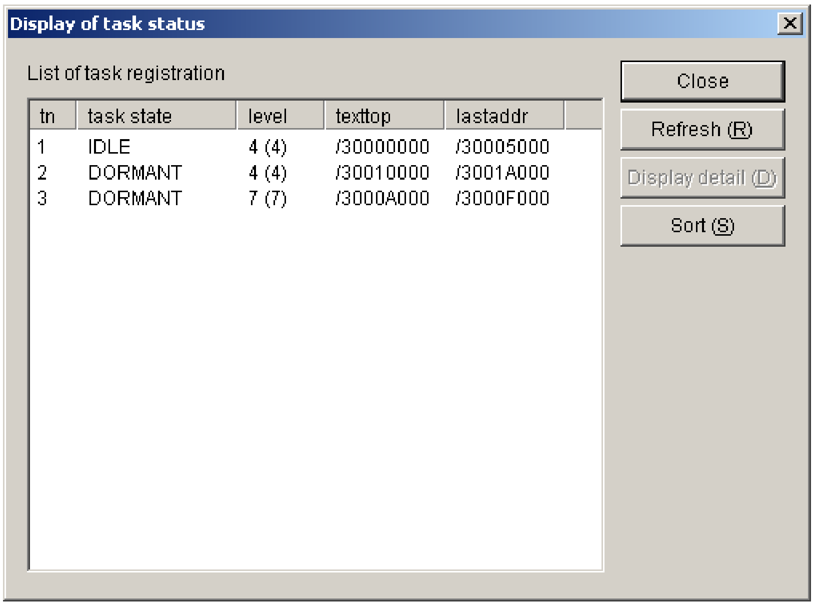
Clicking this button cancels task deletion, and returns to the [CPMS DEBUGGER] window.

3.4 Display of Task Status

<[Display of task status] window>

Function: The “Display of task status” command opens the window to display the status of tasks.

Procedure: Click the Display of task status button in the [CPMS DEBUGGER] window.



tn	task state	level	texttop	lastaddr
1	IDLE	4 (4)	/30000000	/30005000
2	DORMANT	4 (4)	/30010000	/3001A000
3	DORMANT	7 (7)	/3000A000	/3000F000

Figure 3-5 [Display of task status] Window

(1) [List of task registration] box

This box lists information about currently registered tasks.

The information listed is as follows:

No.	Item	Description
1	tn	Task number
2	task state	Task state (*)
3	level	Execution level (initial execution level)
4	texttop	First address of task
5	lastaddr	End address of task

(*) Task states

No.	Task state	Description
1	DORMANT	Task initiation is suspended.
2	IDLE	The task is awaiting initiation.
3	READY	The task is being executed or awaiting execution.
4	WAIT	The task is awaiting an event.
5	SUSPENDED	Task execution is suspended.

(2) Close button

Clicking this button closes the [Display of task status] window, and returns to the [CPMS DEBUGGER] window.

(3) Refresh button

Clicking this button updates the information shown in the [List of task registration] box.

(4) Display detail button

Clicking this button opens the [Display of task status] (detail) window, which displays detailed information about the task with the task number selected in the [List of task registration] box.

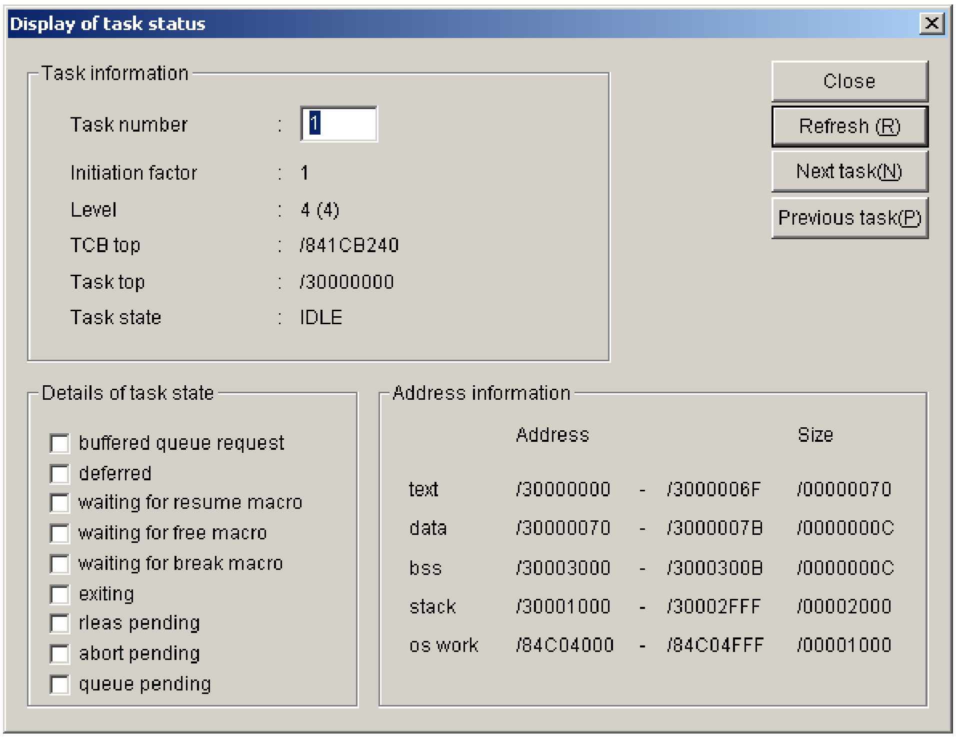
(5) Sort button

Clicking this button sorts the task information shown in the [List of task registration] box in order of the start addresses of tasks. The sorting order alternately switches between ascending and descending order each time this button is clicked.

<[Display of task status] (detail) window>

Function: The [Display of task status] (detail) window displays the detailed status of a task.

Procedure: Click the Display detail button in the [Display of task status] window.



Display of task status

Task information

Task number : 1

Initiation factor : 1

Level : 4 (4)

TCB top : /841CB240

Task top : /30000000

Task state : IDLE

Details of task state

□ buffered queue request
□ deferred
□ waiting for resume macro
□ waiting for free macro
□ waiting for break macro
□ exiting
□ rleas pending
□ abort pending
□ queue pending

Address information

	Address			Size
text	/30000000	-	/3000006F	/00000070
data	/30000070	-	/3000007B	/0000000C
bss	/30003000	-	/3000300B	/0000000C
stack	/30001000	-	/30002FFF	/00002000
os work	/84C04000	-	/84C04FFF	/00001000

Figure 3-6 [Display of task status] (detail) Window

(1) [Task number] box

This box shows the task number (in decimal notation) of the task for which status information is currently displayed in this window.

To display the status information about a different task, enter the task number of that task in this box, and then press the [Enter] key on the keyboard or click the Refresh button.

The range of task numbers that can be displayed or entered in this box is 1 to 229.

(2) Next task button

Clicking this button displays the latest detailed information about the registered task that follows (in order of registration) the specified task.

Unregistered tasks are skipped.

(3) Previous task button

Clicking this button displays the latest detailed information about the registered task that precedes (in order of registration) the specified task.
Unregistered tasks are skipped.

(4) [Initiation factor] field

This field shows a value that indicates the initiation factor of the specified task.
If the specified task is not registered, this field shows “/00000000”.

(5) [Level] field

This field shows the execution level of the specified task (The parenthesized value is the default execution level.)
If the specified task is not registered, this field shows “0 (0)”.

(6) [TCB top] field

This field shows the physical start address of the TCB information of the specified task.

(7) [Task top] field

This field shows the start address of the specified task.
If the specified task is not registered, this field shows “/00000000”.

(8) [Task state] field

This field shows the state of the specified task.
One of the following states is shown:

No.	State	Description
1	DORMANT	Task initiation is suspended.
2	IDLE	The task is awaiting initiation.
3	READY	The task is being executed or awaiting execution.
4	WAIT	The task is awaiting an event.
5	SUSPENDED	Task execution is suspended.
6	NON-EXISTENT	The task is not registered.

(9) [Details of task state] group box

The items in this group box indicate detailed status information (status bit information) about the specified task. The status bit information is denoted by the ON/OFF state of the check box for each item (i.e., parameter).

One or more parameters may be set to ON.

(10) [Address information] group box

The fields in this group box show the addresses and sizes of areas allocated to the specified task. If the specified task is not registered, the [Address] and [Size] fields show "/00000000".

(11) Close button

Clicking this button closes the [Display of task status] (detail) window, and returns to the [CPMS DEBUGGER] window.

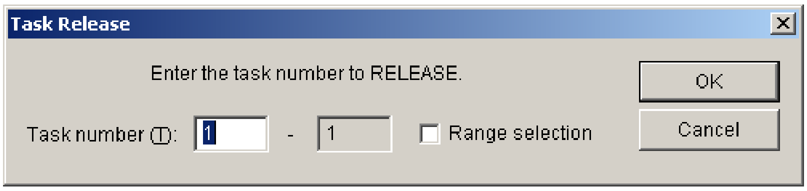
(12) Refresh button

Clicking this button updates the detailed status information displayed for the task specified by the task number entered in the [Task number] box.

3.5 Task Release

Function: The “Task Release” command opens the window to release a task (or tasks) from the Dormant state (initiation-suspended state).

Procedure: Click the Task Release button in the [CPMS DEBUGGER] window.



Task Release

Enter the task number to RELEASE.

Task number (T): 1 - 1 □ Range selection

Figure 3-7 [Task Release] Window

(1) [Task number] box
This box is used to enter the task number of the task to be released from the Dormant state. To enable a range specification of task numbers, select the [Range selection] check box.

(2) [Range selection] check box
Selecting this check box enables a range specification of task numbers. Even if an invalid range is specified, no error message will appear. To confirm the result of command execution, open the [Display of task status] window, and confirm that “READY” is displayed as the task state for each task number in the specified range of task numbers.

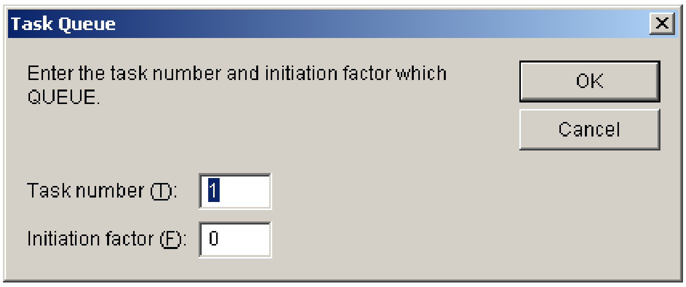
(3) OK button
Clicking this button releases the specified task from the Dormant state. When the [Range selection] check box is selected and a range of task numbers specified, clicking this button releases the tasks with task numbers in the specified range from the Dormant state.
After the tasks are released, the [CPMS DEBUGGER] window appears.

(4) Cancel button
Clicking this button cancels the releasing of tasks from the Dormant state, and returns to the [CPMS DEBUGGER] window.

3.6 Task Queue

Function: The “Task Queue” command opens the window to request initiation of a task.

Procedure: Click the Task Queue button in the [CPMS DEBUGGER] window.



Enter the task number and initiation factor which QUEUE.

Task number (T): 1

Initiation factor (F): 0

Figure 3-8 [Task Queue] Window

(1) [Task number] box

This box is used to enter the task number of the task requested to be initiated.

(2) [Initiation factor] box

This box is used to enter the value that indicates the initiation factor of the task requested to be initiated.

(3) OK button

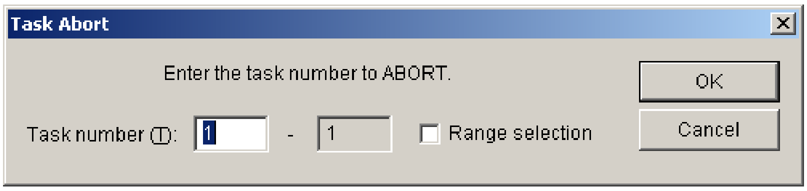
Clicking this button requests initiation of the task with the specified task number. After initiation is requested, the [CPMS DEBUGGER] window appears.

(4) Cancel button

Clicking this button cancels the request for task initiation, and returns to the [CPMS DEBUGGER] window.

3.7 Task Abort

Function: The "Task Abort" command opens the window to suspend initiation of a task (or tasks).
Procedure: Click the Task Abort button in the [CPMS DEBUGGER] window.


Task Abort

Enter the task number to ABORT.

Task number (T): 1 - 1 □ Range selection

Figure 3-9 [Task Abort] Window

(1) [Task number] box
This box is used to enter the task number of the task for which initiation is to be suspended. To enable a range specification of task numbers, select the [Range selection] check box.

(2) [Range selection] check box
Selecting this check box enables a range specification of task numbers. Even if an invalid range is specified, no error message will appear. To confirm the result of command execution, open the [Display of task status] window, and confirm that "DORMANT" is displayed as the task state for each task number in the specified range of task numbers.

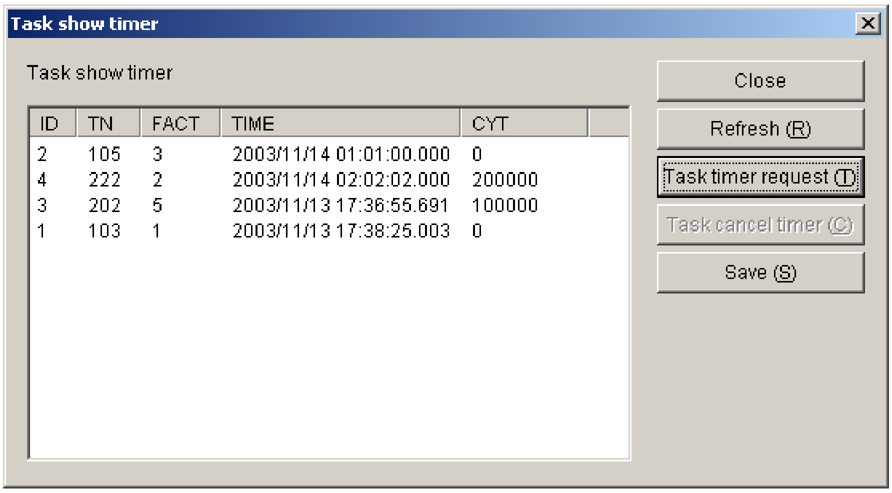
(3) OK button
Clicking this button suspends initiation of the task with the specified task number. When the [Range selection] check box is selected and a range of task numbers specified, clicking this button suspends initiation of the tasks with the task numbers in the specified range. After task initiation is suspended, the [CPMS DEBUGGER] window appears.

(4) Cancel button
Clicking this button cancels the suspension of task initiation, and returns to the [CPMS DEBUGGER] window.

3.8 Task Timer Request

Function: The “Task Timer Request” command opens the window to display the current settings of cyclic task initiation and enables the user to make new settings.

Procedure: Click the Task Timer Request button in the [CPMS DEBUGGER] window.



Task show timer

ID	TN	FACT	TIME	CYT
2	105	3	2003/11/14 01:01:00.000	0
4	222	2	2003/11/14 02:02:02.000	200000
3	202	5	2003/11/13 17:36:55.691	100000
1	103	1	2003/11/13 17:38:25.003	0

Figure 3-10 [Task show timer] Window

(1) [Task show timer] box

This box lists the current settings of cyclic task initiation.

The items of information displayed in the list are the same as those displayed by the sht command of RPDP/S10V.

The items of information listed are as follows:

No.	Item	Description
1	ID	Timer type (*)
2	TN	Task number
3	FACT	Initiation factor
4	TIME	Initiation time (year/month/day hour:minute:second. millisecond)
5	CYT	Cycle time (in milliseconds)

(*) Timer type

ID	Description
1	Initiation based on length of time (timer)
2	Initiation based on time of day (timer)
3	Cyclic initiation based on length-of-time specification (timer)
4	Cyclic initiation based on time-of-day specification (timer)

(2) Close button

Clicking this button closes the [Task show timer] window, and returns to the [CPMS DEBUGGER] window.

(3) Refresh button

Clicking this button updates the information shown in the [Task show timer] box.

(4) Task timer request button

Clicking this button opens the [Task timer request] window, which is used for setting cyclic task initiation.

(5) Task cancel timer button

Clicking this button cancels the cyclic initiation setting for the task selected in the [Task show timer] box.

The content of this processing is the same as the processing performed by the ct command of RPDP/S10V.

<Note> Canceling the cyclic initiation setting for a specific task having initiation factor “0” also cancels the cyclic initiation setting for other tasks with the same task number as that of the specific task.

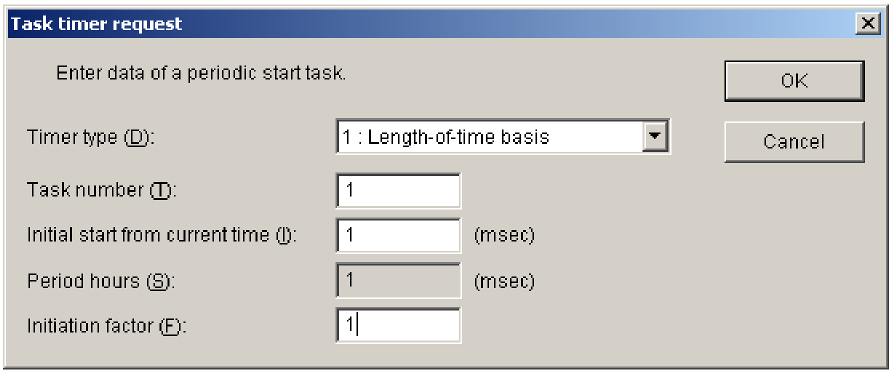
(6) Save button

Clicking this button saves the information displayed in the [Task show timer] box as a text-format file.

<[Task timer request] window>

Function: The [Task timer request] window is used to make cyclic initiation settings for a task.

Procedure: Click the Task timer request button in the [Task show timer] window.



Task timer request

Enter data of a periodic start task.

Timer type (D): 1 : Length-of-time basis

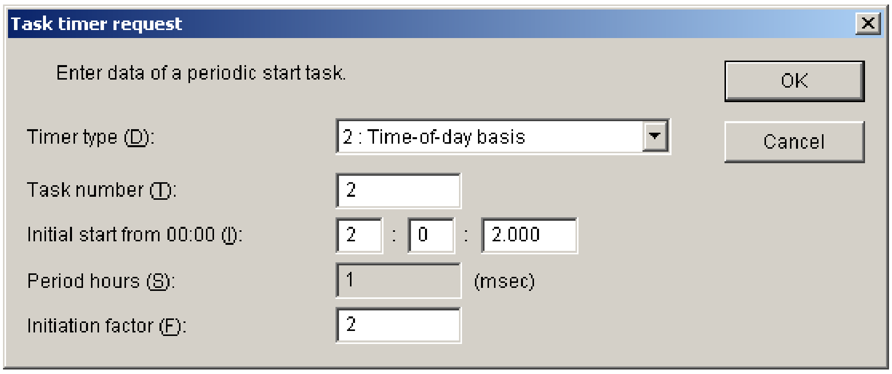
Task number (T): 1

Initial start from current time (I): 1 (msec)

Period hours (S): 1 (msec)

Initiation factor (F): 1

Figure 3-11 [Task timer request] Window (1: Length-of-time basis)



Task timer request

Enter data of a periodic start task.

Timer type (D): 2 : Time-of-day basis

Task number (T): 2

Initial start from 00:00 (I): 2 : 0 : 2.000

Period hours (S): 1 (msec)

Initiation factor (F): 2

Figure 3-12 [Task timer request] Window (2: Time-of-day basis)

(1) [Task number] box

This box is used to enter the task number of a task to be initiated cyclically.

(2) [Initial start from current time] and [Initial start from 00:00] boxes

These boxes are used to specify the length of time to initially start the timer event and the time of day to initially start the timer event, respectively.

In the [Initial start from current time] box, the user can specify a relative length of time from the current time to the start time from 1 to 86400000.

In the [Initial start from 00:00] box, he/she can specify a time of day in the range 00:00:0.000 to 23:59:59.999 as the initial start time of the timer event.

(3) [Period hours] box

This box is used to specify a cycle time when a timer event is to be generated cyclically for the specified task.

The cycle timer can be specified from 1 to 86400000 (in units of ms).

(4) [Initiation factor] box

This box is used to specify the initiation factor of the specified task.

(5) OK button

Clicking this button sets the specified contents of cyclic initiation for the specified task. After this setting is made, the [Task show timer] window appears.

(6) Cancel button

Clicking this button cancels the new setting of cyclic task initiation, and returns to the [Task show timer] window.

3.9 Task Suspend

Function: The “Task Suspend” command opens the window to suspend execution of a task (or tasks).

Procedure: Click the Task Suspend button in the [CPMS DEBUGGER] window.

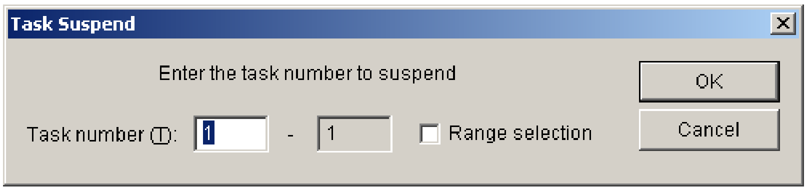


Figure 3-13 [Task Suspend] Window

(1) [Task number] box
This box is used to enter the task number of the task for which execution is to be suspended. To enable a range specification of task numbers, select the [Range selection] check box.

(2) [Range selection] check box
Selecting this check box enables a range specification of task numbers. Even if an invalid range is specified, no error message will appear. To confirm the result of command execution, open the [Display of task status] (detail) window specifying each task number in the specified range, and confirm that the [waiting for resume macro] check box is selected in the [Details of task state] group box.

(3) OK button
Clicking this button suspends execution of the task with the specified task number. When the [Range selection] check box is selected and a range of task numbers specified, clicking this button suspends execution of the tasks with task numbers in the specified range.
After task execution is suspended, the [CPMS DEBUGGER] window appears.

(4) Cancel button
Clicking this button cancels the suspension of task execution, and returns to the [CPMS DEBUGGER] window.

3.10 Task Resume

Function: The "Task Resume" command opens the window to release a task (or tasks) from the Suspended state (execution-suspended state).

Procedure: Click the Task Resume button in the [CPMS DEBUGGER] window.

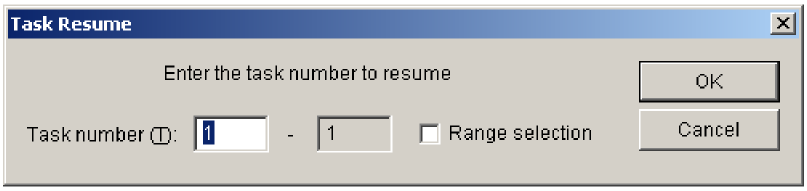


Figure 3-14 [Task Resume] Window

(1) [Task number] box

This box is used to enter the task number of the task to be released from the Suspended state. To enable a range specification of task numbers, select the [Range selection] check box.

(2) [Range selection] check box

Selecting this check box enables a range specification of task numbers. Even if an invalid range is specified, no error message will appear. To confirm the result of command execution, open the [Display of task status] (detail) window specifying each task number in the specified range, and confirm that the [waiting for resume macro] check box is not selected in the [Details of task state] group box.

(3) OK button

Clicking this button releases the task with the specified task number from the Suspended state. When the [Range selection] check box is selected and a range of task numbers specified, clicking this button releases the tasks with the task numbers in the specified range from the Suspended state.

After the tasks are released, the [CPMS DEBUGGER] window appears.

(4) Cancel button

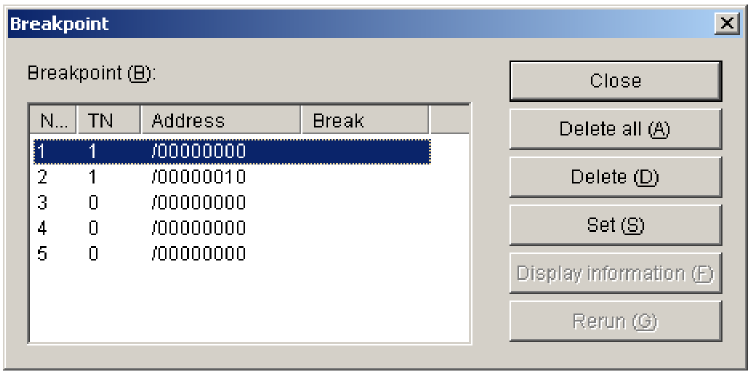
Clicking this button cancels the releasing of tasks from the Suspended state, and returns to the [CPMS DEBUGGER] window.

3.11 Breakpoint

<[Breakpoint] window>

Function: The “Breakpoint” command opens the window to set or delete a breakpoint (or breakpoints).

Procedure: Click the Breakpoint button in the [CPMS DEBUGGER] window.



Breakpoint (B):

N...	TN	Address	Break
1	1	/00000000	
2	1	/00000010	
3	0	/00000000	
4	0	/00000000	
5	0	/00000000	

Figure 3-15 [Breakpoint] Window

(1) [Breakpoint] box

This box shows the current break status of tasks. Up to five breakpoints can be registered. The items of information shown in this box are as follows:

No.	Item	Description
1	TN	Task number
2	Address	Breakpoint address (relative address in program)
3	Break	Break-occurrence status

(2) Close button

Clicking this button closes the [Breakpoint] window, and returns to the [CPMS DEBUGGER] window.

(3) Delete all button

Clicking this button deletes all currently set breakpoints other than those in the task(s) in the break state.

(4) Delete button

Clicking this button deletes the breakpoint selected in the [Breakpoint] box.

(5) Set button

Clicking this button opens the [Breakpoint address] window, which is used for breakpoint setting.

(6) Display information button

Clicking this button opens the [Contents of register] window, which is used for displaying and setting register contents.

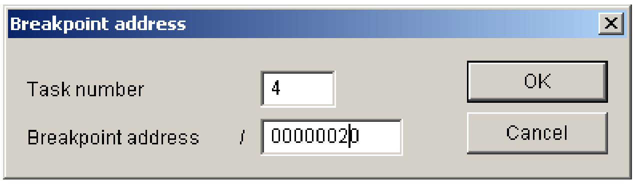
This button is only effective when the program in the CPU of a connected programmable controller (PCs) is in the break state.

(7) Rerun button

Clicking this button restarts the task in the break state.

This button is only effective when the program in the CPU of a connected PCs is in the break state.

<[Breakpoint address] window>



Breakpoint address

Task number: 4

Breakpoint address / 00000020

Figure 3-16 [Breakpoint address] Window

(1) [Task number] box

This box is used to enter the task number of the task for which a breakpoint is to be set.

(2) [Breakpoint address] box

This box is used to specify the breakpoint address (relative address in the relevant program) to be set.

(3) OK button

Clicking this button sets a breakpoint at the specified address.

After the breakpoint is set, the [Breakpoint] window appears.

(4) Cancel button

Clicking this button cancels breakpoint setting, and returns to the [Breakpoint] window.

<Breakpoint setting procedure>

The following describes a sample procedure for setting a breakpoint at a point (A) in a generated task program (task1):

<task1.c>

```
int b1;
int d1 = 11;
static int b2 ;
static int d2 = 101 ;

main()
{
  unsigned short *fw010;
  unsigned short *dw000;

  static int b3 ;
  static int d3 = 1001 ;
  int s1;
  int s2 = 21 ;

  fw010 = (unsigned short *)0xE2020;
  dw000 = (unsigned short *)0x61000;

  *fw010 = *fw010 + *dw000;                                    (A)
  exit(0);

}
```

Execute the shc command with the "-listfile and -show=source,object" options specified to assemble the assembler source in which C-language source files are inserted. To set a breakpoint in source program "task1.c", reference the "task1.1st" file.

In the "task1.c" file, the C-language source used to set the breakpoint is indicated by (1), and assembler instruction (2) is associated with the C-language source. The offset of assembler instruction (2) in the C-language source (task1.c) is 0x0000000A.

<prog1.lst>

```
************ OBJECT LISTING ************

FILE NAME: task1.c

SCT OFFSET     CODE          C LABEL       INSTRUCTION OPERAND       COMMENT

         task1.c      1      int b1;
         task1.c      2      int d1 = 11;
         task1.c      3      static int b2 ;
         task1.c      4      static int d2 = 101 ;
         task1.c      5
         task1.c      6      main()
P    00000000                 _main:                              ; function: main
                                                                  ; frame size=0
         task1.c      7      {
         task1.c      8              unsigned short *fw010;
         task1.c      9              unsigned short *dw000;
         task1.c     10
         task1.c     11              static int b3 ;
         task1.c     12              static int d3 = 1001 ;
         task1.c     13              int s1;
         task1.c     14              int s2 = 21 ;
         task1.c     15
         task1.c     16              fw010 = (unsigned short *)0xE2020;
     00000000 D504                          MOV.L       L11+2,R5     ; H'000E2020
     00000002 E400                          MOV         #0,R4         ; H'00000000
         task1.c     17              dw000 = (unsigned short *)0x61000;
         task1.c     18
         task1.c     19              *fw010 = *fw010 + *dw000;                      (1)
     00000004 D204                          MOV.L       L11+6,R2     ; H'00061000
     00000006 6621                          MOV.W       @R2,R6
     00000008 6251                          MOV.W       @R5,R2
     0000000A 362C                          ADD         R2,R6                       (2)
   task1.c      20             exit(0);
     0000000C D203                          MOV.L       L11+10,R2    ; _exit
     0000000E 422B                          JMP         @R2
     00000010 2561                          MOV.W       R6,@R5
     00000012                  L11:
     00000012 00000002                      .RES.W      1
     00000014 000E2020                      .DATA.L     H'000E2020
     00000018 00061000                      .DATA.L     H'00061000
     0000001C <00000000>                    .DATA.L     _exit
         task1.c     21
   task1.c      22       }
```

Execute the optlink command with the -list option specified to generate an execution file with a mapping file.
The information on SECTION “P” in the mapping file indicates that task1.ocj begins at address 0x30000000.
Because the relative address (in the source program) of the breakpoint to be set is 0x0000000A, the address of the breakpoint to be set is 0x3000000A (0x30000000 + 0x0000000A).
The relative address in the program (0x0000000A) can be obtained by subtracting the start address of the program from the address of the breakpoint to be set (0x3000000A - 0x30000000).

<prog.map>

```
*** Mapping List ***

SECTION                              START      END          SIZE    ALIGN

P
                                      30000000   3000006f      70     4
```

<[Display of register] window>

Function: The [Display of register] window is used to display and set the contents of registers.

Procedure: Click the Display information button in the [Breakpoint] window.

(1) Contents of a register

These boxes show the contents of the registers described below. The user can change the value of a register by entering a new value (in hexadecimal notation) in the corresponding box and pressing the [Enter] key on the keyboard or clicking the OK button.

No.	Item	Description
1	SR	Status register
2	PC	Program register
3	GBR	Global base register
4	PR	Procedure register
5	MACH	System register (MAC register high)
6	MACL	System register (MAC register low)
7	FPUL	Floating-point communication register
8	FPSCR	Floating-point status/control register
9	R0 to R15	General-purpose registers (R15 is used as a stack pointer.)

(2) Floating-point registers

Clicking the Display register button opens the [Floating decimal point register] window, which is used for displaying and setting the contents of the floating-point registers listed below.

No.	Item	Description
1	FR	Single-precision floating-point register
2	XF	Extended single-precision floating-point register
3	DR	Double-precision floating-point register
4	XF	Extended double-precision floating-point register

(3) OK button

Clicking this button changes the current contents of registers for the task in the break state to those specified by the user as the new values of general-purpose and other registers.
After the register contents are changed, the [Breakpoint] window appears.

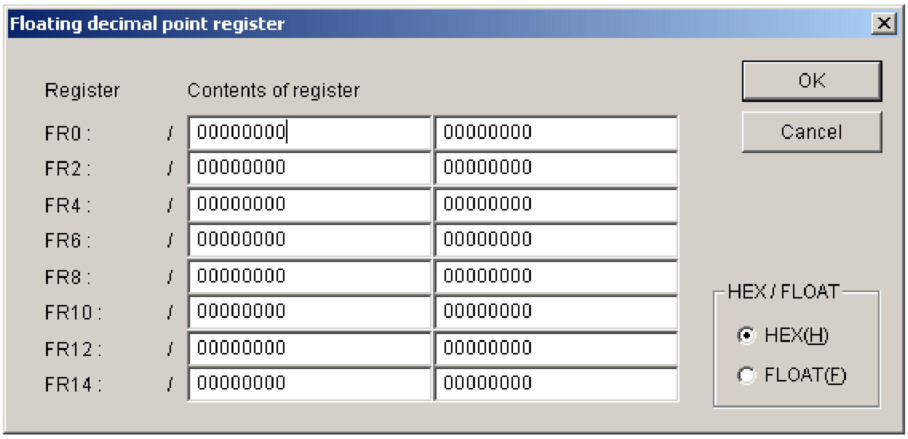
(4) Cancel button

Clicking this button cancels the changes made to register contents, and returns to the [Breakpoint] window.

<[Floating decimal point register] window>

Function: The [Floating decimal point register] window is used to display and set the contents of floating-point registers.

Procedure: Click the Display register button in the [Display of register] window.



Register	Contents of register	
FR0 : /	00000000	00000000
FR2 : /	00000000	00000000
FR4 : /	00000000	00000000
FR6 : /	00000000	00000000
FR8 : /	00000000	00000000
FR10 : /	00000000	00000000
FR12 : /	00000000	00000000
FR14 : /	00000000	00000000

● HEX(H)
○ FLOAT(F)

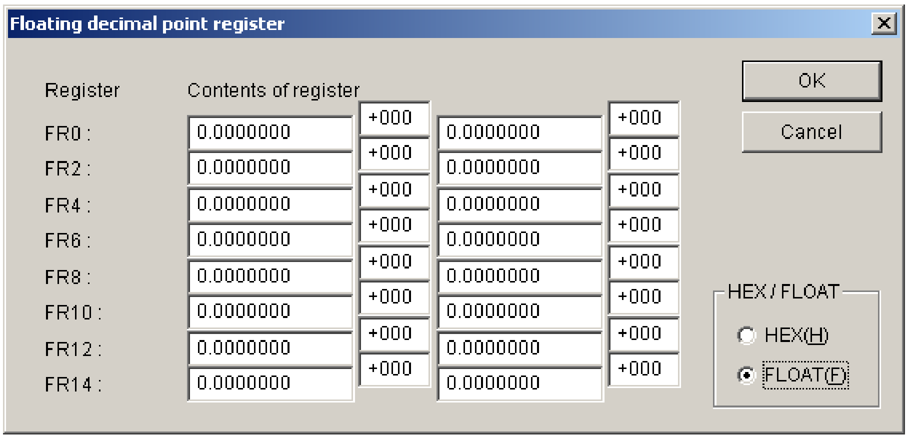
Figure 3-17 [Floating decimal point register] Window

(1) [HEX/FLOAT] box

The radio buttons in this box are used to switch the numeration system of displayed values between hexadecimal numerals (HEX) and real numbers (FLOAT).

For example, selecting the [FLOAT] radio button in the window shown in Figure 3-17 switches the numeral display boxes to those of the window shown in Figure 3-18.

<Example of window displaying the contents of floating-point registers (FRs)>



Register	Contents of register			
FR0 :	0.0000000	+000	0.0000000	+000
FR2 :	0.0000000	+000	0.0000000	+000
FR4 :	0.0000000	+000	0.0000000	+000
FR6 :	0.0000000	+000	0.0000000	+000
FR8 :	0.0000000	+000	0.0000000	+000
FR10 :	0.0000000	+000	0.0000000	+000
FR12 :	0.0000000	+000	0.0000000	+000
FR14 :	0.0000000	+000	0.0000000	+000

○ HEX(H)
● FLOAT(F)

Figure 3-18 [Floating decimal point register] Window (FLOAT)

(2) [Register] column

This column indicates the 0 to 15 registers specified in the [Display of register] window.

(3) [Contents of register] boxes

These boxes show the contents of the indicated registers. The numeration system and setting format can be switched between HEX and FLOAT.

(4) OK button

Clicking this button changes the current contents of registers for the task in the break state to those specified by the user as the new values of floating-point registers.
After the register contents are changed, the [Display of register] window appears.

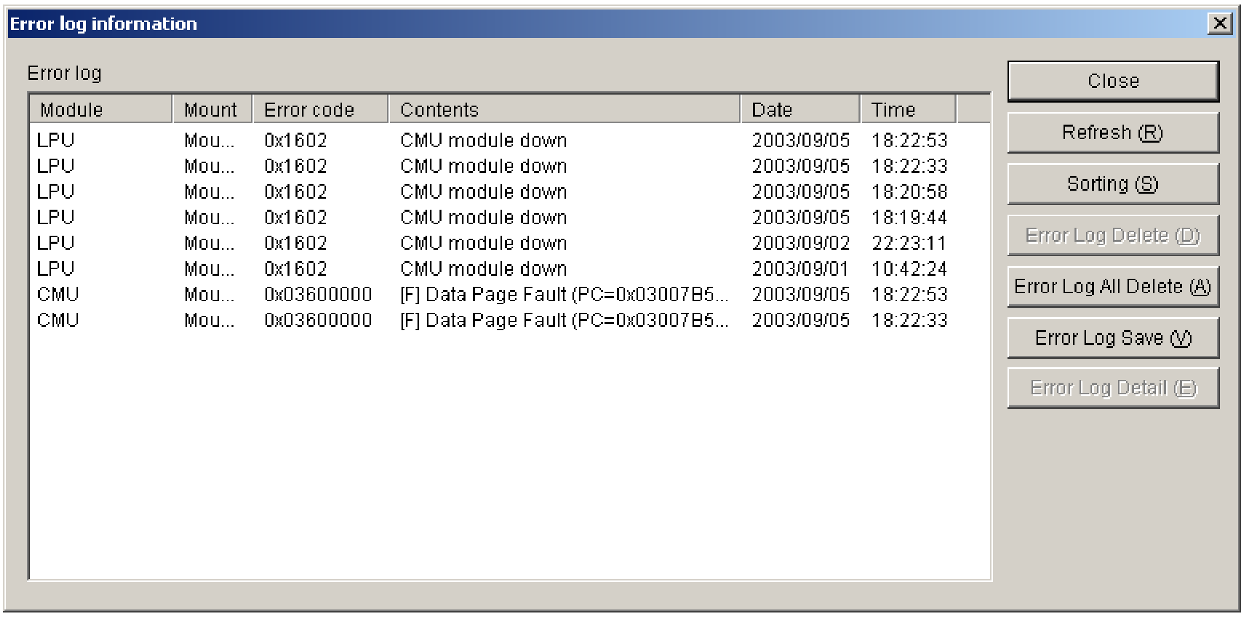
(5) Cancel button

Clicking this button cancels the content changes made to the floating-point registers, and returns to the [Display of register] window.

3.12 Error Log

Function: The “Error Log” command opens the window to display a list of error log information.

Procedure: Click the Error Log button in the [CPMS DEBUGGER] window.



Module	Mount	Error code	Contents	Date	Time
LPU	Mou...	0x1602	CMU module down	2003/09/05	18:22:53
LPU	Mou...	0x1602	CMU module down	2003/09/05	18:22:33
LPU	Mou...	0x1602	CMU module down	2003/09/05	18:20:58
LPU	Mou...	0x1602	CMU module down	2003/09/05	18:19:44
LPU	Mou...	0x1602	CMU module down	2003/09/02	22:23:11
LPU	Mou...	0x1602	CMU module down	2003/09/01	10:42:24
CMU	Mou...	0x03600000	[F] Data Page Fault (PC=0x03007B5...	2003/09/05	18:22:53
CMU	Mou...	0x03600000	[F] Data Page Fault (PC=0x03007B5...	2003/09/05	18:22:33

Figure 3-19 [Error log information] Window

(1) [Error log] box

This box lists the current error log information.

The items of information listed are as follows:

No.	Item	Description
1	Module	Name of option module on which error log information is found
2	Mount	Mounting status of module
3	Error code	Error code
4	Contents	Content of error
5	Data	Date of error occurrence
6	Time	Time of error occurrence

(2) Close button

Clicking this button closes the [Error log information] window, and returns to the [CPMS DEBUGGER] window.

(3) Refresh button

Clicking this button updates the error log information shown in the window.

(4) Sorting button

Clicking this button sorts the error log information in order of the time of error occurrence. The sorting order alternately switches between ascending and descending order each time this button is clicked.

(5) Error Log Delete button

Clicking this button deletes the error log information on the module specified in the [Error log information] window.

(6) Error Log All Delete button

Clicking this button deletes all error log information on the LPUs, CMUs, and option modules.

(7) Error Log Save button

Clicking this button enables the user to save error log information on the LPUs, CMUs, and option modules as a text file. When this button is clicked, the [Save As] window appears to allow the user to specify the file in which to save the information.

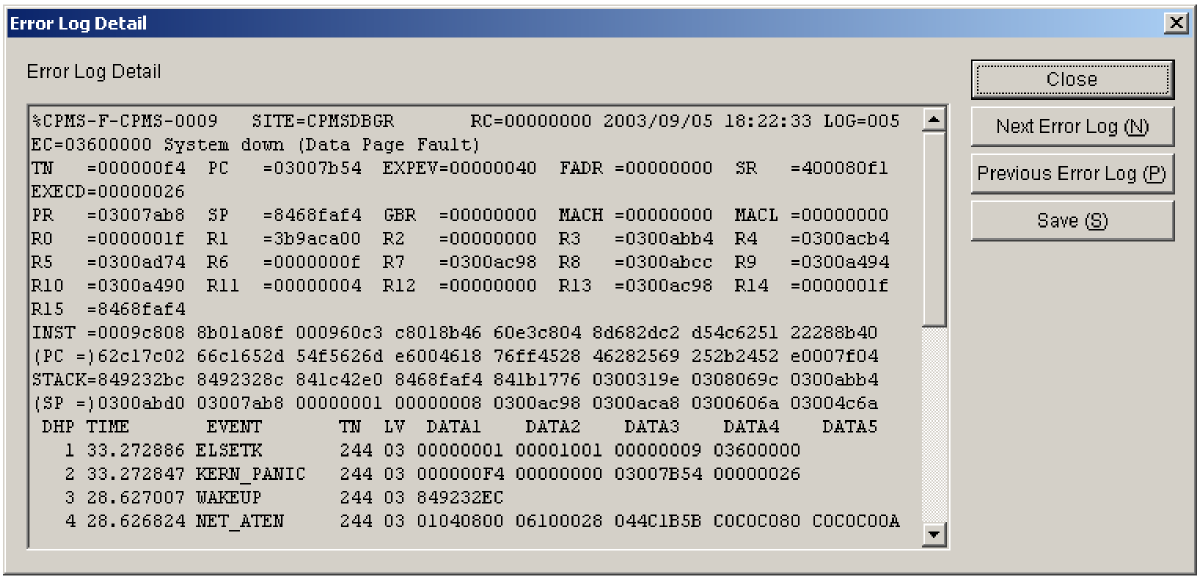
(8) Error Log Detail button

Clicking this button opens the [Error Log Detail] window, which displays details of the error log specified in the [Error log] box. Detail display is only available for CMU error logs.

<[Error Log Detail] window>

Function: The [Error Log Detail] window displays details of a CMU error log.

Procedure: Click the Error Log Detail button in the [Error log information] window.



```
%CPMS-F-CPMS-0009   SITE=CPMSDBGR       RC=00000000 2003/09/05 18:22:33 LOG=005
EC=03600000 System down (Data Page Fault)
TN   =000000f4  PC   =03007b54  EXPEV=00000040  FADR =00000000  SR   =400080f1
EXECD=00000026
PR   =03007ab8  SP   =8468faf4  GBR  =00000000  MACH =00000000  MACL =00000000
R0   =0000001f  R1   =3b9aca00  R2   =00000000  R3   =0300abb4  R4   =0300acb4
R5   =0300ad74  R6   =0000000f  R7   =0300ac98  R8   =0300abcc  R9   =0300a494
R10  =0300a490  R11  =00000004  R12  =00000000  R13  =0300ac98  R14  =0000001f
R15  =8468faf4
INST =0009c808 8b01a08f 000960c3 c8018b46 60e3c804 8d682dc2 d54c6251 22288b40
(PC =)62c17c02 66c1652d 54f5626d e6004618 76ff4528 46282569 252b2452 e0007f04
STACK=849232bc 8492328c 841c42e0 8468faf4 841b1776 0300319e 0308069c 0300abb4
(SP =)0300abd0 03007ab8 00000001 00000008 0300ac98 0300aca8 0300606a 03004c6a
 DHP TIME       EVENT        TN  LV  DATA1    DATA2    DATA3    DATA4    DATA5
   1 33.272886 ELSETK       244 03 00000001 00001001 00000009 03600000
   2 33.272847 KERN_PANIC   244 03 000000F4 00000000 03007B54 00000026
   3 28.627007 WAKEUP       244 03 849232EC
   4 28.626824 NET_ATEN     244 03 01040800 06100028 044C1B5B C0C0C080 C0C0C00A
```

Figure 3-20 [Error Log Detail] Window

(1) [Error Log Detail] box

This box shows details of the error log specified in the [Error log information] window. The details of the error log shown in this box are the same as those displayed by the svelog command of RPDP/S10V.

(2) Close button

Clicking this button closes the [Error Log Detail] window, and returns to the [Error log information] window.

(3) Next Error Log button

Clicking this button displays the CMU error log following the one that is currently displayed.

(4) Previous Error Log button

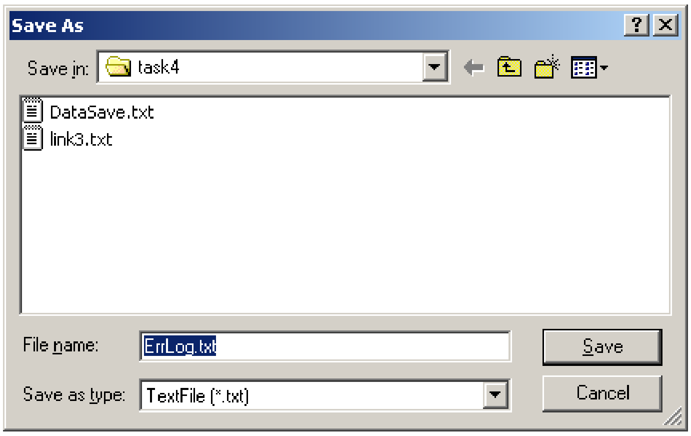
Clicking this button displays the CMU error log preceding the one that is currently displayed.

(5) Save button

Clicking this button saves the details of the error log currently displayed in the [Error Log Detail] window as a text file. "DEBUGGER" is always recorded as the site name at the top of the text file.

<[Save As] window>

Function: The [Save As] window is used to save error log information as a text file.



Save As

Save in: task4

DataSave.txt
link3.txt

File name: ErrLog.txt

Save as type: TextFile (*.txt)

Figure 3-21 [Save As] Window

(1) [Save in] box

This box is used to select the folder and file to save error log information through operation of the combo and input boxes.

(2) [File name] box

This box is used to display a selected save-destination file or enter the name of the file to be saved. This box shows “ErrLog.txt” as the default file name.

(3) [Save as type] combo box

This combo box is used to select the type of file to be saved. “TextFile (*.txt)” or “AllFiles (*.*)” can be selected.

(4) Save button

Clicking this button saves error log information in the specified file, and returns to the previously displayed window.

“DEBUGGER” is always recorded as the site name at the top of the saved file.

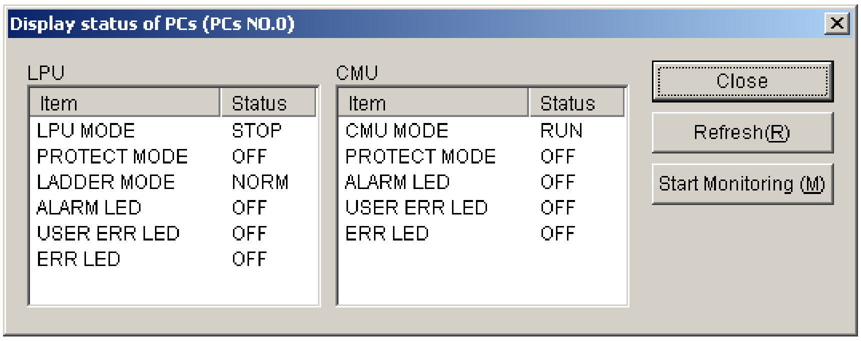
(5) Cancel button

Clicking this button cancels the saving of information to the specified file, and returns to the previously displayed window.

3.13 Display Status of PCs

Function: The "Display Status of PCs" command opens the window to display the status of programmable controllers (PCs).

Procedure: Click the Display Status of PCs button in the [CPMS DEBUGGER] window.



Display status of PCs (PCs NO.0)

LPU

Item	Status
LPU MODE	STOP
PROTECT MODE	OFF
LADDER MODE	NORM
ALARM LED	OFF
USER ERR LED	OFF
ERR LED	OFF

CMU

Item	Status
CMU MODE	RUN
PROTECT MODE	OFF
ALARM LED	OFF
USER ERR LED	OFF
ERR LED	OFF

Figure 3-22 [Display status of PCs] Window

(1) [LPU] and [CMU] boxes

These boxes show the status of currently connected LPUs and CMUs.

The items of status information shown in the boxes are as follows:

No.	Item	Status	Description
1	LPU/CMU MODE	RUN	The LPU or CMU is operating.
		STOP	The LPU or CMU is stopped.
2	PROTECT MODE	ON	The LPU or CMU is in protection mode.
		OFF	The LPU or CMU is not in protection mode.
3	LADDER MODE	NORM	The LPU is operating normally.
		SIMU	The LPU is running in simulation mode.
4	ALARM LED	ON	The ALARM indicator is on.
		OFF	The ALARM indicator is off.
5	USER ERR LED	ON	The USER indicator is on.
		OFF	The USER indicator is off.
6	ERR LED	ON	The ERR indicator is on.
		OFF	The ERR indicator is off.

(2) Close button

Clicking this button closes the [Display status of PCs] window, and returns to the [CPMS DEBUGGER] window.

(3) Refresh button

Clicking this button updates the PCs status information shown in the window.

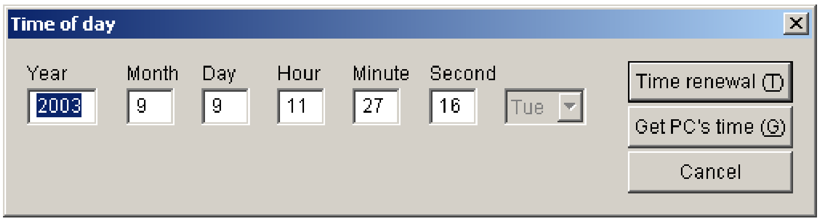
(4) Start Monitoring and Stop Monitoring buttons

These buttons are used to turn system status monitoring on and off, respectively.

3.14 Set Time

Function: The "Set time" command opens the window to display and update the current time set on the CPU of a programmable controller (PCs).

Procedure: Click the Set time button in the [CPMS DEBUGGER] window.



Year	Month	Day	Hour	Minute	Second	
2003	9	9	11	27	16	Tue

Figure 3-23 [Time of day] Window

(1) Time boxes

These boxes indicate the time set on the currently connected CPU (PCs).

(2) Time renewal button

Clicking this button changes the time set on the CPU (PCs) to the time specified in the time boxes.

(3) Get PC's time button

Clicking this button updates the time indicated by the time boxes based on the current time set on the personal computer.

To enable time update, click the Time renewal button.

(4) Cancel button

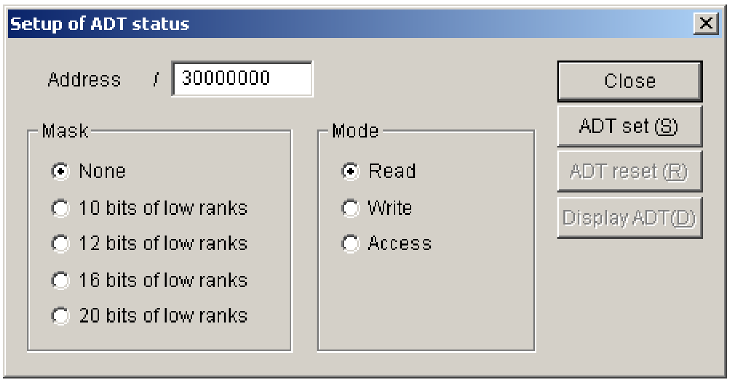
Clicking this button cancels time setting, and returns to the [CPMS DEBUGGER] window.

3.15 Setup of ADT Status

<[Setup of ADT status] window>

Function: The “Setup of ADT status” command opens the window to set an address detection trap (ADT).

Procedure: Click the Setup of ADT status button in the [CPMS DEBUGGER] window.



Setup of ADT status

Address / 30000000

Mask
● None
○ 10 bits of low ranks
○ 12 bits of low ranks
○ 16 bits of low ranks
○ 20 bits of low ranks

Mode
● Read
○ Write
○ Access

Figure 3-24 [Setup of ADT status] Window

(1) [Address] box

This box is used to specify the logical address (in bytes) where an ADT is to be set.

(2) [Mask] group box

The radio buttons in this box are used to select a mask value to mask the specified logical address.

(3) [Mode] group box

The radio buttons in this box are used to select a mode to be set for the ADT.

(4) Close button

Clicking this button closes the [Setup of ADT status] window, and returns to the [CPMS DEBUGGER] window.

(5) ADT set button

Clicking this button sets the specified ADT.

The set ADT applies to the byte, word, and long word accesses made to the logical address specified in the [Address] box. After the ADT is set, the [Display Status of ADT] window appears, and then the [Setup of ADT status] window reappears.

(6) ADT reset button

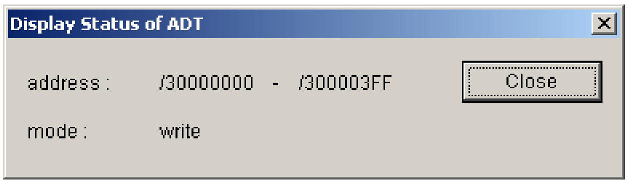
Clicking this button clears the set ADT.

(7) Display ADT button

Clicking this button opens the [Display Status of ADT] window, which displays the ADT setting content.

<[Display Status of ADT] window>

Function: The [Display Status of ADT] window displays the ADT setting content.



address : /30000000 - /300003FF

mode : write

Figure 3-25 [Display Status of ADT] Window

(1) [address] field

This field shows the range of addresses where the ADT is set.

(2) [mode] field

This field shows one of the following modes set for the ADT:

read: Read trap

write: Write trap

access: Read/write trap

(3) Close button

Clicking this button closes the [Display Status of ADT] window, and returns to the [Setup of ADT status] window.

<Notes>

- The ADT can only be set at one position.
- The set ADT is not deleted when the CPU is restarted.
- The ADT can only detect access made via the MMU of the processor. Therefore, the ADT cannot detect access made using a physical address by a memory access subcommand of CPMS Debugger.
- The ADT can be set in the shaded ranges shown in Figure 3-26. The ADT cannot be set in an unmapped area.

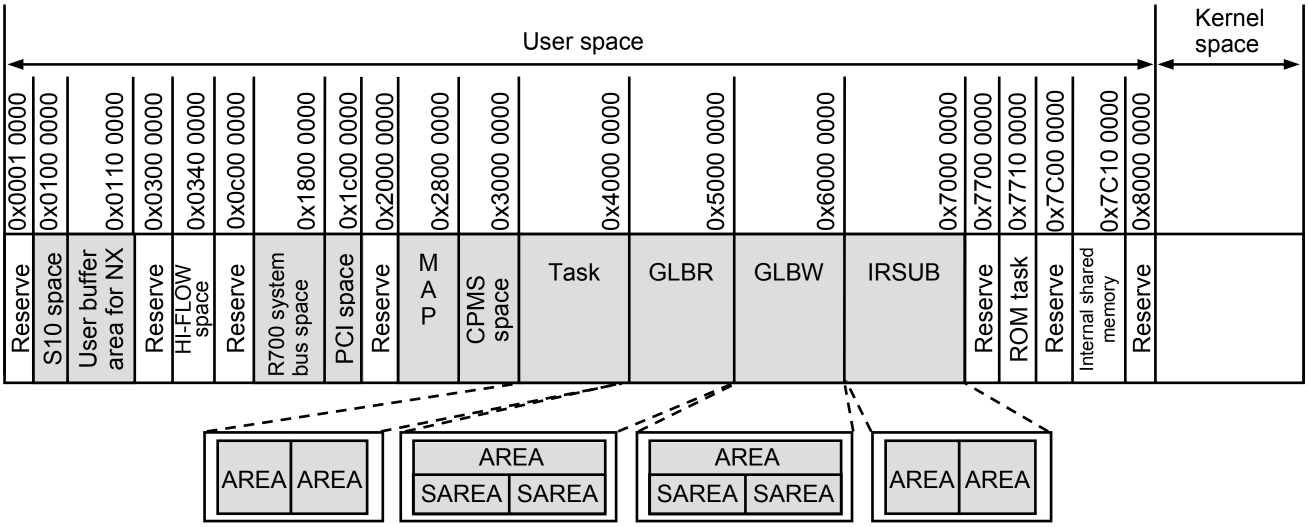
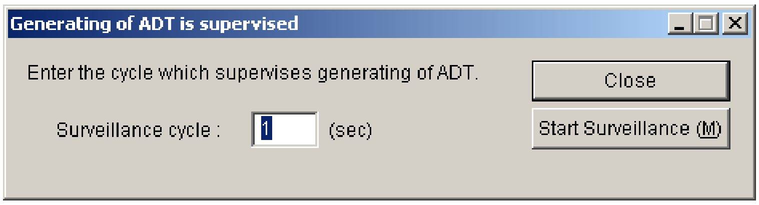


Figure 3-26 ADT-settable Ranges

3.16 ADT Monitor

Function: The “ADT monitor” command opens the window to monitor ADT generation.
Procedure: Click the ADT monitor button in the [CPMS DEBUGGER] window.



Generating of ADT is supervised

Enter the cycle which supervises generating of ADT.

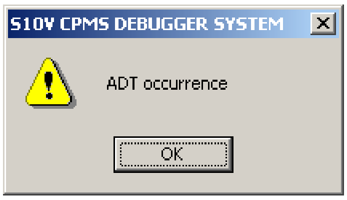
Surveillance cycle : 1 (sec)

Figure 3-27 [Generating of ADT is supervised] Window

(1) [Surveillance cycle] box
This box is used to specify the cycle of ADT generation monitoring from 1 to 60 seconds.

(2) Close button
Clicking this button closes the [Generating of ADT is supervised] window, and returns to the [CPMS DEBUGGER] window.

(3) Start Surveillance button
Clicking this button starts the monitoring of ADT generation in the cycle specified in the [Surveillance cycle] box.
When the generation of ADT is detected, the message shown below appears and monitoring operation ends.



S10V CPMS DEBUGGER SYSTEM

ADT occurrence

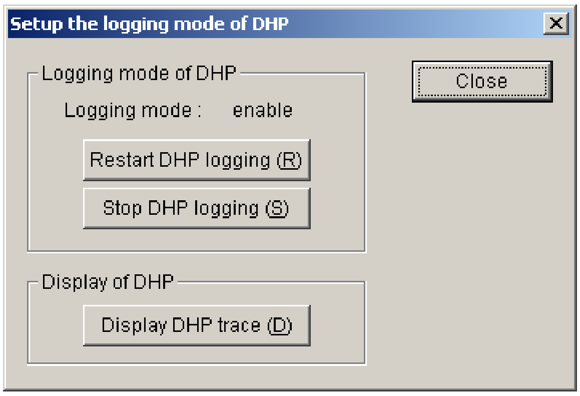
Figure 3-28 ADT Generation Message

3.17 Setup of DHP Status

<[Setup the logging mode of DHP] window>

Function: The “Setup of DHP status” command opens the window to set the DHP logging mode.

Procedure: Click the Setup of DHP status button in the [CPMS DEBUGGER] window.



Setup the logging mode of DHP

Logging mode of DHP

Logging mode : enable

Figure 3-29 [Setup the logging mode of DHP] Window

(1) [Logging mode] field

This field shows the current DHP logging mode.

(2) Restart DHP logging button

Clicking this button sets the DHP logging mode to enable mode.

When mode setting ends normally, the mode indication in the [Logging mode] field is updated.

(3) Stop DHP logging button

Clicking this button sets the DHP logging mode to disable mode.

When mode setting ends normally, the mode indication in the [Logging mode] field is updated.

(4) Display DHP trace button

Clicking this button opens the [Display DHP trace] window, which displays DHP trace information.

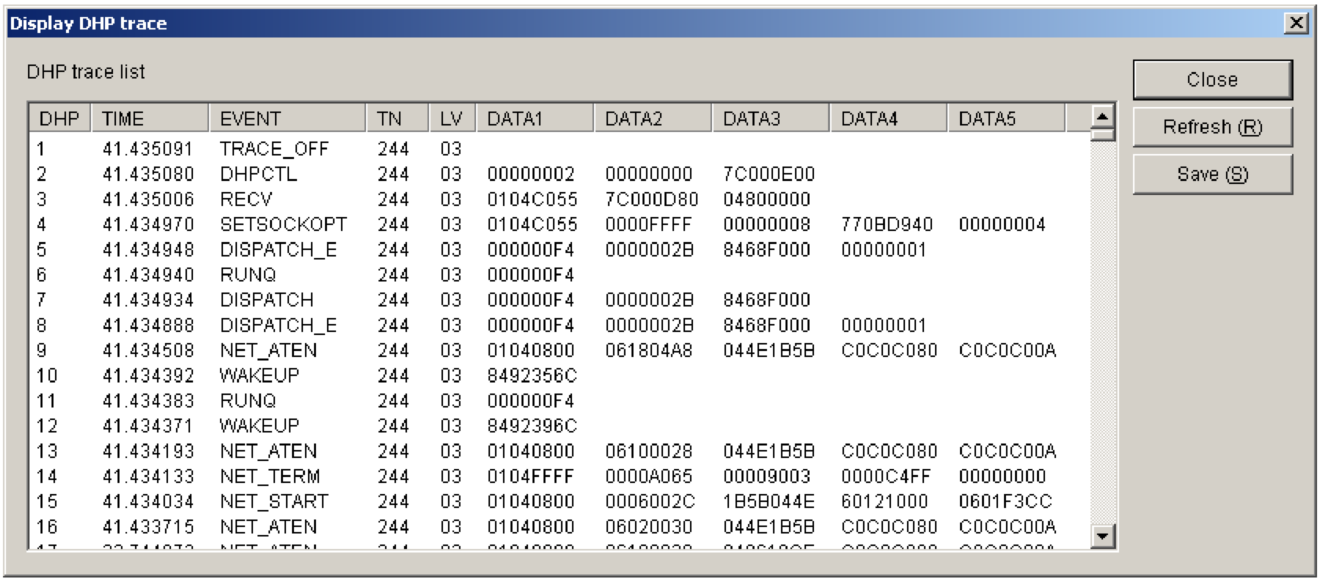
(5) Close button

Clicking this button closes the [Setup the logging mode of DHP] window, and returns to the [CPMS DEBUGGER] window.

<[Display DHP trace] window>

Function: The [Display DHP trace] window displays DHP trace information.

Procedure: Click the Display DHP trace button in the [Setup the logging mode of DHP] window.



DHP trace list

DHP	TIME	EVENT	TN	LV	DATA1	DATA2	DATA3	DATA4	DATA5
1	41.435091	TRACE_OFF	244	03					
2	41.435080	DHPCTL	244	03	00000002	00000000	7C000E00		
3	41.435006	RECV	244	03	0104C055	7C000D80	04800000		
4	41.434970	SETSOCKOPT	244	03	0104C055	0000FFFF	00000008	770BD940	00000004
5	41.434948	DISPATCH_E	244	03	000000F4	0000002B	8468F000	00000001	
6	41.434940	RUNQ	244	03	000000F4				
7	41.434934	DISPATCH	244	03	000000F4	0000002B	8468F000		
8	41.434888	DISPATCH_E	244	03	000000F4	0000002B	8468F000	00000001	
9	41.434508	NET_ATEN	244	03	01040800	061804A8	044E1B5B	C0C0C080	C0C0C00A
10	41.434392	WAKEUP	244	03	8492356C				
11	41.434383	RUNQ	244	03	000000F4				
12	41.434371	WAKEUP	244	03	8492396C				
13	41.434193	NET_ATEN	244	03	01040800	06100028	044E1B5B	C0C0C080	C0C0C00A
14	41.434133	NET_TERM	244	03	0104FFFF	0000A065	00009003	0000C4FF	00000000
15	41.434034	NET_START	244	03	01040800	0006002C	1B5B044E	60121000	0601F3CC
16	41.433715	NET_ATEN	244	03	01040800	06020030	044E1B5B	C0C0C080	C0C0C00A

Figure 3-30 [Display DHP trace] Window

(1) [DHP trace list] box

This box shows a list of current DHP trace information.

The items of information listed are the same as those displayed by the svdhp command of RPDP/S10V.

The items of information listed areas follows:

No.	Item	Description
1	DHP	DHP trace number in the list
2	TIME	Trace time tt.tttttt Seconds Microseconds
3	EVENT	Trace point type
4	TN	Task number
5	LV	Priority level
6	DATA1 to DATA5	Trace data (hexadecimal)

(2) Close button

Clicking this button closes the [Display DHP trace] window, and returns to the [Setup the logging mode of DHP] window.

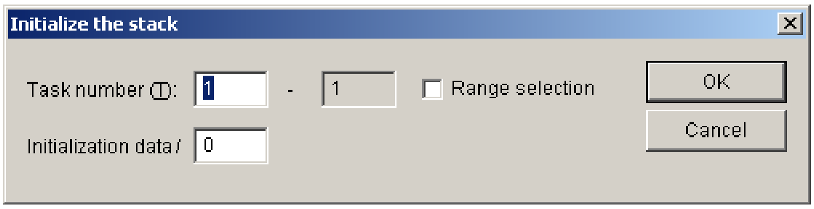
(3) Save button

Clicking this button saves the information displayed in the [DHP trace list] box as a text-format file.

3.18 Initialize the Stack

Function: The “Initialize the stack” command opens the window to initialize the stack for a task (or tasks) using a fixed pattern of data.

Procedure: Click the [Initialize the stack] button in the [CPMS DEBUGGER] window.



Task number (T): 1 - 1

□ Range selection

Initialization data / 0

Figure 3-31 [Initialize the stack] Window

(1) [Task number] box
This box is used to specify the task number (from 1 to 229) of the task whose used stack size is to be displayed.

(2) [Initialization data] box
This box is used to enter a fixed pattern of data (ranging from 0x0 to 0xF) to initialize a stack.

(3) [OK] button
Clicking this button initializes the stack for a specified task.

(4) [Cancel] button
Clicking this button cancels stack initialization, and returns to the [CPMS DEBUGGER] window.

(5) [Range selection] check box
Selecting this check box enables a range specification of task numbers.

<Notes>

- The task subject to stack initialization must be set to the Dormant state before stack initialization.
- When a range of task numbers is specified, CPMS Debugger checks the tasks in the specified range of task numbers to find any task in other than the Dormant state. If a task in other than the Dormant state is found, processing ends abnormally with error message “Task number ** is not dormant” displayed. If processing ends abnormally, stack initialization is also not performed for tasks in the Dormant state. (**: task number)
- Stack initialization is executed in the area from the start address of the page (containing the stack for the specified task) to the last address of the stack. (See Figure 3-32.)

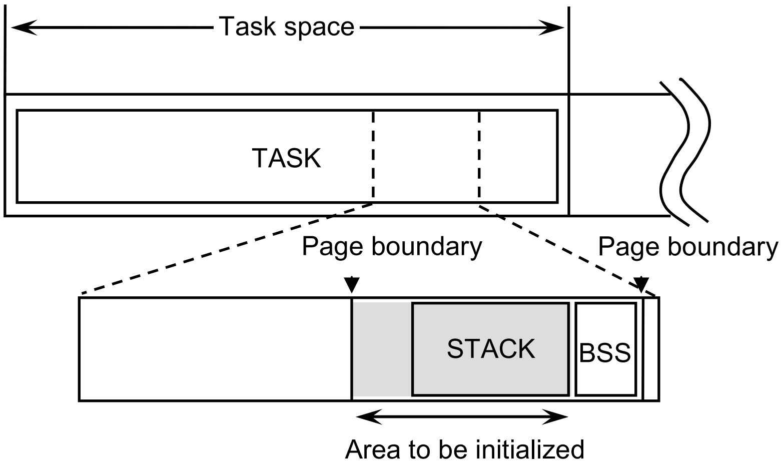
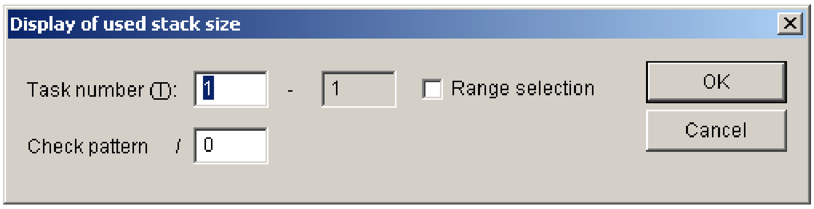


Figure 3-32 Range of Stack Initialization

3.19 Display of Used Stack Size

Function: The “Display of used stack size” command opens the window to display the size of the stack used by a stack (or stacks).

Procedure: Click the Display of used stack size button in the [CPMS DEBUGGER] window.



Display of used stack size

Task number (T): 1 - 1

□ Range selection

Check pattern /: 0

Figure 3-33 [Display of used stack size] Window

(1) [Task number] box

This box is used to specify the task number (from 1 to 229) of the task whose used stack size is to be displayed.

(2) [Check pattern] box

This box is used to enter a check pattern of data (ranging from 0x0 to 0xF) to calculate the used stack size.

(3) [Range selection] check box

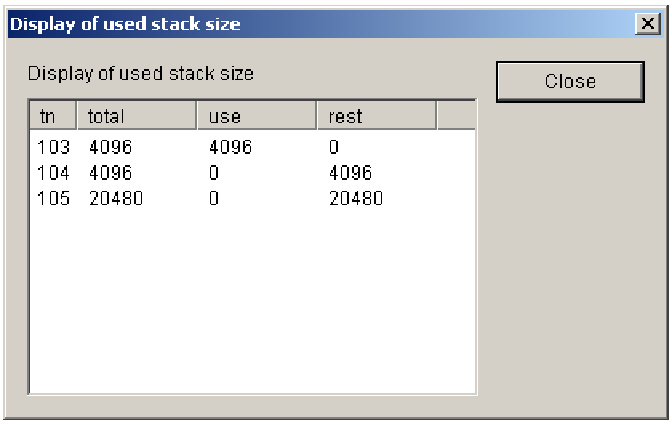
Selecting this check box enables a range specification of task numbers. Even if an invalid range is specified, no error message will appear.

(4) Cancel button

Clicking this button cancels the display of used stack size, and returns to the [CPMS DEBUGGER] window.

(5) OK button

Clicking this button opens the [Display of used stack size] (size display) window.



Display of used stack size

```
tn   total   use    rest
103  4096    4096   0
104  4096    0      4096
105  20480   0      20480
```

Figure 3-34 [Display of used stack size] (size display) Window

<Display items>

No.	Item	Description
1	tn	Task number
2	total	Total stack size
3	use	Used stack size
4	rest	Unused stack size

(1) Close button

Clicking this button closes the [Display of used stack size] window, and returns to the [CPMS DEBUGGER] window.

<Notes>

- The same data pattern as the initialization pattern specified in the [Initialize the stack] window must be specified as the check pattern.
- Calculating the used stack size is based on detection of a data pattern different from the specified check pattern in the stack page area used by the specified task. Therefore, if the stack page area begins with the same pattern as the specified check pattern, the used stack size cannot be calculated and displayed correctly.
- Figure 3-35 shows the correspondence of sizes displayed in the [Display of used stack size] (size display) window to sizes in the task-operating space.

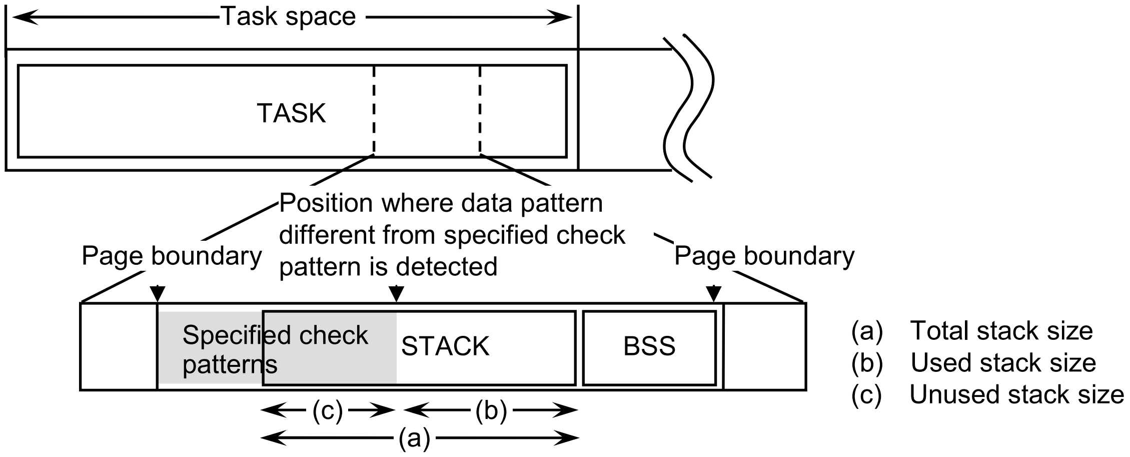
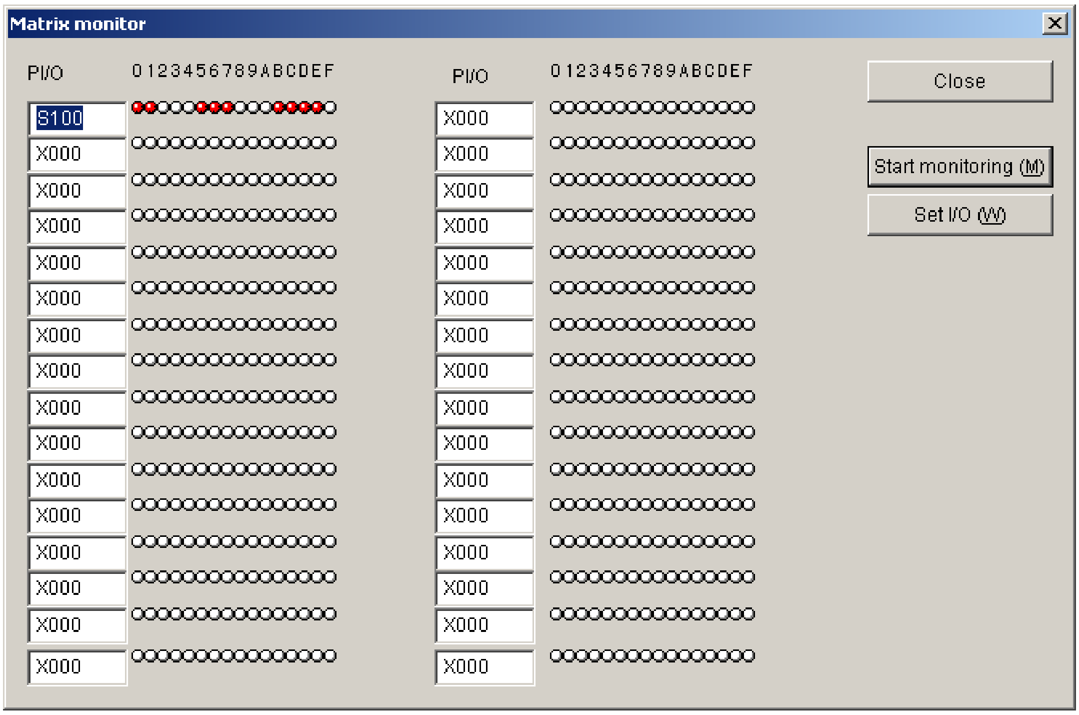


Figure 3-35 Sizes Displayed in the [Display of used stack size] (size display) Window

3.20 Matrix Monitor

Function: The “Matrix monitor” command opens the [Matrix monitor] window.

Procedure: Click the Matrix monitor button in the [CPMS DEBUGGER] window.



PI/O	0123456789ABCDEF	PI/O	0123456789ABCDEF
S100	●●○○○●●●○○○●●●●○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○
X000	○○○○○○○○○○○○○○○○	X000	○○○○○○○○○○○○○○○○

Figure 3-36 [Matrix monitor] Window

(1) Start monitoring button

Clicking this button starts the monitoring of set PI/O values.

(2) Set I/O button

Clicking this button opens the window to set a value of PI/O as shown in Figure 3-37. In this window, specify the bit status to set a PI/O value, and then click the OK button.

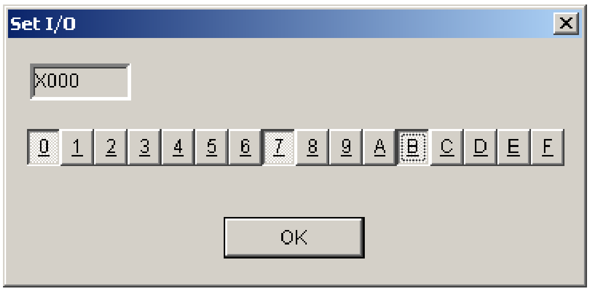


Figure 3-37 [Set I/O] Window

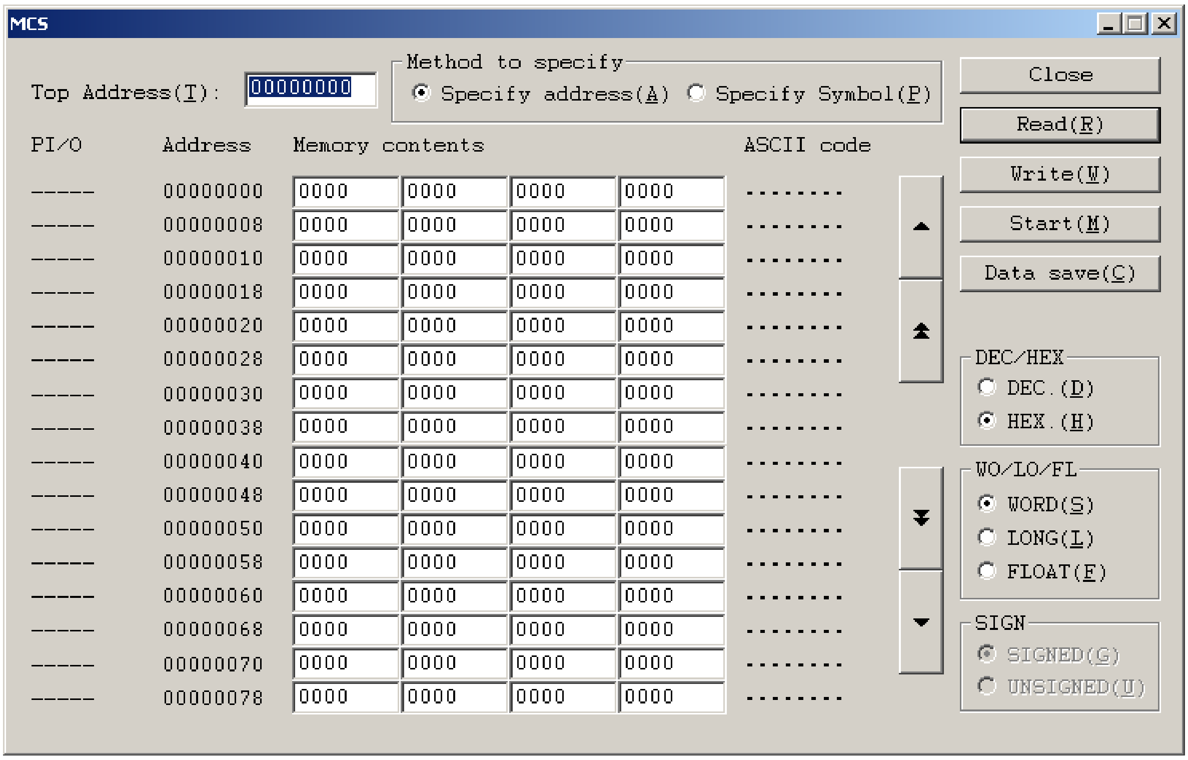
(3) Close button

Clicking this button closes the [Set I/O] window, and returns to the [CPMS DEBUGGER] window.

3.21 MCS

Function: The “MCS” command opens the [MCS] window.

Procedure: Click the MCS button in the [CPMS DEBUGGER] window.



Top Address(T): 00000000

Method to specify: ● Specify address(A) ○ Specify Symbol(P)

DEC/HEX: ○ DEC.(D) ● HEX.(H)

WO/LO/FL: ● WORD(S) ○ LONG(L) ○ FLOAT(F)

SIGN: ● SIGNED(G) ○ UNSIGNED(U)

PI/O	Address	Memory contents				ASCII code
-----	00000000	0000	0000	0000	0000	--------
-----	00000008	0000	0000	0000	0000	--------
-----	00000010	0000	0000	0000	0000	--------
-----	00000018	0000	0000	0000	0000	--------
-----	00000020	0000	0000	0000	0000	--------
-----	00000028	0000	0000	0000	0000	--------
-----	00000030	0000	0000	0000	0000	--------
-----	00000038	0000	0000	0000	0000	--------
-----	00000040	0000	0000	0000	0000	--------
-----	00000048	0000	0000	0000	0000	--------
-----	00000050	0000	0000	0000	0000	--------
-----	00000058	0000	0000	0000	0000	--------
-----	00000060	0000	0000	0000	0000	--------
-----	00000068	0000	0000	0000	0000	--------
-----	00000070	0000	0000	0000	0000	--------
-----	00000078	0000	0000	0000	0000	--------

Figure 3-38 [MCS] Window

(1) [Top Address] box

This box is used to specify the top address of the memory area of which the contents are to be displayed. To select a specification method, select the [Specify address] radio button or [Specify Symbol] radio button in the [Method to specify] group box.

(2) Read button

Clicking this button reads data from the displayed addresses of memory on the connected programmable controller (PCs).

(3) Write button

Clicking this button writes the displayed memory data to the displayed addresses of memory on the connected PCs.

(4) Start button

Clicking this button starts the monitoring of data at the displayed addresses.

(5) Stop button

Clicking this button stops the monitoring of data at the displayed addresses.

(6) Close button

Clicking this button closes the [MCS] window, and returns to the [CPMS DEBUGGER] window.

(7) Data save button

Clicking this button saves the data displayed in the [MCS] box as a text-format file.

(8) [DEC] and [HEX] radio buttons

These radio buttons are used to switch the display format in the [Memory contents] boxes between decimal and hexadecimal notation.

(9) [WORD], [LONG], and [FLOAT] radio buttons

These radio buttons are used to select the display format in the [Memory contents] boxes from among word, long word, and floating point.

(10) [SIGNED] and [UNSIGNED] radio buttons

These radio buttons are used to switch the display format in the [Memory contents] boxes between signed and unsigned.

3.22 Change Connection PCs

Function: The “Change connection PCs” command opens the window to set the type of communication (*) between a programmable controller (PCs) and the personal computer.

Procedure: Click the Change connection PCs button in the [CPMS DEBUGGER] window.

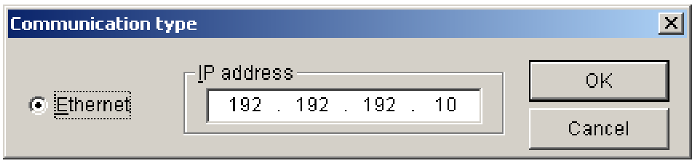


Figure 3-39 [Communication type] Window

(1) [IP address] box

This box is used to enter the IP address of the PCs to be connected.

(2) OK button

Clicking this button connects the personal computer to the PCs at the specified IP address. After this connection is made, the [CPMS DEBUGGER] window appears.

(3) Cancel button

Clicking this button cancels the changed type of communication, and returns to the [CPMS DEBUGGER] window.

(*) Communication type

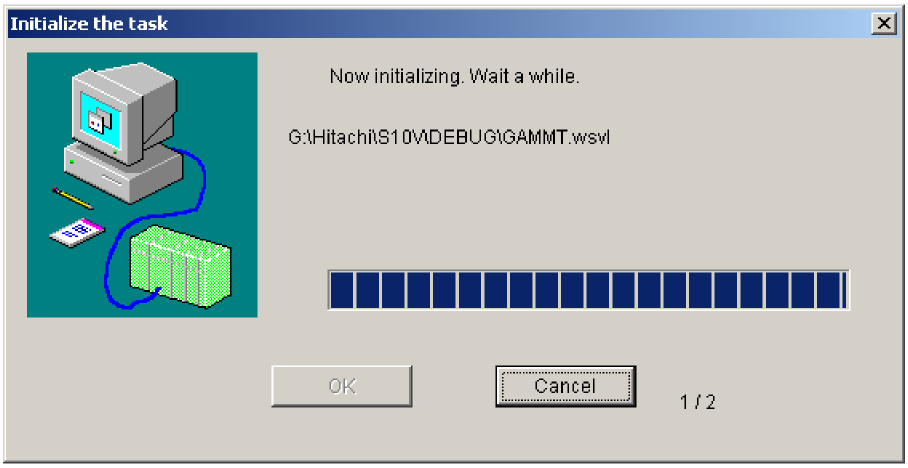
Only the Ethernet that connects the personal computer and the CMUs of PCs is supported for communication between CPMS Debugger (on the personal computer) and each PCs.

The S10V BASE SYSTEM must be used to set the IP address of each CMU.

3.23 Initialize the Task

Function: The “Initialize the task” command opens the window to initialize the task environment (*).

Procedure: Click the Initialize the task button in the [CPMS DEBUGGER] window.



Initialize the task

Now initializing. Wait a while.

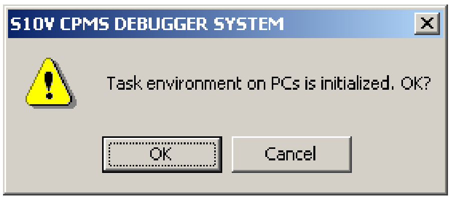
G:\Hitachi\S10V\DEBUG\GAMMT.wsvl

Figure 3-40 [Initialize the task] Window

<Confirmation message 1>

Clicking the Cancel button in the confirmation message 1 window cancels the processing, and returns to the [CPMS DEBUGGER] window.

When the OK button is clicked in the same window, the “confirmation message 2” window appears.



S10V CPMS DEBUGGER SYSTEM

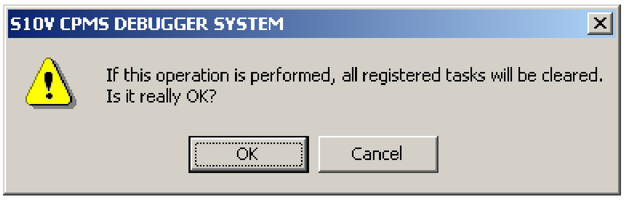
Task environment on PCs is initialized. OK?

<Confirmation message 2>

Clicking the Cancel button in the confirmation message 2 window cancels the processing, and returns to the [CPMS DEBUGGER] window.

Clicking the OK button in the same window aborts all tasks and then initializes the task environment.

After the task environment is initialized, the message window shown below appears, followed by the [CPMS DEBUGGER] window.



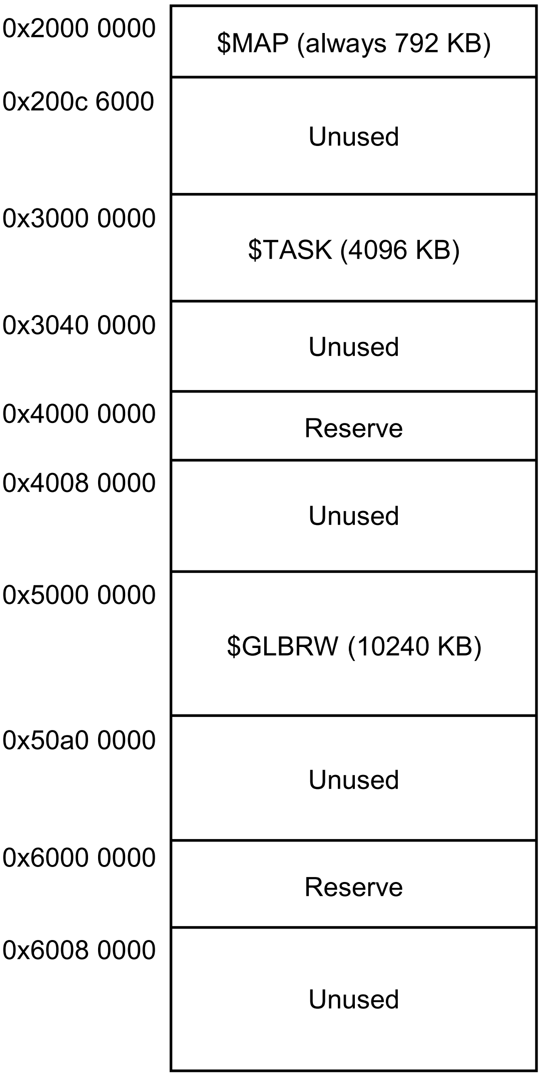
S10V CPMS DEBUGGER SYSTEM

If this operation is performed, all registered tasks will be cleared.
Is it really OK?

(*) Task environment

Initializing the task environment resets the task environment on the connected programmable controller (PCs) to the default environment defined by CPMS Debugger.

The following shows the default environment (area definition map) defined in CPMS Debugger:



0x2000 0000	$MAP (always 792 KB)
0x200c 6000	Unused
0x3000 0000	$TASK (4096 KB)
0x3040 0000	Unused
0x4000 0000	Reserve
0x4008 0000	Unused
0x5000 0000	$GLBRW (10240 KB)
0x50a0 0000	Unused
0x6000 0000	Reserve
0x6008 0000	Unused

$MAP: Area that stores management information on PCs memory.

$TASK: Area that stores a task (program).

$GLBRW: Read/write area that stores global data.

Figure 3-41 Default Environment Defined by CPMS Debugger

APPENDIXES

Appendix A Notes on Using Program Names

The user must be careful when using a subroutine program that has the same name as that of a subroutine prepared by the system. All subroutines prepared by the system are contained in library files. Each subroutine can be easily linked by using the optlnk command with the library option specified. However, when using the optlnk command to link a subroutine program with the same name as a subroutine prepared (in a library file) by the system, be sure to specify the object file defining the subroutine program as an argument of the optlnk command. Otherwise, the subroutine prepared by the system (and not the subroutine program) will be called from the library file incorrectly.

The library files prepared by the system and the subroutine names defined in the files are listed below. Refer to the subroutine names to avoid using duplicate names for the user's own subroutine programs during programming.

If a subroutine program with the same name as a subroutine prepared (in a library file) by the system must be used, specify the optlnk command to link the object file of said subroutine program before the optlnk command that links any of the subroutines in the library file. In this way, the incorrect calling of a subroutine with the same name can be avoided.

Listed below are the library files and subroutines prepared by the system.

Do not use a program name beginning with an underbar (_), which is the prefix indicating a reserved name in the system.

The configuration of libraries is as follows:

Library name	Contents	Remarks
libsh4nbmzz.lib	C-language subroutines Denormalized numbers: Treated as denormalized numbers Value-rounding method: Discarding	For details, refer to the manual for the shc compiler.
libsh4nbmdn.lib	C-language subroutines Denormalized numbers: Treated as 0 Value-rounding method: Discarding	
libcpms.lib	CPMS macro linkage subroutines	For details, refer to "CPMS GENERAL DESCRIPTION AND MACRO SPECIFICATIONS (Manual number SVE-3-201)."

<libsh4nbmzz.lib>

<libsh4nbmdn.lib>

- atof
- fpgetmask
- fpgetround
- fpgetsticky
- fpsetmask
- fpsetround
- fpsetsticky
- frexp
- ldexp
- memchr
- memcpy
- memset
- modf
- sscanf
- sprintf
- strcat
- strchr
- strcmp
- strcpy
- strcspn
- strlen
- strncat
- strncmp
- strncpy
- strpbrk
- strrchr
- strspn
- strtod
- strtol
- vsprintf
- fpcheck
- fpchecko
- acos
- asin
- atan
- atan2
- ceil
- exp
- fabs
- floor
- fmod
- log
- log10
- matherr
- pow
- cos
- sin
- cosh
- sinh
- sqrt
- tan
- tanh

<libcpms.lib>

- abort
- arsum
- asusp
- chap
- chml
- close
- cpms_copy
- ctime
- delay
- dhpctl
- dhpread
- exit
- free
- gfact
- cpms_ginfo
- gtime
- ioctl
- open
- pfree
- post
- printf
- prsrv
- queue
- read
- resume_env
- rleas
- rserv
- rsum
- save_env
- sfact
- stime
- susp
- timer
- usrdhpset
- usrelset
- wait
- write
- chkbmem
- chktaer
- dhpset
- elset
- geterrno
- gettimebase
- gtkmem
- getsysinfo
- gettaskinfo
- usrdhp
- usrel
- wrtmem

<libsysctl.lib>

- cardoff
- slotewrite
- cardon
- sloteclear
- cardstat
- dcmctl
- dcmrb
- dcmstat
- dsuctl
- dsustat
- ledctl
- pioctl
- ptnwrite
- regLRread
- hdutl
- sysRegRead
- sysRegWrite
- sysdo
- wdtset

Appendix B Sizes of Stack Areas Used for Library Functions

The table below lists the sizes of stack areas that are used for library functions.

(1/2)

Library	Function name	Stack size
C-language standard libraries (libsh4nbmdn.lib libsh4nbmzz.lib)	atof	408
	freexp	8
	ldexp	20
	memchr	0
	memset	12
	modf	40
	sscanf	528
	sprintf	752
	strcat	0
	strchr	0
	strcmp	20
	strcpy	24
	strcspn	4
	strlen	0
	strncat	4
	strncmp	4
	strncpy	0
	strpbrk	4
	strchr	12
	strspn	4
	strtod	408
	strtol	68
	vsprintf	752
	acos	196
	asin	184
	atan	156
	atan2	176
	ceil	28
	exp	92
	fabs	0
	floor	28
	fmod	40

(2/2)

Library	Function name	Stack size
C-language standard libraries (libsh4nbmdn.lib libsh4nbmzz.lib)	log	60
	log10	72
	pow	132
	cos	84
	sin	84
	cosh	112
	sinh	144
	sqrt	8
	tan	132
	tanh	156
libcrs.lib	fpgetmask	0
	fpgetround	0
	fpgetsticky	0
	fpsetmask	0
	fpsetround	0
	fpsetsticky	0
	fpcheck	0
	fpchecko	0
libfirad.lib	irglbad	0
	irsubad	0
libcpms.lib	memcpy (*)	28

(*) When a linked program calls the memcpy() function, the program does not call the memcpy() function stored in the C-language standard library, but calls the memcpy() function stored in the CPMS library.

Appendix C Task Creation Procedures

<Standard procedure using RPDP/S10V>

(1) Required software

If the operating system used is either Windows® 2000 or Windows® XP, the following are required:

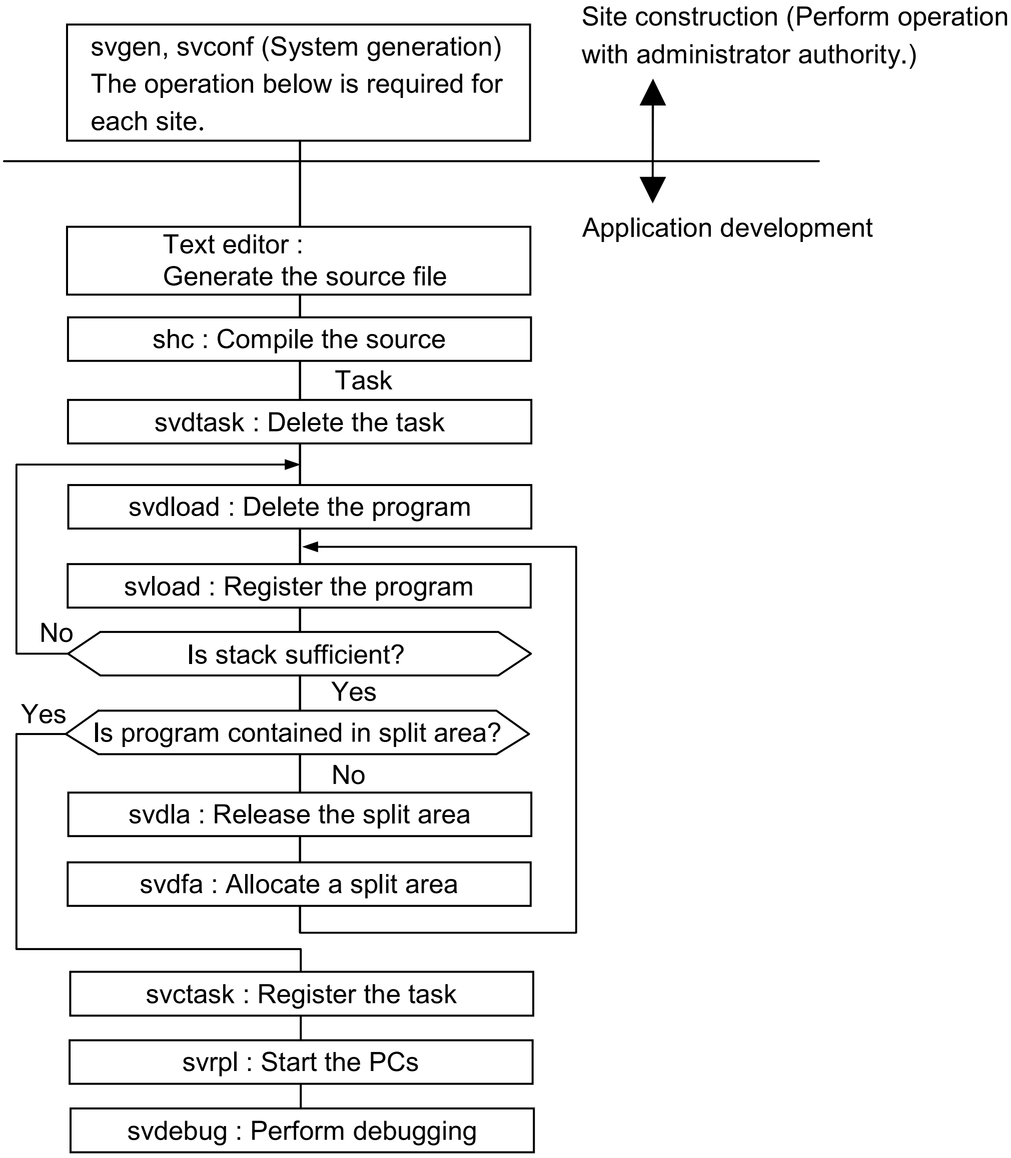
SH compiler (SuperH RISC engine C/C++ Compiler Ver. 7)
RPDP/S10V (Type: S-7895-10)

If it is Windows® 7 (32-bit), the following are required:

SH compiler (SuperH RISC engine C/C++ Compiler Ver. 9)
RPDP/S10V (Type: S-7895-63)

(2) Program development procedure

Follow the RPDP/S10V operation procedure to create the desired task.


svgen, svconf (System generation)
The operation below is required for each site.
Site construction (Perform operation with administrator authority.)
Application development
Text editor :
Generate the source file
shc : Compile the source
Task
svdtask : Delete the task
svdload : Delete the program
svload : Register the program
No
Is stack sufficient?
Yes
Yes
Is program contained in split area?
No
svdla : Release the split area
svdfa : Allocate a split area
svctask : Register the task
svrpl : Start the PCs
svdebug : Perform debugging

<Nonstandard procedure not using RPDP/S10V>

(1) Required software

If the operating system used is either Windows® 2000 or Windows® XP, the following are required:

SH compiler (SuperH RISC engine C/C++ Compiler Ver. 7)

S10V CPMS Debugger (Type: S-7895-07)

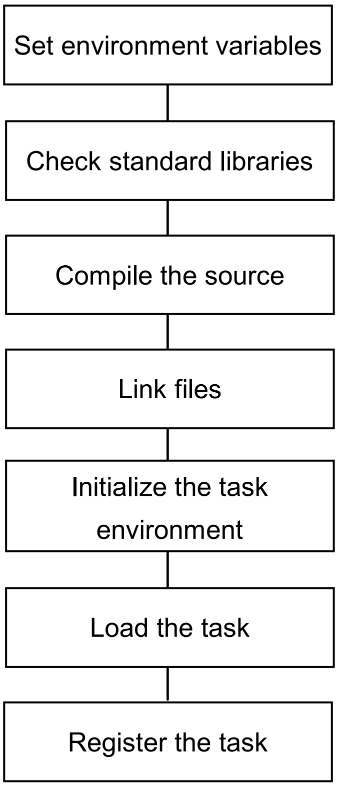
If it is Windows® 7 (32-bit), the following are required:

SH compiler (SuperH RISC engine C/C++ Compiler Ver. 9)

S10V CPMS Debugger (Type: S-7895-62)

(2) Program development procedure

Follow the operation procedure below to create the desired task.



· · · See Item (a), “Setting environment variables.”

· · · See Item (b), “Checking standard libraries.”

· · · See Item (c), “Compiling the source.”

· · · See Item (d), “Linking files.”

· · · See Section 3.23, “Initialize the Task.”

· · · See Section 3.2, “Task Loading and Register.”

· · · See Section 3.2, “task Loading and Register.”

(a) Setting environment variables

The following shows a typical setting of the execution environment for the shc Compiler Package:

Example of shc Compiler Package execution environment setting for S-7895-07P:

```
SHCPU=SH4
SHC_INC=C:\Hew2\Tools\Hitachi\Sh\7_1_1\include;C:\users\subsys\include;
           C:\Hitachi\S10V\Debug\include;C:\Hitachi\S10V\Debug\include\cpms
SHC_LIB=C:\Hew2\Tools\Hitachi\Sh\7_1_1\BIN                              ①
SHC_TMP=C:\Hitachi\S10V\Debug\tmp
HLNK_DIR=C:\Hitachi\S10V\Debug\lib;C:\users\subsys\lib                  ②
HLNK_TMP=C:\Hitachi\S10V\Debug\tmp
PATH=%PATH%;C:\Hew2\Tools\Hitachi\Sh\7_1_1\BIN                          ③
```

① This environment variable specifies "C:\Hew2\Tools\Hitachi\Sh\7_1_1\bin" as the directory to store the shc Compiler when the shc Compiler Package (Ver. 7.1.01) is installed with default settings.

② This environment variable specifies "C:\Hitachi\S10V\Debug\lib" as the directory to store the standard libraries and "C:\users\subsys\lib" as the directory to store user's original libraries.

③ This environment variable specifies "C:\Hew2\Tools\Hitachi\Sh\7_1_1\bin" as the directory to store the shc Compiler.

Example of shc Compiler Package execution environment setting for S-7895-62P:

```
SHCPU=SH4
SHC_INC=C:\Program Files\Renesas\Hew\Tools\Renesas\Sh\9_4_0\include;        ⑦
           C:\users\subsys\include;C:\Hitachi\S10V\Debug\include;
           C:\Hitachi\S10V\Debug\include\cpms
SHC_LIB=C:\Program Files\Renesas\Hew\Tools\Renesas\Sh\9_4_0\BIN           ④ ⑦
SHC_TMP=C:\Hitachi\S10V\Debug\tmp                                            ⑦
HLNK_DIR=C:\Hitachi\S10V\Debug\lib;C:\users\subsys\lib                       ⑤
HLNK_TMP=C:\Hitachi\S10V\Debug\tmp
PATH=%PATH%;C:\ProgramFiles\Renesas\Hew\Tools\Renesas\Sh\9_4_0\BIN        ⑥ ⑦
```

④ This environment variable specifies "C:\Program Files\Renesas\Hew\Tools\Renesas\Sh\9_4_0\bin" as the directory to store the shc Compiler when the shc Compiler Package (Ver. 9) is installed with default settings.

⑤ This environment variable specifies "C:\Hitachi\S10V\Debug\lib" as the directory to store the standard libraries and "C:\users\subsys\lib" as the directory to store the user's original libraries.

⑥ This environment variable specifies "C:\Program Files\Renesas\Hew\Tools\Renesas\Sh\9_4_0\bin" as the directory to store the shc Compiler.

⑦ Starting "C:\Program Files\Renesas\Hew\Tools\Renesas\Sh\9_4_0\shv9400env.bat" will set all of the environment variables SHC_INC, SHC_LIB, SHC_TMP, and PATH -- of these variables, SHC_TMP will be set to the value specified by the environment variable TEMP.

Table C-1 Environment Variables to be Set

No.	Environment variable	Content of setting
1	path	Use the path environment variable to add the directories to store the execution files of the installed compiler package. It is necessary to set the paths to such execution files as the compiler (shc), assembler (asmsh), and optimization linkage editor (optlnk). Therefore, specification of the path environment variable must not be omitted. Specification format: path= <name-of-execution-file-path> [;<name of existing path> ;...]
2	SHC_LIB	Use the SHC_LIB environment variable to specify the directories to store the compiler load modules and system include files. Specification of the SHC_LIB environment variable must not be omitted. Specification format: set SHC_LIB= <name-of-execution-file-path>
3	SHCPU	Use the SHCPU environment variable to specify the target CPU type. For this system, specify "SH4" in the SHCPU environment variable. Note that "SH1" is assumed to be the CPU type if the SHCPU environment variable is omitted. The CPU type can also be specified using the -cpu option. Specification format: set SHCPU= <CPU>
4	SHC_INC	Use the SHC_INC environment variable to specify the directory to store the include files of the compiler. To find a system include file, directories are searched in order of the directory specified by the include option, the directory specified in the SHC_INC environment variable, and the system directory (SHC_LIB). To find a user include file, directories are searched in order of the current directory, the directory specified by the include option, and the directory specified in the SHC_INC environment variable. Specification format: set SHC_INC= <include-path-name> [;<include-path-name> ;...]
5	HLNK_DIR	Use the HLNK_DIR environment variable to specify the directory to store the input files of the optimization linkage editor. To find a file specified by the Input or library option, directories are searched in order of the current directory and the directory specified in the HLNK_DIR environment variable. The library search path for the loader complies with the specification in the HLNK_DIR environment variable. Specification format: set HLNK_DIR= <include-path-name> [;<include-path-name> ;...]
6	SHC_TMP	Use the SHC_TMP environment variable to specify the directory where the compiler generates temporary files. set SHC_TMP= <directory>
7	HLNK_TMP	Use the HLNK_TMP environment variable to specify the directory where the linkage editor generates temporary files. set HLNK_TMP= <directory>

(b) Checking standard libraries

Note the following when using the shc compiler for compilation:

- andling of floating-point numbers

  The shc compiler allows the user to control the denormalizing and rounding of floating-point numbers through the specification of compile options.

  Note that the conditions for supporting the settings of compile options vary depending on the type of standard library. The tables below list the options to control the methods of handling and rounding denormalized numbers and the standard libraries corresponding to individual option settings.

  Specify the library to be used when files are linked. For how to specify the library, see Item (d), “Linking files.”

Table C-2 Options to Control Handling of Floating-point Numbers

	Specification	Option	Default
Handling of denormalized numbers	Treating a denormalized number as 0	-denormalization=off	Treating as 0
	Treating a denormalized number as a denormalized number (*)	-denormalization=on	
Method of rounding result values	Discarding fractions below significant digits	-round=zero	Discarding
	Rounding off fractions below significant digits	round=nearest	

(*) When the program created is executed, each denormalized number is treated as 0 because the SH4 (SH7751), the S10V series CPU, does not support denormalized numbers.

Table C-3 Correspondence between Methods of Handling Floating-point Numbers and Standard Libraries

	-denormalization	-round	Standard library
Option specifications	off	zero	libsh4nbmzz.lib
	on	zero	Not supported
	off	nearest	Not supported
	on	nearest	libsh4nbmdn.lib

Table C-4 Library Contents

Library name	Library contents	Remarks
libsh4nbmzz.lib	C-language subroutines Denormalized numbers: Treated as denormalized numbers Value-rounding method: Discarding	For details, refer to the manual for the shc compiler.
libsh4nbmdn.lib	C-language subroutines Denormalized numbers: Treated as 0 Value-rounding method: Discarding	
libcpms.lib	CPMS macro linkage subroutines	For details, refer to “CPMS GENERAL DESCRIPTION AND MACRO SPECIFICATIONS (Manual number SVE-3-201).”

The table below lists the files provided by RPDP/S10V and the S10V CPMS Debugger System.

Table C-5 Files Provided by S10V CPMS Debugger

(1/2)

File name	Directory storing files at installation of S10V CPMS Debugger (default) (*1)	Content
libcpms.lib	C:\HITACHI\S10V\DEBUG\LIB	CPMS macro linkage subroutines
libcrs.lib	C:\HITACHI\S10V\DEBUG\LIB	Subroutines to control IEEE floating-point processing environment
libsh4nbmdn.lib	C:\HITACHI\S10V\DEBUG\LIB	C-language subroutines for SH compiler
libsh4nbmzz.lib	C:\HITACHI\S10V\DEBUG\LIB	C-language subroutines for SH compiler
libsysctl.lib	C:\HITACHI\S10V\DEBUG\LIB	Subroutines to manage CPMS macro system (*2)
cpms_dhp.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for DHP
cpms_elog.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for user error log
cpms_errno.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for error number definitions
cpms_macro.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file to use CPMS macros
cpms_table.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for task execution environment table
cpms_types.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for type declarations to use macros
cpms_ulsub.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for built-in subroutines
ieeefp.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for floating-point registers
sdio.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for SDIO driver
stdio.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for standard I/O
stdlib.h	C:\HITACHI\S10V\DEBUG\INCLUDE	Include file for standard libraries
cpms_adbbf.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for ADB table
cpms_cardctl.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	(Unused)
cpms_dcm.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	(Unused)
cpms_log.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for error log numbers
cpms_oscb.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for OSCB table
cpms_pucomt.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	(Unused)
cpms_rserv.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for shared resource
cpms_syscb.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for SYSCB table

(*1) It is assumed that S10V CPMS Debugger is installed using drive C.

(*2) The subroutines include wdtsert and TimebaseToSecs.

Table C-5 Files provided by S10V CPMS Debugger

(2/2)

File name	Directory storing files at installation of S10V CPMS Debugger (default) (*1)	Content
cpms_taskenv.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for task execution environment data table
cpms_tcb.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for TCB table
cpms_timer.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file related to timer
cpms_ucb.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for UCB table
cpms_ulsubln.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file to link built-in subroutines
cpms_usr_param.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for parameter definitions
dhpinfo.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for DHP information
errno.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for error number definitions
macro_base_p.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file to use CPMS macros
macro_id.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for CPMS macro number definitions
macro_rpdp_p.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for RPDP
R700_parameter.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for table address definitions
R700_usr_param.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for system definitions
sysctl.h	C:\HITACHI\S10V\DEBUG\INCLUDE\CPMS	Include file for system control

(*1) It is assumed that S10V CPMS Debugger is installed using drive C.

(c) Compiling the source

CPMS Debugger is designed on the premise that the Hitachi’s Microcomputer Development Environment System “SuperH RISC engine C/C++ Compiler Package Ver.7 or later” (hereinafter called the shc compiler) is used as the compiler or assembler.

Details of C compiler options

The following describes the compiling method using the shc compiler and some precautions on compilation:

For detailed shc compiler specifications, refer to the manual for the shc compiler.

● Command format

shc [Δ<option>...][Δ<file-name>[Δ<option>...]...]

(Example) shcΔtest1.c Δtest2.c [Enter]

● Setting environment variables

When using the shc compiler, set proper values in the environment variables listed in Table C-1.

● Generating and saving a compile list (shc)

Before compilation, generate and save a compile list that is required to calculate the size of the stack to be used by the task. To generate a compile list, specify the -listfile option as described below.

Specify the -listfile option at a position before the C-language source file to be compiled.

Note that specifying the -listfile option after the link with C-language source file will generate a compile list only for the last file.

■ Specification of compile list generation

```
-listfile [ = < list-file-name >] -show=source, object
```

If no list file name is specified in the -listfile option, the name of the compile list will combine the source file name and extension “lst”.

(Example)

◆ shc Δ-listfile Δtest1.c Δtest2.c [Enter]

The listfile option is valid for both test1.c and test2.c.

◆ shc Δtest1.c Δtest2.c Δ-listfile [Enter]

The listfile option is valid only for test2.c.

Press the [Enter] key.

(d) Linking files

Link compiled object files as described below to generate the files (absolute modules) having the format that enables the files to be sent to programmable controllers.

Use the optlnk command to link files. The following describes how to start the linker:

Enter the command below using the MS-DOS command prompt.

- Command format

```
optlnk -noprelink -form=absolute -start=P,D,C/address-1,S/address-2,B/address-3
        -output=file name of absolute module to be generated -library=libcpms.lib
        -library=libsh4nbmdn.lib (-list=mapping-file -show=symbol) (-stack)
         user-object,library
```

(Example)

```
 c:\tmp>optlnk -noprelink -form=absolute -start=P,D,C/30004000,S/30005000,
          B/30006000
        -output=task1.elf -library=libcpms.lib -library=libsh4nbmdn.lib -list=task1.map
         -stack task1.obj
```

address-1: 30004000
address-2: 30005000
address-3: 30006000
file name of absolute module to be generated: task1.elf
mapping-file: task1.map
user-object: task1.obj

Table C-6 Optlink Specification Format

No.	Item	Specification format	Function
1	Library file	library=file-name	Specifies the library.
2	Suppression of pre-linker activation	noprelink	Suppresses pre-linker activation to increase processing speed. Pre-linking can be omitted when not using the C++ template and run-time type check functions.
3	Output format	form=absolute	Specifies the format of the linker output file that is applied to load module generation. This parameter is required.
4	Output file	output=file-name	Specifies the linker output file that is used to generate a load module.
5	List file	list=file-name	Outputs a list file that is used to generate a mapping file.
6	List content	show=symbol	Outputs a list file that is used to generate a mapping file.
7	Section start address	start=section/ address[,...]	Allocates the specified section to the specified address. This parameter is used for program allocation. This parameter is required due to the map configuration described later. The user must determine addresses 1 and 2 in consideration of their usage for program or stack size.
8	Stack information file	stack	Outputs the information file on used stack size.

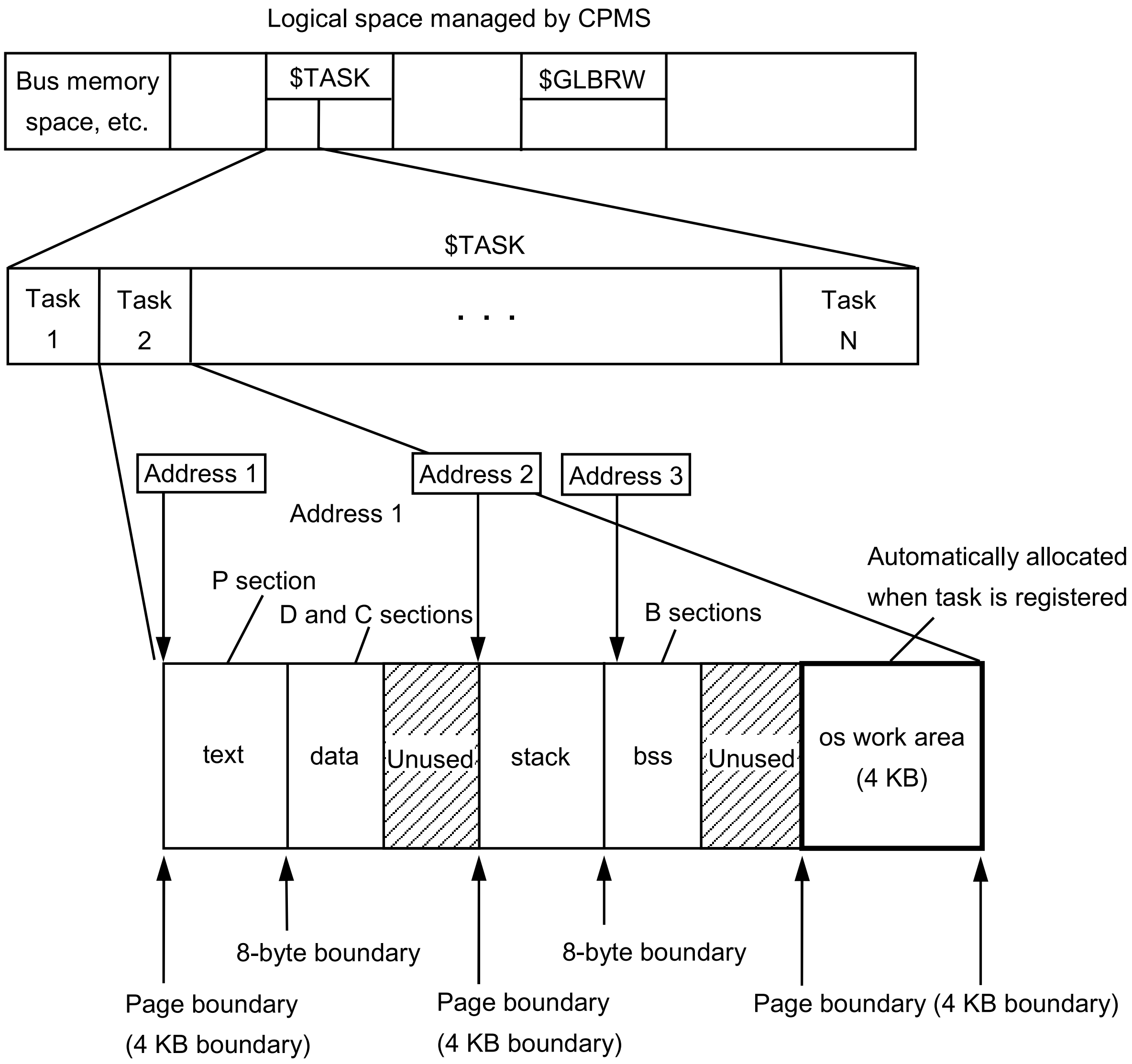
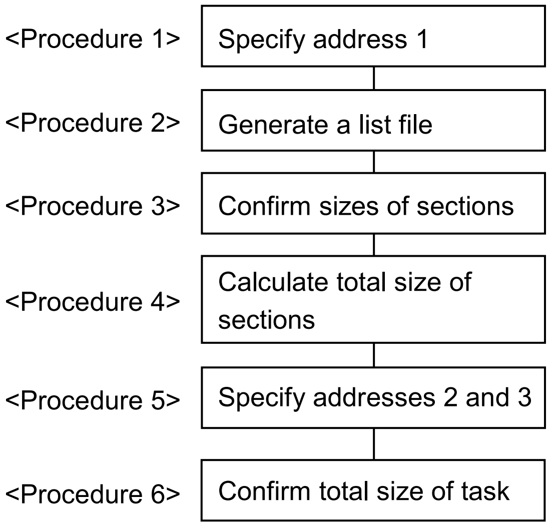


Figure C-1 Task Configuration

The $TASK area stores tasks (programs).
The $GLBRW area is a read/write area used to store global data.
The text area stores the procedures of the program.
The data area stores the initial data of the program.
The stack area is a dynamic work area used by the task.
The bss area is a static work area used by the program.
The os work area is a dynamic work area used by the os. This area is automatically allocated as a page that follows the bss area when the task is registered.
The text area must be aligned on a page boundary (4 KB boundary). The data, bss, and stack areas must each be aligned on an 8-byte boundary.
The stack area must be aligned on a page boundary.
Addresses 1, 2, and 3 must be specified by the user.

Follow the procedure below to obtain addresses 1 and 2.


<Procedure 1> Specify address 1
<Procedure 2> Generate a list file
<Procedure 3> Confirm sizes of sections
<Procedure 4> Calculate total size of sections
<Procedure 5> Specify addresses 2 and 3
<Procedure 6> Confirm total size of task

(Example)
(Source file) File name: task1.c

```
int b1;
int d1 = 10;
static int b2 ;
static int d2 = 100 ;
main()
{
        static int b3 ;
        static int d3 = 1000 ;

          int s1;
          int s2 = 20 ;
          exit(0);
}
```

Be sure to add exit(0); on the last line.

<Procedure 1> (Specifying address 1)

Arbitrarily specify the start address (address 1) of the text area. Address 1 must be on a page boundary (4 KB boundary) and outside any range of addresses used by other tasks. Address 1 must be specified in the optlnk command that is executed for linking files as described in Item (d), “Linking files.”

<Procedure 2> (Generating a list file)

Specify the -list option in the shc command to generate a list file at compilation.

```
F:\tmp>shc task1.c-list
```

<Procedure 3> (Confirming sizes of sections)

Check the value of “frame size” in the generated list file (file name: task1.lst) to confirm the stack size, and confirm the sizes of sections listed under “SECTION SIZE INFORMATION” in the generated list file.

A detailed method of calculating stack size is described later.

```
(Generated list file) File name: task1.lst

SCT OFFSET     CODE          C LABEL       INSTRUCTION OPERAND       COMMENT
                                                                                     Stack size
P     00000000               _main:                                ; function: main
                                                                   ; frame size=0
      00000000 D201                        MOV.L        L11+2,R2     ; _exit
      00000002 422B                        JMP          @R2
～～～
******* SECTION SIZE INFORMATION *******

PROGRAM   SECTION (P):                                             0000000C Byte(s)
CONSTANT SECTION (C):                                              00000000 Byte(s)
DATA          SECTION (D):                                         0000000C Byte(s)
BSS           SECTION (B):                                         0000000C Byte(s)
```

<Procedure 4> (Calculating total size of sections)

Calculate the sum of the sizes of program section (P), constant section (C), and data section (D). In the example below, the total size is 24 bytes. This total size indicates that the text and data areas can be contained on one page (4 KB).

```
<Size of each section>
PROGRAM    SECTION (P):                    0000000C Byte(s)
CONSTANT   SECTION (C):                    00000000 Byte(s)
DATA       SECTION (D):                    0000000C Byte(s)
```

(P) (C) (D)

0xC + 0x0 + 0xC = 0x18 bytes (decimal 24 bytes)

<Procedure 5> (Specifying addresses 2 and 3)

Specify the start address (address 2) of the stack area and the start address (address 3) of the bss area.

The stack and bss areas must be allocated on a page other than that containing the text and data areas.

Although the stack size obtained in procedure 3 is 0, allocate a one-page (4 KB) area to the stack for future use.

Accordingly, the start address of the stack area must follow the page of the text and data areas (in other words, the start address must be at the top of the second page).

The start address of the bss area must follow the page of the stack area (or be at the top of the third page).

Since the size of the bss area was confirmed to be 12 bytes in procedure 3, the bss area can be contained on the third page.

Addresses 2 and 3 must be specified in the optlnk command that is executed for linking files as described in Item (d), “Linking files.”

<Procedure 6> (Confirming total size of task)

The os work area must be aligned on a page boundary (4 KB boundary), and always have a size of 4 KB. Therefore, the os work area is allocated on the fourth page, and the total size of the task amounts to 16 KB.

A new task can be allocated in the area that follows the end address of the os work area.

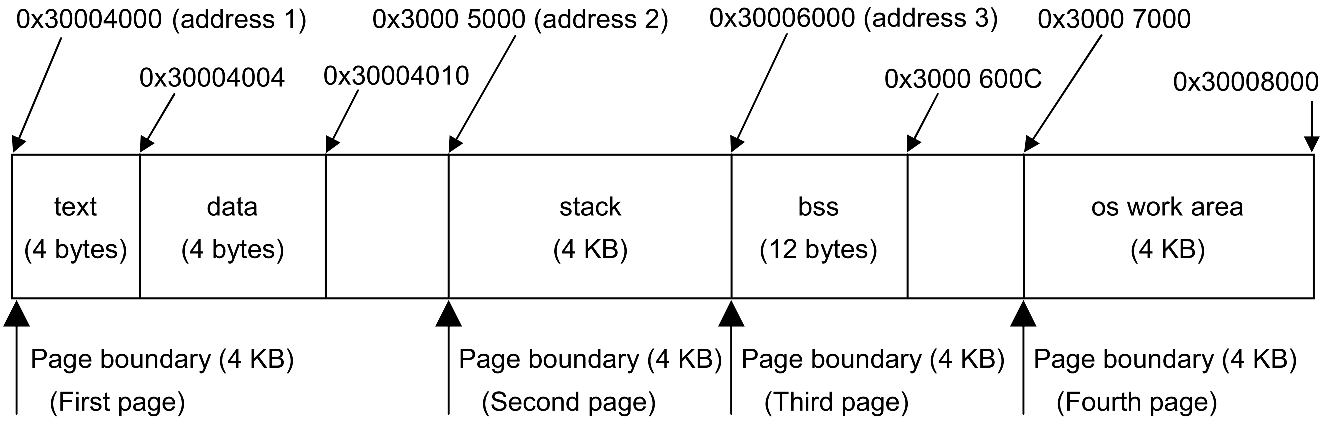


Figure C-2 Map Configuration

<Method of calculating stack size>

To calculate the size of the stack area to be used, calculate the size of the stack area to be used by each function, each component of the program, and then calculate the total size of the stack to be used based on the calling relationship between functions.

(1) Calculating stack area to be used by each function

The size of the stack area to be used by each function can be determined from the value of “frame size” in the object list output by the compiler.

An actual example is given below.

■ Source code

```
extern int h(char , int *, double);
int h(char a, register int *b, double c)
{
            char *d;

            d = &a;
            h(*d,b,c);
            {
                      register int i;

                      i = *d;
                      return i;
            }
}
```

- Object list

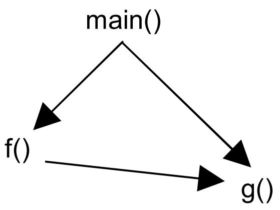
```
SCT OFFSET    CODE         C LABEL      INSTRUCTION OPERAND       COMMENT

P  00000000              _h:                                      ; function: h
                                                                 ; frame size=12
   00000000 2FE6                     MOV.L        R14,@-R15
   00000002 4F22                     STS.L       PR,@-R15
```

In this example, the size of the stack area to be used by the h function is 12 bytes, which is indicated as the value of "frame size" in the "COMMENT" column.

(2) Calculating total stack size based on calling relationship

Calculate the total size of the stack area to be used based on the calling relationship between functions.

Figure C-3 shows an example of calculating the size of stack area to be used based on the calling relationship between functions.



Function name	Size of stack to be used (in bytes)
main	24
f	32
g	24

Figure C-3 Calling Relationship between Functions and Sizes of Stack to be Used

When the g function is called via the f function as shown in the example above, the size of the stack area is as shown in Table C-7.

Table C-7 Example of Calculating Stack Size

Calling route	Stack size (in bytes)
main(24) → f(32) → g(24)	80
main(24) → g(24)	48

As shown in the example, the size of the stack area to be used must be calculated when calling the function at the deepest level of calling, and then a stack area that has at least the maximum size calculated must be allocated.
When standard library functions are to be used, the size of the stack area to be used by the library functions must also be considered. For the sizes of stack areas used by standard library functions, see Appendix B, "Sizes of Stack Areas Used for Library Functions."
To calculate the stack size for a function that is called recursively, use formula "'stack size for function' × 'maximum number of recursive calls'."
Even if the source program does not use any library function, it may include some links to the run-time routines required for program execution. The size of the stack area to be used by a run-time routine can be confirmed by a stack analysis tool (described below in the explanation of the method of confirming the used stack size).

<Method of confirming the used stack size>

When program files are linked by the optlnk command with the -stack option specified, an information file on used stack size can be generated.
The total size of the stack used by the program can be obtained by using the stack analysis tool (an accessory to the compiler package) to analyze the generated information file on used stack size.

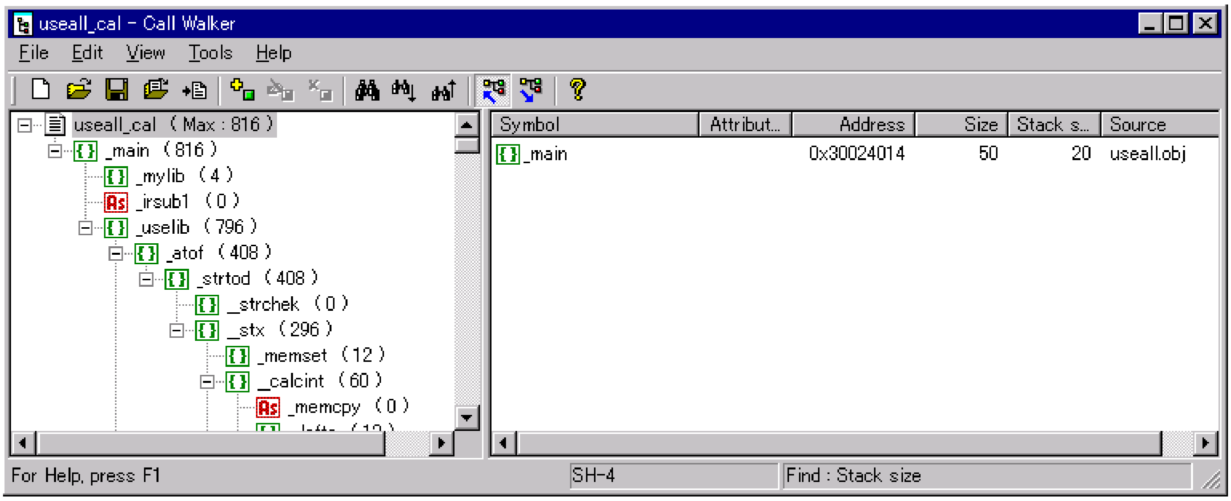
● Generation of an information file on used stack size
  When program files are linked by the optlnk command with the -stack option specified, an information file on used stack size is generated.
  The information file on used stack size is generated in the current directory at linkage, and its file name has extension ".sni".

● Method of using the stack analysis tool
  Follow the procedure below to start the stack analysis tool and display the sizes of stack areas used by programs and subprograms.
  For details on how to use the stack analysis tool, refer to the manual for the shc compiler and the Help of the stack analysis tool.

  ① If the shc compiler used is of Ver. 7, click the [Start] button on the Windows® desk top, select [(All) Programs], select [Hitachi Embedded Workshop 2], and then click [Hitachi Call Walker] to start the stack analysis tool.
  If it is of Ver. 9, click the [Start] button on the Windows® desktop and then select [(All) Programs] – [Renesas] – [High-performance Embedded Workshop] – [Call Walker] to start the stack analysis tool.

  ② Select [Import Stack file...] from the [File] menu of the stack analysis tool to open a dialog box, specify the size information file on used stack generated by the loader in the [File name] field in the dialog box, and then click the [Open] button.

- Example of display by stack analysis tool



```
useall_cal （Max : 816）
  _main （816）
    _mylib （4）
    _irsub1 （0）
    _uselib （796）
      _atof （408）
        _strtod （408）
          _strchek （0）
          _stx （296）
            _memset （12）
            _calcint （60）
              _memcpy （0）
```

Symbol	Attribut...	Address	Size	Stack s...	Source
_main		0x30024014	50	20	useall.obj

For Help, press F1

SH-4

Find : Stack size

- Notes on analyzing the sizes of the stack used by loaded programs and subprograms

  The stack analysis tool indicates that the size of the stack used by a program or subprogram written in assembly language is 0 byte. Therefore, note the following points when using the stack analysis tool to analyze the sizes of stack areas used by loaded programs and subprograms :

  - Size of stack used by memcpy()

    When a loaded program or subprogram calls the memcpy() function, the program does not call the memcpy() function stored in the C-language standard library, but calls the memcpy() function stored in the CPMS library. The stack analysis tool indicates that the size of the stack used by the memcpy() function in the CPMS library is 0 byte. However, the memcpy() function in the CPMS library actually uses 28 bytes of stack.

    Select the [Modify] command from the [Edit] menu of the stack analysis tool, change the stack size of the memcpy() function to 28 bytes, and then recalculate the used stack size.

    The user can determine whether a loaded program or subprogram uses the memcpy() function by using the search function of the stack analysis tool to search for memcpy.

  - Size of stack used by program or subprogram written in assembly language

    Analyze the stack used by a program or subprogram written in assembly language in the same way as analyzing the size of the stack used by memcpy(). Select the [Modify] command from the [Edit] menu of the stack analysis tool, change the stack size, and then recalculate the used stack size.

Appendix D　Error Messages

(1/2)

No.	Error message	Corrective action
1	Task number %d is dormant	Check the task status, and then retry operation.
2	Task number %d is not dormant	Check the task status, and then retry operation.
3	Task number %d is already suspend	Check the task status, and then retry operation.
4	Task number %d is not suspend	Check the task status, and then retry operation.
5	Task number %d is not registered	Check the specified task number.
6	User Task not running	The user task is not running.
7	Invalid address error	Check contents of the CMU error log. (*3)
8	Invalid address set	Check the set address of ADT.
9	Processor connection table is full	Wait until another user ends operation, and then retry operation.
10	Task number %d is not idle	Check the task status, and then retry operation.
11	Cannot get register information	Contact the system administrator. (*1) Check contents of the CMU error log. (*3)
12	Cannot register timer event in TRB	The number of registered timer events exceeded the maximum limit.　Reduce the number of registered timer events.
13	Must specify address in text space	Check the set breakpoint address .
14	Cannot get TCB	Contact the system administrator. (*1) Check contents of the CMU error log. (*3)
15	ADT channel is already set	Delete the ADT, and then execute operation.
16	Break task is not found	The breakpoint may have been deleted by another PC. Check contents of the DHP trace information. (*4) Check the task state and breakpoint setting.
17	systemcall error (%s, errno = %d)	Contact the system administrator. (*1) Check contents of the CMU error log. (*3)
18	DHP data read error	Check contents of the CMU error log. (*3) Contact the system administrator. (*1)
19	Cannot DHP trace ON/OFF	Check contents of the CMU error log. (*3) Contact the system administrator. (*1)
20	connection timeout	Check the network connection status (IP address and communication cable), and then retry operation. (*2)

%d: Error code　%s: System call name

(2/2)

No.	Error message	Corrective action
21	connection refused	Check the network connection status (IP address and communication cable), and then retry operation. (*2) Check contents of the CMU error log. (*3)
22	connection cut	Check the network connection status (IP address and communication cable), and then retry operation. (*2) Check contents of the CMU error log. (*3)
23	connection reset	Check the network connection status (IP address and communication cable), and then retry operation. (*2) Check contents of the CMU error log. (*3)
24	server closed	Check the network connection status (IP address and communication cable), and then retry operation. (*2) Check contents of the CMU error log. (*3)
25	port busy	Wait until another user ends operation, and then retry operation.
26	socket creat error	Check contents of the CMU error log. (*3)
27	no buffer	Check contents of the CMU error log. (*3)
28	network not reached	Check the network connection status (IP address and communication cable), and then retry operation. (*2)
29	network down	An error occurred in the network connection interface of the CMU. Check contents of the CMU error log. (*3)
30	port No error	Contact the system administrator. (*1) Check contents of the CMU error log. (*3)
31	IP address error	Contact the system administrator. (*1) Check contents of the CMU error log. (*3)
32	memory attach failed	Check contents of the CMU error log. (*3)
33	fatal error	Contact the system administrator. (*1) Check contents of the CMU error log. (*3)
34	rc = %d	Check the network connection status (IP address and communication cable), and then retry operation. (*2) Check contents of the CMU error log. (*3)

(*1) If an error message “Contact the system administrator.” appears in the Corrective action column, collect the data for investigation and DHP trace. The data files to be collected are as follows:

<Data for investigation>

C:\Hitachi\S10V\DEBUG\Debugger.log (for S10V CPMS Debugger installed using drive C)

<DHP trace>

File saved by the Save button in the [Display DHP trace] window as described in Section 3.17, “Setup of DHP Status”

(*2) Method of checking network connection status by using the Ping function of the personal computer

Use a general-purpose personal computer with Windows® 2000, Windows® XP, Windows® 7 (32-bit) installed to check the Ethernet connection and IP address setting of the relevant CMU.

Use the Ping command to confirm that the IP connection of the CMU is normal as follows:

① Open the [Start] menu, select [(All) Programs], select [Accessory], and then click [Command prompt] to open the [Command Prompt] window.

② Enter the Ping command to perform a basic test on communication between the personal computer and linked unit. The input format of the Ping command is “Ping [IP-address]” or “Ping [host-name]”.

<Example of IP address specification> Ping 192.192.192.13

When the Ethernet connection between the CMU and personal computer is normal, the following message will appear:

```
Pinging 192.192.192. 13 with 32 bytes of data
Reply from 192.192.192. 13: bytes=32 time=2ms TTL=32
Reply from 192.192.192. 13: bytes=32 time=1ms TTL=32
Reply from 192.192.192. 13: bytes=32 time=1ms TTL=32
Reply from 192.192.192. 13: bytes=32 time=1ms TTL=32
C:\WINDOWS>
```

③ If the Ethernet connection is abnormal, the following (time-out) message will appear:

```
Pinging 192.192.192. 13 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
C:\WINDOWS>
```

(*3) For how to check the error log, see Section 3.12, “Error Log.”

(*4) For how to check the DHP trace, see Section 3.17, “Setup of DHP Status.”